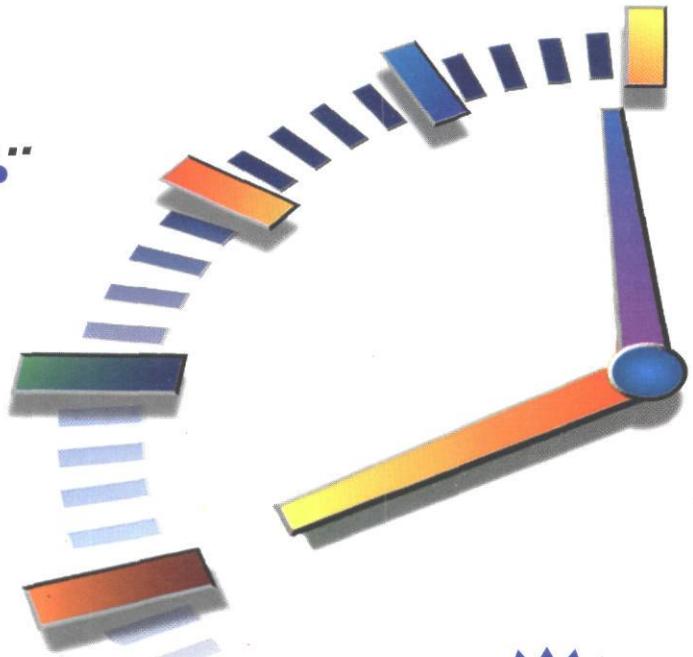
РУКОВОДСТВО ДЛЯ НАЧИНАЮЩИХ

Когда есть время только на ***ответы...***

Двадцать четыре практических занятия



SAMS

Освой самостоятельно

Описан JavaScript 1.5

JavaScript™

Второе издание

Майкл Монкур

SAMS

ВИЛЬЯМС

за 24 часа

http://ITBookz.ru/

Внимание! Файл получен с помощью ссылки, расположенной в каталоге сайта ITBookZ.ru.

Описание сайта:
На сайте Вы найдёте много электронных книг, которые можно бесплатно скачать для **ознакомления**, без регистрации! Сайт полезен человеку, так или иначе занятому в IT сфере, т.е. работающему с компьютером.

Темы электронных книг:

- Программирование
- САПР
- Операционные системы
- WEB-Дизайн
- MS Office
- Сети
- Базы данных
- Безопасность
- 1С
- Графика, Дизайн, 3D
- Железо
- Создание игр
- Математика
- Мультимедиа
- Журналы IT тематики.
- И многое другое!

Море электронных книг по компьютерной тематике!
Постоянное пополнение!
Заходите, выбирайте и скачивайте!
ITBookZ.ru

Авторское право:
Данная книга предоставлена исключительно для **ознакомительных** целей и должна быть удалена с вашего компьютера или любого иного носителя информации сразу после поверхностного ознакомления с содержанием.

Копируя и сохраняя данную книгу, Вы принимаете на себя всю ответственность, согласно действующему международному законодательству, а именно закону об авторском праве и смежными с ним законами.

Публикация данного документа не преследует за собой никакой коммерческой выгоды, а является рекламой бумажного аналога.

Правообладателям:
Все авторские права сохраняются за правообладателем. Если Вы являетесь автором данного документа и хотите дополнить его или изменить, уточнить реквизиты автора или опубликовать другие документы, пожалуйста, свяжитесь с нами через форму обратной связи на сайте. Мы будем рады услышать ваши пожелания и принять меры по устранению недоразумений.

С уважением,
Администрация ITBookZ.ru



http://ITBookz.ru/

http://ITBookz.ru/

SAMS

Освой самостоятельно

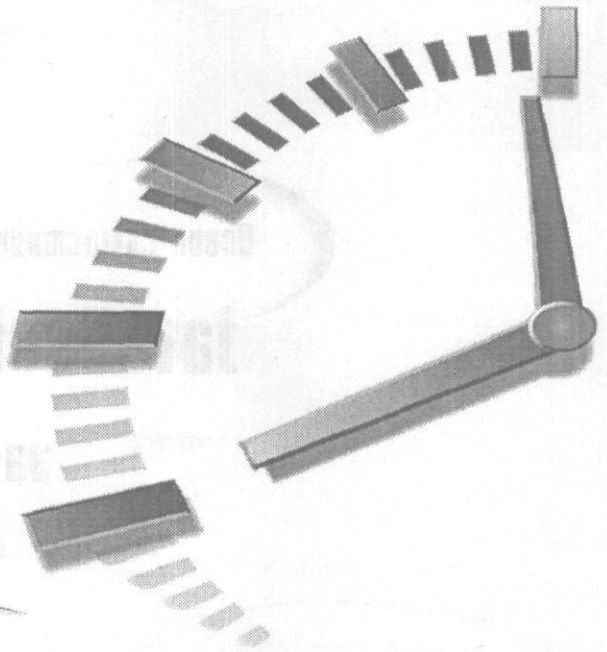
JavaScript

за 24 часа

Второе издание

http://ITBookz.ru/

Michael Moncur



SAMS Teach Yourself

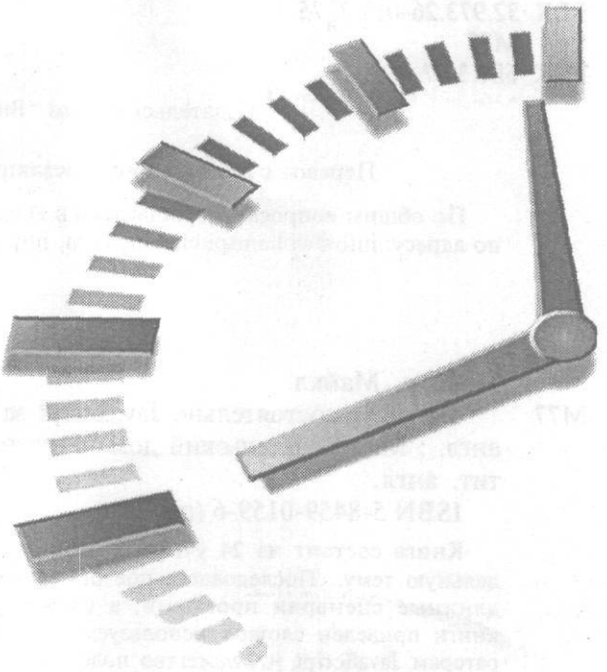
JavaScript in 24 Hours

2nd Edition

SAMS
A Division of Macmillan Computer Publishing
201 West 103rd St., Indianapolis, Indiana, 46290 USA

http://ITBookz.ru/

Майкл Монкур



SAMS

Освой самостоятельно JavaScript за 24 часа

Второе издание

Издательский дом "Вильяме"
Москва ♦ Санкт-Петербург ♦ Киев
2001

http://ITBookz.ru/

ББК 32.973.26-018.2$_{я}$75
М77
УДК 681.3.07

Издательский дом "Вильямс"

Перевод с английского и редакция *И. В. Василенко*

По общим вопросам обращайтесь в Издательский дом "Вильяме"
по адресу: info@williamspublishing.com, http://www.williamspublishing.com

Монкур, Майкл.
М77 Освой самостоятельно JavaScript за 24 часа, 2-е издание. : Пер. с англ. : М. : Издательский дом "Вильяме", 2001. — 320 с. : ил. —Парал. тит. англ.
ISBN 5-8459-0159-6 (рус.)

Книга состоит из 24 учебных занятий, каждое из которых охватывает отдельную тему. Последовательное описание возможностей JavaScript 1.5 делает длинные сценарии простыми, а сложные понятия — доступными. В конце книги приведен словарь используемых терминов, краткий справочник по операторам JavaScript и множество полезных ссылок. Каждое занятие оканчивается тестовыми вопросами и упражнениями, выполнив которые, вы закрепите пройденный материал и расширите свои познания.

В книге описана последняя версия известного языка подготовки сценариев — JavaScript. Многочисленные советы, замечания и предостережения обращают внимание читателя на важные тонкости создания сценариев и возможные ошибки. Примеры программных кодов и иллюстрации упрощают усвоение нового материала и делают книгу доступной для начинающих пользователей любого уровня.

ББК 32.973.26-018.2$_{я}$75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм. Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства Sams Publishing.

Authorized translation from the English language edition published by Sams Publishing, Copyright © 2000
All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.
Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2001

ISBN 5-8459-0159-6 (рус.)
ISBN 0-672-32025-8 (англ.)

© Издательский дом "Вильяме", 2001
© Sams Publishing, 2000

http://ITBookz.ru/

Оглавление

Введение 20

Часть I. Начало начал 21

1-й час. Знакомство с JavaScript 22
2-й час. Создание простых сценариев 31
3-й час. Возможности JavaScript 42
4-й час. Выполнение программ JavaScript 53

Часть II. Методы программирования на JavaScript 59

5-й час. Использование переменных и функций 60
6-й час. Использование массивов и строковых данных 74
7-й час. Тестирование и сравнение значений 86
8-й час. Повторение - мать учения: циклы 95

Часть III. Дополнительные возможности JavaScript 105

9-й час. Использование встроенных объектов 106
10-й час. Работа с объектной моделью документа 116
11-й час. Создание пользовательских объектов 127
12-й час. Обработка событий 135

Часть IV. Управление Web-страницами 127

13-й час. Использование окон и фреймов 146
14-й час . формы введения данных 158
15-й час. Добавление рисунков и анимации 172
16-й час. Создание сценариев для разных броузеров 185

Часть V. Дополнительные средства JavaScript 195

17-й час. Использование таблиц стилей 196
18-й час. Создание динамических страниц с помощью DOM 208
19-й час. Дополнительные средства DOM 217
20-й час. Использование мультимедиа и встроенных утилит 228

Часть VI. Сложные приложения JavaScript 239

21-й час. Отладка приложений JavaScript 240
22-й час. Улучшение Web-страниц 253
23-й час. Создание сценария игры 265
24-й час. Тенденции развития технологий Web 281
Приложение А. Ресурсы JavaScipt 288
Приложение Б. Средства разработчика сценариев JavaScript 290
Приложение В. Словарь терминов 293
Приложение Г. Краткий справочник по JavaScript 295
Предметный указатель 304

http://ITBookz.ru/

Содержание

Введение 20

Часть I. Начало начал 21

1-й час.
Знакомство с JavaScript 22

Основы работы в JavaScript 23
Сценарии и программы 23
История JavaScript 23
Добавление сценария JavaScript на Web-страницу 24
Броузеры, поддерживающие JavaScript 26
Netscape и Internet Explorer 26
Версии JavaScript 26
Соперники JavaScript 27
Java 27
ActiveX 27
VBScript 28
CGI 28
Резюме 28
Вопросы и ответы 29
Семинар 29
Контрольные вопросы 29
Ответы 30
Упражнения 30

2-й час.
Создание простых сценариев 31

Инструменты создания сценария 32
Отсчет времени 33
Начало сценария 33
Добавление операторов JavaScript 33
Сохранение данных в переменных 34
Вычисление значения 34
Вывод результата на экран 34
Вставка сценария на Web-страницу 35
Тестирование сценария 36
Изменение сценария 37
Выявление и устранение ошибок 38
Скрытие сценариев от старых броузеров 39
Резюме 40
Вопросы и ответы 40
Семинар 41
Контрольные вопросы 41
Ответы 41
Упражнения 41

http://ITBookz.ru/

3-й час.
Возможности JavaScript 42
Улучшение пользовательского интерфейса узла 43
Использование строки состояния 43
Средства перемещения по документу 43
Окна с сообщениями и другие элементы 44
Рисунки и анимация 45
Изменение форм 45
Определение версии броузера 47
Внедряемые модули 48
Сложные сценарии 49
Копирование сценария 49
Резюме 51
Вопросы и ответы 51
Семинар 52
Контрольные вопросы 52
Ответы 52
Упражнения 52
4-й час.
Выполнение программ JavaScript 53
Использование функций 54
Выполнение задач с помощью функций 54
Объекты 54
Обработка событий 55
Условные операторы 56
Циклы 56
Последовательность выполнения сценариев 56
Опять о комментариях 57
Резюме 57
Вопросы и ответы 58
Семинар 58
Контрольные вопросы 58
Ответы 58
Упражнения 58
Часть II. Методы програллллирования HaJavaScript **59**
5-й час.
Использование переменных и функций 60
Использование функций 61
Определение функции 61
Вызов функции 62
Возвращаемое значение 63
Использование переменных 64
Выбор имени переменой 65
Глобальные и локальные переменные 65
Определение значений переменным 67
Типы данных в JavaScript 68
Преобразование типов данных 69
Сохранение пользовательских данных в переменных 69
Резюме 71
Вопросы и ответы 71

http://ITBookz.ru/

Семинар 72
Контрольные вопросы 72
Ответы 72
Упражнения 73

6-й час.
Использование массивов и строковых данных 74

Использование объектов String 75
Создание объекта String 75
Определение значения 75
Определение длины строки 77
Изменение регистра текста 77
Подстроковые переменные 77
Использование части строковой переменной 78
Возвращение одного символа 78
Поиск подстроковой переменной 78
Использование числовых массивов 79
Создание числового массива 79
Управление элементами массива 80
Использование строковых массивов 80
Создание строковых массивов 80
Разделение строковой переменной 80
Сортировка элементов массива 81
Отображение бегущих строк 81
Резюме 83
Вопросы и ответы 84
Семинар 84
Контрольные вопросы 84
Ответы 84
Упражнения 85

7-й час.
Тестирование и сравнение значений 86

Оператор if 87
Условные операторы 87
Совместное использование логических и условных операторов 88
Оператор else 89
Использование условных выражений 89
Задание нескольких условий 90
Проверка введенных данных 91
Резюме 93
Вопросы и ответы 93
Семинар 93
Контрольные вопросы 93
Ответы 94
Упражнения 94

8-й час.
Повторение — мать учения: циклы 95

Использование циклов 96
Использование циклов while 97
Использование цикла do...while 98
Управление циклами 98

http://ITBookz.ru/

Создание бесконечного цикла 98
Прерывание цикла 99
Продолжение выполнения цикла 99
Использование цикла for...in 100
Управление массивами 100
Резюме 102
Вопросы и ответы 103
Семинар 103
Контрольные вопросы 103
Ответы 103
Упражнения 104

Часть III. Дополнительные возможности JavaScript 105

9-й час.
Использование встроенных объектов 106

Что такое объект 107
Создание объектов 107
Значения и свойства объекта 107
Методы 107
Ключевое слово with 108
Объект Math 108
Округление и усечение 108
Генерация случайных чисел 109
Управление датами 109
Создание объекта Date 110
Определение значения объекта Date ПО
Получение значений объекта Date 110
Временные зоны 111
Изменение формата представления даты 111
Применение объектов Math на практике 111
Резюме 114
Вопросы и ответы 114
Семинар 114
Контрольные вопросы 114
Ответы 115
Упражнения 115

10-й час.
Работа с объектной моделью документа 116

Объектная модель документа 117
История DOM 118
Объекты window 118
Управление Web-документами 118
Получении информации о броузере 119
Добавление в документ текста 120
Очистка и обновление содержимого Web-страницы 120
Использование ссылок и анкеров 121
Получение сведений о работе броузера 121
Объект location 122
Получение сведений о броузере 122
Создание кнопок Back и Forward 123
Резюме 124

http://ITBookz.ru/

Вопросы и ответы 125
Семинар 125
Контрольные вопросы 125
Ответы 125
Упражнения 126

11-й час.
Создание пользовательских объектов 127

Упрощение сценариев с помощью объектов 128
Определение объекта 128
Добавление в объект метода 129
Создание экземпляра объекта 129
Настройка встроенных объектов 130
Сохранение данных в объектах 131
Резюме 133
Вопросы и ответы 133
Семинар 133
Контрольные вопросы 133
Ответы 134
Упражнения 134

12-й час.
Обработка событий 135

Роль обработчика событий в JavaScript 136
Объекты и события 136
Создание обработчика событий 136
Обработчики событий в JavaScript 137
Использование объекта event 137
События, связанные с мышью 138
В и Из 138
Щелчки и отпускания 138
События, связанные с клавишами 140
Событие onLoad 140
Добавление описания ссылки 141
Резюме 143
Вопросы и ответы 143
Семинар 144
Контрольные вопросы 144
Ответы 144
Упражнения 144

Часть IV. Управление Web-страницами **145**

13-й час.
Использование окон и фреймов **146**

Управление окнами с помощью объектов 147
Создание нового окна 147
Открытие и закрытие окон 148
Временные задержки 148
Обновление страницы с задержкой 150
Отображение диалоговых окон 151
Создание сценария отображения диалогового окна 152
Управление фреймами 153
Использование объектов фреймов 153

http://ITBookz.ru/

Массив frames 154
Создание навигационного фрейма 154
Резюме 156
Вопросы и ответы 156
Семинар 157
Контрольные вопросы 157
Ответы 157
Упражнения 157

14-й час.
формы введения данных 158

Основы работы с формами 159
Определение формы 159
Использование объекта form 159
Свойства объекта form 160
Отправка данных и очистка формы 160
Определение событий формы 160
Создание элементов форм 160
Текстовое поле 161
Текстовые панели 161
Управление текстом в формах 162
Кнопка 162
Флажок 163
Переключатель 163
Раскрывающийсясписок 164
Отображение данных на форме 165
Отправка данных формы в виде почтового сообщения 166
Проверка правильности заполнения формы 168
Резюме 170
Вопросы и ответы 170
Семинар 171
Контрольные вопросы 171
Ответы 171
Упражнения 171

15-й час.
Добавление рисунков и анимации 172

Использование разделенного рисунка 173
Динамические рисунки 175
Управление массивом images 175
Предварительная загрузка рисунка 176
Создание изменяющихся рисунков 176
Создание простой анимации 178
Создание рисунков 178
Создание документа HTML 179
Определение переменных 179
Выполнение анимации 180
Компоновка сценария 181
Резюме 183
Вопросы и ответы 183
Семинар 183
Контрольные вопросы 183
Ответы 184

http://ITBookz.ru/

Упражнения 184

16-й час.
Создание сценариев для разных броузеров 185

Получение сведений о броузере 186
Отображение сведений о броузере 186
Броузеры независимых производителей 187
Поддержка JavaScript броузером 189
Создание сценария для определенного броузера 190
Создание разных страниц 190
Создание универсальной страницы 191
Сценарии в броузерах, не поддерживающих JavaScript 191
Сценарий для разных броузеров 192
Резюме 193
Вопросы и ответы 193
Семинар 194
Контрольные вопросы 194
Ответы 194
Упражнения 194

Часть V. Дополнительные средства JavaScript 195

17-й час.
Использование таблиц стилей 196

Стили и внешний вид 197
Определение и использование стилей CSS 197
Создание правил 198
Выравнивание текста 198
Изменение цвета и рисунка фона 199
Управление шрифтами 199
Границы и поля 200
Создание простой таблицы стилей 200
Использование внешних таблиц стилей 202
Управление таблицами стилей в JavaScript 202
Создание динамических стилей 203
Резюме 206
Вопросы и ответы 206
Семинар 206
Контрольные вопросы 206
Ответы 207
Упражнения 207

18-й час.
Создание динамических страниц с помощью DOM 208

Структура DOM 209
Элемент 210
Родительские и дочерние объекты 210
Уровень структуры 210
Два метода определения слоев 210
Определение свойств слоев 211
Расположение слоя 212
Управление старыми броузерами 212
Создание анимации с помощью слоев 213
Резюме 215
Вопросы и ответы 215

http://ITBookz.ru/

Семинар 215
Контрольные вопросы 215
Ответы 216
Упражнения 216

19-й час.
Дополнительные средства DOM 217

Работа с элементами DOM 218
Основные свойства элемента 218
Свойства связей элементов 218
Методы документа 219
Методы элемента 219
Скрытие и отображение объектов 219
Изменение текста на странице 221
Добавление текста на страницу 222
Великолепное бегущее сообщение 223
Резюме 226
Вопросы и ответы 226
Семинар 227
Контрольные вопросы 227
Ответы 227
Упражнения 227

20-й час.
Использование мультимедиа и встроенных утилит **228**

Что такое LiveConnect 229
Типы MIME 229
Использование LiveConnect 230
Управление объектами утилит 230
Проверка утилит 230
Перечень утилит 231
Использование в надстройках объектов 232
Воспроизведение музыки с помощью мыши 233
Вставка звуковых файлов 233
Отображение клавиатуры 233
Воспроизведение звуков 233
Составление программы 234
Резюме 235
Вопросы и ответы 235
Семинар 235
Контрольные вопросы 236
Ответы 237
Упражнения 237

Часть VI. Сложные приложения JavaScript **239**

21-й час.
Отладка приложений JavaScript 240

Как избежать ошибок 241
Правильная техника программирования 241
Как избежать простых ошибок 241
Основные средства отладки 243
Консоль JavaScript 243
Автоматическое отображение консоли 243

http://ITBookz.ru/

Сообщения в строке состояния 244
Отображение сообщений об ошибках в Internet Explorer 244
Сообщения об ошибках и строка состояния 244
Отладчик JavaScript 245
Установка отладчика 245
Окно отладчика 245
Вставка разрывов и прерываний 246
Проверка переменных 247
Выполнение сценария 247
Отладка сценария 247
Тестирование программы 248
Исправление ошибки 248
Повторная проверка сценария 249
Резюме 251
Вопросы и ответы 251
Семинар 251
Контрольные вопросы 251
Ответы 252
Упражнения 252

22-й час.
Улучшение Web-страниц 253

Создание базового документа HTML 254
Использование раскрывающихся списков 255
Имена страниц 255
Создание структуры данных и документа HTML 256
Создание функции панели перемещения 257
Добавление описания ссылок 257
Добавление графических ссылок 259
Создание рисунков 259
Создание обработчика события 259
Составление полного программного кода 260
Резюме 262
Вопросы и ответы 262
Семинар 263
Контрольные вопросы 263
Ответы 264
Упражнения 264

23-й час.
Создание сценария игры 265

Планирование программы 266
Создание рисунков 266
Выбор переменных 266
Создание документа HTML 267
Составление сценария 269
Управление кнопками Draw или Deal 269
Тасование колоды 269
Раздача карт 270
Сдача карт 270
Добор новой карты 271
Определение счета игрока 271
Готовый документ со сценарием 274

http://ITBookz.ru/

Резюме 279
Вопросы и ответы 279
Семинар 279
Контрольные вопросы 279
Ответы 279
Упражнения 280

24-й **час.**

Тенденции развития технологий Web 281

Углубленное изучение JavaScript 282
Будущие технологии Web 282
Будущие версии JavaScript 282
Будущее DOM 282
XML 283
XSL 283
Планирование стандартов 284
Совместимость сценариев 284
Совместимость с HTML 284
Понимание документа 285
Последние советы 285
Резюме 286
Вопросы и ответы 286
Семинар 286
Контрольные вопросы 286
Ответы 287
Упражнения 287

Приложение А. Ресурсы JavaScipt 288

Книги 288
Web-узелы, посвященные JavaScript 288
Развитие Web 289

Приложение Б. Средства разработчика сценариев JavaScript 290

Редакторы HTML и текстовые процессоры 290
HomeSite 290
FrontPage 291
NetObject ScriptBuilder 291
BBEdit 291
Текстовые редакторы 291
Проверка кода HTML 291

Приложение В. Словарь терминов 293

Приложение Г. Краткий справочник по JavaScript 295

Встроенные объекты 295
Array 295
String 296
Math 297
Date 298
Основы модели DOM 298
Window 299
Location 299
History 299
Document 300
Navigator 300

http://ITBookz.ru/

Создание и настройка объектов 300
Создание объектов 300
Настройка объектов 301
Операторы JavaScript 301
Комментарии 301
Прерывание 301
Продолжение 301
Цикл for 301
Цикл for...in 302
Функция 302
Условие if...else 302
Оператор return 302
Оператор var 302
Цикл while 303
Встроенные функции JavaScript 303
Функция eval 303
Функция parseInt 303
Функция parseFloat 303
Предметный указатель 304

http://ITBookz.ru/

Об авторе

Майкл Монкур (Michael Moncur) известен как блестящий Web-дизайнер и автор многих книг. С Internet он работает с момента основания службы Gopher. Он автор такой прекрасной книги, как *Laura Lemay's Web Workshop: JavaScript* издательства *Sams* и соавтор книг *JavaScript Unleashed* и *Sams Teach Yourself CGI Programming with Perl 5 in a Week*. Кроме того, он написал несколько книг по сетевым технологиям и материалам подготовки сертифицированных специалистов *Microsoft* и *Netscape*. В свободное от работы время (несколько часов в год) он создает музыкальные произведения и пиротехнические приспособления.

http://ITBookz.ru/

Посвящения

Посвящается моей семье и особенно Лауре (Laura). Огромное спасибо за понимание и поддержку в трудную минуту.

http://ITBookz.ru/

Благодарности

Я хотел бы поблагодарить всех сотрудников *Sams* за громадную помощь при написании этой книги. В частности **Скотта Мейерса** (Scott Meyers) и **Марка Табера** (Mark Taber), заставивших меня взяться за работу и поддерживающих меня во время написания книги. Редактор **Сюзан Хоббс** (Susan Hobbs) сделала книгу грамотно написанной и оформленной, а технический консультант **Сунил Хазари** (Sunil Hazari) тщательно проверил сценарии и текст книги.

Также хотелось бы поблагодарить всех сотрудников редакции, принимавших участие в создании предыдущего издания книги. В первую очередь **Дэвида Майхью** (David Mayhew), **Сина Медлока** (Sean Medlock) **Мишель Винер** (Michelle Wyner). Особенные благодарности передаю **Дэвиду и Шерри Рогелберг** (David and Sherry Rogelberg) за технические консультации во время реализации проекта.

Особых похвал заслужила моя жена **Лаура** и родители **Гари и Сюзан Монкур** (Gary and Susan Moncur), а также остальные мои родственники (Мет (Matt), **Мелани** (Melanie), **Ян** (Ian) и **Кристин** (Kristen)) и друзья (**Чак Перкинс** (Chuck Perkins), **Мет Стриб** (Matt Strebe), **Кори Сторм** (Cory Storm), **Роберт Персоне** (Robert Parsons), **Дилан Винслоу** (Dylan Winslow), **Скотт Дурбин** (Scott Durbin), **Рэй Джонс** (Ray Jones), **Джеймс Челис** (James Chellis), **Курт Сиферт** (Curt Sifferi) и **Генри Дж. Тилман** (Henre J. Tillman)).

Без их поддержки я не справился бы с написанием этой книги.

http://ITBookz.ru/

Введение

World Wide Web начинала свое существование в виде простого хранилища информации; сегодня она преобразовалась в жутко громоздкую структуру, позволяющую не только получать необходимую информацию, но общаться, развлекаться и обучаться. По мере развития Всемирной паутины изменяются и средства управления ею. Простые языки разметки документа, такие как HTML, наконец стали позволять интегрировать в себя настоящие языки программирования. Среди них вы найдете и JavaScript.

Не пугайтесь, услышав слово "программирование". Для многих оно ассоциируется с бессонными ночами, проведенными перед экраном монитора в поисках того кода, который все же позволит выполнить поставленную задачу. (Есть, правда, и те, кому нравятся подобные "посиделки".)

Хотя JavaScript — это полноценный язык программирования, разобраться в нем очень просто. Если вы не занимались программированием, то изучение JavaScript станет хорошим введением в увлекательный мир создания программ. Для того чтобы успешно создавать программы на JavaScript, достаточно обладать скромными познаниями в области программирования. Первую свою программу вы создадите уже в главе "2-й час. Создание простых сценариев".

Если вы уже знакомы с HTML и знаете, как с его помощью создаются Web-страницы, вам не составит особого труда изучить и JavaScript. Программы на JavaScript могут содержать всего одну строку или достигать объемов целого приложения. В этой книге вы найдете как простые сценарии, так и целые приложения, например карточные игры.

Вы уже достаточно опытны в создании Web-страниц и знаете, что Web постоянно изменяется. Не правда ли, вам сложно уследить за всеми новыми средствами программирования? Эта книга познакомит вас с JavaScript, который станет еще одним инструментом в вашем наборе средств создания высококачественных Web-документов. Я надеюсь, вам понравится изучать его.

Написание этой книги не было для меня обременительным. Но это не означает, что ее следует воспринимать несерьезно. Я надеюсь, вы со всей ответственностью и прилежностью приступите к изучению интересного для вас материала.

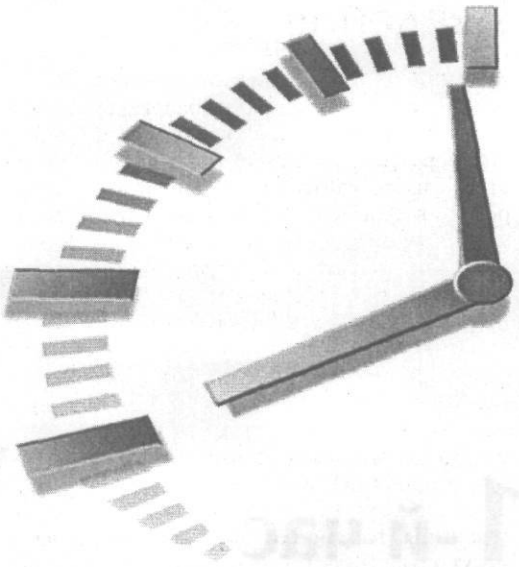
Как использовать эту книгу

Эта книга состоит из 24 практических уроков. Каждый из них содержит описание отдельного раздела о JavaScript и рассчитан на изучение в течение часа. В первых главах вы познакомитесь с основами JavaScript. На последних занятиях вы познакомитесь с более сложными понятиями и методами программирования. Вы можете изучать один урок в день или быстрее, в зависимости от ваших способностей и желания. (Если вы в состоянии не спать на протяжении 24 часов и сохранять нормальное самочувствие, то можете изучить всю книгу всего за один день. В этом случае вы точно будете знать, каково быть автором компьютерной книги.)

Вопросы и ответы, семинар и упражнения

В конце каждой главы вы найдете три раздела. В разделе "Вопросы и ответы" приведены часто задаваемые вопросы по пройденному материалу и ответы на них. Раздел "Семинар" содержит три вопроса, которые позволят вам проверить усвоенные знания, раздел "Упражнения" предназначен для закрепления ваших знаний на практике.

http://ITBookz.ru/



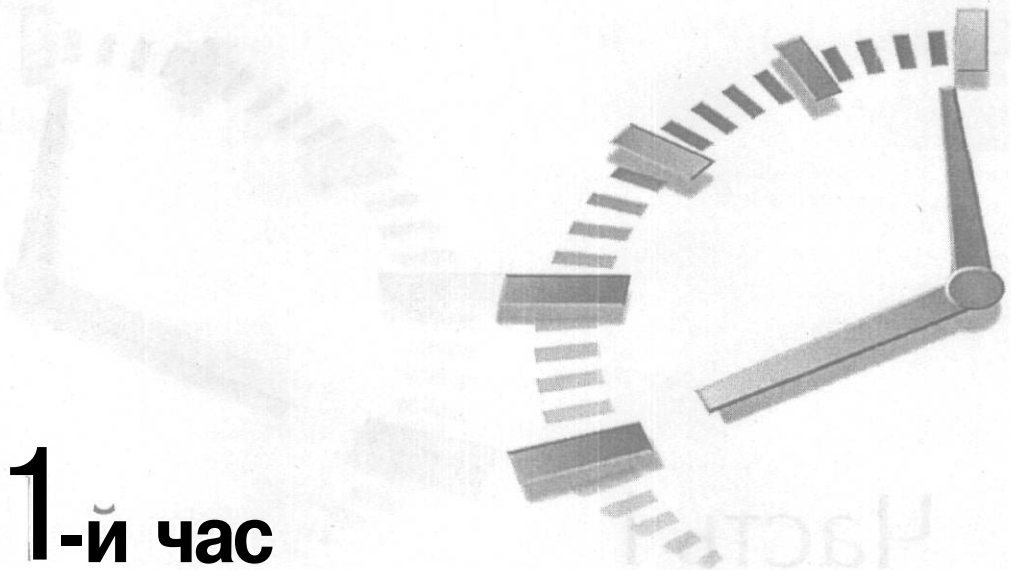
Часть I

Начало начал

Темы занятий

1. **Знакомство с JavaScript**
2. **Создание простых сценариев**
3. **Возможности JavaScript**
4. **Выполнение программ JavaScript**

http://ITBookz.ru/



1-й час

Знакомство с JavaScript

World Wide Web (WWW) в начале своего существования позволяла получать только текстовую информацию. Первые версии спецификации HTML не были оснащены средствами интегрирования в документы графических объектов. Несмотря на то, что Web достаточно молода (по сравнению, например, с телевидением), темпы ее развития в несколько раз превышают темпы развития других информационных технологий.

Сегодня Web-узлы содержат самые различные виды информации: графические объекты, звуковые клипы, анимационные изображения, видеофильмы и просто текстовые документы. Языки создания сценариев в Web, подобные JavaScript, позволяют достаточно просто улучшать внешний вид Web-страниц и устанавливать тесную взаимосвязь Web с пользователями.

В первой главе этой книги кратко описана концепция написания сценариев в Web-документах на языке JavaScript. Здесь также описывается методика применения JavaScript, Java и других языков программирования в Web для создания необходимых документов. В этой главе рассматриваются следующие темы.

- **Сценарии в Web и их предназначение**
- **Разница между программированием и написанием сценариев**
- **Знакомство с JavaScript**
- **Методы внедрения команд JavaScript на Web-страницу**
- **Управление JavaScript различными броузерами**
- **Разница между JavaScript и альтернативными языками**

http://ITBookz.ru/

Основы работы в JavaScript

В любых фантастических фильмах (и других фильмах о высоких технологиях) компьютеры управляются командами на английском языке. Насколько это будет справедливо в будущем, покажет время. В настоящее время базис всех команд языка программирования составляют синтаксические конструкции языков BASIC, C и Java.

Если вы знакомы с методами создания Web-документов с помощью HTML, то считайте, что, по меньшей мере, один язык программирования вы уже освоили. Задавая дескрипторами HTML форматирование документа, вы даете указания броузеру выполнить необходимые действия, который правильно отображает Web-страницу на экране монитора.

Поскольку HTML — это простой язык разметки документов, он не позволяет пользователю управлять внешним видом документа, а автоматически отображает указанным образом Web-документ на экране. Интерактивные задачи требуют применения более сложных языков программирования. Такие языки программирования и называются *языками подготовки сценариев.*

В то время как большинство языков программирования очень сложные, языки подготовки сценариев до невозможности простые. Они имеют простой синтаксис, позволяют выполнять простые операции и легки для обучения. Языки подготовки сценариев в Web позволяют создавать сценарии, внедряемые впоследствии в интерактивные Web-страницы.

Сценарии и программы

Фильм (или видеоклип) соответствует определенному сценарию — последовательности действий (сцен), которые разыгрывают актеры. Сценарий в JavaScript может содержать как одну строку, так и большой листинг объемом в небольшое приложение. (В последнем случае программы JavaScript запускаются только в броузерах или других программах, поддерживающих JavaScript.)



Какая же разница между написанием сценариев и программированием? В этой книге мы почти все время будем говорить о создании сценариев. Будьте готовы к тому, что после изучения настоящей книги вы захотите (и будете вполне готовы) научиться программировать на JavaScript.

Некоторые языки программирования необходимо *компилировать,* или преобразовывать, в машинный код, который впоследствии и выполняется. JavaScript, в противовес им, — это *интерпретируемый* язык программирования. Броузер выполняет каждую строку сценария последовательно, после выполнения предыдущей.

У интерпретируемых языков программирования есть одно большое преимущество: создание и изменение сценариев выполняется очень просто. Изменение сценария JavaScript проводить так же просто, как и редактирование обычного документа HTML. Все проведенные изменения вступают в силу непосредственно после загрузки Web-страницы в окне броузера.



Интерпретируемые языки программирования имеют также и большой недостаток — созданные на них программы выполняются относительно медленно. Именно по этой причине управлять с их помощью графическими объектами не всегда удобно. Они также требуют использования специального интерпретатора (в случае JavaScript — это броузер).

История JavaScript

JavaScript разработан компанией *Netscape Communication Corporation,* которая создала известный на весь мир Web-броузер Netscape Navigator. JavaScript — это первый разработанный язык подготовки сценариев, все еще достаточно популярный, чтобы обратить на него свое внимание.

http://ITBookz.ru/



JavaScript имел рабочее название LiveScript и впервые был представлен как часть Netscape Navigator 2.0 в 1995 году. Позже его переименовали в JavaScript, чтобы подчеркнуть его сходство с Java.

JavaScript практически так же прост в изучении, как и HTML. Сценарии JavaScript напрямую вставляются в документы HTML. С помощью JavaScript вы решите следующие задачи.

- Добавите в документ бегущие строки и сообщения о его изменении.
- Измените форму введения данных и проведете необходимые вычисления. (Например, чтобы форма заказа товаров в интерактивном магазине отображала суммарную стоимость покупок.)
- Отобразите сообщения, предназначенные для пользователя (как на самой странице, так и в виде отдельного диалогового окна).
- Создадите анимированные изображения, которые изменяются при наведении на них указателя мыши.
- Добавите интерактивный баннер, который намного привлекательнее, чем статическое изображение.
- Определите используемый броузер и настроите в соответствии с ним Web-страницу.
- Обнаружите используемые внедряемые модули и уведомите пользователя об их статусе.

Это только некоторые из общих задач, которые позволяет выполнить JavaScript. На самом деле, с его помощью вы можете решить и более сложные задачи, включая и создание отдельных приложений. По мере изложения материала в этой книге мы будем переходить к созданию все более сложных сценариев.

Добавление сценария JavaScript на Web-страницу

Как вы уже знаете, HTML — это язык разметки документов, с помощью которого создаются Web-документы. Чтобы напомнить вам, что из себя представляет программа документа HTML я приведу простой пример (листинг 1.1).

Листинг 1.1. Простой документ HTML

```
<HTMЬ>
<HEAD>
<TITLE>Наша начальная страница</TITLE>
</HEAD>
<BODY>
<H1>Птицефабрика по разведению страусов</H1>
<P>Добро пожаловать на нашу первую
Web-страницу. Она еще не создана
полностью </P>
</BODY>
</HTML>
```

Этот документ состоит из заголовка, обозначенного дескрипторами <HEAD>, и тела, обозначенного дескрипторами <BODY>. Для того чтобы добавить сценарий JavaScript на Web-страницу, используется пара дескрипторов <SCRIPT>.

http://ITBookz.ru/

Пара дескрипторов <SCRIPT> указывает броузеру рассматривать текст программы как сценарий. Обнаружив дескриптор </SCRIPT>, броузер возвращается к выполнению обычной программы HTML. В большинстве случаев операторы JavaScript выполняются только внутри сценария (обозначенного дескрипторами <SCRIPT> и </SCRIPT>). Исключение составляют только обработчики событий, о которых речь пойдет ниже.

Используя дескриптор <SCRIPT>, вы создадите на Web-странице только простые сценарии (в нашем примере, приведенном на листинге 1.2, он состоит только из одной строки). Если вы хотите выполнить сценарий листинга 1.2, то не вводите номера строк — они введены только для удобства отслеживания введенных дескрипторов.

Листинг 1.2. Простой документ HTML с простым сценарием

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Наша начальная страница</TITLE>
4:  </HEAD>
5:  <BODY>
6:  <H1>Птицефабрика по разведению страусов</H1>
7:  <P>Добро пожаловать на нашу первую
8:  Web-страницу. Она еще не создана
9:  полностью </P>
10: <SCRIPT LANGUAGE="JavaScript">
11: document.write(document.lastModified);
12: </SCRIPT>
13: </BODY>
14: </HTML>
```

Оператор document.write, рассматриваемый ниже, позволяет отображать результат сценария на Web-странице. В нашем примере отображается дата изменения документа.



Заметьте, что дескриптор <SCRIPT> в листинге 1.2 содержит параметр Language="JavaScript". Этот параметр определяет используемый язык подготовки сценария. В нем вы можете также указывать и номер версии языка (об этом вы узнаете немного позже).

В приведенном примере сценарий размещается в теле программы Web-документа. Сценарии могут размещаться в четырех различных частях программы документа HTML.

- В теле программы. В этом случае результат сценария отображается на Web-странице при ее загрузке в броузере.
- В заголовке программы между парой дескрипторов <HEAD>. Сценарий, размещенный в заголовке, не выполняется сразу же при загрузке страницы, а используется другими сценариями. В этом случае он используется как функция – группа операторов JavaScript, выполняемых как одно целое.
- В дескрипторе HTML. Такая конструкция называется обработчиком событий и позволяет выполнять сценарий JavaScript вместе с дескриптором. Обработчик событий представляет собой отдельный тип сценария, который не требует использования дескриптора <SCRIPT> для его обозначения. Детальнее с обработчиками событий вы познакомитесь в главе "4-й час. Выполнение программ JavaScript".
- В отдельном файле. JavaScript позволяет создавать собственные файлы с расширением .js, содержащие готовые сценарии. В этом случае сценарий указывается в программе в виде имени файла, приведенного между дескрипторами <SCRIPT>. Подобные файлы JavaScript используются только в Netscape Navigator версии 3.0 и выше и Internet Explorer версии 4.0 и выше.

http://ITBookz.ru/

Броузеры, поддерживающие JavaScript

Подобно HTML, JavaScript требует поддержки броузером. Разные броузеры выполняют сценарии по-разному. В отличие от HTML, отсутствие в броузере средств поддержки JavaScript оборачивается трагедией для пользователя. В отличие от неправильного форматирования (при отсутствии средств поддержки используемой версии HTML), в случае невыполнения сценария на экране отображается сообщение об ошибке или броузер вообще прекращает свою работу.

Давайте проведем небольшой экскурс в мир броузеров и выясним, какие из них поддерживают JavaScript.

Netscape и Internet Explorer

В настоящее время пользователи преимущественно используют два броузера — Netscape Navigator и Internet Explorer. На протяжении долгих лет Netscape Navigator был более популярным средством просмотра Web-документов. Но за последние годы Internet Explorer заметно повысил свои возможности, что и вызвало больший интерес к нему со стороны пользователей. Оба броузера в той или иной степени поддерживают JavaScript.



Давайте договоримся в этой книге говорить только о JavaScript 1.5, последней и самой доработанной версии этого языка программирования, для выполнения всех примеров вам необходимо пользоваться Netscape Navigator версии 4.5 и выше, хотя большинство сценариев, приведенных в книге, прекрасно выполняются и в Internet Explorer, и Netscape Navigator версии 4.0 и выше.

Версии JavaScript

Язык JavaScript начал свое существование с момента выхода в мир броузера Netscape Navigator 2.0. Существует четыре версии JavaScript.

- JavaScript 1.0. Первая версия языка, поддерживаемая Internet Explorer 3.0 и Netscape Navigator 2.0.
- JavaScript 1.1. Поддерживается Netscape Navigator 3.0 и Internet Explorer 4.0 (почти полностью).
- JavaScript 1.2. Поддерживается Netscape Navigator 4.0 и Internet Explorer 4.0 (частично).
- JavaScript 1.3. Поддерживается Netscape Navigator 4.5.
- JavaScript 1.5. Поддерживается Netscape 6.0. Большинство средств этой спецификации поддерживается Internet Explorer версии 5.5 и выше.

Каждая последующая версия броузера поддерживает версии JavaScript, используемые в предыдущих версиях. Броузеры, которые поддерживают новые версии JavaScript, позволяют выполнять сценарии, созданные в ранних версиях JavaScript.

Язык JavaScript соответствует стандартам ECMA, Европейской ассоциации стандартизации. Эта организация разработала спецификацию ECMA-262, которой должны удовлетворять все языки программирования. Стандартизированный язык программирования иногда называют еще и ECMAScript. JavaScript 1.3 полностью удовлетворяет требованиям ECMA-262, а JavaScript 1.5 — ECMA-262 третьей редакции.



Еще один язык подготовки сценариев, о котором вы могли слышать, — это JScript. Это ответ *Microsoft* на выпуск *Netscape* языка JavaScript. Он частично совместим с JavaScript.

http://ITBookz.ru/

Соперники JavaScript

JavaScript — не единственный язык подготовки сценариев в Web. В некоторых ситуациях он не позволяет добиться всего того, чего можно выполнить с помощью других языков. Язык Java, описанный выше, имеет более широкие возможности, чем JavaScript. В этом разделе мы рассмотрим языки программирования в Web, активно используемые наряду с JavaScript.

Java

Язык Java разработан компанией *Sun Microsystems* и предназначен для создания *аплетов*, или программ, выполняемых на Web-страницах.

Java — это компилируемый язык программирования. При компилировании программа преобразуется в машинный код для *виртуального компьютера,* а не реального. Машинный код виртуального компьютера интерпретируется Web-броузером. Это обусловливает то, что все аплеты Java выполняются одинаково на всех компьютерах: Windows, Mac и Unix и в любых броузерах.



Java — это, кроме всего прочего, очень густо населенный остров в Индонезийском архипелаге.

Давайте в самом начале выясним предмет этой книги. Java — это прекрасный объект для изучения, но не он рассматривается в этой книге. Хотя названия многих команд в JavaScript и похожи на таковые в Java, по сути эти языки программирования остаются разными.

ActiveX

ActiveX — это спецификация, разработанная *Microsoft,* которая позволяет запускать на Web-странице обычные приложения Windows. Программы ActiveX создаются в таких языках программирования, как Visual C++ и Visual Basic. Перед тем как запустить их на сервере, их необходимо обязательно откомпилировать.

Приложения ActiveX, называемые *элементами управления,* загружаются и выполняются Web- броузером, подобно аплетам Java. В отличие от аплетов Java, элементы управления устанавливаются навсегда при загрузке. Нет необходимости загружать их повторно.

Главное преимущество ActiveX состоит в многофункциональности. В некоторых случаях это становится и главным недостатком: иногда программисты-шутники используют ее средства для создания Web-страниц, отключающих компьютер пользователей, которые их загружают, или форматирующих жесткие диски их компьютеров.

К счастью, в ActiveX предусмотрена возможность отслеживать источник элементов управления и таким образом предотвращать нежелательное вмешательство. Для того чтобы предохранить вашу систему от "шуток" коллег, просто используйте проверенные источники.

Кроме того, ActiveX имеет два больших недостатка. Первый заключается в относительной сложности обучения. Второй состоит во все той же несовместимости с броузерами независимых производителей — ее поддерживают только броузеры Internet Explorer, запущенные на платформе Windows.

http://ITBookz.ru/

VBScript

VBScript, известный также как язык сценариев, созданный на основе Visual Basic, — это наиболее ярко выраженный соперник JavaScript компании *Microsoft*. Подобно тому как синтаксис JavaScript сильно напоминает синтаксис Java, синтаксис VBScript незначительно отличается от синтаксиса Visual Basic, популярного языка программирования в среде Windows.

Как и JavaScript, VBScript — это простой язык подготовки сценариев; операторы VBScript так же легко внедряются в документ HTML, как и операторы JavaScript. Все сценарии VBScript начинаются с дескриптора <SCRIPT LANGUAGE="VBScript">.

VBScript умеет многое из того, что подвластно JavaScript. В некоторых случаях они даже подобны. Этот язык подготовки сценариев имеет следующие преимущества.

- Для тех, кто знаком с Visual Basic, проще изучить VBScript.
- Он полностью совместим с технологией ActiveX, стандартом *Microsoft* для внедряемых в документы Web приложений.

Главный недостаток VBScript заключается в поддержке его только броузерами Internet Explorer. С другой стороны, JavaScript поддерживается как Netscape Navigator, так и Internet Explorer. Может, именно поэтому JavaScript популярнее, и его можно встретить повсеместно, практически в любых Web-документах.

CGI

CGI (Common Gateway Interface — Общий шлюзовой интерфейс) — это не язык программирования, а спецификация, позволяющая программам запускаться на Web-сервере. Программы CGI создаются во многих языках программирования: Perl, C, Visual Basic и т.п.

Программы CGI широко распространены в Web. Даже при заполнении формы анкеты и отправке информации на Web-узел данные принимаются приложением CG1.

Главное отличие между JavaScript и CGI состоит в том, что CGI выполняется на Web-сервере, а приложения JavaScript — на клиенте (в Web-броузере). Основной недостаток CGI — это неустранимая необходимость обмена данными между клиентом и сервером, что вызывает большие временные задержки (и недовольство пользователей).

Но если посмотреть с другой стороны, то можно обнаружить, что без CGI иногда просто не обойтись. Например, CG1 позволяет считывать и записывать файлы на сервере. Приложения JavaScript позволяют считывать и обрабатывать данные только с формы. Сохранить полученный результат невозможно. Использовать файловые ресурсы Web-сервера может JavaScript, запущенный только на нем.

Наряду с традиционным CGI-программированием в жизнь Web активно начинают интегрироваться такие спецификации, как Active Server Pages (Активные страницы сервера), JavaScript Pages (Страницы JavaScript), Cold Fusion Markup Language (Объективный язык разметки документов) и PHP. Не стоит забывать и о сценариях JavaScript, выполняемых на сервере.



CGI и выполнение сценариев на сервере не рассматриваются в этой книге. Детально изучить методы программирования сценариев CGI и их выполнения на сервере вы можете в других книгах нашего издательства.

Резюме

В течение этого часа вы познакомились с языками подготовки сценариев в Web и, в частности, с языком JavaScript. Вы также узнали, каким образом сценарий внедряется на Web-страницу, и чем отличается JavaScript от других языков программирования в Web.

http://ITBookz.ru/

Вопросы и ответы

Если я все равно собираюсь изучать Java или CGI, стоит ли начинать изучать JavaScript?

Конечно. JavaScript — это идеальное средство для изменения многих приложений, например форм ввода данных. Кроме того, Java и CGI не позволяют выполнить некоторые простые задачи, которые по силам реализовать только JavaScript.

Может ли Web-страница содержать больше, чем один дескриптор <SCRIPT>?

Да. На самом деле, все большие программы, приведенные в этой книге, содержат два или несколько сценариев.

Можно ли создать сценарий, который будет одинаково выполняться и в Internet Explorer, и в Netscape Navigator?

Да, но это не всегда просто. Некоторые команды JavaScript поддерживаются обоими броузерами. Если вы запускаете простой сценарий, то он будет выполняться одинаково в обоих броузерах. В сложных же сценариях могут появляться команды, которые выполняются по-разному в разных броузерах. Сначала JavaScript определяет броузер, а затем выполняет программный код, соответствующий обнаруженной программе.

Как определяется версия Netscape?

Если вы не указали номер версии в дескрипторе <SCRIPT>, то старайтесь создавать простой сценарий, который будет выполняться в Netscape Navigator 2.0. В противном случае лучше указать номер версии.

Что делать, когда система не поддерживает JavaScript вообще?

Используете символы комментариев HTML, чтобы исключить строки с кодом JavaScript из выполняемого кода документа HTML. Об этом рассказано в главе 2.

Семинар

Контрольные вопросы

1. Почему JavaScript и Java имеют похожие названия?
   a) JavaScript — это упрощенная версия Java
   b) Синтаксис JavaScript очень похож на синтаксис Java
   c) Они оба были разработаны на острове Java
2. Какой компьютер выполняет программу JavaScript при просмотре пользователем страницы, содержащей эту программу?
   a) Компьютер клиента, на котором запущен броузер Web
   b) Web-сервер
   c) Центральный компьютер, расположенный в недрах офиса компании *Netscape*
3. Какие из указанных языков поддерживаются и Netscape Navigator, и Internet Explorer?
   a) VBScript
   b) ActiveX
   c) JavaScript

http://ITBookz.ru/

Ответы

1, b) Хотя эти языки и разные, их синтаксис во многом похож.

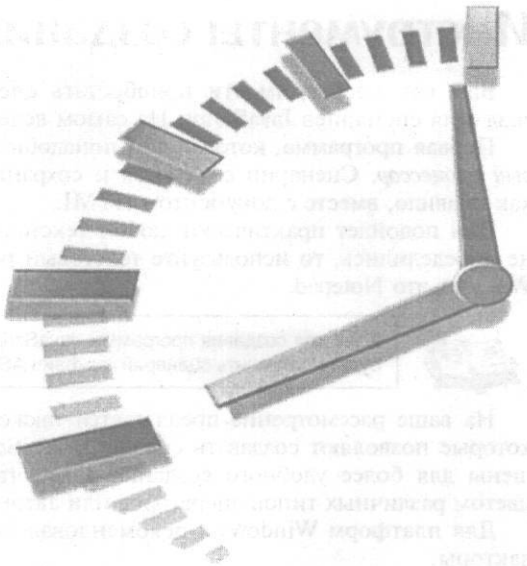
2, a) Программа JavaScript выполняется Web-броузером (существует, правда, и версия JavaScript, которая запускается на сервере, но мы ее изучать не будем).

3, c) JavaScript поддерживается и Internet Explorer, и Netscape Navigator, хотя выполняют они сценарии и по-разному.

Упражнения

Если вы хотите повысить свои познания о JavaScript, перед изучением следующего урока ознакомьтесь с последними обновлениями этого языка. Посетите Web-узел *Netscape*, предназначенный для разработчиков программ, по адресу http://developer.netscape.com/

http://ITBookz.ru/



2-й час

Создание простых сценариев

Как вы уже знаете из материала предыдущего часа "Знакомство с JavaScript", JavaScript — это язык подготовки сценариев для Web-документов. Команды JavaScript вставляются напрямую в документ HTML, и сценарий выполняется непосредственно при загрузке его в окне броузера.

В течение этого часа создадим простой сценарий, отредактируем его и протестируем в Web-броузере. Решая эту задачу, вы узнаете о методах создания сценариев и их предназначении. В этой главе будут рассмотрены следующие темы.

- **Применение программных средств для создания и тестирования сценариев**
- **Начало и завершение сценария**
- **Форматирование операторов JavaScript**
- **Отображение результата выполнения сценария**
- **Включение сценария в Web-документ**
- **Тестирование сценария в Netscape**
- **Изменение сценария**
- **Ошибки в сценариях**
- **Скрытие сценариев в старых броузерах**

http://ITBookz.ru/

Инструменты создания сценария

Вам нет необходимости приобретать специальное программное обеспечение для создания сценариев JavaScript. На самом деле у вас есть все необходимое.

Первая программа, которая вам понадобится для создания сценария, — это *текстовый процессор*. Сценарии создаются и сохраняются в виде обычных текстовых файлов, как правило, вместе с документом HTML.

Вам подойдет практически любой текстовый процессор или редактор. Если вы еше не определились, то используйте текстовый редактор, который включен в состав ОС; в Windows это Notepad.



Если для создания программы JavaScript вы используете текстовый процессор, то не забудьте сохранить сценарий как файл ASCII, а не как документ.

На ваше рассмотрение предлагается также огромное количество редакторов HTML, которые позволяют создавать сценарии JavaScript. Некоторые из них специально оснащены для более удобного создания в них программ JavaScript (например, выделение цветом различных типов операторов или автоматическое создание простых сценариев).

Для платформ Windows я рекомендовал бы использовать следующие текстовые редакторы.

- HomeSite. Великолепный редактор HTML, в который включена поддержка JavaScript.
- Microsoft FrontPage 2000. Наглядный редактор HTML компании *Microsoft*. Средство Script Builder, включенное в его состав, позволяет просто создавать небольшие сценарии.
- TextPad. Простой, но мощный текстовый процессор, который содержит некоторые дополнительные средства, отсутствующие в Notepad.

Для платформ Macintosh, BBEdit, BBEdit Lite и Alpha подойдет любой редактор HTML, позволяющий создавать Web-страницы и сценарии.



В Приложении Б к этой книге приведены адреса Web-узлов, с которых можно загрузить другие редакторы HTML и текстовые процессоры.

Для дальнейшей работы с JavaScript вам понадобятся еще два средства — Web-броузер и компьютер. Поскольку книга посвящена JavaScript 1.5, рекомендуется установить последнюю версию Netscape Navigator. Как вы, наверное, уже знаете, Netscape Navigator прекрасно устанавливается в платформах Windows, Macintosh и Unix. Копию этого броузера вы можете загрузить с Web-узла *Netscape,* расположенного по адресу:

`http://www.netscape.com/`

Броузеры Microsoft доступны на узле `http://www.microsoft.com/`.

Вы должны обладать броузером Netscape версии 4.5 и выше или Internet Explorer версии 4.0 и выше. Некоторые из дополнительных средств (например DOM), описанных в главе 17, предполагают использование Netscape Navigator 6.0 или Internet Explorer 5.0 или выше.



Если вы планируете разместить ваш сценарий в Internet, вам также понадобится Web-сервер или разрешение на его использование. Все примеры, приведенные в этой книге, можно запускать просто с жесткого диска компьютера.

http://ITBookz.ru/

Отсчет времени

Одно из привычных использований JavaScript — это отображение даты и времени. Поскольку сценарий JavaScript выполняется в броузере, необходимо время отображать в соответствии с временным поясом места жительства пользователя. Давайте для начала создадим сценарий вычисления всеобщего скоординированного времени.



Всеобщее скоординированное время соответствует времени нулевого меридиана или времени по Гринвичу. Это время абсолютного уровня отсчета дат. Город Гринвич расположен невдалеке от Лондона.

В качестве демонстрации возможностей JavaScript по управлению датами и временем давайте создадим простой сценарий отображения текущего времени в описанном выше формате на Web-странице.

Начало сценария

Наш сценарий, как и большинство других программ JavaScript, начнем с дескриптора <SCRIPT>. Как вы уже знаете из главы 1, пара дескрипторов <SCRIPT>—</SCRIPT> используется для обозначения в программе HTML сценариев.



В программном коде сценария вводите только операторы используемого языка подготовки сценария и никакие другие. Если броузер обнаружит в этой области дескрипторы HTML, которые не принадлежат указанному языку подготовки сценария, то на экране появится сообщение об ошибке.

Откройте текстовый процессор и введите пару дескрипторов <SCRIPT>, как это показано в листинге 2.1 (В этом листинге и во всех остальных, приведенных в настоящей книге, каждая строка кода нумеруется для удобства.)

Листинг 2.1. Шаблон сценария JavaScript

```
1:          <SCRIPT LANGUAGE="JavaScript">
2:          </SCRIPT>
```

Поскольку в этом сценарии будут использоваться средства JavaScript, не поддерживаемые версией 1.0, при указании языка подготовки сценариев остановитесь на JavaScript 1.1. Этот сценарий будет выполняться в Netscape Navigator версии 2.0 и выше, а также Internet Explorer 2.0 и выше.

Добавление операторов JavaScript

Чтобы успешно создать сценарий, вам необходимо сначала правильно определить местное и абсолютное время, а затем отобразить их в броузере. Преобразование времени из одного формата в другой осуществляется исключительно встроенными средствами JavaScript.

http://ITBookz.ru/

Сохранение данных в переменных

В самом начале сценария необходимо определить *переменную,* в которой будет сохраняться текущая дата и дата начала нового тысячелетия. Детально о переменных речь пойдет в главе "5-й час. Использование переменных и функций". Сейчас же воспринимайте их как контейнеры, содержащие определенную информацию (число или, в нашем случае, дату).

После дескриптора <SCRIPT> введите приведенные ниже строки. Обратите внимание на использование строчных и прописных символов — JavaScript не только их различает, но и использует по-разному:

```
now = new Date();
```

Этот оператор создает переменную now, которая принимает значение текущей даты. Этот оператор и другие, используемые в сценарии, построены на основе объекта Data, необходимого для задания дат. Детальнее о использовании дат в JavaScript вы узнаете в главе "9-й час. Использование встроенных объектов".



В конце каждой строки программы JavaScript ставится разделитель — точка с запятой. Именно он указывает броузеру на конец строки программы. Вы можете не ставить точку с запятой, но я, по старой памяти, буду продолжать это делать.

Вычисление значения

JavaScript определяет дату как количество миллисекунд, отсчитанных от 1 января 1970 года. К счастью, в JavaScript встроены функции преобразования времени и даты в самые разные форматы; вам нет необходимости вручную преобразовывать миллисекунды в привычные вам секунды, часы, сутки, месяцы и годы.

Чтобы завершить сценарий, перед закрывающим дескриптором </SCRIPT> введите следующие две строки.

```
localtime=now.toString();
utctime=now.toGMTString();
```

Эти операторы создают две новые переменные: localtime и utctime. Первая содержит текущее время и дату, а вторая — абсолютные их значения.



В конечном счете обе переменные содержат текстовые значения. Например, January 1, 2001 12:00 PM. Выражаясь языком программистов, сохраненный в переменной текст называется строкой. Детально о строковых переменных рассказано в главе "6-й час. Использование массивов и строковых данных".

Вывод результата на экран

Теперь, когда вы имеете две переменные, принимающие требуемое значение в секундах, самое время побеспокоиться о выводе результата на экран. В противном случае они окажутся в сценарии просто бесполезными. В JavaScript существует несколько способов отображения данных на экране. Один из самых простых заключается в применении оператора document.write.

Оператор document.write позволяет отображать текст, числа и другие типы данных. Поскольку создаваемый сценарий вы планируете вставить в Web-страницу, нужно озаглавить рассчитанное значение следующим образом:

```
document.write("<b>Текущее время: </b>" + localtime + "<BR>");
document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
```


http://ITBookz.ru/

Этот оператор указывает броузеру вывести введенный в кавычках текст, а после него значение соответствующей переменной.



Обратите внимание на символ "+", введенный между отображаемым текстом и переменными. В данном случае он указывает на отображение текста и значения переменным в одной строке. Если использовать символ "+" между двумя переменными, то их значения будут просто суммироваться, и на экране отобразятся не два отдельных значения, а их сумма.

Вставка сценария на Web-страницу

Теперь вы имеете готовый сценарий, готовый для вставки на Web-страницу. Сначала проверьте программный код сценария. Для этого сравните его с приведенным в листинге 2.2. Если вы не все понимаете в этом листинге, то обратитесь к листингу 2.3, который содержит только комментарии к листингу 2.2.

Листинг 2.2. Сценарий определения текущего времени

```
<SCRIPT LANGUAGE="JavaScript">
now = new Date();
localtime=now.toString();
utctime=now.toGMTString();
document.write("<b>Текущее время: </b>" + localtime + "<BR>");
document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
</SCRIPT>
```

Листинг 2.3. Описание программы сценария определения текущего времени

```
С этого дескриптора начинается листинг
Определение текущей даты и сохранение ее в переменной now
  Текущее время сохраняется в переменной localtime
Абсолютное время сохраняется в переменной utctime
Отображает значений переменной localtime на Web-странице
Отображает значений переменной utctime на Web-странице
Конец сценария
```

Для того чтобы выполнить сценарий, необходимо вставить его в программу документа HTML. Самый простой документ HTML содержит дескрипторы <HTML>, <HEAD> и <BODY>.

Если вы добавите эти дескрипторы и описательный заголовок Web-страницы к имеющемуся листингу сценария, то получите вполне загружаемый в окне броузера документ HTML (листинг 2.4).

Листинг 2.4. Документ HTML со вставленным сценарием определения текущего времени

```
<HTML>
<HEAD><TITLE>Отображение даты</TITLE></HEAD>
<BODY>
<H1>Текущее время и дата</H1>
<p>
<SCRIPT LANGUAGE="JavaScript">
now = new Date();
localtime=now.toString();
```


http://ITBookz.ru/

```
9:          utctime=now.toGMTString();
10:         document.write("<b>Текущее время: </b>" + localtime + "<BR>");
11:         document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
12:     </SCRIPT>
13:     </BODY>
14:     </HTML>
```

Вот вы и получили документ HTML со сценарием JavaScript. Сохраните его с расширением .html или .htm. (В Windows 3.1 необходимо использовать только расширение .htm.)



Некоторые текстовые процессоры, подобные Notepad, по умолчанию сохраняют свои файлы с расширением .txt. Поэтому будьте внимательны и всегда проверяйте расширение сохраняемого файла.

Тестирование сценария

Для того чтобы протестировать сценарий, необходимо просто загрузить полученную Web-страницу в броузере. Запустите программу Netscape Navigator, откройте меню File (Файл) и воспользуйтесь командой Open Page (Открыть страницу). Щелкните на кнопке Choose File (Обзор) и укажите расположение файла на жестком диске. Выделив файл, щелкните на кнопке Open (Открыть).

Если вы ввели программу HTML правильно, то увидите на экране Web-страницу, подобную изображенной на рис. 2.1. (Отображаемое у вас значение не будет похоже на мое, если, конечно, вы не умеете путешествовать во времени.)

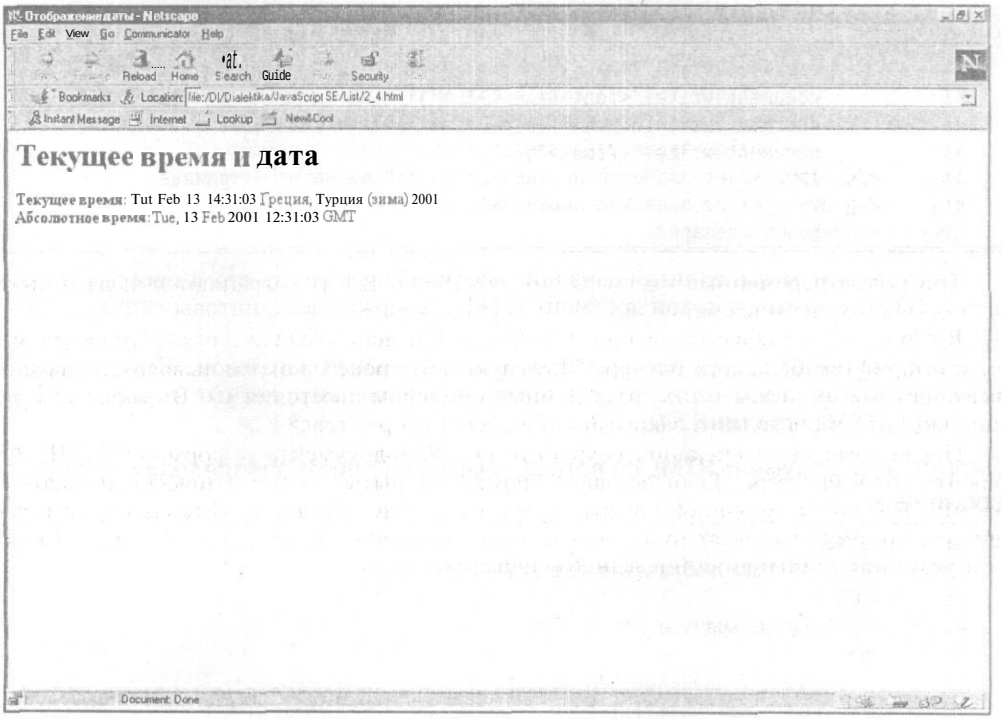


Рис. 2.1. Результат вычисления текущего времени

http://ITBookz.ru/

Изменение сценария

Загрузив Web-страницу, вы, наверное, обратили внимание, что отображаемое время имеет не очень привлекательный вид. Если вы не приверженец максимальной простоты и хотите отобразить значения красиво, вам придется немного преобразовать программу сценария. Давайте отобразим время так, как это делает большой электронный будильник. Для этого необходимо воспользоваться некоторыми дополнительными средствами JavaScript и HTML.

Для отображения времени нам потребуется ввести три новые переменные: для часов, минут и секунд. И в этом случае все сложные вычисления выполнят встроенные функции JavaScript. В листинге 2.5 приведен измененный код сценария.

Листинг 2.5. Измененный документ HTML со вставленным сценарием определения времени

```
1:  <HTML>
2:  <HEAD><TITLE>Отображение даты</TITLE></HEAD>
3:  <BODY>
4:  <H1>Текущее время и дата</H1>
5:  <p>
6:       <SCRIPT LANGUAGE="JavaScript">
7:       now = new Date();
8:       localtime=now.toString();
9:       utctime=now.toGMTString();
10:      hours=now.getHours();
11:      mins=now.getMinutes();
12:      secs=now.getSeconds();
13:      document.write("<b>Текущее время: </b>" + localtime + "<BR>");
14:      document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
15:      document.write("<font size='+5'>");
16:      document.write(hours + ":" + mins + ":" + secs);
17:      document.write("</font>");
18: </SCRIPT>
19: </BODY>
20: </HTML>
```

Три новые переменные заданы в строках 10-12. В них сохраняются часы, минуты и секунды текущего времени.

В строке 15 добавляется дескриптор . Он используется для отображения значений шрифтом большого размера. В следующей строке на экран выводится значение всех трех новых переменных, разделенных символом двоеточия (:). В строке 17 с помощью JavaScript вводится закрывающий дескриптор .

После изменения сценария сохраните код Web-документа в формате HTML. Откройте его в броузере. Если оставить броузер открытым и периодически щелкать на кнопке Reload, то можно обновлять время в его окне. Протестируйте сценарий и убедитесь, что он показывает то же время, что и сценарий листинга 2.5. На рис. 2.2 показан результат выполнения последнего сценария.

http://ITBookz.ru/

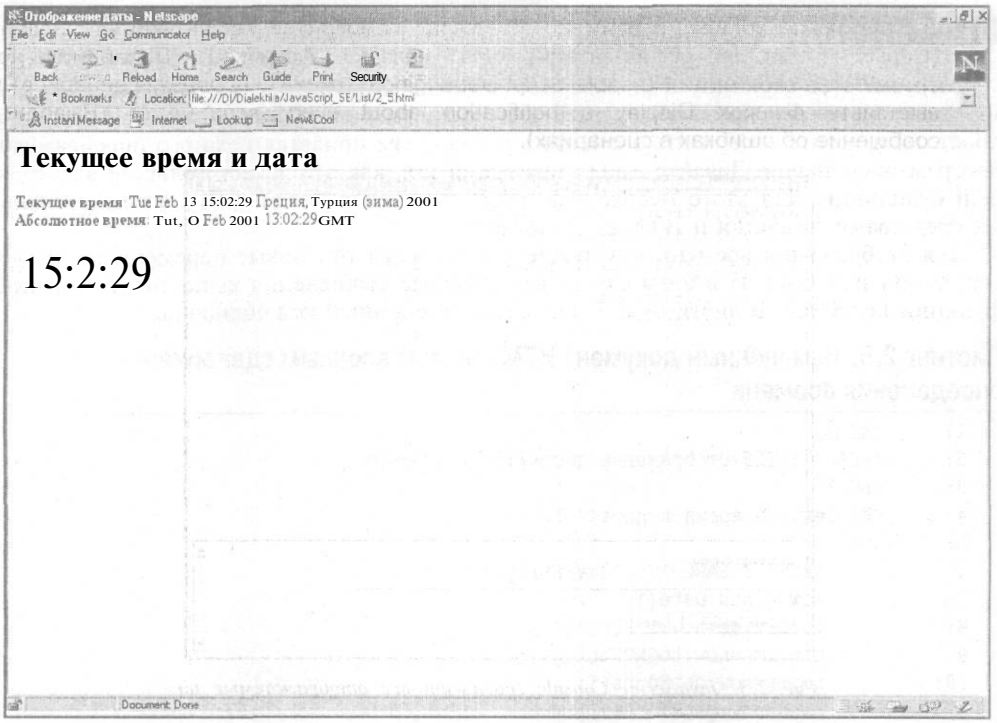


Рис. 2.2. Результат выполнения улучшенного сценария определения времени и даты

Выявление и устранение ошибок

Вот мы и создали полезный сценарий JavaScript и даже успели его изменить. Давайте теперь изменим сценарий таким образом, чтобы он содержал ошибку.

Вы можете удивиться: зачем добавлять ошибку в прекрасно работающий сценарий? Ответ: рано или поздно вы все равно встретитесь с ошибками в сценариях. Поскольку на простом примере их выявлять намного проще, давайте познакомимся с поведением Netscape Navigator при обнаружении ошибки в программе JavaScript.

Создать ошибку в программном коде очень просто. Измените оператор, созданный в прошлом разделе, — удалите закрывающую скобку. Тогда оператор будет выглядеть следующим образом:

```
document.write("</font>";
```

Сохраните документ HTML и обновите его в окне броузера. В зависимости от версии броузера вы получите либо сообщение об ошибке, либо сценарий просто не будет выполняться.

При отображении сообщения об ошибке вам легче устранить ее. Если сообщение об ошибке не отображается, вам необходимо настроить броузер таким образом, чтобы он автоматически тестировал ваш сценарий. Опять-таки, выполняемые операции зависят от используемого броузера.

- В Netscape Navigator версии 4.5 и выше включено средство JavaScript Console, которое отображает сообщения об ошибках. Чтобы отобразить консоль JavaScript, в поле введения адреса броузера введите `javascript:`. Рабочее окно консоли представлено на рис. 2.3. В нем отображаются те же сообщения об ошибках, что отображались на экране.

http://ITBookz.ru/

- В Internet Explorer версии 4.0 и выше выберите Tools⇨Internet Options (Сервис⇨Свойства обозревателя). На вкладке Advanced (Дополнительно) снимите флажок опции Disable Script Debugging (Отменить проверку сценариев) и выставьте флажок Display a notification about every script error (Выводить сообщение об ошибках в сценариях).

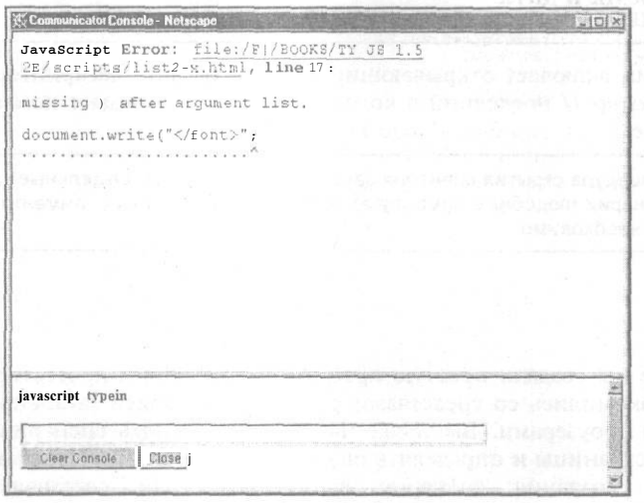


Рис. 2.3. JavaScript Console содержит все отображаемые на экране сообщения об ошибках

Сообщение missing) after argument list, правильно определяет источник ошибки. Будьте готовы к тому, что сообщение об ошибке не всегда правильно определяет ее причину. В любом случае, в окне консоли JavaScript указывается строка, в которой встречается ошибка.



Обратите внимание на поле в нижней части окна консоли JavaScript. Оно позволяет вам ввести оператор JavaScript, который будет выполнен немедленно. Таким образом вы можете познакомиться со многими возможностями JavaScript.

В то время, как в Internet Explorer для каждой ошибки отображается отдельное сообщение, в Netscape Navigator 4.5 в окне консоли все ошибки приводятся в одном списке. Именно по этой причине для отладки сценариев JavaScript лучше использовать Netscape Navigator.

Скрытие сценариев от старых броузеров

Поскольку старые броузеры не выполняют сценарии JavaScript и не понимают дескриптор <SCRIPT>, вам необходимо всячески изгаляться, чтобы правильно отобразить Web-страницу. В большинстве случаев программный код сценария просто отображается в средней части страницы (иногда вы просто его не замечаете).

Чтобы избежать подобного поведения броузера, используются дескрипторы комментариев. Они дают указание старым броузерам игнорировать программу сценария. Новые броузеры достаточно "умны", чтобы понять, что программа сценария, определенная как комментарий, на самом деле должна выполняться.

Комментарии в HTML начинаются с дескриптора <!-- и заканчиваются дескриптором -->. В листинге 2.6 приведен пример простого сценария, скрытого комментарием.

http://ITBookz.ru/

Листинг **2.6.** Скрытие сценария комментарием

```
1:    <SCRIPT LANGUAGE="JavaScript">
2:    <!--
3:    document.write("Ваш броузер поддерживает JavaScript");
4:    // -->
5:    </SCRIPT>
```

Этот сценарий включает открывающий и закрывающий дескрипторы комментариев HTML. Оператор // последний в комментарии. Он защищает комментарий HTML от определения его как ошибки в коде JavaScript.



Процедура скрытия сценария JavaScript не идеальна. Отдельные символы в вашем сценарии (подобные символу >) могут обозначить конец комментария раньше, чем это необходимо.

Резюме

В этом уроке мы создали простую программу JavaScript и протестировали ее в броузере. Вы познакомились со средствами создания сценариев JavaScript — текстовыми процессорами и броузерами. Вы также научились изменять сценарии, обновлять измененные Web-страницы и определять ошибки выполнения программы.

В процессе создания сценария вы использовали составляющие единицы JavaScript — переменные, оператор document.write и функции отображения даты и округления чисел. Более подробно о них вы узнаете из последующих уроков книги.

Вопросы и ответы

Это книга о JavaScript 1.5, почему же мы тогда в дескрипторе <SCRIPT> не определили его версию?

Поскольку сценарий не содержит никаких специальных элементов, присущих только JavaScript 1.5, нет необходимости обязательно указывать версию языка. Если в программе HTML ввести дескриптор <SCRIPT LANGUGE="JavaScript 1.5">, то сценарий будет выполняться только в Netscape Navigator 6.0 и выше. Поскольку же сценарий создается в JavaScript без указания версии, его можно выполнять в Netscape Navigator версии 3.0 и выше, Internet Explorer и других броузерах, совместимых с технологией JavaScript.

При запуске сценария он не выполняется, а его программа отображается в окне броузера. Что я сделал неправильно?

Такое поведение броузера вызывается тремя причинами. Первая: вы забыли перед первой строкой сценария вставить дескриптор <SCRIPT>. Убедитесь также, что правильно указан номер версии <SCRIPT LANGUAGE="JavaScript">. Вторая: ваш файл сохранен в формате .txt. В этом случае просто переименуйте его в файл документа HTML (.htm или .html). Третья: ваш броузер не поддерживает JavaScript или его поддержка отключена в диалоговом окне свойств броузера.

Зачем нужны дескриптор <p> и <b> в сценарии. Я думал, что дескрипторы HTML используются только вне программы сценария.

Поскольку указанный дескриптор введен в кавычках, он не нарушает правил написания сценариев. Для добавления форматирования дескрипторы HTML можно использовать в коде сценария, но только при введении в кавычках.

http://ITBookz.ru/

Мне понравилось оформление Web-страницы с помощью сценария. Можно ли сохранить его в виде отдельного файла, чтобы применить к другой Web-странице?

Да. Создайте файл с расширением .js и скопируйте в него все операторы сценария (без дескрипторов <SCRIPT>). Вместо дескриптора <SCRIPT> введите следующий дескриптор: <SCRIPT LANGUAGE="JavaScript" SRC="имя_файла.js">. Эта возможность реализована только в JavaScript версии 1.1 и выше.

Семинар

Контрольные вопросы

1. Какая программа используется для создания и изменения сценариев JavaScript?
   a) Броузер
   b) Текстовый процессор
   c) Ручка и чистый лист бумаги
2. Какие переменные используются в сценариях JavaScript?
   a) Сохраняющие числа, даты и другие значения
   b) Определяемые случайным образом
   c) Из школьного курса алгебры
3. Что вводится в конце сценария JavaScript?
   a) **Дескриптор** <SCRIPT LANGUAGE="JavaScript">
   b) **Дескриптор** </SCRIPT>
   c) **Оператор** END

Ответы

1, b) Для создания сценариев JavaScript может использоваться любой текстовый процессор. Вы также можете обратиться и к редактору HTML.

2, a) Переменные используются для сохранения чисел, дат и других значений.

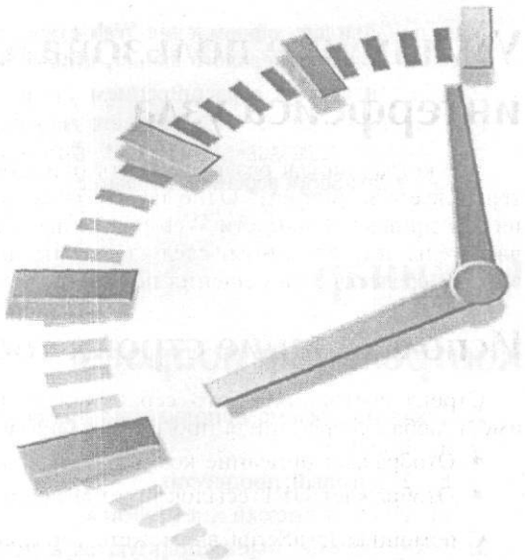
3, b) Сценарий заканчивается дескриптором </SCRIPT>.

Упражнения

Если вы хотите закрепить знания, полученные в этом уроке, то выполните следующее.

- Добавьте в сценарий определения даты и времени комментарии. Убедитесь, что сценарий выполняется в броузере.
- Добавьте в электронные часы раздел миллисекунд. Используйте оператор getMilliseconds. Проверьте правильность выполнения сценария.
- Создайте большие электронные часы для отображения абсолютного времени. Чтобы правильно определить время, используйте операторы getUTCHours, getUTCMinutes и getUTCSeconds. Этот сценарий должен выполняться в JavaScript версии 1.2 и выше.

http://ITBookz.ru/



3-й час

Возможности JavaScript

В главе "2-й час. Создание простых сценариев" мы создали работающую программу JavaScript. Правильно выполнив все инструкции предыдущего урока, вы создали Web-страницу, в которой указывается текущее время в двух форматах. Но вы наверняка не будете добавлять подобные строки на все Web-страницы. Вам не терпится познакомиться с действительно полезными возможностями JavaScript.

Перед тем как перейти к детальному изучению возможностей JavaScript, я хотел бы еще раз напомнить вам о предназначении этого языка подготовки сценариев. В течение этого часа вы познакомитесь с часто создаваемыми с помощью JavaScript приложениями. (Некоторые из них детально будут рассмотрены в следующих уроках.) В этой главе рассматриваются следующие темы.

- **Улучшение пользовательского интерфейса узла с помощью JavaScript**
- **Создание надоедливых бегущих строк**
- **Добавление с помощью JavaScript изображений и анимации**
- **Использование JavaScript для изменения форм и автоматизации их заполнения**
- **Определение типа и версии броузера с помощью JavaScript**
- **Совместная работа JavaScript с внедряемыми модулями**
- **Создание с помощью JavaScript сложных приложений**
- **Копирование сценария на Web-страницу**

http://ITBookz.ru/

Улучшение пользовательского интерфейса узла

Вас когда-нибудь раздражал хоть один Web-узел (если нет, то вы просто мало путешествуете в Internet)? Одно из наиболее частых применений JavaScript — это повышение привлекательности Web-страницы. Среди элементов, которые повышают привлекательность страницы, следует назвать анимационные рисунки и бегущие строки, а также средства перемещения по документу и отображения информации.

Использование строки состояния

Строка состояния — это серая панель в нижней части окна броузера, которую имеет любая современная программа. Она выполняет следующие основные функции.

- Отображает описание команд меню и кнопок панели инструментов
- Отображает URL ссылок при наведении на них указателя мыши

С помощью JavaScript вы можете управлять строкой состояния. Вы, наверное, уже замечали бегущие сообщения, которые отображаются при загрузке некоторых Web-страниц. Хотя в некоторых случаях они и раздражают (или мне это просто кажется), в большинстве ситуаций с их помощью можно узнать много полезной информации.



О том, как написать сценарий, позволяющий добавить в строку состояния бегущие сообщения, вы узнаете в главе "6-й час. Использование массивов и строковых данных". Если же вам не терпится узнать об этом раньше, то ознакомьтесь с предпоследним разделом этой главы.

Кроме добавления бегущих сообщений, вы можете использовать строку состояния некоторыми другими способами. Еще один пример частого использования JavaScript для модернизации строки состояния: замена адреса ссылки, на которую наведен указатель мыши, описанием Web-страницы.

Средства перемещения по документу

С помощью JavaScript вы сделаете свой Документ HTML удобным для просмотра и перемещения. Вы, конечно, уже встречали на Web-страницах раскрывающиеся списки, позволяющие перемещаться между различными страницами Web-узла. В списке вы указываете заголовок страницы и щелкаете на кнопке загрузки страницы. Указанная страница загружается в этом же окне броузера. В некоторых случаях страница загружается автоматически при выделении ее заголовка.

Подобные задачи почти всегда решаются с помощью JavaScript. Детально о добавлении раскрывающихся списков на Web-страницу мы поговорим в главе "14-й час. Формы введения данных". JavaScript позволяет выполнять и более сложные задачи. Например, Web-узел Netcenter компании *Netscape,* показанный на рис. 3.1, содержит раскрывающийся список, помогающий проводить поиск необходимой информации.

В поле введения условий можно проводить сложный поиск информации на этом же узле. Задав необходимые условия поиска и нажав кнопку Search (Поиск), вы посылаете запрос в необходимую поисковую систему.

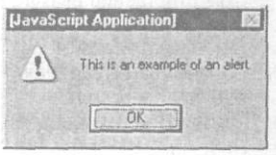
http://ITBookz.ru/



ft«c. *3.1. Враскрывающемся списке узла Netcenter компании Netscape выбирается поисковое средство*

Окна с сообщениями и другие элементы

JavaScript оснащен набором средств, предназначенных для уведомления пользователей о выполняемых операциях. Например, вы можете создать сообщение с предупреждением, напоминанием или подтверждением. Простой пример окна с предупреждением приведен на рис. 3.2. Предупреждение, выводимое на экран, уведомляет пользователя о ошибке, неправильно выполненной операции или отсутствии необходимой информации.



Если вам необходимо вывести на экран много информации, которая не помещается в простом окне с сообщением, создайте с помощью JavaScript новое окно броузера. При необходимости точно указывают все параметры окна — его размер, вид панелей инструментов и т.п.

Рис. 3.2. Окно с предупреждением, созданное с помощью JavaScript

При создании нового окна броузера в нем загружается Web-страница с указанным вами адресом или созданный вами документ HTML. Вы также можете оставить окно броузера пустым, не отображая в нем ни одной Web-страницы. JavaScript может оперировать и с фреймами, разделяющими окно броузера на отдельные области.



В главе "10-й час. Работа с объектной моделью документа" вы найдете описание методов создания окон с сообщениями, В главе "13-й час. Использование окон и фреймов" мы поговорим о создании пользовательского окна броузера и управлении фреймами.

http://ITBookz.ru/

Новые версии Netscape Navigator и Internet Explorer поддерживают динамические документы HTML, созданные на языке JavaScript. Динамические Web-страницы состоят из отдельных слоев (также известных как размещаемые объекты). Главное отличие слоев от фреймов заключается в их перекрываемости (подобно окнам ОС Windows) и перемещаемости в главном окне броузера. JavaScript позволяет управлять отдельными слоями и создавать поистине потрясающие анимационные эффекты.



Детально о динамических Web-страницах мы поговорим в главе "18-й час. Создание динамических страниц с помощью DOM".

Рисунки и анимация

Главные элементы, которые украшают Web-страницы и делают их более привлекательными, создаются на основе рисунков. Прочитав эту книгу, вы, возможно, не научитесь создавать рисунки, но точно научитесь управлять ими с помощью JavaScript.

При загрузке в окне броузера документа HTML текст загружается параллельно с рисунками. Рисунки на Web-странице не изменяются до тех пор, пока вы не отобразите в окне другой документ HTML. JavaScript полностью меняет это правило. JavaScript 1.1 и более поздние его версии имеют средство управления *динамическими рисунками,* позволяющее заменять на Web-странице одно изображение другим, не перезагружая ее полностью.

Одно из основных применений подобных рисунков состоит в изменении рисунка при наведении на него указателя мыши. Таким образом, как правило, выделяются рисунки, используемые в качестве гиперссылок на другие Web-страницы.



Рисунки, которые изменяются при наведении на них указателя мыши, известны как динамические, или переменные рисунки, или просто как рисунки, изменяющиеся при наведении указателя мыши. Подробно о них будет рассказано в главе "15-й час. Добавление рисунков и анимации".

JavaScript позволяет также отображать рисунки в указанной последовательности, создавая таким образом анимационные картинки. Если при создании подобных изображений не пользоваться ограничениями (например, при создании динамической компьютерной игры), то Web-страница может стать очень большой и громоздкой.



Анимация не входит в число преимуществ JavaScript. Java в этом отношении выглядит предпочтительнее. Быстро сменяющие друг друга изображения лучше создавать с помощью Java (но кто сказал, что это просто). Анимационные картинки и динамические рисунки в формате GIF можно также создавать и в графических редакторах. Сложные динамические изображения и видеоклипы требуют использования специальных приложений, таких как ShockWave и QuickTime.

Изменение форм

Один из главных элементов получения информации от пользователей в Web — это формы ввода данных. Они используются для введения пользователями информации, передаваемой впоследствии в базу данных Web-узла. Эта возможность выглядит привлекательнее, чем простое ознакомление с содержимым Web-страницы. Формы позволяют проводить самые различные операции: от заказа товаров в электронных магазинах до получения сведений о популярности узла.

http://ITBookz.ru/

Традиционно формы вводения данных управляются CGI (общим шлюзовым интерфейсом). Сценарий запускается на сервере. После введения пользователем на форме всей необходимой информации необходимо щелкнуть на кнопке Submit (Отправить). При этом сценарий отправляется на выполнение на сервере. После выполнения сценария его результаты отправляются обратно в броузер и Web-страница обновляется.



CGI — это не язык программирования, а стандарт, определяющий взаимодействие программ и языков программирования на Web-сервере. Программы CGI обычно создаются на языке Perl, более многофункциональном языке подготовки сценариев, чем JavaScript, а также C или Visual Basic.

Несмотря на то, что CGI — это надежная система, она имеет и свои недостатки. Главный недостаток заключается в необходимости передачи данных между броузером и сервером после заполнения формы и при обработке результатов. Это может занять от нескольких секунд до нескольких минут, что часто раздражает пользователей.

Как правило, сценарии CGI используются для *проверки правильности введения данных* на форме. Например, если пользователь вводит в поле телефонного номера только четырехзначное число, форма считается незаполненной и отправляется на повторное заполнение.

Этот метод проверки правильности введенных данных часто разочаровывает пользователей медленной обработкой данных, и впоследствии они негативно относятся к Web-страницам, содержащим формы ввода данных. Особенно часто недовольство вызывает возвращение формы на повторное заполнение при обнаружении ошибок в отображенных на ней сведениях.

Как вы уже догадались, JavaScript позволяет упростить процедуру введения информации на формах. Программы JavaScript обрабатывают данные автономно, не привлекая к этому сервер, и выводят сообщение об ошибке в отдельном окне. При этом указатель мыши наводится на то поле, которое было заполнено неправильно.



Программы JavaScript, предназначенные для проверки правильности введенных на форме данных, будут рассмотрены в главе "14-й час. Формы введения данных".

В некоторых случаях JavaScript используется для полного управления формой, включая и получение результатов обработки данных, введенных на форме, в виде почтового сообщения. Сценарий JavaScript позволяет также передавать данные после их проверки сценарию CGI.

В JavaScript интегрированы средства автоматизирования введения данных на форме. На рис. 3.3 показан пример формы, которая частично уже заполнена при выполнении сценария JavaScript.



JavaScript управляет формами быстрее, чем CGI, что вызвано автономностью их выполнения. Вы можете спросить, зачем тогда необходимо использовать CGI? Дело в том, что формы, запускаемые на клиенте, всегда остаются простыми и не имеют всех тех возможностей, которые имеют сценарии CGI. (Скоро появится версия JavaScript, запускаемая на сервере. Она должна стать достойным конкурентом CGI.)

http://ITBookz.ru/

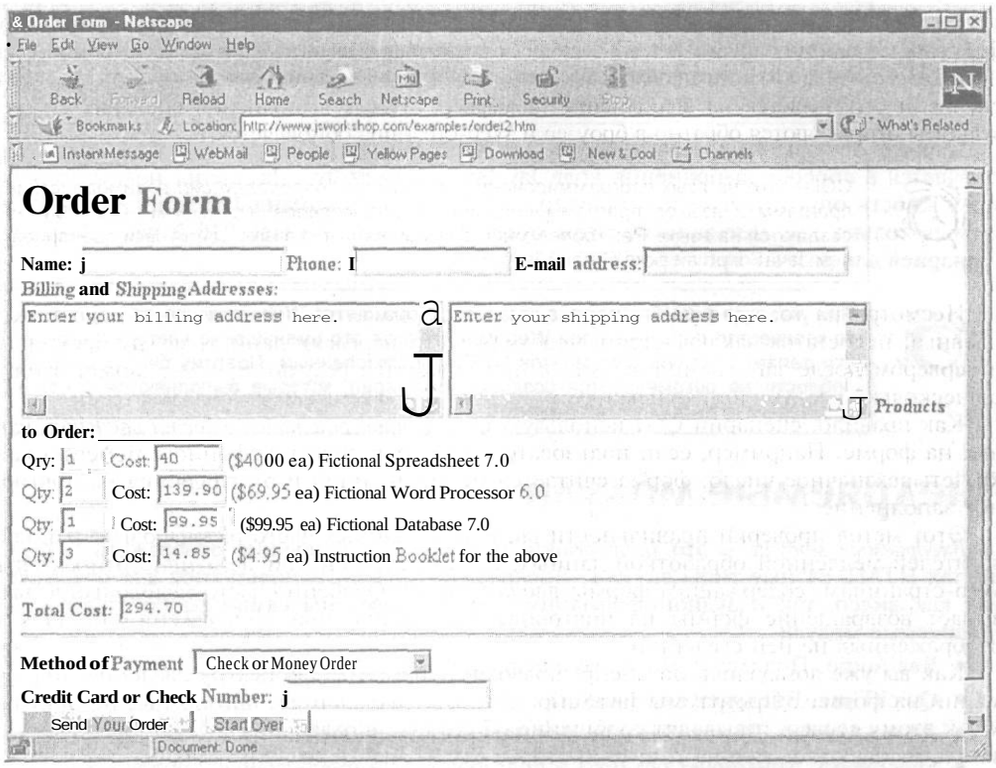


Рис. 3.3. Форма заказа товаров, частично заполненная при выполнении сценария JavaScript

Определение версии броузера

Стандарт HTML разрабатывался как универсальный, единый для всех платформ. Другими словами, если придерживаться стандарта HTML, то страница будет одинаково выглядеть в любом броузере, установленном на любой платформе.

Если вы долгое время занимались Web-дизайном, то знаете, что не всегда одни и те же страницы отображаются одинаково в разных броузерах. Новые броузеры от *Netscape* и *Microsoft* лучше всего поддерживают стандарт HTML. Это, правда, не означает, что они одинаково отображают данные. Примите в расчет и то, что броузеры имеют различные версии, которые также по-разному отображают те или иные элементы Web-страницы.

В то время как простые Web-документы отображаются одинаково в обоих броузерах, новые элементы, такие как динамические изображения, отображаются в них по-разному. Чтобы выйти из этой затруднительной ситуации, необходимо для каждого броузера выпускать свою версию Web-страницы.

Некоторые узлы и были сконструированы подобным образом. При открытии их Web-страниц на экране появляется запрос на указание используемого броузера и его версии. С помощью JavaScript этот процесс автоматизируется. Тип броузера определяется автоматически при загрузке Web-страницы. При этом либо загружается страница, соответствующая определенному броузеру, либо имеющаяся страница изменяется таким образом, чтобы выглядеть правильно.

http://ITBookz.ru/



В связи с разницей в отображении Web-страницы в разных броузерах необходимо загрузить ее и в Internet Explorer, и в Netscape Navigator. Если JavaScript используется для определения броузера, обязательно установите в системе их оба (желательно самой последней версии).

Главное отличие в броузерах *Microsoft и Netscape* (в различных их версиях тоже) заключается в способе выполнения кода JavaScript. Поэтому в JavaScript предусмотрена возможность определения типа броузера и выполнения команд JavaScript, поддерживаемых только им. Детально об этом будет рассказано в главе "16-й час. Создание сценариев для разных броузеров".



При указании типа броузера, *в* котором отображается Web-страница, вы уменьшаете количество пользователей Web-узла. Иногда это уменьшение настолько велико, что делает создание документов HTML бессмысленным. Поэтому без крайней надобности не рекомендуется создавать сценарии, которые выполняются в одном броузере и не выполняются в другом.

Внедряемые модули

Внедряемые модули — это надстройки броузера, позволяющие отображать в документах HTML разные типы данных. Внедряемые модули используются для отображения как видео, так и аудиоинформации. Ниже приведены самые распространенные внедряемые модули.

- RealAudio. Поддерживает потоки аудиоданных.
- QuickTime. Внедряет видеоклипы.
- Adobe Acrobat. Дополнительно форматирует документы.
- ShockWave. Анимирует объекты и поддерживает интерактивные приложения.

Как видно, внедряемые модули используются для добавления в документы HTML данных практически любых типов. Но существует одна маленькая проблема: наряду с интегрированными в броузер внедряемыми модулями, выпускаются внедряемые модули, которые необходимо дополнительно устанавливать вручную.

Это вызывает почти ту же проблему несовместимости, что и проблема с версиями броузеров. Или вы дополнительно устанавливаете необходимый внедряемый модуль, или создаете альтернативную Web-страницу, которая не требует его инсталляции (в этом случае вы размещаете на узле две разные версии одной страницы).

И опять, JavaScript немного облегчает ваши страдания. Он позволяет определить при непосредственной загрузке Web-страницы наличие в броузере необходимого модуля. Если таковой отсутствует, загружается альтернативная Web-страница, не требующая его использования. Если модуль обнаружен, то загружается исходная страница.

Кроме определения необходимого внедряемого модуля, JavaScript позволяет управлять его содержимым. Средство управления внедряемым модулем в JavaScript называется LiveConnect.



Кроме всего прочего, LiveConnect позволяет программам JavaScript соединяться с аплетами Java и предоставлять в их распоряжение переменные и команды JavaScript.

Например, вы можете внедрить на Web-страницу звуковой клип и с помощью JavaScript воспроизводить его в заранее определенные моменты. В зависимости от типа внедряемого модуля вы, с помощью сценария, можете управлять различными параметрами воспроизведения звука.

http://ITBookz.ru/



О том, как на практике использовать JavaScript для определения присутствия в броузере внедряемого модуля и управления объектами внедряемых модулей, мы поговорим в главе "20-й час. Использование мультимедиа и встроенных утилит".

Сложные сценарии

В этой главе вы уже познакомились с наиболее примечательными способами использования JavaScript. Все перечисленные возможности по отдельности реализуются на практике с помощью одной или нескольких команд, составляющих простой сценарий.

Но не дайте себя обмануть и не думайте о JavaScript как о языке создания только простых сценариев. С его помощью реализуются и сложные, полнофункциональные программы, хотя и имеющие некоторые ограничения.

В главе "23-й час. Создание сценария игры" вы создадите программу простой компьютерной игры. Она состоит из большого числа строк и достаточно непростого для понимания сценария.

Копирование сценария

Теперь вы поняли, как много полезного можно выполнить с помощью JavaScript. В следующих главах книги вы детальнее познакомитесь со всеми описанными выше возможностями.

Конечно, вам не терпится немедленно приступить к изменению вашего Web-узла. В некоторых ситуациях важнее не понять структуру сложного сценария и создать его аналог в одном из документов HTML, а поступить проще и скопировать его без зазрения совести.

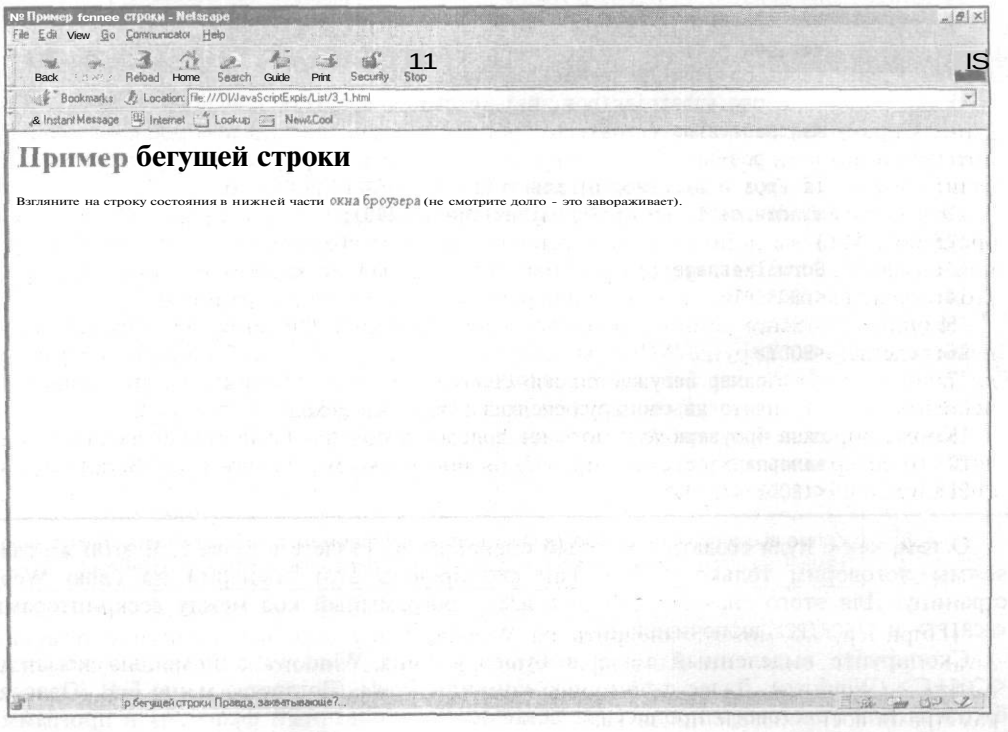


Рис. 3.4. Бегущая строка на Web-странице

http://ITBookz.ru/

Например, вы хотите добавить в строку состояния Web-страницы бегущую строку. Вместо того чтобы создавать сценарий с нуля, вы можете просто скопировать готовый сценарий с уже существующей страницы.



Будьте осторожны. Подавляющее большинство созданных Web-страниц защищены законом об авторских правах. Перед тем как использовать часть документа HTML, созданного другим разработчиком, получите у него на это разрешение. В некоторых случаях сценарии на узлах предоставляются для свободного использования. Некоторые из них указаны в Приложении А. "Ресурсы JavaScipt".

Например, вы хотите скопировать сценарий бегущей строки с уже готовой Web-страницы, расположенной по адресу:

```
http://www.jsworkshop.com/scroll.html
```

и изменить ее в соответствии с вашими потребностями. (Не волнуйтесь — это моя Web-страница, поэтому ожидать прихода стражей порядка вам не следует.)

На рис. 3.4 показана эта Web-страница в окне Netscape Navigator.

Чтобы скопировать сценарий из программного кода этой страницы, используйте команду Page Source (В виде HTML) меню View (Вид). На экране появится диалоговое окно с программным кодом HTML страницы. Листинг 3.1 отображает этот код.

Листинг 3.1. Программный код HTML страницы со сценарием бегущей строки

```
1:  <HTML>
2:  <HEAD><TITLE>Пример бегущей строки</TITLE>
3:  <SCRIPT LANGUAGE="JavaScript">
4:  var msg = "Это пример бегущей строки. Правда, захватывающе?";
5:  var spacer = "...                                    ...";
6:  var pos = 0;
7:  function ScrollMessage(){window.status =
8:  msg.substring(pos, msg.length) + spacer +
    msg.substring(0,pos);
9:  pos++;
10: if (pos > msg.length) pos = 0;
11: window.setTimeout("ScrollMessage()",200);
12: }
13: ScrollMessage();
14: </SCRIPT>
15: </HEAD>
16: <BODY>
17: <H1>Пример бегущей строки</H1>
18: Взгляните на строку состояния в нижней части
19: окна броузера (не смотрите долго - это
20: завораживает).
21: </BODY></HTML>
```

О том, как с нуля создать подобный сценарий вы узнаете в главе 6. В этой же главе мы поговорим только о том, как скопировать этот сценарий на свою Web-страницу. Для этого сначала выделите весь программный код между дескрипторами <SCRIPT> и </SCRIPT> включительно.

Скопируйте выделенный текст в буфер обмена Windows *с* помощью команды <Ctrl+C> (Windows). Далее с помощью команды Paste (Вставить) меню Edit (Правка) броузера или текстового процессора вставьте скопированный фрагмент в программный код любой Web-страницы.

http://ITBookz.ru/

Как и в приведенном примере, сценарий нужно заключить в пару дескрипторов <HEAD>. Удостоверьтесь, что скопирован весь сценарий и что дескриптор </SCRIPT> расположен перед дескриптором </HEAD>.

После того как сценарий скопирован, приступите к изменению бегущей надписи. Для этого введите собственное сообщение между символами кавычек в строке var msg=. Это первая строка после дескриптора <SCRIPT>.

После вставки сценария в документ HTML и изменения бегущего сообщения сохраните программный код Web-страницы и загрузите ее в окне броузера. В строке состояния должна появиться строка состояния с введенным вами текстом. (Если вы ее не наблюдаете, значит вы по ошибке неправильно изменили часть программного кода. Детально о предназначении каждого оператора сценария рассказано в главе 6.)

Резюме

В течение этого часа кратко рассмотрены возможности JavaScript и способы его применения. Вы узнали об использовании JavaScript для создания окон с сообщениями, рисунков и форм, а также о применении его для определения типа броузера и внедряемых модулей.

Кроме того, вы узнали, как копировать сценарий из одного программного кода в другой. Начиная с главы "4-й час. Выполнение программ JavaScript", вы познакомитесь с описанными выше возможностями более подробно.

Вопросы и ответы

Мне встречались бегущие строки, расположенные в верхней части Web-страницы, а не в строке состояния. После щелчка на определенной их части можно было перейти к другой Web-странице. Позволяет ли JavaScript создавать такие элементы документа HTML?

Большинство подобных объектов создается с помощью Java, обладающего более широкими возможностями. С помощью JavaScript вы можете создавать бегущие строки в теле Web-страницы только в том случае, если они размещаются в поле формы или на слое динамического документа HTML.

Известно, что с помощью JavaScript можно создавать рисунки, которые изменяются при наведении на них указателя мыши. Как насчет рисунков, которые изменяются после щелчка на них?

Действительно, JavaScript позволяет создавать и подобные рисунки. Детально о них рассказано в главе 15.

JavaScript позволяет распознавать разные версии броузеров. Можно ли при этом узнать, поддерживает ли броузер JavaScript?

Честно говоря, вы не можете этого сделать, поскольку броузер, который не поддерживает JavaScript, не запустит ваш сценарий. Тем не менее, броузеры, которые поддерживают JavaScript, позволяют определить номер его версии. В броузерах, которые не поддерживают JavaScript, используйте строки комментариев и дескриптор <NOSCRIPT>. Детально об этом рассказано в главе 1.

Я скопировал один из сценариев в мою Web-страницу, но он не выполняется. Неужели я сделал ошибку?

Существует много причин, по которым сценарий не хочет выполняться. Возможно, вам больше повезет, когда вы детально познакомитесь с возможностями JavaScript и сможете проанализировать скопированный сценарий.

http://ITBookz.ru/

Семинар

Контрольные вопросы

1. Чего *нельзя* выполнить с помощью JavaScript, запускаемого на клиенте?
   a) Проверить правильность введения формы
   b) Отправить содержимое формы в виде почтового сообщения
   c) Сохранить содержимое формы в файл базы данных на сервере
2. Что такое CGI?
   a) Язык подготовки сценариев для Web-серверов
   b) Спецификация, позволяющая запускать программы на Web-сервере
   c) Компания, производящая Web-серверы

Ответы

1, c) JavaScript не позволяет управлять файлами базы данных на сервере

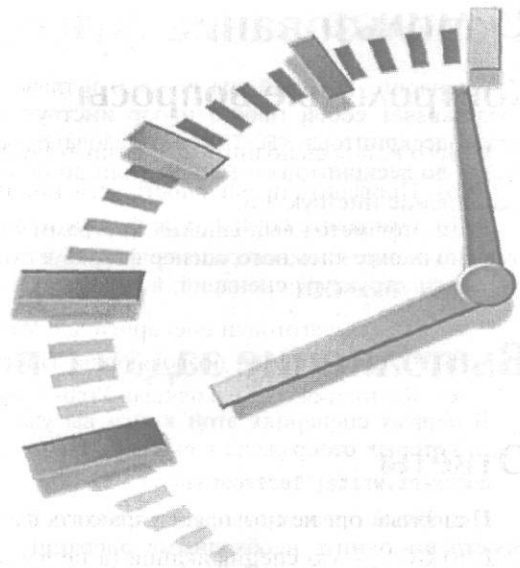
2, b) CGI — это спецификация (а не язык), позволяющая профаммам запускаться на Web-сервере

Упражнения

Если вы хотите детальнее ознакомиться с возможностями JavaScript перед изучением следующих глав, выполните следующие упражнения.

- Зайдите в Web и попытайтесь найти примеры элементов и объектов, создаваемых с помощью сценариев JavaScript
- Введите в поле адреса броузера Netscape команду `about:plugins`. При этом в его окне отобразится список установленных внедряемых модулей и типов файлов, управляемых ими.

http://ITBookz.ru/



4-й час

Выполнение программ JavaScript

Вот мы и добрались до последней главы части I. В первых трех главах вы знакомились с предназначением и возможностями JavaScript, а также научились создавать простые сценарии и вставлять их в программу Web-страницы.

На протяжении этого часа вы познакомитесь с основными элементами JavaScript, которые используются практически в любом сценарии. Эта глава поможет вам быстрее изучить излагаемый в следующих главах материал, описывающий методы использования функций и операторов.

В этой главе рассмотрены следующие темы.

- **Создание сценариев с использованием функций**
- **Применение объектов в программах JavaScript**
- **Управление событиями с помощью JavaScript**
- **Введение программ, альтернативных сценариям JavaScript, для выполнения в старых броузерах**

http://ITBookz.ru/

Использованиефункций

Сценарий, который мы создали в главе "2-й час. Создание простых сценариев", представляет собой просто набор инструкций. При загрузке Web-страницы броузер после дескриптора <SCRIPT> последовательно выполняет все инструкции сценария. Дойдя до дескриптора </SCRIPT> (или до ошибочной инструкции), броузер прекращает выполнение инструкции.

Если этот метод выполнения программ JavaScript применим к простым сценариям, то выполнение сложного сценария требует совершенно иных методов. Для того чтобы упростить структуру сценария, в JavaScript используются *функции*.

Выполнение задач с помощью функций

В первых сценариях этой книги вы уже познакомились с операторами JavaScript, часть которых отображена в скобках. Например:

```
document.write("Тестирование")
```

Подобные операторы представляют собой функции. Функции позволяют быстро и просто выполнить необходимые операции, например отобразить значение переменной в окне броузера. В JavaScript внедрено огромное количество функций. С большинством из них вы познакомитесь по мере изучения этой книги. Оператор, который позволяет выполнить функцию, называется *оператором вызова функции*.

Функции имеют собственные параметры (вводятся в скобках), которые и определяют то, что они выполняют. Кроме того, функция часто возвращает значение в специально определенную в сценарии переменную. Например, приведенная ниже функция выводит на экран запрос на введение текста и сохраняет его в виде строковой переменной:

```
text=prompt("Введите произвольный текст");
```

При необходимости вы можете создавать собственные функции. Собственные функции создаются исключительно по двум причинам. Первая: разделение большого сценария на отдельные части, более простые для понимания и управления. Вторая: множественное использование функции для выполнения подобных и повторяющихся задач.



Методы определения, вызова функции и возвращения ею значений описаны в главе "5-й час. Использование переменных и функций".

Объекты

В главе 2 вы узнали, что переменные — это контейнеры, содержащие текст, число или другой тип данных. Кроме переменных, в JavaScript используются *объекты*. Подобно переменным, объекты предназначены для сохранения данных. Разница между ними заключается в том, что объекты могут содержать больше одного значения.

Каждый элемент данных, сохраненный в объекте, называется *свойством*. Например, вы можете в виде объекта сохранять контактную информацию о пользователях. Тогда в качестве свойств используется имя пользователя, его адрес, телефонный номер и другая информация.

Для разделения имени объекта от свойства используется разделитель. Например, для объекта пользователя Bob свойства определяются как `Bob.address` и `Bob.phone`.

http://ITBookz.ru/

Объекты могут также содержать и *методы*. Методы — это функции, которые обрабатывают свойства объекта. Например, объект нашего пользователя может содержать метод display(), отображающий на экране сведения о пользователе. В терминологии JavaScript оператор Bob.display() отображает сведения об объекте Bob.



Функция document.write, описанная ранее в этой главе, на самом деле представляет собой метод объекта document. Детально объекты описаны в главе "10-й час. Работа с объектной моделью документа".

Не расстраивайтесь, если вы чувствуете себя некомфортно, знакомясь с этими определениями. Вы познакомитесь со всеми этими элементами в следующих главах. За этот час вы познакомитесь только с их определениями. В JavaScript поддерживается три типа объектов.

- *Встроенные объекты*. Эти объекты встроены непосредственно в JavaScript. В главе "2-й час. Создание простых сценариев" уже рассмотрены два из таких объектов: Date и Math (). В главе "6-й час. Использование массивов и строковых данных" будут рассмотрены объекты Array и String.
- *Объекты броузера*. Эти объекты представляют элементы броузера и текущего документа HTML. Например, функция alert(), описанная раньше в этой книге, на самом деле представляет собой метод объекта window. Детальнее об этом объекте вы узнаете в главе "10-й час. Работа с объектной моделью документа".
- *Пользовательские объекты*. Это создаваемые вами объекты. Например, вы можете создать пользовательский объект для каждого пользователя, описанный раньше. Пользовательские объекты детальнее рассмотрены в главе "11-й час. Создание пользовательских объектов".

Обработка событий

Как уже рассказывалось в главе 1, не все сценарии в программе HTML определяются дескриптором <SCRIPT>. Сценарии также используются и как *обработчики событий*. Хотя этот термин и относится к языкам программирования высокого уровня, ничего сложного в нем нет. Обработчик события — это сценарий, который управляет событиями.

В реальной жизни все время происходят какие-то событие. Например, в своем ежедневнике вы записываете: "Посетить зубного врача" или "День рожденья жены". Множество событий в жизни вы просто не замечаете, например покупка билета или неожиданный приход друзей.

Независимо от того, как происходят события — по расписанию или нет, вы хотите, чтобы они проходили нормально, без неожиданных инцидентов. Обработчик событий в реальной жизни может содержать следующую инструкцию: В день рождения жены с самого утра подарить ей подарок и потребовать большой шоколадный торт.

Обработчики событий в JavaScript поступают подобным образом. Они дают указания броузеру выполнить определенные действия при наступлении определенных событий. События JavaScript не так ярко выражены, как события реальной жизни. Например, *щелчок кнопкой мыши* или *конец загрузки страницы*. Но это не делает их малозначащими в программах JavaScript.

Много событий JavaScript (подобных щелчку кнопкой мыши) генерируют пользователи. Вместо того чтобы выполнять команды сценария в строго определенной последовательности, сценарий управляется событиями, задаваемыми пользователем.

Обработчик событий связывается с определенным объектом броузера и определяется в дескрипторе объявления объекта, например текстовые и графические ссылки

http://ITBookz.ru/

имеют событие onMouseOver, которое происходит при наведении указателя мыши на ссылку. Вот как выглядит дескриптор графической ссылки с обработчиком событий:

```
<IMG SRC="button.gif" onMouseOver="highlight();">
```

Вы определяете обработчик событий как параметр дескриптора HTML и добавляете оператор JavaScript, управляющий событием, в кавычках. Идеально для управления событием использовать функцию, поскольку имена функций короткие и позволяют выполнять одновременно несколько операторов.



Детально обработчики событий рассмотрены в главе "12-й час. Обработка событий".

Условные операторы

В то время, как обработчики событий уведомляют сценарий о выполнении некого действия, условные операторы позволяют проверить выполняемость определенного условия, например правильно ли ввел пользователь свой почтовый адрес.

Условные операторы JavaScript позволяют сравнивать самые разные значения и переменные. Типичное условие отображается при задании оператора if:

```
if (a==1) alert("Значение счетчика 1.");
```

Этот оператор позволяет сравнить значение переменной a с единицей. Если значение переменной равно 1, то на экран выводится предупреждение, введенное в кавычках. Обычно ни один сложный сценарий не обходится без условных операторов.



Детально о условных операторах рассказано в следующей главе.

Циклы

Еще одно полезное средство JavaScript часто используется не только при создании сценариев, но и в любом языке программирования, — это циклы. Циклы просто незаменимы при выполнении повторяющихся операций. Например, приведенный ниже код позволяет десять раз отобразить в окне броузера одно и то же сообщение:

```
for (i=1; i<=10; i++) {
  alert("Да! Это еще одно сообщение!");
}
```

Оператор for — это основной оператор цикла. Циклы с таким оператором используются в JavaScript чаще всего. Естественно, в своих сценариях вы будете использовать циклы для выполнения более полезных задач.



Детально циклы рассмотрены в главе "8-й час. Повторение — мать учения: циклы"

Последовательность выполнения сценариев

В главе 1 этой книги я уже упоминал, что большой Web-документ часто содержит несколько сценариев JavaScript, а поэтому и несколько дескрипторов <SCRIPT>. Возникает справедливый вопрос, как броузер определяет очередность выполнения сценариев? Чтобы не запутаться при добавлении на Web-страницу сценариев JavaScript, следуйте приведенных ниже правилам.

http://ITBookz.ru/

- Дескриптор <SCRIPT>, введенный в элементе <HEAD> документа HTML, всегда выполняется первым. Хотя этот сценарий и не позволяет отображать на экране данные, он прекрасно подходит для определения функций.
- Сценарий, добавленный в тело документа HTML, выполняется всегда после сценариев, введенных в его заголовке. Несколько сценариев в теле документа HTML выполняются в порядке их следования.
- Обработчики событий запускаются при выполнении соответствующих событий. Например, обработчик onLoad выполняется сразу после загрузки Web-страницы. Чтобы правильно задать функцию, используемую в обработчике событий, старайтесь всегда определять ее в заголовке документа HTML.

Опять о комментариях

Из главы 2 вы знаете, как с помощью комментариев скрыть сценарий от броузера. Другое применение комментариев — отображать в программе HTML объяснения и инструкции.

Комментарии позволяют включить в код сценария необходимую документацию. Она понадобится тому разработчику, который, возможно, попытается разобраться в вашем сценарии (или вам, если вы вернетесь к его рассмотрению после длительного перерыва). Чтобы включить комментарий в сценарий JavaScript, начните строку с двух символов косой черты:

```
//это комментарий
```

Если комментарий необходимо вставить в середине строки, поступите следующим образом:

```
a = a + 1; //это комментарий
```

JavaScript поддерживает C-подобные комментарии, начинающиеся с /* и заканчивающиеся */• Эти комментарии могут занимать больше, чем одну строку, что демонстрирует приведенный ниже пример:

```
/*Этот сценарий содержит самые различные команды
и операторы, а также комментарии */
```



Поскольку эти комментарии подчиняются только синтаксису JavaScript, они вводятся в области сценария. Их нельзя вводить в программном коде HTML.

Резюме

В этом уроке вы изучили некоторые основные методы программирования в JavaScript. Вы узнали, как определяется функция, используются объекты и обработчики событий, скрываются сценарии и применяются комментарии. Вы также научились исключать сценарии из программы HTML при отображении Web-страницы в старом броузере.

Вот мы и достигли конца части I этой книги. В ней вы познакомились с фундаментальными понятиями JavaScript и узнали об основных методах программирования на нем.

http://ITBookz.ru/

Вопросы и ответы

Я слышал о том, что C и Java — это объектно-ориентированные языки программирования. Если JavaScript поддерживает объекты, то он тоже должен быть объектно-ориентированным языком?

Да, хотя это и не совсем так. Объекты JavaScript не имеют всех тех свойств, что объекты Java и C.

Выполнение отдельных сценариев для каждого случая жизни кажется утомительных занятием. Почему бы не воспользоваться обработчиками событий?

Обработчики событий — это идеальный вариант (а JavaScript — единственный язык) создания ответного действия на нажатие кнопки, определение опции или введения данных. Использовать их в одном сценарии очень удобно. Чем создавать несколько сценариев, выполняющихся в каждом конкретном случае, лучше создать большой сценарий, управляемый обработчиками событий.

Семинар

Контрольные вопросы

1. Примером чего будет сценарий, который выполняется после щелчка на ссылке?
   a) Объекта
   b) Обработчика событий
   c) Невозможного
2. Какие из приведенных свойств характерны для функции JavaScript?
   a) Содержит параметры
   b) Возвращает значения
   c) Оба
3. Какой сценарий выполняется в броузере первым?
   a) Введенный в заголовке документа HTML
   b) Введенный в теле документа HTML
   c) Вызываемый обработчиком событий

Ответы

1, b) После щелчка пользователем на ссылке выполняется обработчик событий

2, c) Функция содержит параметры и возвращает в сценарий определенные значения

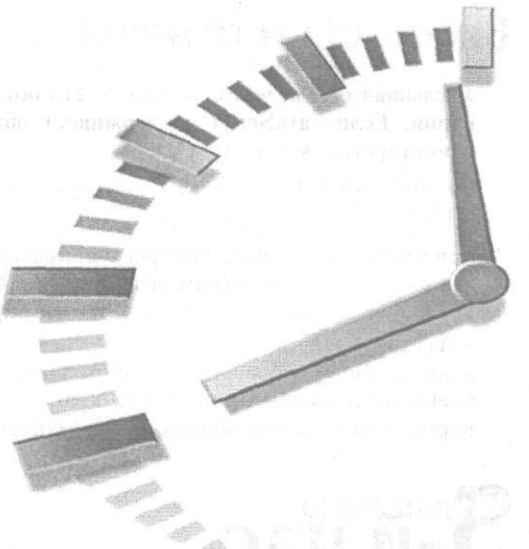
3, a) Определенный в заголовке Web-страницы

Упражнения

Чтобы подробнее познакомиться с возможностями JavaScript, описанными в этой главе, выполните следующие упражнения.

- Еще раз ознакомьтесь со сценариями, описанными в главе 2, и найдите в них все используемые функции.
- Добавьте в сценарии главы 2 комментарии, объясняющие назначение операторов.

http://ITBookz.ru/

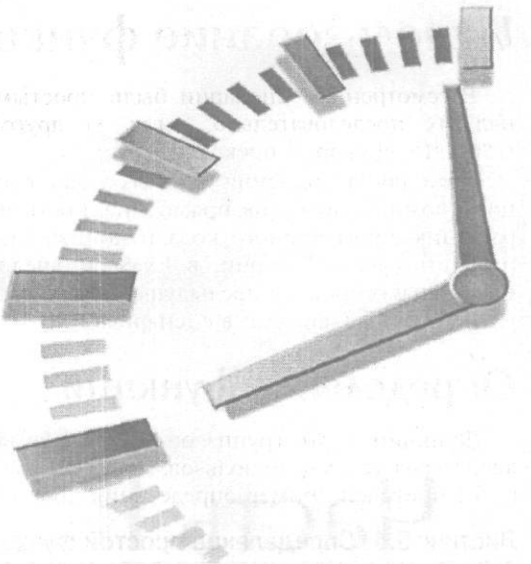


Часть II
Методы программирования на JavaScript

Темы занятий

5. Использование переменных и функций
6. Использование массивов и строковых данных
7. Тестирование и сравнение значений
8. Повторение — мать учения: циклы

http://ITBookz.ru/



5-й час

Использование переменных и функций

Вы переходите к изучению второй части этой книги. В ней вы найдете описание основных методов и средств создания сценариев, чаще всего используемых в программах JavaScript.

В этой главе речь пойдет о переменных и выражениях. В ней рассмотрены следующие темы.

- **Объявление и вызов функции**
- **Возвращение функцией значения**
- **Присваивание имен переменным и их объявление**
- **Использование глобальных и локальных переменных**
- **Определение значений переменных**
- **Сохранение в JavaScript различных типов данных**
- **Преобразование одних типов данных в другие**
- **Использование переменных и констант в выражениях**
- **Использование переменных для сохранения введенных пользователем данных**

http://ITBookz.ru/

Использование функций

Рассмотренные сценарии были простыми наборами инструкций. Броузер выполнял их последовательно, одну за другой. Достигнув закрывающий дескриптор </SCRIPT>, сценарий прекращался.

Если для выполнения простых задач этот подход вполне оправдан, то при реализации сложных проектов прямолинейность программирования приводит только к разрастанию программного кода и понижению эффективности. Чтобы правильно организовать ваши сценарии, в JavaScript внедрена поддержка функций, с которыми вы уже познакомились в предыдущей главе. В этой главе вы детально рассмотрим методы управления функциями в сценариях.

Определение функции

Функции — это группа операторов JavaScript, которые выполняются как одно целое. Перед тем как использовать функцию, вам необходимо ее определить. В листинге 5.1 приведен пример определения простой функции.

Листинг 5.1. Определение простой функции

```
1:    function Greet()    {
2:         alert("Внимание!");
3:    }
```

Листинг 5.1 определяет функцию, которая отображает на экране сообщение с предупреждением. Функция начинается с ключевого слова function. В приведенном примере функция называется Greet. Обратите внимание на параметры, введенные после названия функции. Как вы уже, наверное, знаете, в скобках можно приводить аргументы функции.

Первая и последняя строка определения функции содержит скобки {}. Эти скобки используются для определения операторов функции. Броузер по этим скобкам определяет начало и конец функции. В скобках, в нашем примере, записана всего одна строка. В ней введена встроенная функция alert, позволяющая отображать на экране сообщение с предупреждением. Сообщение содержит текст Внимание!.

Теперь поговорим о кавычках. Используемая функция Greet выполняет одно простое действие: отображает на экране сообщение. Если не ввести текст сообщения, то функция становится бесполезной.

Чтобы сделать функцию более гибкой, используются *параметры,* или *аргументы.* Это переменные, которые запрашиваются функцией при каждом ее вызове. Например, в нашу функцию можно вставить параметр who, который определяет пользователя, для которого отображается сообщение. Листинг 5.2 содержит программный код измененной функции.

Листинг 5.2. Функция с аргументом

```
1:    function Greet(who)  {
2:         alert("Внимание!" + who);
3:    }
```

Конечно, чтобы использовать эту функцию, необходимо вставить ее в программу документа HTML. Традиционно функция в программе документа HTML определяется в области заголовка. Поскольку область заголовка выполняется самой первой в программе, функция определяется до ее использования. Листинг 5.3 содержит определение функции Greet в заголовке программы документа HTML.

http://ITBookz.ru/

Листинг 5.3. Определение функции Greet в программе HTML

```
1:    <HTML>
2:    <HEAD>
3:    <TITLE>Функции</TITLE>
4:    <SCRIPT LANGUAGE="JavaScript">
5:    function Greet(who) {
6:         alert("Внимание!" + who);
V:    >
8:    </SCRIPT>
9:    </HEAD>
10:   <BODY>
11:   Тело страницы
12:   </BODY>
13:   </HTML>
```

Вызов функции

Теперь вы знаете, как определить функцию и вставить ее в программу HTML. Тем не менее, если вы загрузите программу листинга 5.3 в броузер, то обнаружите, что ровным счетом ничего не происходит. И все потому, что функция определена и готова к использованию, но не используется.

Для того чтобы использовать функцию в программе, необходимо ее *вызвать*. Для вызова функции в качестве оператора необходимо указать ее имя. В скобках после названия функции указываются параметры и значения. Например, функция Greet может содержать следующий параметр:

```
Greet("Фред")
```

Эта строка дает указание броузеру выполнить функцию Greet с указанными параметрами. Таким образом в функцию подставляется параметр "Фред" вместо переменной who.



Функции могут иметь не только один параметр. Чтобы определить функцию с несколькими параметрами, введите их в скобках, после названия функции, отделяя запятыми.

Листинг 5.4 содержит программу полноценного документа HTML, содержащую как определение функции, так и ее вызов в теле страницы. Чтобы продемонстрировать применимость функции, мы вызвали ее два раза, отобразив сообщения двум разным пользователям.

Листинг 5.4. Пример использования функции

```
1:    <HTML>
2:    <HEAD>
3:    <TITLE>Функции</TITLE>
4:    <SCRIPT LANGUAGE="JavaScript">
5:    function Greet(who) {
6:         alert("Внимание!" + who);
7:    }
8:    </SCRIPT>
9:    </HEAD>
10:   <BODY>
```


http://ITBookz.ru/

```
11:    <H1>Пример функции</H1>
12:    <P>Сообщение выводится два раза</P>
13:    <SCRIPT LANGUAGE="JavaScript">
14:    Greet("Фред")
15:    Greet("Дик")
16:    </SCRIPT>
16:   </BODY>
17:    </HTML>
```

Листинг содержит два одинаковых дескриптора <SCRIPT>. Первый раз при определении функции, второй — при ее вызове. Функция вызывается два раза для разных пользователей. Теперь, когда вы имеете готовый сценарий, попытайтесь загрузить его в броузере. При этом вы должны получить на экране нечто подобное изображенному на рис. 5.1.



Заметьте, что второе сообщение не отображается на экране до тех пор, пока вы не щелкнете на кнопке OK. После вызова функции выполнение сценария прерывается до удаления сообщения с экрана.

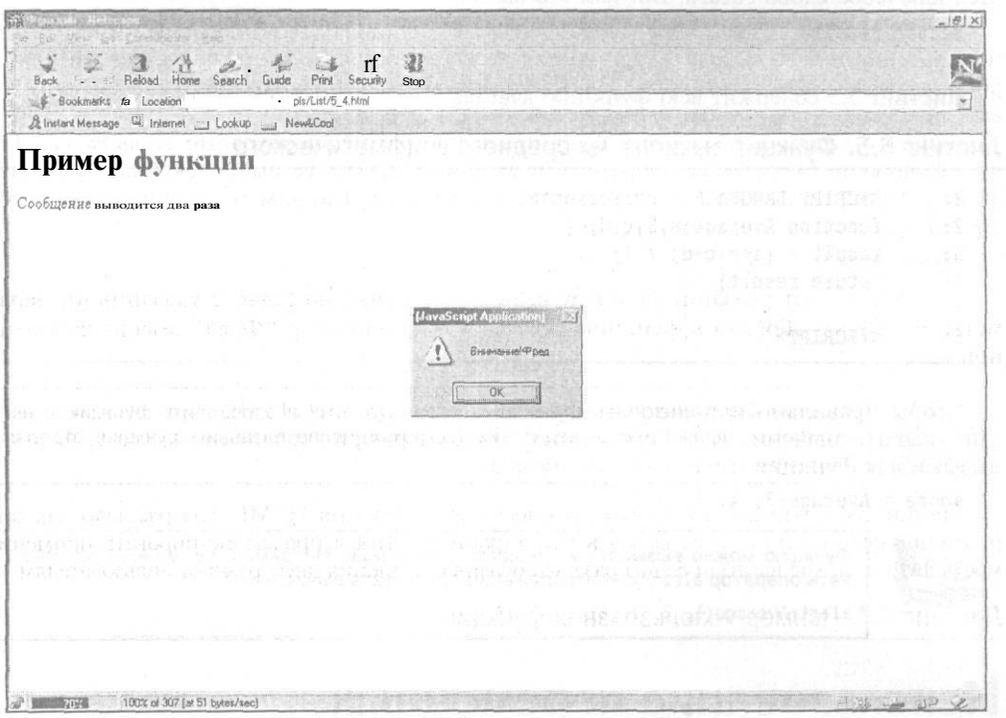


Рис. 5.1. Пример выполнения функции Greet

Возвращаемое значение

Наряду с выполненной вами функцией, отображающей на экране сообщение для пользователя, существуют функции, возвращающие в сценарий определенные значения. Это позволяет использовать функцию для вычислений. Для примера рассмотрим функцию, которая определяет среднее значение четырех чисел.

http://ITBookz.ru/

Функция начинается с ключевого слова function, названия функции и параметров. В качестве параметров будут использоваться четыре числа: a, b, c и d. Эти параметры принимают значения усредняемых чисел. Первая строка функции выглядит следующим образом:

```
function Average(a,b,c,d){
```



В первую строку функции включена и открывающая фигурная скобка. Это стандартный символ; его вы можете вводить и в отдельной строке.

Затем вам необходимо провести вычисление среднего значения указанных параметров. Среднее значение определяется как сумма всех значений, разделенная на их количество (в нашем случае 4). Вот как это выглядит в программе сценария:

```
result = (a+b+c+d) / 4;
```

Этот оператор создает переменную result и присваивает ей значение среднего арифметического четырех чисел. (Скобки ставятся, чтобы сделать операцию суммирования привилегированной и выполнить ее перед операцией деления.)

Для того чтобы возвратить результат в сценарий, содержащий функцию, используется ключевое слово return. Вот как это выглядит:

```
return result;
}
```

Листинг 5.5 содержит всю функцию Average.

Листинг 5.5. Функция вычисления среднего арифметического

```
1:    <SCRIPT LANGUAGE="JavaScript">
2:    function Average(a,b,c,d) {
3:    result = (a+b+c+d) / 4;
4:    return result;
5:    }
6:    </SCRIPT>
```

Чтобы правильно использовать функцию в программе HTML, при вызове функции укажите значения всех переменных. Например, при усреднении чисел 3, 4, 5 и 6 вызываемая функция имеет следующий вид:

```
score = Average(3, 4, 5, 6);
```



Функцию можно вызывать и как часть выражения. Например, вы можете использовать оператор alert для отображения результата вычислений:

```
alert(Average(1, 2, 3 ,4))
```

Использование переменных

В предыдущих главах этой книги вы уже сталкивались с переменными. Вы, наверное, уже сможете использовать переменные без посторонних подсказок. Тем не менее, некоторым особенностям использования переменных вам еще стоит поучиться.

http://ITBookz.ru/

Выбор имени переменой

Переменные — это именованные контейнеры данных (например, чисел, текста или объектов). Как вы уже знаете, каждая переменная имеет свое уникальное имя. Существует несколько неписаных правил определения переменным имен.

- Имена переменных могут содержать все буквы алфавита, как строчные, так и прописные, а также цифры (0—9) и символ подчеркивания.
- Имя переменной должно начинаться с буквы или символа подчеркивания.
- В имена переменных различаются регистры букв. Например, переменные totalnum, Totalnum и TotalNum в JavaScript различаются и могут принимать различные значения.
- Не существует официального ограничения на длину имени переменной, но оно должно располагаться в одной строке кода программы. (Будьте готовы к многократному введению в программе одного и того же длинного имени.)

Следуя приведенным выше правилам, можно для примера привести правильные имена переменных:

```
total number_of_fish
LastInvoiceNumber
tempi
a
_var39
```



При определении имени переменой выбирайте либо логическое, описательное название, либо сокращенное, символическое. Не бойтесь использовать длинные описательные имена переменных. Если, конечно, вы вундеркинд и даже через несколько лет сможете вспомнить, что означают имена переменных а, Ь, х и x1, то можете обойтись и символическими названиями.

Глобальные и локальные переменные

Некоторые языки программирования требуют объявления в начале программы всех используемых в ней переменных. В JavaScript используется ключевое слово var для определения области программы объявления переменных. Во многих случаях переменные просто не объявляются. Объявление переменной проводится только тогда, если ей присваивается определенное значение.

Чтобы понять, в каких случаях необходимо объявлять переменные, вам нужно определить *область действия переменной*. Область действия переменной — это часть сценария, в которой используется переменная. Существует два типа переменных.

- *Глобальные переменные.* Используются во всем сценарии (и других сценариях одного и того же документа HTML). Они могут использоваться и как аргументы разных функции.
- *Локальные переменных.* Используются только как аргументы одной функции. Они применяются только в той функции, в которой были созданы.

Чтобы создать глобальную переменную, объявите ее в главном сценарии, а не в функции. В приведенном ниже примере для объявления переменной используется ключевое слово var:

```
var students=25;
```


http://ITBookz.ru/

Этот оператор объявляет переменную students и определяет ей значение 25. Если этот оператор использовать вне функции, то будет создана глобальная переменная, если же внутри — то локальная. В нашем примере ключевое слово var вводить не обязательно. Следующей оператор равносилен предыдущему:

```
students=25;
```



Если вы не хотите вводить команду var, удостоверьтесь, что это точно можно делать. Хорошей привычкой считается вводить команду var всегда. Таким образом вы сделаете свой сценарий простым в понимании и не вызовете дополнительных ошибок.



В большинстве своем все используемые в этой книге переменные — глобальные.

Локальные переменные используются только в одной функции. Все объявленные вами переменные (или используемые в первый раз) в функции считаются локальными. Например, все параметры функции — это локальные переменные.

Будьте внимательны и обязательно перед введением локальной переменной функции используйте команду var. Это дает указание JavaScript создать локальную переменную, даже если уже существует глобальная переменная с таким же именем.

Чтобы более наглядно объяснить используемые в JavaScript типы данных и методы объявления переменных, рассмотрим листинг 5.6. Это измененный программный код функции Greet(), приведенной в главе "4-й час. Выполнение программ JavaScript".

Листинг 5.6. Сценарий, в котором используются глобальные и локальные переменные

```
<HTML>
<HEAD>
<TITLE>Локальные и глобальные переменные</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var  name1="Фред"
var  name2="Дик"
function Greet(who) {
     alert("Внимание!" + who);
          var  name2="Таня";
}
</SCRIPT>
</HEAD>
<BODY>
<H1>Пример функции</H1>
<P>Сообщение выводится два раза</P>
<SCRIPT LANGUAGE="JavaScript">
Greet(name1)
Greet(name2)
</SCRIPT>
</BODY>
</HTML>
```

В сценарии листинга 5.6 используются следующие переменные.

- name1 и name2 — глобальные переменные, определенные в заголовке;
- who — локальная переменная, созданная в функции Greet().

http://ITBookz.ru/

Если вы внимательно рассмотрели сценарий, то уже заметили, что в функции `Greet()` тоже создается переменная name2. Поскольку при ее задании используется команда var, она не пересекается с глобальной переменной `name2`. (Если бы это происходило, то имя второго пользователя, которому выводится сообщение, изменилось бы.)



Если вы считаете использование двух переменных с одним и тем же именем неправильным, то переименуйте одну из них. В дальнейшем всегда используйте и для глобальных, и для локальных переменных разные имена.

Примите к сведению, что глобальные переменные объявлены в заголовке программы документа HTML. На самом деле переменные позволено объявлять в любом месте программы, но удобнее всего это делать в заголовке, поскольку он выполняется первым. Если использовать переменную перед ее объявлением (или определением ей значения), то ей будет определено нулевое значение.

Я думаю, вы уже поняли разницу между глобальной и локальной переменной. Если вы все еще не уверены в этом — не расстраивайтесь. Всегда используйте при объявлении переменной команду var, тогда вы избежите ошибок.

Определение значений переменным

В главе "2-й час. Создание простых сценариев" для определения переменной значения используется символ равенства. Например, в следующем выражении переменной lines присваивается значение 40:

```
lines = 40;
```

Справа от знака равенства можно вводить самые произвольные выражения, содержащие и переменные. Для добавления переменной единицы используется следующий оператор:

```
lines = lines + 1;
```

Поскольку положительное и отрицательное приращение используется в программировании очень часто, JavaScript содержит два специальных оператора, позволяющих выполнить эти операции. Первый — это оператор положительного приращения +=:

```
lines +=1;
```

Подобно ему вводится и оператор отрицательного приращения:

```
lines -=1;
```

Если вы считаете, что это достаточно сложно, то можете воспользоваться и операторами инкремента и декремента (++ и --). Следующий оператор увеличивает значение переменной на единицу:

```
lines ++;
```

А этот оператор уменьшает значение переменной на единицу:

```
lines --;
```

Операторы ++ и -- можно использовать и перед именем переменной. Например, ++lines. Следует заметить, что эти два выражения не идентичны. Разница заключается в моменте выполнения операции приращения.

- Если оператор стоит после имени переменной, то приращение осуществляется *после* вычисления текущего выражения.
- Если оператор стоит перед именем переменной, то приращение осуществляется *до* вычисления текущего выражения.

http://ITBookz.ru/

Эта разница и определяет использование операторов декремента и инкремента в одном и том же выражении. В качестве примера попытаемся присвоить переменной lines значение 40. Следующие два выражения отображают два разных результата:

```
alert(lines++);
alert(++lines);
```

В первом случае отображается значение 40, а затем значение переменной lines увеличивается на единицу. Во втором же случае сначала значение переменной увеличивается на единицу и поэтому на экране отображается значение 41.



Эти операторы введены только для удобства введения программного кода. Если вы предпочитаете работать с понятными выражениями, то поступайте просто — используйте простые арифметические операторы (+1 и -1) и добьетесь того же результата.

Типы данных в JavaScript

В некоторых языках программирования необходимо при объявлении переменных определять их тип данных. В JavaScript это делать не надо. Тип данных определяется только в некоторых исключительных случаях. Именно поэтому вам стоит познакомиться более детально с типами используемых в JavaScript данных.

- В JavaScript используются следующие основные типы данных.
- *Числовой.* Например, 3, 25 или 1.4142138. В JavaScript используются целочисленные значения и с плавающей точкой.
- *Булев,* или логический. Он принимает только два значения: true (правда) и false (ложь). Обычно он используется для определения выполнения заданного условия.



Детальнее о булевом типе данных и о выполнении условий мы поговорим в главе "7-й час. Тестирование и сравнение значений".

- *Строковый.* Например, "Я прилежно изучаю урок". Он состоит из одного или многих текстовых символов. (Если поступать по правилам, то строковый тип данных назначается строковым объектам, о которых будет рассказано в главе "6-й час. Использование массивов и строковых данных".)
- *Нулевой.* Определяется ключевым словом null. Это значение принимает неопределенная переменная. Например, оператор `document.write(fig)` принимает это значение, если переменная `fig` раньше не определялась.

Хотя JavaScript и сохраняет за переменной тот тип данных, который ей определен, существует вероятность изменения типа данных переменной. Предположим, вы объявили переменную следующим образом:

```
total = 31;
```

Этот оператор объявляет переменную total и назначает ей значение 31. Это значение имеет числовой тип данных. Предположим, вы затем изменяете тип данных переменной:

```
total = "Альбатрос";
```

Теперь переменной total определен строковый тип данных. При выполнении последнего оператора сообщение об ошибке на экран не выводится. Он записан правильно, хотя и существует несогласованность в определении типа данных переменной total.

http://ITBookz.ru/



Хотя в JavaScript позволительно создавать подобные несогласованности в определении переменной типа данных, сделать ошибку все же можно. Например, если переменная total раньше использовалась в математических вычислениях, то результат последнего оператора приведет к конфликту, хотя сообщение об ошибке на экран не выводится.

Преобразование типов данных

JavaScript позволяет во всех возможных случаях проводить преобразование одних типов данных в другие. Пусть вы в программе используете оператор:

```
document.write("Общая сумма: " + total);
```

Если переменная total имеет значение 40, то на экране отобразится следующая строка: Общая сумма: 40. Поскольку функция document.write управляет строковым типом данных, любые нетекстовые значения (в нашем примере total) преобразуются в текстовые. И только после этого результат выводится в окне броузера.

Этот метод вывода данных на экран применяется также для числовых значений с плавающий точкой и булевых переменных. Но существуют некоторые ситуации, в которых он не используется. Например, следующее выражение выполняется правильно в случае, если переменная total имеет значение 40:

```
average = total / 3;
```

Если же переменная total будет иметь строковый тип данных, то выполнение этого оператора приведет к возникновению ошибки.

В некоторых ситуациях сроковый тип данных определен числу и необходимо преобразовать его в числовой. Для этих целей в JavaScript используются две функции.

- parseInt(). Преобразует текстовый тип данных в целочисленный.
- parseFloat(). Преобразует текстовый тип данных в числовой с плавающей точкой.

Обе функции считывают число в виде текста и преобразуют его в числовой тип данных. Например, вам необходимо преобразовать предложение 30 злющих белых медведей в числовое значение:

```
stringvar = "30 злющих белых медведей"
numvar = parseInt(stringvar);
```

После выполнения этих операторов переменная numvar принимает значение 30. Нечисловая часть предложения игнорируется и отбрасывается.



Функции преобразования типов данных ищут числа только в начале строки текста. Если число не найдено, функция возвращает строковое значение NaN, указывая на то, что текст не содержит числовых значений.

http://ITBookz.ru/

Сохранение пользовательских данных в переменных

Одно из наиболее распространенных применений переменных — это сохранение пользовательских данных. В качестве примера давайте создадим сценарий, который запрашивает у пользователя данные и создает документ HTML, содержащий их.

В этом сценарии будет создана настраиваемая начальная страница пользователя. (Конечно, она не будет столь красочная, как вы привыкли наблюдать в Web, но зато работающей.) Для запрашивания у пользователя информации в JavaScript используется функция prompt. Она выполняется обратно функции alert.

В начале сценария у пользователя запрашивается имя, фамилия и название страницы. Это реализуется с помощью следующих операторов:

```
first = prompt("Введите ваше имя.");
second = prompt("Введите вашу фамилию.");
title = prompt("Введите название страницы.");
```

Теперь вы можете приступать к настройке страницы с помощью переменных. Начнем с названия страницы, введенного пользователем:

```
document.write("<H1>" + title + "</H1>");
```

Этот оператор добавляет в документ HTML название страницы, заключенное в дескрипторы <H1> (заголовок первого уровня). Таким же образом добавляется имя пользователя и его фамилия:

```
document.write("<H2>Создано: " + first + " " +second + "</H2>");
```

Чтобы сделать сценарий полным, добавьте дескрипторы <SCRIPT>. Листинг 5.7 содержит полный программный код документа HTML.

Листинг 5.7. Сценарий настраиваемого документа HTML

```
<HTML>
<HEAD>
<TITLE>Настраиваемая страница</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
first = prompt("Введите ваше имя");
second = prompt("Введите вашу фамилию");
title = prompt("Введите название страницы");
document.write("<H1>" + title + "</H1>");
document.write("<H2>Создано: " + first + " " + second + "</H2>");
</SCRIPT>
<P>Страница находится в процессе создания </P>
</BODY>
</HTML>
```

Чтобы протестировать этот сценарий, загрузите документ HTML в броузере. При этом в его окне отобразится запрос на введение сразу трех значений. После их введения отображается ваша начальная страница. Ее вид показан на рис. 5.2.

http://ITBookz.ru/

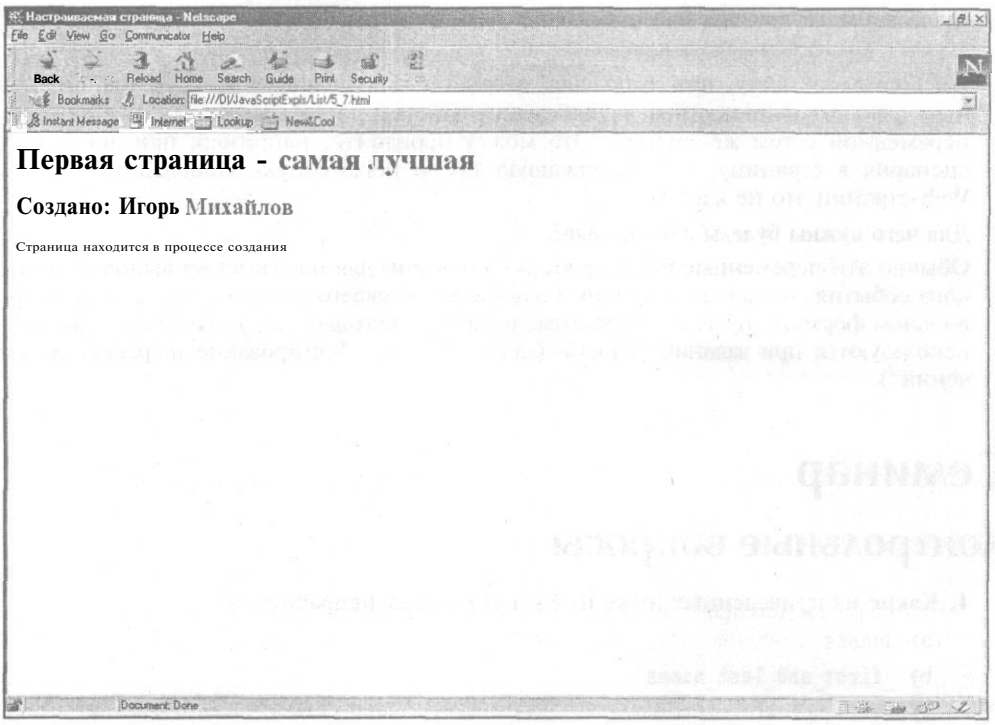


Рис. 5.2. Настраиваемая начальная страница пользователя в Netscape Navigator

Резюме

В течение этого часа вы познакомились с функциями, переменными и методами их управления средствами JavaScript. Вы узнали, как давать переменным названия, объявлять их. Кроме того, вы научились различать глобальные и локальные переменные.

Вы познакомились с используемыми в JavaScript типами данных, а также с функциями преобразования одних типов данных в другие. В следующей главе вы продолжите изучать типы данных. В ней описаны массивы и строковые переменные.

Вопросы и ответы

Netscape отображает на экране сообщение об ошибке missing `semicolon` before statement (перед оператором пропущена точка с запятой). **Что это значит?**

Вы, очевидно, что-то неправильно ввели. JavaScript очень чувствителен к регистру символов, поэтому особое внимание обратите на написание названий переменных и функций. Чаще всего эта ошибка происходит при введении `Function` вместо `function`.

Каково назначение ключевого слова `var`? **Следует ли его всегда использовать при объявлении переменных?**

Команда `var` используется для объявления локальной переменной функции и только в том случае, если глобальной переменной с тем же именем не существует. Если вы сомневаетесь, то добавляйте ее при объявлении любой переменной. Ошибки в этом не будет. Таким образом вы, наоборот, обезопасите сценарий от случайных ошибок.

http://ITBookz.ru/

Существует ли причина при объявлении локальной переменной, сходной с глобальной переменной, использовать ключевое слово var?

По большому счету, нет. Ключевое слово var при объявлении переменной главным образом используется во избежание конфликта при нахождении еще одной переменной с тем же именем. Это может произойти, например, при добавлении сценария в страницу, уже содержащую такую переменную. Небольших, простых Web-страниц это не касается.

Для чего нужны булевы переменные?

Обычно эти переменные используются в сценарии для определения выполнения некого события, например введения пользователем своего телефонного номера в правильном формате. Булевы переменные идеально подходят для этого. Они также часто используются при задании условий (глава "7-й час. Тестирование и сравнение значений").

Семинар

Контрольные вопросы

1. Какие из приведенных ниже имен переменных неправильны?
   a) `2names`
   b) `_first_and_last_names`
   c) `FirtsAndLast`
2. Если выражение var fig=2 приведено в программном коде функции, то какой тип данных имеет переменная?
   a) Глобальная
   b) Локальная
   c) Константа
3. Что будет результатом выполнения выражения 31 + "злобный белый медведь"?
   a) Сообщение об ошибке
   b) 32
   c) 31 злобный белый медведь

Ответы

1, a) 2names — это неправильное имя переменной в JavaScript, поскольку оно начинается с цифры. Остальные имена переменных правильные, хотя вариант b) представляет очень длинное имя.

2, b) Поскольку переменная объявлена в функции, то она локальная. Ключевое слово var подразумевает создание новой локальной переменной.

3, c) При выполнении указанного выражения будет получен результат 31 злобный белый медведь. (На самом деле полярные медведи не такие уж и злобные и не собираются в такие большие группы.)

http://ITBookz.ru/

Упражнения

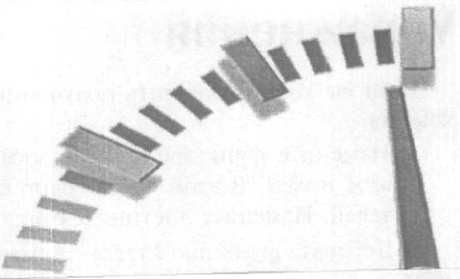
Если вы хотите укрепить полученные в этом уроке знания, выполните следующие задания.

- Измените функцию Greet таким образом, чтобы она содержала два параметра who1 и who2. В этом случае одно сообщение будет выводиться для двух пользователей. Измените листинг 5.4 так, чтобы он использовал эту функцию.
- Добавьте функцию Average в заголовок документа HTML. С помощью функции alert отобразите среднее в окне сообщения. Проверьте правильность полученного результата.
- Удалите в листинге 5.6 ключевое слово var перед объявлением переменной name2 в функции Greet(). Отличается ли содержимое выводимого на экран от первоначального? Если да, то почему.
- Добавьте несколько запросов в листинг 5.2. Например, добавьте запрос на введение телефонного номера или адреса электронной почты.


```
4:     </HEAD>
5:     <BODY>
6:     <H1>Тестовая строка</H1>
7:     <SCRIPT LANGUAGE="JavaScript">;
8:     test1 = "Это тест.  ";
9:     test2 = "Это только тест";
```


их значений на отдельные значения, сохраняемые *о* подстроковы... получения части строковой переменной используется метод substring(), а метод charA применяется для возвращения отдельного ее символа.

http://ITBookz.ru/



```
10:     test = test1 + test2;
11:      alert(test);
12:     </SCRIPT>
13:     </BODY>
14:     </HTML>
```

В этом сценарии текстовые значения определяются двум различным строковым объектам, которые впоследствии объединяются и назначаются еще одному строковому объекту. Если выполнить этот программный код в Netscape Navigator, то на экране отобразится сообщение, подобное показанному на рис. 6.1.

Использование части строковой переменной

Метод `substring()` возвращает часть значения строкового объекта, определенного двумя индексами, указанными в скобках. Для примера приведем оператор, который используется для отображения 4–6 символов значения переменной test:

```
document.write(test.substring(3,6));
```

Вы можете спросить, почему в скобках введены числа 3 и 6. Чтобы понять это, вам необходимо изучить правила, согласно которым определяются индексы в скобках метода `substring()`.

- Индексирование текста начинается с 0. Поэтому четвертый символ будет иметь индекс 3.
- Второй индекс определяется исключительно. Поскольку шестой символ имеет индекс 5, то в скобках указывается индекс 6.
- Оба индекса указываются в произвольном порядке. В нашем примере сначала указан меньший индекс. Вариант `(6,3)` приведет к тому же результату.

Приведем еще один пример. Пусть английскому алфавиту определяется переменная alpha:

```
alpha = "ABCDEFGHIGKLMNOPQRSTUVWXYZ";
```

Следующие операторы демонстрируют использование метода `substring()`:

- `alpha.substring(0,4)` возвращает значение ABCD
- `alpha.substring(10,12)` возвращает значение KL
- `alpha.substring(12,10)` возвращает значение KL. Поскольку 10<12, он используется в качестве начального индекса
- `alpha.substring(6,7)` возвращает значение, G
- `alpha.substring(24,26)` возвращает значение YZ
- `alpha.substring(0,26)` возвращает весь алфавит
- `alpha.substring(6,6)` возвращает нулевое значение или пустую строку. Если индексы, приведенные в скобках, одинаковы, то всегда возвращается нулевое значение.

Возвращение одного символа

Метод `charA()` используется для возвращения отдельного символа значения строкового объекта. Для его определения в скобках следует указать индекс или расположение символа. Индексирование значения строкового объекта начинается с 0. Ниже приведен пример выполнения его для объекта alpha:

- `alpha.charA(0)` возвращает значение A
- `alpha.charA(12)` возвращает значение M
- `alpha.charA(25)` возвращает значение Z
- `alpha.charA(27)` возвращает пустую строку, поскольку символа с таким индексом просто не существует.

Поиск подстроковой переменной

http://ITBookz.ru/

вого объекта и в скобках укажите текст, по которому проводится поиск. В приведенном ниже примере в значении объекта test ищется текст текст:

```
location = test.indexOf("текст");
```



Для большинства объектов JavaScript необходимо точно соблюдать регистр символов слов, по которым проводится поиск. Вводите слово точно так, как оно определено в строковой переменной.

Значение, которое принимает переменная location, — это диапазон индексов символов строки, которые составляют искомое слово. Первый индекс значения строковой переменной традиционно равен 0.

При необходимости позволяется добавлять в качестве условия поиска и начальный индекс, с которого проводится поиск текста. Например, следующее выражение позволяет отыскать слово рыба в значении строкового объекта temp, начиная с 20-го символа:

```
location = temp.indexOf("рыба",19);
```



Предназначение второго параметра метода indexOf() — это поиск нескольких одинаковых текстовых фрагментов в значении строкового объекта. Зная расположение первого текстового фрагмента, вы задаете условия поиска таким образом, чтобы избежать его нахождения и выполнить поиск второго такого же фрагмента.

Второй метод, lastIndexOf(), выполняется подобным образом, но возвращает индекс *последнего* найденного текстового фрагмента в значении строковой переменной. Например, ниже приведен оператор отыскания индекса последнего расположения имени Фред в значении объекта names:

```
location = names.lastIndexOf("Фред");
```

Как и в случае с методом indexOf(), второй параметр используется для задания начального индекса поиска. Но в этом случае поиск проводится в направлении убывания индексов, а не увеличения.

Использование числовых массивов

Массив — это набор элементов, содержащих значения, сохраненный под одним именем. Например, массив score может использоваться для сохранения счета сыгранных матчей. Массивы могут состоять из чисел, строковых переменных, объектов и других типов данных.

Создание числового массива

В отличие от большинства используемых в JavaScript типов данных, массивы необходимо объявлять перед использование. В приведенном ниже примере объявлен массив, состоящий из 30 элементов:

```
scores = new Array(30);
```

Чтобы определить значения массива, укажите его индекс в скобках. Индексирование элементов массива начинается с 0, поэтому элементы объявленного выше массива имеют индексы 0—29. Следующие операторы определяют значения первых четырех элементов массива:

```
scores[0] = 39;
scores[1] = 40;
scores[2] = 100;
scores[3] = 49;
```


http://ITBookz.ru/

Подобно строковым переменным массивы имеют свойство length. Оно определяет количество элементов, из которых состоит массив. Оно же определяется при создании массива. В следующем примере отображается число элементов массива scores:

```
document.write(scores.length);
```

Управление элементами массива

Содержимое массива определяется значениями его элементов. При управлении элементами массива используются те же методы, что и при управлении значениями и переменными. Например, следующий оператор позволяет отобразить значения первых четырех элементов массива scores:

```
scoredisp = "Статистика: " + scores[0] + "," + scores[1] + "," + scores[2] + "," +
scores [3];
document.write(scoredisp);
```



Глядя на последний пример, вы можете подумать, что отображать значения элементов больших массивов очень сложно, особенно содержащих больше сотни элементов. Идеально для управления массивами использовать циклы, позволяющие повторять выполнение одних и тех же операций для разных значений. Детально о циклах будет рассказано в главе "8-й час. Повторение - мать учения: циклы".

Использование строковых массивов

Вот вы и познакомились с массивами чисел. В JavaScript, наряду с числовыми массивами, активно используются и *строковые массивы.* Это позволяет просто и эффективно управлять большими объемами текстовой информации.

Создание строковых массивов

Объявление строковых массивов проводится тем же способом, что и объявление числовых массивов (в JavaScript не существует принципиальной разницы между числовыми и строковыми массивами):

```
names = new Array(30);
```

Следующим образом объявляются его элементы:

```
names[0] = "Виталий";
names[1] = "Ирина";
```

Элементы строкового массива используются везде, где позволено использовать строковые переменные. Все описанные выше методы выполняются и для элементов строковых массивов. Например, следующий оператор позволяет распечатать первые пять символов значения первого элемента массива names:

```
document.write(names[0].substring(0,5));
```

Разделение строковой переменной

JavaScript содержит метод split(), позволяющий разделять строку на составные части. Для того чтобы правильно его использовать, укажите необходимый объект и символ, по которому проводится разделение:

```
test = "Виталий";
parts = test.split("и");
```


http://ITBookz.ru/

В этом примере строковая переменная test принимает значение Виталий. Метод split() разделяет значение на три составные части по символам и. После выполнения этого метода массив определен следующим образом:

- parts[0] = "В"
- parts[1] = "тал"
- parts[2] = "й"

JavaScript обладает еще одним методом управления элементами строкового массива: join(). Он выполняет операцию, обратную методу split(). Следующий оператор объединяет элементы массива parts в одну строку:

```
fullname = parts.join("и");
```

Параметр в скобках определяет символ, по которому проводится объединение. В нашем случае в качестве этого символа выступает буква и. В результате получим исходное имя Виталий. Если символ указывать нет необходимости, обязательно введите в качестве параметра запятую.

Сортировка элементов массива

JavaScript содержит метод sort(), используемый для сортировки элементов массива. Он возвращает упорядоченную версию исходного массива. Упорядочение проводится как по алфавиту (для строковых значений), так и по возрастанию или убыванию (для числовых значений). В следующем примере упорядочен массив, состоящий из четырех английских имен:

```
names[0] = "Public, John Q.";
names[1] = "Tillman, Henry J.";
names[2] = "Clinton, Bill";
names[3] = "Mouse, Micky";
sortednames = names.sort();
```

Последний оператор создает в массиве sortednames упорядоченные по алфавиту элементы массива names.

Отображение бегущих строк

В главе "3-й час. Возможности JavaScript" вы узнали, что JavaScript используется для создания бегущих сообщений, отображаемых в строке состояния окна броузера. В нем приведен также полный листинг программного кода сценария, позволяющего это сделать. Зная методы управления строковыми переменными, вам ничего не строит написать подобную программу самостоятельно.

Для начала давайте определим сообщение, которое будет отображаться в бегущей строке. Для сохранения текста сообщения используем строковую переменную msg. В самом начале сценария определите значение переменной (укажите ваше собственное значение, а не копируйте мое):

```
msg = "Это пример бегущего сообщения. Впечатляет?"
```

Далее определите следующую строковую переменную spacer. Ее значение будет отображаться между копиями значений переменной msg:

```
spacer = "...                                • • .";
```


http://ITBookz.ru/

Кроме того, вам понадобится еще одна переменная — числовая переменная, имеющая числовое значение места расположения строки. Назовите ее pos и определите начальное значение 0.

Создание бегущего сообщения проводится с помощью функции ScrollMessage(). Листинг 6.2 содержит ее программный код.

Листинг 6.2. Код функции ScrollMessage()

```
1:          function ScrollMessage( ){
2:              window. status =msg. substring (pos, msg. length)
            + spacer + msg.substring(0,pos) ;
3:                pos++;
4:          if (pos > msg. length) pos = 0;
5:          window. setTimeout("ScrollMessage( )",200);
6:          }
```

Ниже приведено построчное описание программного кода функции.

- Строка 1 содержит название функции, которое определяет выполняемую ею операцию.
- Строка 2 содержит текст, который будет отображаться в бегущем сообщении строки состояния. Сообщение состоит из части значения строковой переменной msg (определенной от индекса pos до конца строки), пробела и части значения строковой переменной msg (определенной с начала строки до индекса pos).
- Строка 3 содержит оператор увеличения значения переменной pos на единицу.
- Строка 4 содержит оператор проверки отношения значения переменной pos к длине строки msg. Если оно больше, то переменная pos принимает новое значение, равное 0. (Детально о операторе if вы узнаете в следующей главе.)
- Строка 5 содержит метод window. setTimeout, определяющий время задержки, через которое будет выполняться оператор. В нашем случае функция ScrollMessage выполняется с задержкой в 2 секунды.
- Строка 6 содержит закрывающую скобку, символизирующую конец программного кода функции.

Чтобы сделать пример работающим, дополните его дескрипторами <SCRIPT> и необходимыми дескрипторами HTML, позволяющими создать полноценную Web-страницу. Листинг 6.3 содержит программный код Web-страницы с бегущим сообщением в строке состояния окна броузера.

Листинг 6.3. Пример документа HTML с бегущим сообщением в строке состояния

```
1:          <HTML>
2:          <HEAD><TITLE>Пример бегущей строки</TITLE>
3:          <SCRIPT LANGUAGE="JavaScript">
4:          var msg = "Это пример бегущей строки.    Впечатляет?";
5:          spacer = ". . .                  . . .";
6:          pos = 0;
7:          function ScrollMessage ( ){window.status =
8:              msg. substring (pos, msg. length) + spacer +
            msg.substring(0,pos);
9:                pos++;
10;         if (pos > msg. length) pos = 0;
```


http://ITBookz.ru/

```
11:         window.setTimeout("ScrollMessage()",200);
12:         }
13:         ScrollMessage();
14:         </SCRIPT>
15:         </HEAD>
16:         <BODY>
17:         <H1>Пример бегущей строки</H1>
18:         Взгляните на строку состояния в нижней частиокна броузера (не смотрите
                долго - это завораживает).
19:         </BODY></HTML>
```

На рис. 6.2 показан результат выполнения программы листинга 6.3.

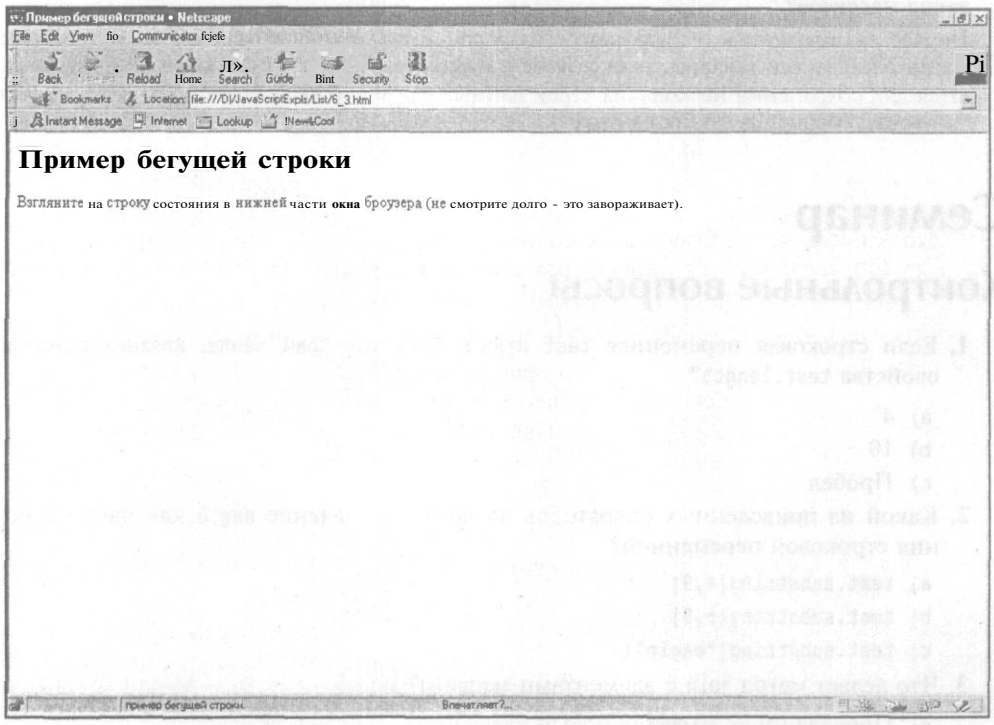


Рис. 6.2. Вот она, бегущая строка

Резюме

В этом уроке вы узнали о сохранении строковых значений в JavaScript. Вы познакомились с методами разделения текстового значения и определения своих переменных отдельным частям. Кроме того, вы научились управлять массивами переменных и сохранять в них числа и текст.

Вы применили свои знания о строковых переменных для создания бегущего сообщения. В следующем уроке вы узнаете об операторе if и методах сравнения в JavaScript значений переменных и массивов.

http://ITBookz.ru/

Вопросы и ответы

Можно ли сохранять в виде массива любые типы данных? Могу ли я, например, создать массив знаменательных дат моей жизни?

Конечно. В виде массива в JavaScript сохраняются любые используемые типы данных.

Используются ли в JavaScript двумерные массивы?

Двумерные массивы имеют два индекса (строки и столбца). JavaScript напрямую не поддерживает подобные типы массивов, но позволяет использовать объекты, заменяющие их функции. Детально с ними вы познакомитесь в главе "11-й час. Создание пользовательских объектов".

Я вручную определил элементам массива значения. В чем же преимущество использования массивов?

Первое преимущество — это автоматическое выполнение подобных операций надо всеми элементами массива, проводимое с помощью циклов. Массивы также используются для сохранения нескольких строк данных формы. (Об этом мы поговорим в главе "14-й час. Формы введения данных").

Семинар

Контрольные вопросы

1. ЕСЛИ строковая переменная test имеет значение Край земли, каково значение свойства `test.length`?
   a) 4
   b) 10
   c) Пробел
2. Какой из приведенных операторов возвращает значение eagle как часть значения строковой переменной?
   a) `test.substring(4,9)`
   b) `test.substring(5,9)`
   c) `test.substring("eagle")`
3. Что делает метод join с элементами массива?
   a) Присоединяет в массив новый элемент
   b) Объединяет массив с другим массивом
   c) Объединяет значения элементов массива в одно значение.

Ответы

1, b) Длина строкового значения составляет 10 символов

2, a) Правильный оператор `test.substring(4,9)`. Поскольку индексирование текста начинается с нуля, второй вариант не насчитывает необходимого количества символов

3, c) Метод join используется для объединения значений элементов одного массива

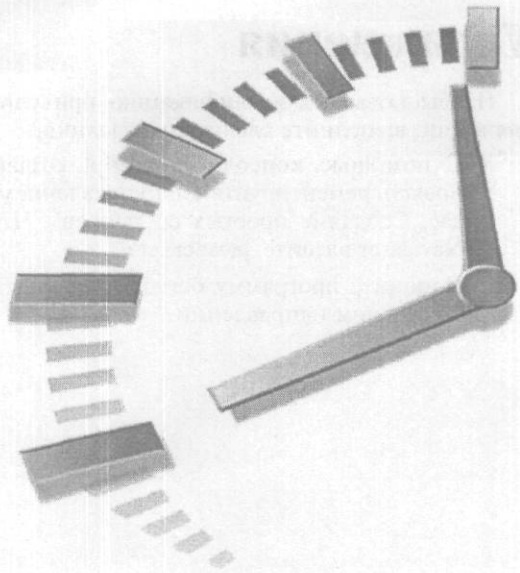
http://ITBookz.ru/

Упражнения

Чтобы повысить квалификацию при управлении массивами и строковыми переменными, выполните следующие задания.

- С помощью консоли JavaScript создайте несколько строковых переменных и поэкспериментируйте с их управлением. Консоль JavaScript описана в главе "2-й час. Создание простых сценариев". Чтобы запустить ее в поле адреса Netscape Navigator введите `javascript:`.
- Измените профамму бегущего сообщения таким образом, чтобы оно двигалось в обратном направлении.

http://ITBookz.ru/



7-й час

Тестирование и сравнение значений

Прилежно изучив две предыдущие главы, вы сумеете создать переменные самых разных типов. В этой главе вы узнаете еще о нескольких применениях переменных – их сравнении, проверке и оценке.

В течение этого часа вы познакомитесь с дополнительными операторами JavaScript. Здесь рассмотрены следующие темы.

- **Проверка переменных с помощью оператора** if
- **Использование операторов для сравнения переменных**
- **Использование операторов сравнения для задания условий**
- **Добавление альтернативного условия с помощью оператора** else
- **Создание выражений с операторами сравнения**
- **Проверка нескольких условий**
- **Проверка правильности запрошенных у пользователя данных**

http://ITBookz.ru/

Оператор if

Одно из основных преимуществ всех языков программирования — это возможность проверки и сравнения значений. Это позволяет программисту задавать разные сценарии поведения программы в зависимости от введенных пользователем данных.

Оператор if чаще других используется в JavaScript для сравнения данных. Этот оператор используется практически во всех известных языках программирования. Его конструкция позаимствована из грамматики английского языка. Например, давайте рассмотрим простое английское предложение.

If the phone rings, answer it. (Если телефон зазвонит, поднимите трубку.)

Это предложение состоит из двух частей: условия *If the phone rings (Если телефон зазвонит)* и действия *answer it (поднимите трубку)*. Оператор if в JavaScript используется подобным образом. Вот пример стандартного оператора с командой if:

```
if (a == 1} window.alert("1 найдено!");
```

В этом операторе вначале также задается условие (если а равно единице), а затем действие (отобразить сообщение). При выполнении оператора проверяется значение переменной. Если оно равно единице, то печатается указанное сообщение. В противном случае сообщение не выводится.

В предыдущем примере в случае правильности условия выполняется только одно действие. При необходимости можно также при выполнении условия задавать несколько операторов, заключив их в фигурные скобки {}. Это и продемонстрировано в листинге 7.1.

Листинг 7.1. Оператор if, используемый с несколькими операторами действия

```
1:    if (a == 1) {
2:          window.alert("1 найдено!");
3:          a = 0;
4;    }
```

Этот блок операторов также проводит проверку значения переменной а. Если оно равно 1, то отображается сообщение и переменной а определяется новое значение 0.

Условные операторы

ЕСЛИ вторая часть оператора if, в которой указываются действия в случае выполнения условия, может содержать только уже известные вам операторы, то первая его часть, в большинстве случаев, использует неизвестный вам синтаксис. В ней вводятся *выражения условия*.

Выражения условия содержат два сравниваемых значения (в предыдущем примере это переменная а и число 1). В качестве значения может выступать переменная, константа или целое выражение.



Любая часть выражения сравнения может содержать переменную, константу или выражение. Вы можете сравнивать переменную с константой или две переменные. (Можно, конечно, сравнивать и две константы, но в этом нет смысла.)

Между двумя сравниваемыми операторами вводится *условный оператор*. Этот оператор задает условие, которому должны удовлетворить оба значения. Ниже приведены все используемые в JavaScript условные операторы.

- == (равно)
- != (не равно)
- < (меньше)
- > (больше)
- <= (меньше или равно)
- => (больше или равно)

http://ITBookz.ru/



Будьте внимательны и не путайте оператор сравнения (==) с оператором определения значения (=). Хотя оба оператора обозначены символом равенства, первый используется для *сравнения* двух значений, а второй — для *определения* значения переменной. Это одна из самых распространенных ошибок при программировании на JavaScript.

Совместное использование логических и условных операторов

Во многих случаях необходимо одновременно проверить одну переменную на выполнение нескольких условий или несколько переменных. Для этого в JavaScript имеются *логические, или булевы, операторы.* В приведенном ниже примере проверяются разные условия и выполняется одно и то же действие:

```
if (phone == " ") window.alert("Ошибка!");
if (email == " ") window.alert("Ошибка!");
```

С помощью логического оператора этих два оператора объединяются в один:

```
if (phone == " " !! email == " ") window.alert("Ошибка!");
```

В этом выражении использован логический оператор ИЛИ (||). На обычном языке это выражение можно определить как: "Если телефонный номер или адрес электронной почты содержит пробелы, то на экране отображается сообщение об ошибке".

Еще один логический оператор, с которым вы будете часто сталкиваться — это оператор И (&&). Изучите следующее выражение:

```
if (phone == " " && email == " ") window.alert("Ошибка!");
```

В этом выражении вместо оператора || использован оператор &&. В этом случае сообщение об ошибке будет отображаться на экране только в том случае, если выполняются *оба* условия (т.е. и телефонный номер, и адрес электронной почты содержит пробелы). По правде говоря, оператор ИЛИ использовать в данном случае предпочтительнее.



В случае использования оператора && сначала проверяется первое условие. Если оно не выполняется, то второе условие проверяться не будет. В этом нет надобности. Используйте эту особенность логических операторов для повышения производительности.

Третий логический оператор, с которым вам необходимо познакомиться, — это оператор НЕТ (!). Он используется для инвертирования выражения, другими словами, в случае невыполняемости условия будут выполняться указанные оператор(ы) действия. Ниже приведен пример оператора НЕТ:

```
if (phone != " ") window.alert("Правильно!");
```

В этом случае оператор ! используется как часть оператора сравнения (!=). Этот оператор инвертирует условие. Поэтому, если телефонный номер *не* содержит пробелов, то выводится сообщение о правильности его введения.



Логические операторы используются очень часто. Именно поэтому так просто сделать в них ошибку и неправильно задать условие. Например, условие a < 10 && a >20 на первый взгляд правильное. Но если вы его проанализируете, то получите следующее: "если a меньше 10 и больше 20". Одновременно оба эти условия выполняться не могут. Поэтому необходимо вместо оператора && использовать оператор !||.

http://ITBookz.ru/

Оператор else

Дополнительный оператор, используемый вместе с оператором if, — это ключевое слово else. Как и в обычном языке (else в переводе с английского означает иначе), этот оператор определяет действия, которые выполняются в случае не выполнения условия. В листинге 7.2 приведен пример использования ключевого слова else.

Листинг 7.2. Пример совместного использования операторов if и else

```
1:    if (a == 1) {
2:         alert("1 найдено");
3:         a = 0;
4:    }
5:    else {
6:        alert("Неправильное значение: " = a);
1:    }
```

Этот сценарий отображает сообщение и назначает переменной a новое значение 0. Если же условие не выполняется (если a не равно 1), то отображается другое сообщение.



Подобно ключевому слову if, ключевое слово else может содержать и один оператор действия. В этом случае фигурные скобки использовать нет необходимости.

Использование условных выражений

В дополнение к оператору if язык JavaScript оснащен дополнительной возможностью построения условных выражений. Они имеют немного странный на первый взгляд синтаксис, используемый и в других языках программирования (например, в C). Условное выражение имеет следующую конструкцию:

```
Переменная = (условие) ? если выполняется : если не выполняется;
```

Это выражение позволяет определить переменной одно из двух значений. Одно в случае выполнения условия, второе — при его невыполняемости. Приведем пример его использования:

```
value = (a == 1 ) ? 1 : 0;
```

Для тех, кто запутался в сложной структуре оператора, приведем его аналог с помощью ключевого слова if:

```
if (a == 1)
  value = 1;
else
  value = 0;
```

Другими словами, значение после вопросительного знака будет определяться переменной в том случае, если условие выполняется. Если условие не выполняется, то переменной определяется значение, введенное после двоеточия. Двоеточие в этом выражении играет роль оператора else, который используется не во всех случаях.

Это сокращенное выражение предпочтительней использовать вместо условных операторов при определении значения переменной. Ниже приведен пример отображения значения переменной counter:

```
document.write("Найдено" + counter + (counter == 1) ? " слово." : "слова."));
```

Если переменная counter имеет значение 1, то на экран выводится сообщение Найдено 1 слово. Если counter имеет значение 2 и больше, то на экран выводится сообщение Найдено 2 слова.

http://ITBookz.ru/

Задание нескольких условий

Как правило, для задания комплексного условия используется многострочная конструкция из нескольких операторов if. Листинг 7.3 содержит пример подобной структуры.

Листинг 7.3. Задание нескольких условий с помощью операторов if

```
1:    if (button=="следующая страница") window.location=next.htm;
2:    if (button=="предыдущая страница") window.location=prev.htm;
3:    if (button=="начальная страница") window.location=home.htm;
4:    if (button=="назад") window.location=back.htm;
```

Хотя это и распространенный метод задания условий, при большом их количестве он становится обременительным. Чтобы избавить вас от многократного введения операторов if, в JavaScript добавлен оператор switch. Он один позволяет задавать целый блок условий, приводящих к выполнению самых разных действий. В листинге 7.4 приведен программный код оператора switch, выполняющий те же действия, что и код листинга 7.3.



Оператор switch впервые использовался в JavaScript 1.2. Поэтому, перед тем как использовать его, удостоверьтесь, что ваш броузер поддерживает его. Чтобы избежать возможных ошибок, также указывайте версию JavaScript: <SCRIPT LANGUAGE="JavaScript 1.2">.

Листинг 7.4. Задание нескольких условий с помощью оператора switch

```
1:        switch(button) {
2:            case "следующая страница" :
3:                 window.location="next.html";
4:                 break;
5:             case "предыдущая страница" :
6:                 window.location="prev.html";
7:                 break;
9:            case "начальная страница" :
10:                window.location="home.html";
11:                break;
12:            case "назад" :
13:                window.location="back.html";
14:                break;
15:             default :
16:                window.alert("Не та кнопка.");
```

Оператор switch состоит из нескольких основных элементов.

- Начальный оператор switch. Этот оператор определяет сравниваемое значение (в приведенном выше примере button), которое вводится в скобках.
- Фигурные скобки {}. Выполняют те же функции, что и скобки в операторе if.
- Один или несколько операторов case. Каждая из строк, которая начинается этим оператором, содержит значение, сравниваемое с исходным. Если значения совпадают, выполняется оператор, указанный после ключевого слова case. В противном случае сравнивается следующее значение case.
- Ключевое слово break, используемое для определения конца действия текущего оператора case.

http://ITBookz.ru/

- Необязательный оператор default. Выполняется по умолчанию, если ни один из операторов case не содержит правильного значения.



После каждого оператора case можно использовать несколько операторов действий. Заключать их в скобки нет необходимости. Если условие текущего оператора case справедливо, то выполняются все операторы, введенные до ключевого слова break.

Проверка введенных данных

В качестве примера использования изученного только что оператора switch приведем программу Web-страницы, которая, в зависимости от введенного пользователем значения, выполняет разные действия. Пусть у пользователя запрашивается ключевое слово, определяющее Web-страницу.

Если ключевое слово совпадает с одним из введенных в сценарии, то в окне броузера будет отображена указанная страница. Если ключевое слово не совпадает с содержащимися в сценарии, в окне броузера отобразится страница по умолчанию.

Чтобы запросить у пользователя необходимое значение используется функция prompt(). Укажите в качества параметра этой функции вопрос, отображаемый для пользователя. Вот как приблизительно это должно выглядеть:

```
where = prompt("Куда заглянем сегодня?");
```

Далее необходимо ввести оператор switch, определяющий различные варианты перехода:

```
switch (where) {
  case "Netscape" :
       window.location="http://www.netscape.com";
       break;
  case "Microsoft" :
       window.location="http://www.microsoft.com";
       break;
  case "Yahoo" :
       window.location="http://www.yahoo.com";
       break;
```

Добавьте также оператор default, позволяющий отобразить в окне пользователя страницу по умолчанию:

```
  default :
       window.location="http://www.mcp.com";
}
```

Поскольку это последний оператор в структуре switch, нет необходимости использовать после него ключевое слово break. Скобка закрывает код оператора switch.

Листинг 7.5 содержит полный программный код документа HTML. Чтобы протестировать Web-страницу, загрузите ее в броузере. На экране появится запрос, подобный изображенному на рис. 7.1. Введите одно из ключевых слов и проследите, правильная ли страница загружается в окне броузера. Если указать неправильное ключевое слово, будет загружена Web-страница по умолчанию.

http://ITBookz.ru/

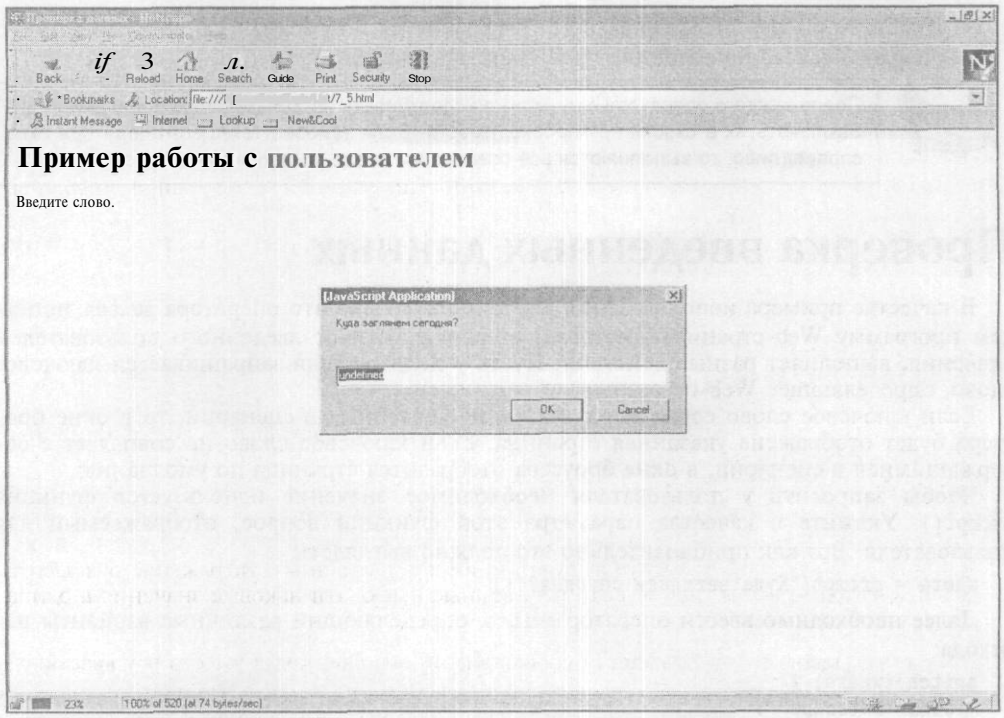


Рис. 7.1. ***Введите страницу, которую хотите посетить***

Листинг 7.5. Сценарий проверки введенных данных

```
<HTML>
<HEAD><TITLE>Проверка данных</TITLE>
</HEAD>
<BODY>
<H1>Пример работы с пользователем</H1>
Введите слово.<BR>
<SCRIPT LANGUAGE="JavaScript 1.2">
where = window.prompt("Куда заглянем сегодня?");
switch (where) {
     case "Netscape" :
window.location="http://www.netscape.com";
 break;
     case "Microsoft" :
window.location="http://www.microsoft.com";
 break;
     case "Yahoo" :
 window.location="http://www.yahoo.com";
 break;
     default :
 window.location="http://www.mcp.com";
 }
 </SCRIPT>
 </BODY>
 </HTML>
```


http://ITBookz.ru/

Резюме

В этой главе вы узнали об использовании в JavaScript операторов if и else. Вы также познакомились с быстрым методом задания нескольких условий и узнали о способе определения для переменной разных значений с помощью оператора switch. Вы применили полученные знания для создания документа HTML, запрашивающего у пользователя информацию.

В следующей главе речь пойдет о еще одном фундаментальном методе обработки информации — циклах. Они используются в JavaScript для автоматического выполнения повторяющихся действий.

Вопросы и ответы

Что произойдет, если попытаться сравнить два значения, которые относятся к разным типам данных (например, число и текст)?

JavaScript позволяет сравнивать только однотипные данные. Поэтому в этом случае число будет преобразовано в текст, который и будет сравниваться впоследствии с другим текстом. В JavaScript 1.3 и выше используется специальный оператор ===, позволяющий сравнивать не только значения, но и их типы данных. Выражение будет справедливым только в том случае, если переменные имеют одинаковые значения *и* одинаковые типы данных.

Почему на экране не отображается сообщение об ошибке, когда вместо == ввести =?

В некоторых случаях сообщение об ошибке все же выводится. Например, в операторе if (a = 1) переменной a определяется значение 1. Условие оператора выполняется и значение переменной запрашивается.

Почему в сценарии листинга 7.5 определена версия JavaScript 1.2?

Оператор switch впервые был введен в JavaScript 1.2. Определив эту версию языка, вы автоматически устраняете возможные проблемы несовместимости.

Семинар

Контрольные вопросы

1. Что делает оператор if `(fig==1)`?
   a) Определяет переменной `fig` значение 1
   b) Отображает сообщение об ошибке, поскольку названия фруктов в качестве имен переменных использоваться не могут
   c) Проверяет, равна ли переменная единице, и выполняет в случае совпадения необходимые действия
2. Какой из указанных операторов означает "Не равно"?
   a) !
   b) !=
   c) <>
3. Что делает оператор switch?
   a) Определяет значение переменной в наборе данных
   b) Присоединяет и отключает переменную
   c) Делает оператор if длиннее и сложнее

http://ITBookz.ru/

Ответы

1, c) Этот оператор проверяет, равна ли переменная a единице, и выполняет необходимые действия

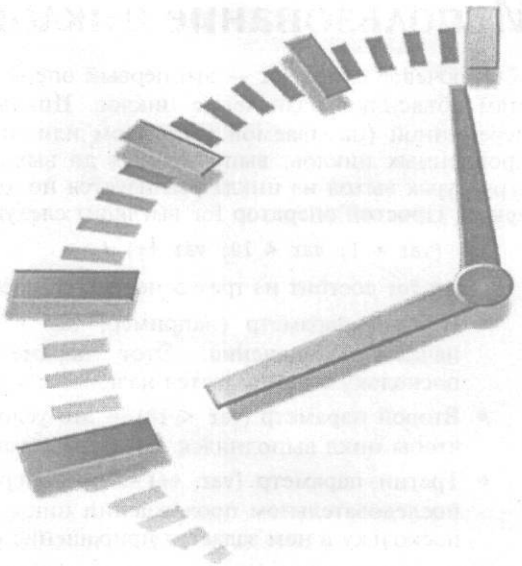
2, b) Оператор != означает *не равно*

3, a) Оператор switch определяет значение переменной в наборе данных и выполняет в каждом случае разные действия

Упражнения

Если вы хотите повысить свое умение использовать оператор if в разных ситуациях, выполните следующее упражнение. Вы, наверное, уже заметили, что в листинге 7.5 текстовые значения необходимо вводить символами строго определенного регистра. Если, например, вы введете netscape вместо Netscape, то не отобразите страницу www.netscape.com. Измените программный код таким образом, чтобы он позволял вводить слова, независимо от регистра символов. (Используйте метод `toLowerCase()`, описанный в главе "6-й час. Использование массивов и строковых данных", а затем сравните преобразованное значение со сравниваемым значением, введенным строчными символами.)

http://ITBookz.ru/



8-й час

Повторение - мать учения: циклы

Этой главой заканчивается часть II книги. Мы подошли к одному из самых главных средств автоматической обработки информации в JavaScript. С помощью циклов вы сможете создавать программы, выполняющие повторяющиеся операции.

В этой главе рассмотрены следующие темы.

- **Повторное выполнение операторов в цикле** `for`
- **Использование оператора** while **для задания цикла**
- **Применение циклов** do **и** while `for`
- **Создание бесконечных циклов и их назначение**
- **Выход из цикла**
- **Использование цикла** `for...in` **для обработки массивов**

http://ITBookz.ru/

Использование циклов

Ключевое слово for — это первый оператор, на котором начинающим программистам объясняют назначение циклов. Циклы for требуют использования временной переменной (называемой счетчиком или индексом), которая определяет количество пройденных циклов, выполненных до выхода из повторяющейся структуры. В этих структурах выход из цикла реализуется по достижении счетчиком определенного значения. Простой оператор for выглядит следующим образом:

```
for (var = 1; var < 10; var ++) {
```

Цикл for состоит из трех основных элементов.

- Первый параметр (например, var = 1) определяет счетчик и указывает его начальное значение. Этот параметр называется *начальным выражением,* поскольку в нем задается начальное значение счетчика.
- Второй параметр (var < 10) — это условие, которое должно быть справедливым, чтобы цикл выполнялся. Он называется *условием* цикла.
- Третий параметр (var ++) — это оператор, который выполняется при каждом последовательном прохождении цикла. Он называется *выражением инкремента,* поскольку в нем задается приращение счетчика.

После определения всех параметров цикла вводится открывающая фигурная скобка, символизирующая начало тела цикла. Закрывающая фигурная скобка вводится в конце тела цикла. Все операторы, введенные в скобках, выполняются при каждом прохождении цикла.

Как и в случае оператора if, если вы в теле цикла используете только один оператор, добавлять фигурные скобки не нужно.

Вам это покажется простой хвальбой, но, применив в программе один раз цикл for, вы не сможете отказаться от его использования. Простой пример использования цикла for приведен в листинге 8.1.

Листинг 8.1. Пример использования цикла for

```
1:    for (i=1; i<10; i++) {
2:         document.write("Это строка № ",i,"<BR>");
3:     }
```

Этот пример использования цикла для отображения сообщения при каждом прохождении цикла. Результат выполнения цикла показан ниже:

```
Это строка № 1
Это строка № 2
Это строка № 3
Это строка № 4
Это строка № 5
Это строка № 6
Это строка № 7
Это строка № 8
Это строка № 9
```

Заметьте, что цикл выполняется только девять раз. Это происходит потому, что использовано условие i<10. Когда счетчик принимает значение 10, условие становится ложным. Если необходимо выполнить цикл десять раз, измените это условие на i<=10 или i<11.

http://ITBookz.ru/



Вам придется привыкнуть к тому, что переменная i чаще всего используется как счетчик в циклах. Это одна из традиций программирования, которая своими корнями уходит в древнюю эпоху ископаемых языков программирования, таких как Forth. Вам нет необходимости придерживать традиции, но использовать все время одну переменную в качестве счетчика цикла предпочтительнее. (Детально о языке Forth вы можете узнать, посетив узел организации *Forth Interest Group* по адресу www.forth.org.)

Структура цикла for в JavaScript позаимствована из Java, который, в свою очередь, имеет много общего с языком C. Хотя в этих циклах и принято традиционно использовать счетчик, изменяющий свое значение, в качестве условия выполнения цикла можно задавать любую комбинацию логических и условных операторов. Циклы, в которых не используются счетчики, лучше задавать оператором while, описанным в следующем разделе.

Использование циклов while

Еще одно ключевое слово, с помощью которого в JavaScript организуются циклы, — это while. В отличие от циклов for, цикл while не требует использования счетчиков. Наоборот, они выполняются до тех пор, *пока* выполняется указанное условие. Тем не менее, если условие не выполняется вообще, то цикл выполняться тоже не будет.

Оператор while содержит в скобках все необходимые параметры. В фигурных скобках традиционно указываются все выполняемые в случае истинности условия операторы. Листинг 8.2 содержит пример использования цикла while.

Листинг 8.2. Простой цикл while

```
1:    while(total < 10) {
2:    n++;
3:    total  +=values[n];
4:    }
```

В этом цикле также используется счетчик, определяющий номера элементов массива. Вместо того чтобы прекратиться при достижении счетчика определенного значения, цикл прерывается, когда сумма значений массива становится больше 10.

Если немного подумать, то можно создать подобный цикл и с помощью оператора for:

```
for (n=0; total<10; n++) {
total+= values[n];
}
```

На самом деле цикл for — это не что иное, как специальный тип циклов while, который автоматически выполняет инициализацию и приращение переменной счетчика. Все циклы for реализуются и с помощью циклов while. Выбор типа цикла лежит полностью на вас. Он в первую очередь определяется поставленной перед вами задачей и ... вашей предрасположенностью к одному из них.

http://ITBookz.ru/

Использование цикла do...while

JavaScript 1.2 содержит еще один тип циклов: do..while. Этот тип циклов сильно похож на своего родителя — цикл while. Единственная разница заключается в расположении условия. Условие в цикле do...while приводится в конце кода. Листинг 8.3 содержит код типичного цикла do...while.

Листинг 8.3. Пример цикла do...while

```
1:    do {
2:    n++;
3:    total += values[n];
4:    }
5:    while(total < 10);
```

Вы, наверное, уже заметили, что этот цикл выполняет те же действия, что и цикл листинга 8.2. Но существует маленькое отличие в выполняемых ими операциях. В этом программном коде условие проверяется в конце листинга, а в программном коде листинга 8.2 — в начале. Это означает, что этот цикл будет выполнен по меньшей мере один раз, чего не скажешь о его прародителе.



Как и в циклах for и while, цикл do...while не требует добавления фигурных скобок к телу цикла, если в нем введен всего один оператор.

Управление циклами

Хотя использовать циклы для выполнения простых операций не так уж и сложно, при создании громоздких повторяющихся структур лучше придерживать определенных неписаных правил. В следующих разделах вы познакомитесь с операторами break и continue, позволяющими упростить управление циклами.

Создание бесконечного цикла

Циклы for и while позволяют достаточно детально управлять выполнением операций. В некоторых случаях, при неосторожности и невнимательности, это порождает сложные проблемы. Взгляните на листинг 8.4.

Листинг 8.4. Пример бесконечного цикла

```
1:    while(j<10) {
2:    n++;
3:    values[n] = 0;
4:    }
```

В этом примере умышленно сделана одна ошибка. Условие выполнения цикла определяется переменной j. Но, в то же время, эта переменная не изменяется и не влияет на условие. Таким образом получен бесконечный цикл. Выполнение этого цикла может прервать только пользователь или возникновение ошибки.

Бесконечные циклы прерываются пользователем только при закрытии броузера. Некоторые бесконечные циклы не позволяют закрывать броузер или вызвать ошибку.

http://ITBookz.ru/

Как правило, появление бесконечных циклов можно предотвратить. Их тяжело обнаружить, поскольку JavaScript при их возникновении не выводит на экран сообщения об ошибке. Поэтому при создании любого цикла в сценарии будьте внимательны и проверяйте его тщательнее, не позволяя стать бесконечным.



В зависимости от типа и версии броузера бесконечный цикл может вызвать "зависание" броузера, когда пользователь не может им управлять. Всегда предусматривайте в циклах процедуру выхода на случай их преобразования в бесконечные периодично повторяющиеся структуры.

Иногда бесконечный цикл необходимо создать умышленно. Это вызвано необходимостью постоянного выполнения программы до прерывания ее пользователем. Процедура выхода из бесконечного цикла также предусматривает использование оператора break, описанного в следующем разделе. Вот как создается бесконечный цикл:

```
while (true) {
```

Поскольку значение true — это не условие, JavaScript будет бесконечно продолжать искать условие выхода из цикла.

Прерывание цикла

Существует один надежный способ выхода из цикла. В тело цикла после операторов действия необходимо добавить оператор break. Листинг 8.5 иллюстрирует пример использования оператора break.

Листинг 8.5. Прерывание цикла оператором break

```
1:    while (true) {
2:    n++;
3:    if (values[n] == 1) break;
4:    {
```

Оператор while задает бесконечный цикл. Оператор if проверяет значения элементов массива. Если среди значений находится единица, то выполнение цикла прерывается.

При обнаружении оператора break интерпретатор JavaScript упускает выполнение остальных операторов и переходит к выполнению оператора, следующего первым после тела цикла. Оператор break используется в любом типе циклов, как в бесконечном, так и в конечном. Это позволяет создавать процедуры выхода из программы в случае возникновения ошибок или нахождения необходимых данных.

Продолжение выполнения цикла

Повысить управляемость циклами позволяет еще один оператор. continue — это оператор, который позволяет прервать выполнение операций текущей итерации цикла и продолжить их выполнение со следующей итерации. В листинге 8.6 приведен пример использования этого оператора.

Листинг 8.6. Прерывание выполнения операций цикла

```
1:    for (i=1;i<21; i++) {
2:    if (score[i]==0) continue;
3:    document.write ("Номер студента ",i, "оценка: ", score[i], "\n");
4:    }
```


http://ITBookz.ru/

В этом листинге приведен пример цикла for, используемого для вывода оценок 20 студентов, данные о которых сохранены в массиве score. Оператор if используется для сравнения значения оценки с нулем. Предполагается, что оценка 0 определяет студента, который не сдавал тест. В этом случае выполнение цикла продолжается, но результат отсутствующего на тесте студента не распечатывается.

Использование цикла for...in

Мы переходим к рассмотрению последнего типа циклов в JavaScript. Цикл for...in более гибкий, нежели привычные циклы for и while. Он специально разработан для выполнения операций со свойствами объектов.

Например, объект navigator имеет свойства, описывающие параметры броузера (глава "16-й час. Создание сценариев для разных броузеров"). Для отображения свойств объекта также проще всего использовать цикл for...in:

```
for (i in navigator) {
document.write("Свойство: ", + i);
document.write("Значение: ", + navigator[i]);
}
```

Как и обычный цикл for этот тип цикла требует использования индекса (в нашем примере это i). Каждая итерация цикла приводит к определению нового значения переменной-индексу, соответствующего другому свойству объекта. Таким образом очень удобно управлять свойствами объектов.



Этот тип циклов удобно применять для обработки элементов массивов, хотя основное его предназначение — это управление свойствами объектов. Детально о создании собственных объектов и их свойств вы узнаете в главе "11-й час. Создание пользовательских объектов".

Управление массивами

Чтобы окончательно понять, для чего используются циклы, давайте вместе создадим сценарий, который управляет массивом значений. (По мере создания сценария вы поймете, как сложно было бы создать его, не используя циклы.)

Создадим простой цикл, который запрашивает имена пользователей. После введения всех имен пользователей отобразим их в виде нумерованного списка. В самом начале сценария объявим некоторые переменные:

```
names = new Array();
i=0;
```

Массив имен содержит имена пользователей, введенные вами на запрос сценария. Поскольку вы не знаете, как много имен он будет содержать, не определяйте его размер. Переменная i используется в качестве счетчика цикла.

Для создания запроса на введение имен используйте оператор prompt. Чтобы автоматизировать процедуру ввода, используйте все тот же цикл. Если в массив имен будет заноситься по меньшей мере одно имя, то используйте цикл do:

```
do {
  next = prompt("Введите следующее имя");
  if (next > " ") names[i] = next;
  i = i + 1;
  }
  while (next > " ");
```


http://ITBookz.ru/



Если сценарий необходимо сделать как можно меньше, то вместо выражения i = i + 1 используйте оператор ++.

Этот цикл отображает запрос на введение значения строковой переменной next. После введения имени (и если оно больше пробела) оно сохраняется в виде следующего элемента массива names. Запрос отображается на экране до тех пор, пока пользователь не прекратит вводить имена или щелкнет на кнопке Cancel (Отмена).

Далее давайте, ради полноты сценария, отобразим число введенных в массив имен:

```
document.write("<H2>" + (names.length) + " введенных имя.</H2>");
```

Этот оператор отображает свойство length массива имен names, для выразительности выделенное заголовком второго уровня.

Далее отобразим все сохраненные в массиве имена в порядке их введения. Поскольку мы обрабатываем массив, давайте используем цикл for...in:

```
document.write("<OL>");
for (i in names) {
  document.write("LI" + names[i] + "<BR>");
}
document.write("</OL>");
```

Теперь у вас есть цикл for...in, который позволяет просматривать элементы массива. Счетчик принимает значения номеров элементов массива. Имя пользователя распечатывается с дескриптором <LI>, представляющим элемент массива в виде элемента упорядоченного списка.

Теперь вы имеете все необходимые элементы для создания работающего сценария. Листинг 8.7 содержит программный код всего документа HTML, в который вставлен наш сценарий.

Листинг 8.7. Сценарий ввода имен пользователей и распечатывания их в виде списка

```
1:    <HTML>
2:    <HEAD>
3:    <TITLE>Пример использования цикла</TITLE>
4:    </HEAD>
5:    <BODY>
6:    <H1>Пример использования цикла</H1>
7:     <P>Введите несколько имен. Они будут
8:    отображены в виде упорядоченного списка.</P>
9:    <SCRIPT LANGUAGE="JavaScript 1.2">
10:   names = new Array();
11:   i = 0;
12:   do {
13:        next = prompt("Введите следующее имя");
14:       if (next > " ") names[i] = next;
15:       i = i + 1;
16:        }
17:   while (next > " ");
18:   document.write("<H2>" + (names.length) + " введенных имен </H2>");
19:   document.write("<OL>");
20:   for (i in names) {
21:       document.write("<LI>" + names[i] + "<BR>");
```


http://ITBookz.ru/

```
22:         }
23:     document.write("</OL>");
24:     </SCRIPT>
25:     </BODY>
26:     </HTML>
```

Если загрузить этот документ в броузер, то на экране появится запрос на введение имени. Введите все необходимые имена, а затем щелкните на кнопке Cancel (Отмена), На рис. 8.1 показан результат выполнения сценария документа HTML.

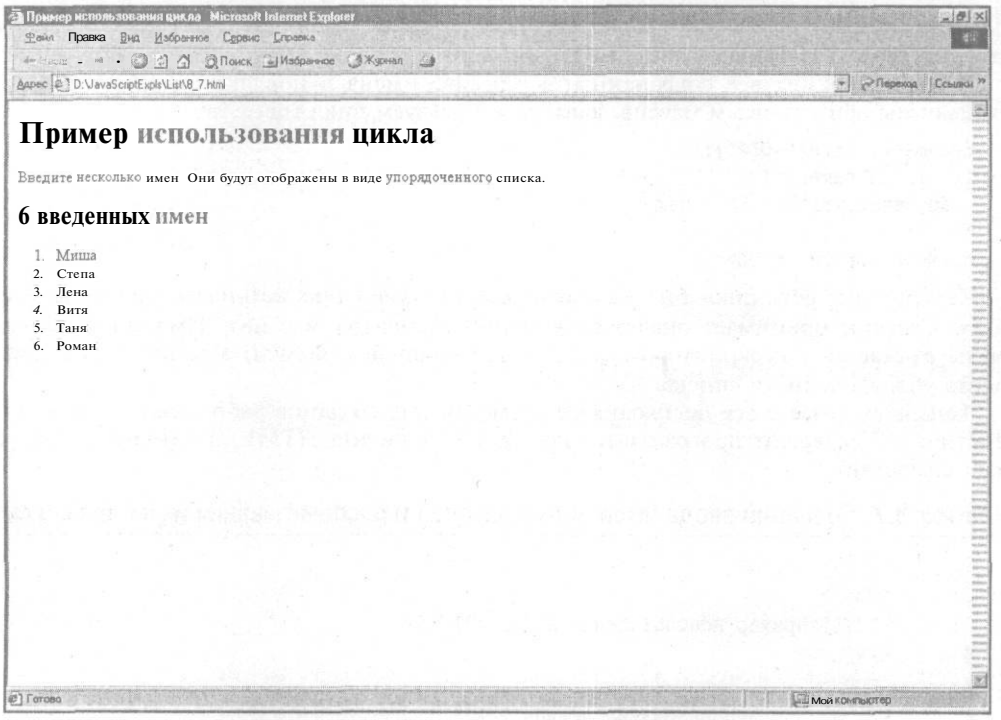


Рис. 8.1. Выполненный сценарий листинга 8.7

Резюме

За этот час вы научились использовать операторы for и while для создания циклов. Вы также познакомились с необычным циклом `for...in`, позволяющим просто и удобно обрабатывать свойства объектов и элементы массива.

Вот вы и изучили часть II книги, которая познакомила вас с основными элементами программного кода JavaScript. В части III книги вы познакомитесь с дополнительными элементами JavaScript — объектами и обработчиками.

http://ITBookz.ru/

Вопросы и ответы

Как правило, любой цикл `for` может заменить другие циклы (while, do **и т.д.). Зачем тогда необходимо столько разных конструкций циклов?**

Вы правы. В большинстве случаев цикл `for` используется по умолчанию. Другие операторы позволяют создавать циклы, удобные для решения одних задач и громоздкие — для других.

Я работаю с JavaScript 1.1. Существует ли в нем альтернатива циклу `do...while`?

Да. Используйте для создания цикла оператор while, а в тело цикла добавьте оператор if для проверки условия. Если условие выполняется, добавьте оператор break для прерывания цикла.

Введение имен по повторяющемуся запросу очень утомительно и скучно. Существует ли более простой метод запрашивания информации у пользователей?

Идеальный метод — это форма ввода данных. Детально о ней будет рассказано в главе "14-й час. Формы введения данных".

Семинар

Контрольные вопросы

1. В каком из типов циклов условие проверяется в конце тела цикла?
   a) for
   b) while
   c) do...while
2. Что делает оператор break, вставленный в тело цикла?
   a) Прерывает работу компьютера
   b) Начинает цикл сначала
   c) Прерывает выполнение цикла
3. Оператор while `(3 == 3)` — это пример
   a) Опечатки
   b) Бесконечного цикла
   c) Неправильного оператора JavaScript

Ответы

1, a) В цикле `do...while` условие проверяется в конце

2, b) Оператор break прерывает работу цикла

3, c) Этот оператор определяет бесконечный цикл

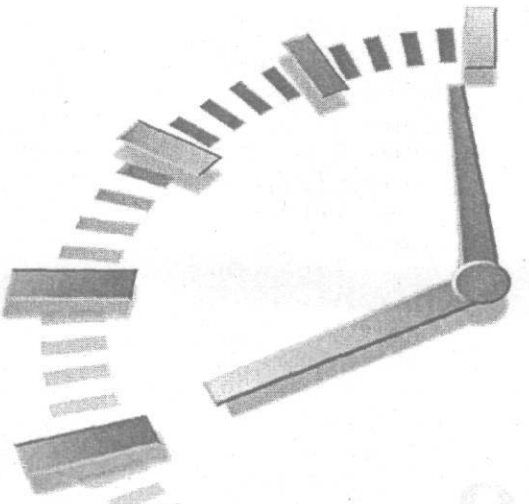
http://ITBookz.ru/

Упражнения

Чтобы повысить ваш уровень познаний о циклах, выполните следующие упражнения.

- Измените листинг 8.7 таким образом, чтобы имена упорядочивались в алфавитном порядке. Используйте для этого метод sort, применяемый к массивам. Детально он описан в главе "6-й час. Использование массивов и строковых данных".
- Измените листинг 8.7 таким образом, чтобы на экране отображался запрос только десяти имен. Что произойдет, если щелкнуть на кнопке Cancel (Отмена) вместо введения имени.

http://ITBookz.ru/

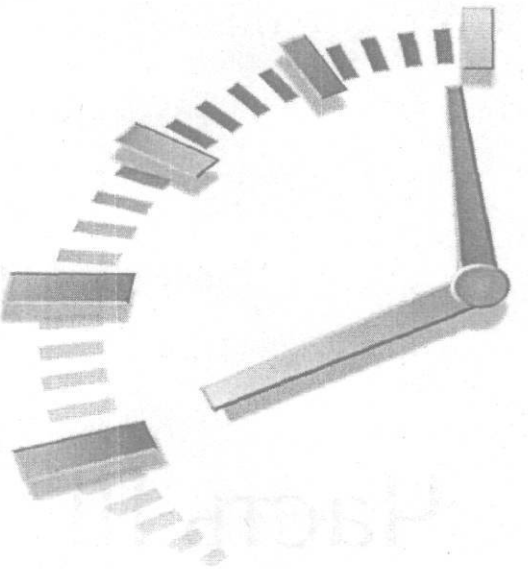


Часть III
Дополнительные возможности JavaScript

Темы занятий

9. Использование встроенных объектов
10. Работа с объектной моделью документа
11. Создание пользовательских объектов
12. Обработка событий

http://ITBookz.ru/



9-й час

Использование встроенных объектов

Мы переходим к изучению части III этой книги. (Если вы изучили предыдущие главы, не отрываясь от книги, то немного отдохните перед тем, как продолжить изучение.) В этой части вы узнаете о двух дополнительных средствах JavaScript — объектах и обработчиках.

В течение этого часа вы познакомитесь с основными возможностями использования объектов в программировании. В качестве примера будут рассмотрены встроенные объекты Math и Date. В этом уроке рассмотрены следующие темы.

- **Что такое объект JavaScript и как он используется**
- **Использование свойств объекта**
- **Управление методами объекта**
- **Использование методов объекта** Math
- **Управление датами с помощью объекта** Date
- **Создание приложения JavaScript с применением математических функций**

http://ITBookz.ru/

Что такое объект

Как вы уже знаете из предыдущих глав, *объект* сочетает в себе несколько типов данных (свойств) и функций (методов), управляющих этими данными. В этой главе вы детально познакомитесь со встроенными объектами Math и Date. Но сначала давайте посмотрим, как используются объекты в JavaScript.

Создание объектов

JavaScript имеет специальные функции, называемые конструкторами, которые используются для создания объектов. Например, встроенная функция String используется для создания объектов String. Строковая переменная создается следующим образом:

```
myname=new String("Получилось");
```

Ключевое слово new дает указание JavaScript создать новый объект — или, в технической терминологии, новый экземпляр объекта String. Этот экземпляр строкового объекта имеет значение Получилось и отзывается на имя myname.

Подобный синтаксис можно применять для создания любого объекта: String, Date, Array или даже пользовательского. (Объекты Math составляют исключение. О них я расскажу в следующих разделах главы.)



Если вам просто не терпится прямо сейчас создать пользовательский объект, то пропустите пару глав и перейдите к рассмотрению главы "11-й час. Создание пользовательских объектов".

Значения и свойства объекта

Каждый объект имеет одно или несколько *свойств,* или атрибутов. Каждое свойство — это отдельная переменная, сохраненная в объекте. Каждой переменной определяется значение. Свойства используются для сохранения данных определенного типа.

Вы уже использовали некоторые свойства объектов в предыдущих уроках. Например свойство length массивов и строковых переменных. Чтобы обратиться к свойству, вводится имя объекта, разделитель (точка), а после него имя свойства. Например, длина массива names определяется следующим образом:

```
names.length
```

В качестве свойств объектов могут выступать другие объекты. Например, каждый элемент массива — это свойство специального типа, обозначенное своим индексом. Длина первого элемента массива, например, определяется следующим образом:

```
names[0].length
```

Методы

Как вы знаете из главы "4-й час. Выполнение программ JavaScript", функции – это набор операторов, которые выполняются как единое целое. Методы — это функции, которые сохраняются в виде свойств объектов.

Вы уже использовали методы. Например, метод toUpperCase объекта String преобразует произвольный текст в текст, введенный строчными символами. Ниже приведен пример использования метода:

```
value.toUpperCase();
```


http://ITBookz.ru/

Подобно обычным функциям, методы возвращают значения. Например, следующий оператор округляет числовое значение с помощью метода round объекта Math и заносит результат в переменную final:

```
final = Math.round(num);
```

Ключевое слово with

Ключевое слово with вам еще не встречалось. С его помощью программы JavaScript становятся проще или, по меньшей мере, их становится проще вводить.

Ключевое слово with определяет объект. После него в скобках вводятся операторы действия. Для каждого оператора любое, указанное без имени объекта, свойство по умолчанию считается свойством указанного вместе с ключевым словом with объекта.

Предположим, у вас есть строковый объект lastname. Чтобы выполнить необходимые операции с ним и упростить программу, добавьте в ее код ключевое слово with:

```
with (lastname) {
  window.alert("Длина фамилии: " + length);
  toUpperCase();
}
```

В этом примере свойство length и метод toUpperCase соответствуют объекту lastname, хотя он явно определен только один раз после ключевого слова with.

По правде говоря, ключевое слово with не несет никакой функциональной нагрузки. Оно предназначено только для упрощения введения программного кода. Тем не менее оно очень распространено и часто используется при управлении объектами, создании громоздких процедур и обработке свойств встроенных объектов, таких как Math.

Объект Math

Объект Math — это встроенный в JavaScript объект, содержащий математические константы и функции. Вам нет необходимости создавать объект Math, поскольку он уже создан и готов к использованию. Свойства объекта Math содержат математические константы, а методы — математические функции.



Поскольку свойства и методы объекта Math управляются с помощью целого набора всевозможных операторов, очень удобно использовать вместе с этим объектом ключевое слово with, описанное раньше в этой главе.

Округление и усечение

Три часто используемых метода объекта Math позволяют округлять десятичные дроби до целых значений.

- Math.ceil(). Округляет число до ближайшего большего целого
- Math.floor(). Округляет число до ближайшего меньшего целого
- Math.round(). Округляет число до ближайшего целого

Все эти методы имеют только один аргумент — округляемое значение. Это не всегда удобно: иногда возникает ситуация, когда необходимо округлять не к целому числу, а к определенному десятичному знаку (например, при расчете денежных величин). Листинг 9.1 демонстрирует, как это упущение можно устранить.

http://ITBookz.ru/

Листинг 9.1. Округление число до двух десятичных знаков

```
1:    function round(num) {
2:        return Math.round(num * 100) / 100;
3:    }
```

Округляемое значение сначала умножается на 100. Таким образом число целых разрядов увеличивается на два. Затем это число округляется и делится на 100. Таким образом получается округленное значение, имеющее два десятичных разряда.

Генерация случайных чисел

Один из часто используемых методов объекта Math нельзя не описать в этом уроке. Это метод Math.random(), позволяющий генерировать случайные числа. Этот метод не требует использования дополнительных параметров. Он возвращает произвольное десятичное число в диапазоне от нуля до единицы.

Следует заметить, что обычно необходимо получить случайное число в диапазоне от 1 до некого значения переменной а. Это число можно получить с помощью специальной функции. Листинг 9.2 демонстрирует, как получить произвольное число в диапазоне от 1 до указанного вами числа.

Листинг 9.2. Получение случайного числа в указанном диапазоне

```
1:    function rand(num) {
2:        return Math.floor(Math.random() * num) + 1;
3:    }
```

Эта функция получает произвольное число следующим образом: указанное число умножается на случайное значение, сгенерированное методом Math.random(), и полученное число преобразуется в целое с помощью метода Math.floor().

Управление датами

Еще один встроенный объект, с которым вам необходимо обязательно познакомиться, — это Date. Этот объект позволяет использовать в сценариях JavaScript даты и числа. Вы можете создать объект Date в любой момент, когда вам понадобится задать дату или применить метод, управляющий датами:

Вы уже имели опыт использования объекта Date в главе "2-й час. Создание простых сценариев" при создании сценария расчета текущего времени. Объект Date не имеет ни одного свойства. Чтобы получить или определить значение объекту Date, необходимо использовать методы, описанные ниже.



Даты в JavaScript сохраняются в миллисекундах, начиная с00:00 01.01.1970. Эта дата называется эпохальной. Даты до этого момента времени использовать в JavaScript нельзя. Это означает, что в объекте Date нельзя сохранять дни рождения ваших родителей и, тем более, дедушек и бабушек. Это вызвано тем, что в сценариях намного чаще приходится иметь дело с настоящим и будущим, нежели с прошлым.

http://ITBookz.ru/

Создание объекта Date

Объект Date создается так же, как и другие объекты JavaScript, — с помощью ключевого слова new. При создании объекта также можно указать и дату, сохраняемую в нем. Для этого используется один из следующих форматов записи:

```
birthday = new Date();
birthday = new Date("June 20, 1999 08:00:00");
birthday = new Date(6, 20, 1999);
birthday = new Date(5, 20, 1999, 8, 0, 0);
```

Выберите один из форматов, в зависимости от вводимого значения и его дальнейшего использования. Если при создании объектов не использовать ни одного параметра (как в первом случае), то в объекте сохраняется текущая дата. Определить дату в объекте Date позволяет метод set, описанный в следующем разделе.

Определение значения объекта Date

Для определения значения объекта Date используется целый набор методов set.

- setDate () определяет день месяца
- setMonth() определяет месяц. Месяц в JavaScript определяется числом в диапазоне от 0 до 11.
- setYear() определяет год
- setTime() определяет время (и дату), отсчитанное в миллисекундах от начала первого января 1970 года
- setHours(), setMinutes() и setSeconds() определяет время в часах, минутах и секундах соответственно

В качестве примера приведем оператор, который определяет год объекта Date с названием holiday как 99:

```
holiday.setYear(99);
```

Получение значений объекта Date

Метод get используется для получения значений, сохраненных в объекте Date. Это единственный способ получения значения объекта Date, поскольку свойств он не имеет. Ниже приведены все используемые в JavaScript методы.

- getDate(). Возвращает день месяца
- getMonth(). Возвращает месяц
- getYear(). Возвращает год
- getTime(). Возвращает время, отсчитанное в миллисекундах от начала первого января 1970 года
- getHours(), getMinutes() и getSeconds(). Возвращает время в часах, минутах и секундах соответственно



Наряду с методами setFullYear и getFullYear, требующими введения года в четырехзначном виде, в JavaScript используются и более простые методы setYear и getYear, позволяющие вводить год сокращенно, в виде двух цифр. Чтобы избежать "проблемы 2000 года", необходимо использовать четырехзначный формат записи года.

http://ITBookz.ru/

Временные зоны

И наконец, несколько функций позволяют использовать в JavaScript даты разных временных поясов.

- `getTimezoneOffset()`. Возвращает значение сдвига во времени относительно нулевого меридиана (нулевого временного пояса). В этом случае локальное время определяется по определенному в Windows временному поясу. (Перед использованием этого метода убедитесь в правильности определения параметров в аплете Date/Time (Язык и стандарты) папки Control Panel (Панель управления).)
- toGMTString(). Преобразует значение объекта Date в текстовое значение в формате UTC (всеобщего скоординированного времени).
- `toLocalString()`. Преобразует значение объекта Date в текст в формате локального временного пояса.

Изменение формата представления даты

Два специальных метода JavaScript позволяют изменять формат записи значения объекта Date. Вместо того чтобы применять эти методы к созданному объекту Date, используйте их вместе со встроенным объектом Date.

- `Date.parse()`. Преобразует текстовую запись даты, например Jun 20, 1996, в значение объекта Date (число миллисекунд, отсчитанных от начала 1/1/1970).
- `Date.UTC()`. Выполняет обратное предыдущему методу действие. Преобразует значение объекта Date (введенное в миллисекундах) в привычный вид (число: месяц:год время).

Применение объектов Math на практике

Метод `Math.random`, описанный выше в этой главе, генерирует случайное значение в диапазоне 0—1. Тем не менее, для компьютера очень сложно сгенерировать действительно произвольное значение. (Человеку сделать это еще сложнее, поэтому для решения этой задачи привлекаются компьютеры.)

На сегодня компьютеры достигли больших высот в методах генерирования случайных чисел. Насколько хорошо выполняет эту операцию функция `Math.random`? Один из способов — это сгенерировать достаточно большой набор случайных чисел и получить их среднее арифметическое.

По теории среднее бесконечного числа случайных чисел должно равняться 0,5. Чем больше чисел вы сгенерируете, тем больше будет приближаться их среднее значение к величине 0,5.

В качестве примера использования метода `Math.random` создадим сценарий, который позволяет проверить точность этой функции. Для этого сгенерируем 5000 случайных чисел и вычислим их среднее значение.

Если вы до сих пор не ознакомились с главой "8-й час. Повторение — мать учения: циклы" и уже вытянули калькулятор, не спешите — с помощью цикла все случайные числа будут усреднены автоматически. (Это выполняется настолько быстро, что вы забудете о существовании калькулятора до конца своих дней.)

В самом начале сценария объявим переменную total. Эта переменная будет сохранять сумму всех случайных чисел. Поэтому определим ей начальное значение 0:

```
total = 0;
```


http://ITBookz.ru/

Дальше зададим цикл, который будет выполнять одни и те же операторы 5000 раз. Поскольку мы знаем точное количество итераций, давайте использовать цикл for:

```
for (i=0; i<5000; i++) {
```

В теле цикла необходимо ввести операторы генерирования произвольного числа и вычисления текущего значения переменной total. Вот какие операторы будут выполняться каждую итерацию цикла:

```
  num = Math.random();
  total += num;
}
```

В зависимости от суммарной производительности вашего компьютера генерирование 5000 случайных чисел займет несколько минут. Чтобы быть уверенным, что во время выполнения сценария компьютер не завис, а проводит вычисления, давайте выведем сообщение о количестве чисел, которые еще осталось сгенерировать:

```
window.status = "Сгенерировано " + i + " чисел. Общая сумма: " + total;
```

В конце сценария необходимо рассчитать среднее значение (значение переменной total делится на 5000). Определим точность полученного значения до трех десятичных знаков. Для этого применим способ, описанный в этой главе:

```
average = total / 5000;
average = Math.round(average * 1000) / 1000;
document.write("<H2>Среднее арифметическое случайно полученных чисел: " + average +
"</H2>");
```



Если ваш компьютер не отличается высокой производительностью (имеет тактовую частоту ниже 60 МГц) измените точность вычисления среднего и количество итераций (т.е. усредняемых случайных чисел). В противном случае вам придется ждать результатов очень долго.

Чтобы протестировать полученный сценарий, вставьте его в документ HTML, обозначив дескрипторами <SCRIPT>. Листинг 9.3 содержит полный программный код работающей Web-страницы с созданным сценарием.

Листинг 9.3. Сценарий вычисления среднего значения случайных чисел

```
1:          <HTML>
2:          <HEAD>
3:          <TITLE>Пример объекта Math</TITLE>
4:          </HEAD>
5:           <BODY>
6:          <H1>Пример объекта Math</H1>
7:          <P>Насколько случайны числа, полученные
8:           генератором случайных чисел7 Подсчитаем
9:          среднее 5000 случайных чисел.</P>
10:         <SCRIPT LANGUAGE="JavaScript">
11:         total = 0;
12:       for (i=0; i<5000; i++) {
13:             num = Math.random();
14:             total += num;
15:            document.status = "Сгенерировано " + i + " чисел";
16:        }
17:         average = total / 5000;
18:         average = Math.round(average * 1000) / 1000;
```


http://ITBookz.ru/

```
19:          document.write("<H2>Среднее арифметическое случайно полученных
                  чисел: " + average + "</H2>");
20:          </SCRIPT>
21:          </BODY>
22:          </HTML>
```

Чтобы проверить правильность выполнения сценария, загрузите Web-страницу в броузер. В строке состояния будет отображаться прогресс вычислений. После вычисления среднего 5000 случайных чисел на экран будет выведен результат. Я получил результат 0,494, что показано на рис. 9.1.

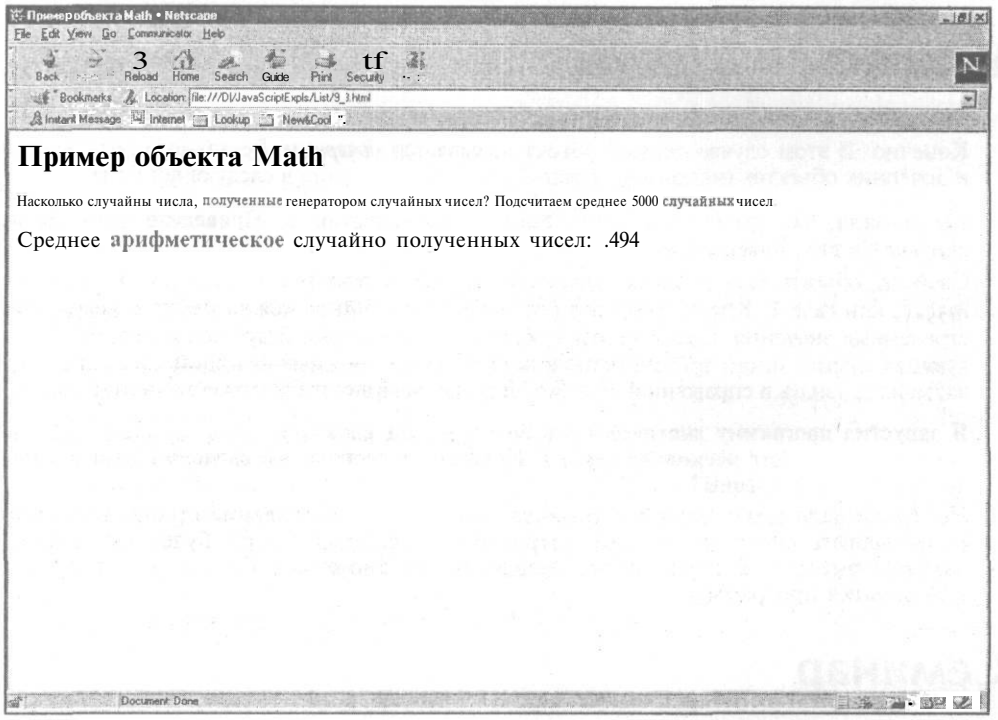


Рис. 9.1. Тестирование точности генератора случайных чисел



Среднее число, полученное вами, это *среднее арифметическое.* Этот тип среднего числа не позволяет добиться большой точности определения правильности работы генератора случайных чисел. На самом деле этот тип среднего числа показывает распределение чисел относительно значения 0,5. Например, если усреднить 2500 чисел 0,4 и 2500 чисел 0,6, то получится 0,5. Но эти числа не есть случайными. Именно поэтому с помощью этого метода нельзя точно определить правильность работы генераторы случайных чисел.

http://ITBookz.ru/

Резюме

За этот час вы познакомились с фундаментальными элементами JavaScript — объектами, которые будут изучаться вами во всех уроках главах части III. Вы узнали и предназначении объектов Math и Date, а также научились использовать их на практике. Кроме того, вы познакомились со встроенным в JavaScript генератором случайных чисел и создали сценарий проверки точности его работы.

В следующей главе вы познакомитесь с часто используемыми объектами в JavaScript — объектами броузера. Они позволяют управлять разными элементами окна броузера и документа HTML.

Вопросы и ответы

Может ли один объект быть свойством другого объекта?

Конечно. В этом случае первый объект называется *дочерним*. Детально о родительских и дочерних объектах (например, `window.location`) вы узнаете в следующей главе.

Вы сказали, что свойства объекта Math — это константы. Приведите хотя бы несколько из них, пожалуйста.

Свойства объекта Math содержат распространенные математические константы, например `Math.PI` или `Math.E`. Кроме того, среди свойств этого объекта можно найти и малораспространенные значения. Среди методов объекта Math (которые будут использоваться в следующих главах) много тригонометрических и других сложных функций. Детально о них вы можете узнать в справочной системе Netscape Navigator и Приложении А этой книги.

Я запустил программу листинга 9.3 в броузере. На сложение двух случайных чисел у компьютера уходит несколько секунд. Неужели персональные вычислительные машины настолько медленны?

На самом деле самая сложная команда цикла — это обновление строки состояния. Если удалить соответствующий оператор из сценария, то он будет выполняться намного быстрее. В этом случае, правда, вы не сможете наблюдать за прогрессом выполнения программы.

Семинар

Контрольные вопросы

1. Какой из приведенных объектов не используется вместе с ключевым словом new?
   a) Date
   b) Math
   c) String
2. В каком виде сохраняется дата в объекте Date?
   a) В виде количества миллисекунд от начала первого января 1970 года
   b) В виде количества дней от начала первого января 1900 года
   c) В виде количества секунд от начала создания компании Netscape
3. Какой диапазон чисел генерирует метод `Math.random`?
   a) 0-100
   b) От 1 до указанного значения
   c) 0-1

http://ITBookz.ru/

Ответы

1, b) Объект Math — статический. Вы не можете создать новый объект Math

2, a) Даты отсчитываются от начала первого января 1970 года

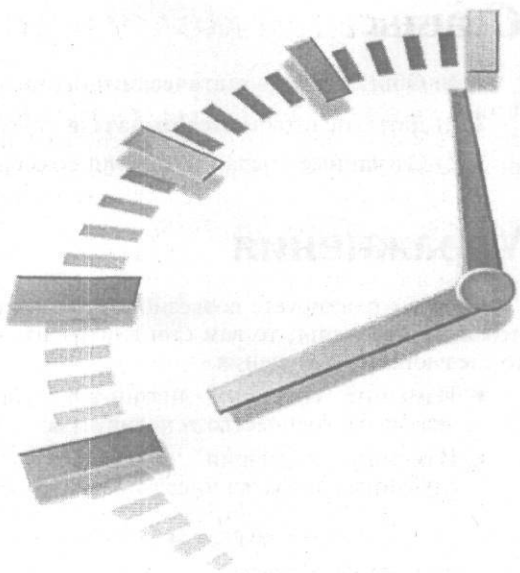
3, c) Случайные числа в JavaScript создаются в диапазоне от 0 до 1

Упражнения

Если вы планируете серьезно заняться математической обработкой данных и дат с помощью JavaScript, то вам стоит повысить свою квалификацию. Для этого выполните следующие упражнения.

- Измените сценарий листинга 9.3 таким образом, чтобы пользователь мог указывать количество генерируемых случайных чисел.
- Измените сценарий листинга 9.3 так, чтобы он рассчитывал среднее случайных 15 тысяч чисел и каждого набора по 5000 чисел.

http://ITBookz.ru/



10-й час

Работа с объектной моделью документа

За прошлый час вы познакомитесь с объектами, позволяющими сохранять данные любых типов. Среди всех средств JavaScript, которые вы используете в сценариях, объектная модель документа (Document Object Model — DOM), пожалуй, самая необычная. Она позволяют управлять не только документом HTML, но и самим броузером.

В этой главе вы познакомитесь с иерархической структурой объектов броузера. В ней рассмотрены следующие темы.

- **Доступ к различным объектам DOM**
- **Управление окном броузера с помощью объекта** window
- **Управление Web-документом с помощью объекта** document
- **Использование объектов для создания ссылок и анкеров**
- **Управление URL с помощью объекта** location
- **Получение информации о броузере с помощью объекта** navigator
- **Создание** кнопок Back (Назад) и Forward (Вперед)

http://ITBookz.ru/

Объектная модель документа

Преимущество JavaScript, даже по сравнению со сложными языками программирования, состоит в том, что он позволяет управлять броузером. Сценарий позволяет загрузить в окне броузера новую Web-страницу, управлять внешним ее видом и преобразовывать документ, а также запускать новые окна броузера.

Для того чтобы справиться со всеми этими операциями, в JavaScript включена специальная иерархическая структура — объектная модель документа (DOM). В соответствии с ней все объекты сценария организованы в древовидную структуру, отвечающую за все без исключения элементы Web-страницы: окно, документ, рисунок и т.п.

Объекты DOM, как и другие объекты, имеют *свойства,* описывающие Web-страницу, и методы, позволяющие управлять частью документа.

Объекты броузера упорядочены в иерархическом порядке, создавая структуру родительских и дочерних объектов. При определении объекта вы сначала вводите имя родительского объекта, после него ставите разделитель (точку), а затем имя дочернего объекта. Например, объекты, соответствующие рисункам Web-страницы, соответствуют дочерним объектам родительского объекта document. В следующем примере определен объект image9, который является дочерним по отношению к объекту document, который, в свою очередь, тоже дочерний — только уже по отношению к объекту window:

```
window.document.image9
```

Объект window находится на самой верхушке иерархической структуры. Рис. 10.1 демонстрирует иерархическую структуру объектов DOM и разнообразие всевозможных объектов.

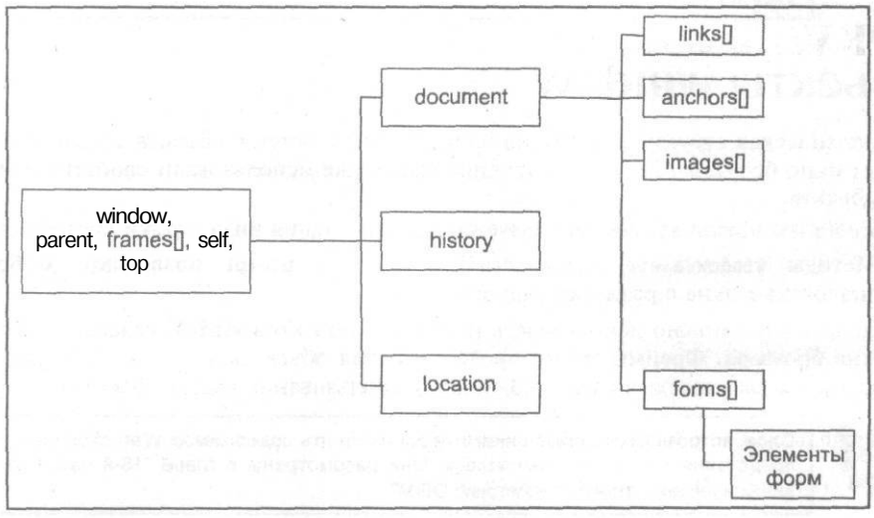


Рис. 10.1. Иерархическая структура объектов DOM в JavaScript



На этой блок-схеме представлены только основные объекты броузера, которые будут описаны в этом уроке. Остальные объекты иерархической структуры также рассмотрены в этой книге, но не так подробно.

http://ITBookz.ru/

История DOM

Начиная с версии 1.1 (реализованной в Netscape 2.0), в JavaScript включены объекты броузера, позволяющие управлять частью документа HTML и элементами броузера. Но на то время объекты броузера еще не стали стандартом. Хотя Netscape Navigator и Internet Explorer уже использовали эту модель, не существовало гарантий, что новый объект будет в них поддерживаться стопроцентно. К тому же одни и те же объекты выполнялись в этих двух броузерах по-разному.

В настоящее время также существует разница в объектных моделях броузеров Netscape и Microsoft. Но для вас припасена и хорошая новость. С момента выпуска Netscape Navigator 3.0 и Internet Explorer 4.0 все основные объекты (описанные в этой главе) поддерживаются обоими броузерами и выполняются одинаково.

Совсем недавно консорциум *W3C* (World Wide Web Consortium — Консорциум по стандартизации Web) разработал стандарт DOM. В этот стандарт включены не только базовые объекты, но и объекты управления частями документа HTML, а также объекты, поддерживаемые в XML.

Стандарт DOM частично поддерживается как в Netscape, так и Internet Explorer. Netscape Navigator 6.0 и Internet Explorer 5.0 максимально усовершенствованы для поддержки нового стандарта.

Хотя новый стандарт никак не влияет на описанные в этой главе примеры, в будущем вы будете, несомненно, благодарны *W3C* за введение стандарт DOM.



Базовая иерархическая структура объектов, описанная в этой главе, соответствует DOM нулевого уровня. DOM 1 и DOM 2 включают в себя все основные объекты модели DOM 0. Детально уровнями модели DOM вы познакомитесь в последней части этой книги.

Объекты window

Иерархическая структура объектов броузера начинается с объекта window. Он представляет окно броузера. Вы в предыдущих главах уже использовали свойства и методы этого объекта.

- Свойство window. status используется для изменения вида строки состояния.
- Методы window.alert, window.confirm и window.prompt позволяют отображать диалоговые окна с разными запросами.

Одновременно можно использовать несколько объектов window, каждый для открытого окна броузера. Фреймы также представляются объектами window. Детально о окнах и фреймах вы узнаете в главе "13-й час. Использование окон и фреймов".



Слои, которые позволяют динамически изменять содержимое Web-страницы, также представляются объектами window. Они рассмотрены в главе "18-й час. Создание динамических страниц с помощью DOM".

Управление Web-документами

Объект document представляет объект Web-документа или Web-страницы. Документы или страницы отображаются в окне броузера, поэтому не удивительно, что объект document дочерний по отношению к объекту window. Изменения, проведенные с помощью объекта document, будут отображаться в окне броузера, а поэтому сказываться на объекте window.

http://ITBookz.ru/



Для отображения текста на Web-странице используется метод document.write. Примеры программных кодов, представленные в первых главах книги, рассчитаны на запуск только одного окна броузера, поэтому в них не было необходимости вводить имя родительского объекта window.document.write. В многооконной среде использовать длинную запись метода просто необходимо.

Использование нескольких окон броузера или фреймов подразумевает существование нескольких объектов window с одним дочерним объектом document. Для использования объекта документа вы вводите имя объекта окна и после точки имя объекта документа.

В следующих разделах вы познакомитесь с некоторыми свойствами и методами объекта document, часто используемыми при написании сценариев.

Получении информации о броузере

Некоторые свойства объекта document используются для получения сведений о броузере и текущем документе.

- Свойство URL используется для определения адреса текущей Web-страницы. Адрес вводится одним словом. Изменить это свойство нельзя. Если необходимо открыть в окне броузера другую страницу, используйте объект window.location, описанный дальше в этой главе.
- Свойство title содержит заголовок текущей страницы, определенный после дескриптора <TITLE>.
- Свойство referrer определяет адрес Web-страницы, просматриваемой до отображения текущего Web-документа. Как правило, на ней есть ссылка на текущую страницу.
- Свойство lastModified содержит дату последнего изменения Web-страницы. Эта дата сохраняется на сервере и присылается при отображении Web-страницы в броузере.

В приведенном ниже листинге 10.1 продемонстрирован документ HTML, в котором отображается дата последнего его изменения.

Листинг 10.1. Отображение даты последнего изменения документа

```
1:  <HTML><HEAD><TITLE>Тестирование документа</TITLE></HEAD>
2:  <BODY>
3:  Эта страница последний раз была изменена:
4:  <SCRIPT>
5:  document.write(document.lastModified);
6:  </SCRIPT>
7:  <BR>
8:  </BODY>
9:  </HTML>
```

Эта программа позволяет пользователям узнать дату последнего изменения Web-страницы. При использовании этого сценария не забудьте при обновлении страницы изменить дату ее последнего изменения.



В некоторых случаях свойство lastModified не выполняется в Web-страницах. Даты последнего изменения сохранены на сервере, а некоторые серверы не сохраняют дату изменения страницы.

http://ITBookz.ru/

Добавление в документ текста

Простейший метод объекта document — это тот, который используется чаще других. Фактически вы его уже многократно использовали. Метод document.write отображает текст документа в окне броузера. Этот оператор используется тогда, когда необходимо добавить результат выполнения операций на Web-страницу.

Подобный ему метод document.writeln также позволяет отобразить в окне броузера текст и содержит в конце строки оператор /п. Этот оператор указывает на то, что в этой строке больше ничего отображаться не будет. Поэтому метод document.writeln используется в тех случаях, когда необходимо в одной строке вывести только результат.



Помните, что некоторые дескрипторы <HTML> не позволяют выводить результат в отдельной строке. К ним, например, относится дескриптор <PRE>. Если вы хотите вывести результат в отдельной строке, используйте дескриптор <BR>.

Эти методы используются только в теле Web-страницы и выполняются при ее загрузке. Вы не можете добавить результат в уже загруженную страницу без ее перезагрузки. О том, как вставить в документ новые данные, рассказано в следующем разделе



Текст на Web-странице также изменяется с помощью новых объектов DOM, поддерживаемых в новых броузерах. Детально о них будет рассказано в главе "19-й час. Дополнительные средства DOM".

Метод document.write используется только в дескрипторе <SCRIPT>, размещенном в теле документа HTML. Его можно использовать и в функции, вызов которой обязательно осуществляется в теле документа HTML.

Очистка и обновление содержимого Web-страницы

Для объекта document определены методы open и close. Несмотря на существование подобных методов в объекте window, объект document использует их для выполнения других операций. Эти методы используются не для открытия или закрытия документов или окон броузера, как может показаться вначале. Метод open открывает новый *поток,* очищающий содержимое документа и подготавливающий его для добавления данных с помощью методов write и writeln.

При использовании метода document.open содержимое текущего документа удаляется. Любые данные, отображаемые в нем, теряются безвозвратно. После этого вы можете приступать к отображению в документе новых данных.

Данные, добавляемые в документ после метода document.open, не отображаются до тех пор, пока не использован метод document.close, закрывающий поток. После нахождения этого оператора JavaScript приступает к выполнению блока операторов write, введенных между указанными методами.



Если применять метод document.open к текущему окну, то ваш сценарий, как часть программы HTML, будет прерван и очищен. Именно поэтому данный метод хорошо использовать при управлении несколькими окнами или фреймами. Детально о них рассказано в главе 13.

В команде document.open можно дополнительно определить тип данных MIME. Это позволит вам создавать документ любого типа, включая рисунки и документы, используемые приложениями. Детально о внедряемых модулях рассказано в главе "20-й час. Использование мультимедиа и встроенных утилит".

http://ITBookz.ru/



MIME — это аббревиатура от *Multipurpose Internet Mail Extensions* (Многоцелевые расширения электронной почты в Internet). Это стандарт Internet для используемых типов документов. Web-сервер посылает в броузер сведения о типе MIME документа, чтобы тот мог правильно отобразить Web-документ. Типичный документ имеет тип HTML (соответствует типу text/html).

Использование ссылок и анкеров

Еще один дочерний объект объекта document, о котором хотелось рассказать, — это link. В одном объекте document одновременно может существовать несколько объектов link. Каждый из них содержит сведения о ссылке на другую страницу или анкер.



Анкер — это именованный элемент документа HTML, переход к которому осуществляется мгновенно. Он определяется следующим дескриптором: **<A** `NAME="part2">`. Ссылка же на него задается так: <A `HREF="#part2">`.

Объект link управляется с помощью массива links. Каждый элемент массива представляет собой объект link текущей страницы. Свойство массива `document.links.length` определяет количество ссылок на странице.

Каждый объект link (или элемент массива links) имеет целый набор свойств, определяющих адрес страницы, на которую добавлена ссылка. Эти же свойства имеет и объект location, описанный в этой главе. В свойствах объекта link указывается также номер ссылки и ее название. В приведенном ниже примере адрес URL первой ссылки определяется переменной `link1`:

```
link1 = links[0].href;
```

Объекты анкеров также выступают в роли дочерних по отношению к объекту document. Каждый объект анкера соответствует анкеру документа HTML — именованному элементу документа HTML.

Подобно ссылкам, анкеры управляются с помощью массива anchors. Каждый элемент этого массива — это отдельный объект anchor. Свойство `document.anchors.length` определяет количество анкеров документа HTML.

Получение сведений о работе броузера

Самый интересный дочерний объект объекта document — это, пожалуй, history. Этот объект содержит сведения о страницах, которые отображались и отображаются в окне броузера, а также содержит методы перехода к ним.

Объект history обладает четырьмя свойствами.

- `history.` length. Указывает длину списка адресов посещаемых страниц, сохраняемых в объекте history. Другими словами, это количество посещаемых за текущий сеанс Web-страниц.
- `history.current`. Содержит одно значение — адрес URL текущей страницы, открытой в окне.
- `history.next`. Тоже содержит одно значение — адрес URL страницы, к которой перейдет пользователь, если щелкнет на панели инструментов броузера на кнопке Forward (Вперед). Поскольку эта кнопка не активна до тех пор, пока хотя бы один раз не воспользоваться кнопкой Back (Назад), это свойство до определенного момента использовать нельзя.
- `history.previous`. Тоже содержит одно значение — адрес URL страницы, к которой перейдет пользователь, если щелкнет на панели инструментов броузера на кнопке Back (Назад).

http://ITBookz.ru/

Объект history можно также обрабатывать и как массив. Каждый элемент массива содержит адрес URL из списка посещаемых страниц. Текущая страница представлена элементом history[0]. Настало время описать методы объекта history. Их у него три.

- history.go. Открывает страницу по указанному адресу в списке адресов посещаемых страниц. положительные и отрицательные числа соответствуют расположению адресов в массиве history. Например, элемент history.go(-2) соответствует двойному щелчку на кнопке Back (Назад).
- history.back. Загружает страницу, которая отображалась в окне броузера перед текущей. Этот метод аналогичен щелчку на кнопке Back (Назад).
- history.forward. Загружает страницу, которая отображается в списке адресов посещаемых страниц после текущей (если она указана). Этот метод аналогичен щелчку на кнопке Forward (Вперед).



Методы history.back и history.forward в некоторых версиях Netscape Navigator выполняются неправильно. По этой причине лучше всегда использовать метод history.go(history.go(-1) и history.go(1)).

Объект location

Третий дочерний объект объекта document — это location. Этот объект содержит сведения о документе HTML, который в текущий момент открыт в окне. Например, следующий оператор дает указание загрузить страницу в текущем окне:

```
window.location.href="http://www.starlightech.com";
```

Свойство href, используемое в операторе, содержит полный адрес URL страницы, загруженной в окне. Вы можете при необходимости указывать только часть адреса URL, например location.protocol определяет протокольную часть адреса (чаще всего http:).



Хотя свойство location .href содержит тот же адрес URL, что и свойство document.URL, уже описанное в этой главе, последнее изменить вы не можете. Для загрузки новой страницы всегда используйте свойство location.href.

Объект location имеет два метода.

- location.reload. Перезагружает текущий документ. Аналогичен по функции кнопке Reload (Обновить), расположенной на панели инструментов броузера.
- location.replace. Замещает текущую страницу указанной. Действует аналогично указанию свойств объекта location вручную. Разница состоит с том, что этот метод не изменяет историю броузера. Другими словами, кнопку Back (Назад) нельзя будет использовать для перехода к исходной странице.

Получение сведений о броузере

Объект navigator не принадлежит к иерархической структуре объекта броузера. Этот объект содержит данные о версии броузера. Он используется для определения типа броузера и платформы компьютера, на которой он запущен. Зная правильную версию броузера, вы сможете лучшим образом написать сценарий и добиться максимальной его производительности.



Объект navigator назван в честь броузера Netscape Navigator, который изначально был единственным, поддерживающим JavaScript. Несмотря на свое название, этот объект прекрасно поддерживается и в Internet Explorer.

http://ITBookz.ru/

Программный код, воспринимаемый только определенным типом броузеров, добавлять в сценарий не рекомендуется. Детально с объектом navigator вы познакомитесь в главе "16-й час. Создание сценариев для разных броузеров".

Создание кнопок Back и Forward

Одно из самых частых применений методов back и forward — это добавление на Web-страницу соответствующих кнопок, позволяющих перемещаться по списку посещаемых страниц.

В качестве примера использования объекта history приведем сценарий, отображающий в документе HTML кнопки Back (Назад) и Forward (Вперед), позволяющие улучшить перемещаемость пользователя в Web. В качестве кнопок Back и Forward будут использоваться графические объекты. Выберите из вашего личного архива или создайте подходящие на ваш взгляд рисунки для кнопок.

Ниже приведена часть кода, соответствующая кнопке Back:

```
<A HREF="javascript:history.go(-1);">
  <IMG BORDER=0 SRC="left.gif">
</A>
```

В этом коде используется метод javascript: URL, позволяющий отображать предыдущую страницу после щелчка на кнопке Back. В качестве кнопки Back используется рисунок со стрелкой влево. Код задания кнопки Forward почти такой же:

```
<A HREF="javascript:history.go(1);">
  <IMG BORDER=0 SRC="right.gif">
</A>
```

Вот и все. Остальное вы уже проделывали многократно. Просто добавьте стандартные дескрипторы HTML. Листинг 10.2 содержит готовый код программы документа HTML с кнопками Back и Forward. На рис. 10.2 показано окно Netscape Navigator с загруженным в нем документом. Обязательно проверьте работоспособность кнопок Back и Forward.

Листинг 10.2. Документ HTML, содержащий кнопки перемещения по загруженным страницам

```
1:  <HTML>
2:  <HEAD><TITLE>Использование кнопок Back и Forward</TITLE>
3:  </HEAD>
4:  <BODY>
5:  <H1>Использование кнопок Back и Forward</H1>
6:  <HR>
7:  Вы можете перемещаться по уже отображаемым
8:  страницам с помощью кнопок Back и Forward.
9:  Они подобны кнопкам на панели инструментов.
10: <HR>
11: <A HREF="javascript:history.go(-1);">
12:     <IMG BORDER=0 SRC="left.gif">
13: </A>
14: <A HREF="javascript:history.go(1);">
15:     <IMG BORDER=0 SRC="right.gif">
16: </A>
17: <HR>
18: </BODY>
19: </HTML>
```


b) DOM первого уровня
c) DOM второго уровня

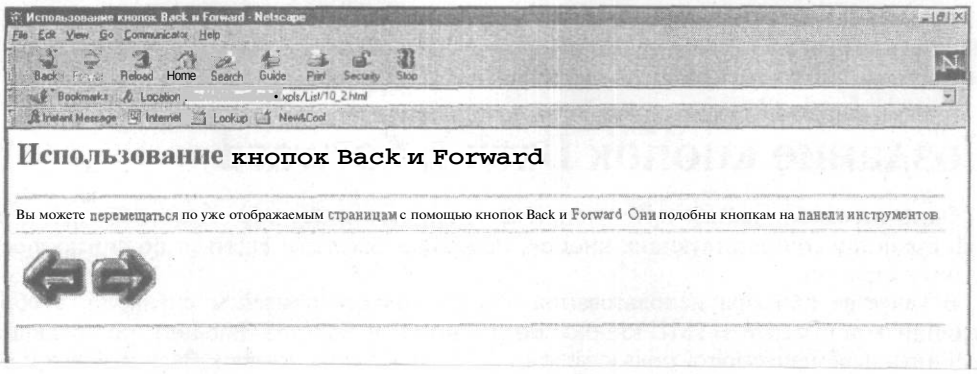
Ответы

1, b) Для отображения новой страницы в окне броузера используется объект window.location

2, a) Сведения о версии броузера содержит объект navigator

3, b) DOM 0.

http://ITBookz.ru/

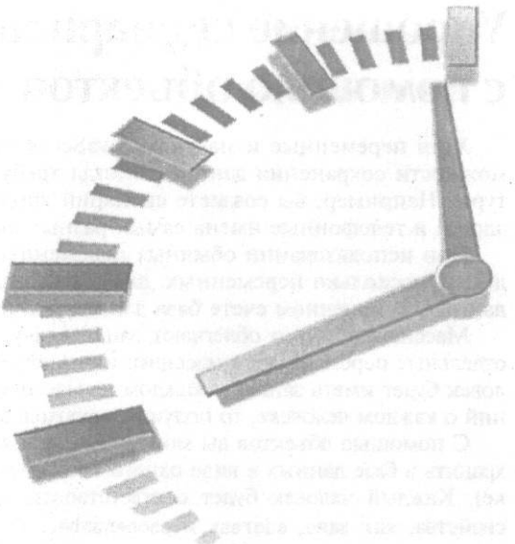


Упражнения

Чтобы повысить квалификацию в области использования объектов броузера, выполните следующие упражнения.

- Добавьте в программный код использования кнопок код кнопки Reload (Обновить), расположив ее между кнопками Back и Forward. (Эта кнопка соответствует методу `location.reload()`.)
- Измените листинг 10.2 так, чтобы при загрузке Web-страницы отображалось число элементов списка адресов посещаемых страниц.

http://ITBookz.ru/



11-й час

Создание пользовательских объектов

В предыдущих двух главах вы познакомились со встроенными в JavaScript объектами, такими, например, как Date. Вы также узнали о существовании объектов броузера и научились ими управлять. Эти типы объектов чаще других используются в JavaScript. Но иногда возникает потребность в создании собственных объектов.

За этот час вы узнаете, как создавать собственные объекты, а заодно познакомитесь с обработчиками JavaScript, позволяющими управлять самыми разными типами объектов. В этом уроке рассмотрены следующие темы.

- **Упрощение сценариев с помощью объектов**
- **Определение объектов**
- **Добавление метода в объект**
- **Создание экземпляра объекта**
- **Использование объекта для сохранения и управления данными**

http://ITBookz.ru/

Упрощение сценариев с помощью объектов

Хотя переменные и массивы JavaScript предоставляют *более чем* достаточные возможности сохранения данных, иногда требуется использовать весьма сложные структуры. Например, вы создаете сценарий управления базой данных, содержащей имена, адреса и телефонные имена самых разных людей.

При использовании обычных переменных для каждого человека вам придется создавать несколько переменных: для имени, для адреса и т.д. и все их сохранять в базе данных. В конечном счете база данных станет громоздкой и сложно управляемой.

Массивы немного облегчают вашу задачу. В базу данных теперь будут заноситься не отдельные переменные, а массивы: имен, адресов, телефонных номеров и т.д. Каждый человек будет иметь запись в каждом из массивов. Если база данных содержит много сведений о каждом человеке, то получить необходимые данные будет просто, но неудобно.

С помощью объектов вы можете все переменные, относящиеся к одному человеку, сохранять в базе данных в виде одного элемента (подобно одной личной карточке в картотеке). Каждый человек будет соответствовать отдельному объекту Card, имеющему такие свойства, как name, address и phonenumber. В объект также можно добавить специальные методы, позволяющие обрабатывать и отображать на экране необходимые данные.

В следующих разделах вы создадите объект Card со всеми необходимыми свойствами и методами. Позже в этой главе этот объект используется для отображения на экране сведений о нескольких членах базы данных.

Определение объекта

Первый шаг в использовании объекта — это определить его и его свойства. Мы уже договорились назвать объект Card. Каждый объект Card будет иметь следующие свойства:

- name
- address
- workphone
- homephone

Первым делом зададим функцию, создающую новые объекты Card. Эта функция называется *конструктором* объектов. Вот как выглядит конструктор объектов Card:

```
function Card(name,address,work,home) {
  this.name = name;
  this.address = address;
  this.workphone = work;
  this.homephone = home;
}
```

Конструктор — это простая функция, инициализирующая объект и определяющая значения всех его свойств. Эта функция имеет несколько параметров, которые приравниваются к свойствам объекта. Поскольку функция названа Card, наш объект тоже будет называться Card.

Обратите внимание на ключевое слово this. Вы будете использовать его при определении любого объекта. Оно ставится перед каждым свойством текущего объекта, т.е. объекта, который создается.

http://ITBookz.ru/

Добавление в объект метода

Теперь необходимо добавить в наш объект хотя бы один метод. Поскольку все объекты Card имеют одинаковые свойства, удобно иметь функцию, позволяющую выводить содержимое объекта в определенном формате. Назовем эту функцию PrintCard.

Функция PrintCard будет использоваться в качестве метода объекта Card. Поэтому не нужно определять ее параметры. Наоборот, используйте в коде функции ключевое слово this для определения свойств объекта. Вот как выглядит функция PrintCard:

```
function PrintCard() {
  line1="Имя: " + this.name + "<BR>\n";
  line2="Адрес: " + this.address + "<BR>\n";
  line3="Тел.(р): " + this.workphohe + "<BR>\n";
  line4="Тел.(д): " + this.homephone + "<BR>\n";
  document.write(line1,line2,line3,line4);
}
```

Эта функция запрашивает значения свойств объекта из базы данных и выводит каждое новое значение в отдельной строке.

Теперь у вас есть функция, которая позволяет отображать сведения о человеке (объекте), но она еще не определена как метод объекта Card. Все, что вам необходимо сделать, — это добавить описание функции PrintCard в определение объекта Card. Это реализуется следующим образом:

```
function Card(name,address,work,home) {
  this.name = name;
  this.address = address;
  this.workphohe = work;
  this.homephone = home;
  this.PrintCard = PrintCard;
}
```

Добавленный оператор выглядит подобно определению свойства объекта, но ссылается он на функцию PrintCard. Пока функция PrintCard будет определена в сценарии, метод PrintCard выполняется для объектов Card.

Создание экземпляра объекта

Настало время использовать на практике созданное определение объекта и его метод. Для того чтобы применить определение объекта на практике, необходимо создать новый объект. Это выполняется с помощью ключевого слова new. Это слово уже использовалось вами при создании объектов Date и Array.

Следующий оператор позволяет создать новый объект Card, имеющий название tom:

```
tom=new Card("Том Джонс", "123 Elm Street", "555-1234", ""555-9876");
```

Как вы видите, создать новый объект очень просто. Все, что необходимо сделать, — это вызвать функцию Card и определить ее атрибуты в том же порядке, что и при определении объекта.

После выполнения этого оператора создается новый объект, сохраняющий информацию о Томе Джонсе. Этот объект называется *экземпляром* объекта Card. Подобно существованию нескольких строк в программе, объект может иметь несколько экземпляров.

Вместо того чтобы определять всю необходимую информацию о Томе Джонсе с помощью ключевого слова new, вы можете задать ее после создания экземпляра объекта. В листинге 11.1 приведен пример создания пустого экземпляра объекта Card и последующего определения его свойств.

http://ITBookz.ru/

Листинг 11.1. Создание объекта и определение его свойств

```
holmes = new Card();
holmes.name = "Шерлок Холмс";
holmes.address = "221B Baker Street";
holmes.workphone = "555-2345";
holmes.homephone = "555-3456";
```

После создания экземпляра объекта, одним из приведенных выше способов, вы можете отобразить необходимые сведения с помощью метода PrintCard. Следующий оператор, например, отображает сведения, сохраненные в объекте tom:

```
tom.PrintCard();
```

Настройка встроенных объектов

JavaScript оснащен средствами, позволяющими расширять возможности встроенных объектов. Например, если вы считаете, что объект String не соответствует всем вашим запросам, расширьте его, добавив новые свойства и методы. Этот прием особенно часто используется при создании больших сценариев с многочисленными строковыми переменными.

Добавление свойств и методов в уже существующий встроенный объект проводится с помощью ключевого слова; prototype. (Другими словами, создается *прототип* уже существующего объекта. prototype — это, в данном случае, название конструктора объекта.) Ключевое слово prototype позволяет вам изменить определение объекта вне программного кода функции конструктора.

В качестве примера давайте добавим в объект String один метод. Создадим метод heading, преобразующий строку в заголовок HTML. Для начала определим строковую переменную:

```
title = "Начальная страница Фреда";
```

Следующий оператор позволяет отображать значение строковой переменной title в виде заголовка HTML первого уровня:

```
document.write(title.heading(1));
```

Листинг 11.2 содержит программный код добавления метода heading в объект String и отображения содержимого объекта в окне броузера.

Листинг 11.2. Добавление метода в объект String

```
<HTML>
<HEAD><TITLE>Добавление метода</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript1.1">
function addhead(level) {
    html = "H" + level;
    text = this.toString();
    start = "<" + html + ">";
    stop = "</" + html + ">";
return start + text + stop;
}
String.prototype.heading = addhead;
```


http://ITBookz.ru/

```
14:         document.write("Это тест".heading(1));
15:         </SCRIPT>
16:         </BODY>
17:         </HTML>
```

В этом коде сначала определяется функция addhead(), выступающая в роли метода объекта String. Она в качестве параметра использует только номер уровня заголовка. Переменные start и stop используются для задания пары дескрипторов <HTML> (открывающего и закрывающего).

После определения функция назначается методом объекта String с помощью ключевого слова prototype. Впоследствии этот метод можно использовать вместе с объектом String, наряду с другими его свойствами и методами. Это проиллюстрировано в последних строках сценария посредством отображения содержимого объекта String в виде заголовка HTML первого уровня.

Сохранение данных в объектах

Вы уже создали новый объект, позволяющий сохранять в базе данных контактную информацию о сотрудниках фирмы. В качестве последнего примера применения объектов, их свойств, функций и методов используем созданный объект для отображения на Web-странице сведений о нескольких сотрудниках.

В документе HTML необходимо обязательно включить определение функции PrintCard, а также функцию конструктора объекта Card. Давайте создадим еще три экземпляра объекта Card и выведем их содержимое на Web-страницу. Листинг 11.3 содержит полный программный код этого документа.

Листинг 11.3. Документ HTML, использующий объект Card

```
1:          <HTML>
2:          <HEAD>
3:          <TITLE>Личные карточки</TITLE>
4:          <SCRIPT LANGUAGE="JavaScript">
5:          function PrintCard() {
6:          line1="<B>Имя: </B>" + this.name + "<BR>\n";
7:          line2="<B>Адрес: </B>" + this.address + "<BR>\n";
8:          line3="<B>Тел.(р): </B>" + this.workphohe + "<BR>\n";
9:          line4="<B>Тел.(д): </B>" + this.homephone + "<BR>\n";
10:         document.write(line1,line2,line3,line4);
11:         }
12:         function Card(name,address,work,home) {
13:              this.name = name;
14:              this.address = address;
15:              this.workphohe = work;
16:             this.homephone = home;
17:              this.PrintCard = PrintCard;
18:         }
19:         </SCRIPT>
20:         </HEAD>
21:         <BODY>
22:         <H1>Личные карточки</H1>
23:         Сценарий начинается здесь.<HR>
24:         <SCRIPT LANGUAGE="JavaScript">
```


http://ITBookz.ru/

```
25:         //Создайте объекты
26:         sue=new Card( "Сью Сазерс", "123 Elm Street", "555-1234", "555-9876");
27:         phred=new Card ("ФредМадсен", "233 Oak Lane", "555-2222", "555-4444");
28:         henry=new Card( "Генри Тилмен", "233 Walnut Circle", "555-1299",
                "555-1344");
29:         //Отобразите их
30:         sue.PrintCard();
31:         phred.PrintCard();
32:         henry.PrintCard();
33:         </SCRIPT>
34:         <HR>Конец сценария
35:         </BODY>
36:         </HTML>
```

Заметьте, что функция PrintCard() немного изменена, чтобы улучшить внешний вид отображаемых данных. Результат выполнения сценария и загрузки Web-страницы из листинга 11.3 показан на рис. 11.1.



Этот пример позволяет управлять только крохотной базой данных, поскольку для каждого сотрудника необходимо вводить свои данные. Тем не менее, объект Card можно использовать для управления записями базы данных, сохраненной на сервер. В этом случае выполняемая работа может оказаться просто неоценимой.

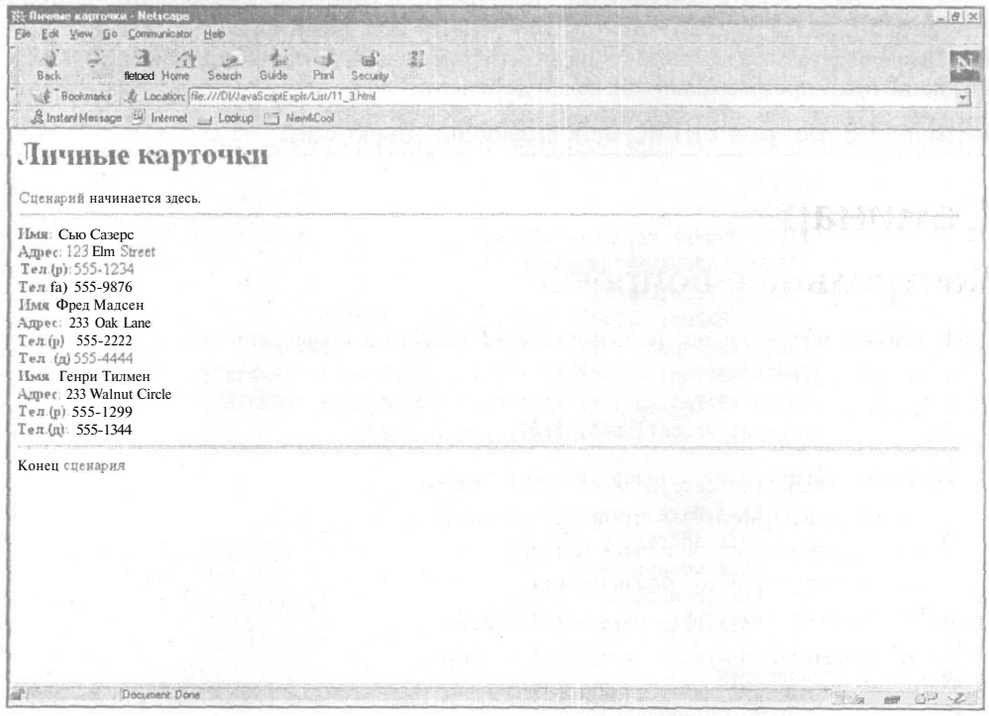


Рис. 11.1. Netscape Navigator отображает крохотную базу данных сотрудников

http://ITBookz.ru/

Резюме

За этот час вы научились создавать в JavaScript собственные объекты. Вы узнали, как создавать определения объектов, функций, добавлять в объект свойства и методы, создавать экземпляры объектов и управлять ими. Кроме того, вы изменили встроенный объект JavaScript.

Это предпоследний урок части III. В последней главе этой части вы познакомитесь с обработчиками событий, позволяющими сценарию реагировать на проводимые пользователем действия, — щелчки мышью, нажатия клавиш и т.п.

Вопросы и ответы

В главе 10 были описаны родительские и дочерние объекты. Можно ли включить дочерние объекты в созданный мною объект?

Да. Создайте функцию конструктора этого объекта и добавьте соответствующее свойство в родительский объект. Например, если вы создали объект Nicknames, содержащий псевдонимы сотрудников, и хотите сделать его дочерним по отношению к объекту Card, тогда в конструкторе объекта Card добавьте следующую строку: `this.nick = new Nicknames();`

Можно ли создать массив пользовательских объектов?

Да. Сначала создайте конструктор объекта, а затем определите массив с указанным количеством элементов. С помощью цикла создавайте новые объекты и присваивайте их новым элементам массива (например, `cardarray[1]= new Card`).

Какие броузеры поддерживают пользовательские объекты?

Приведенные в этом уроке примеры выполняются в JavaScript 1.1. Поэтому вам необходимо использовать Netscape Navigator 3.0 или Internet Explorer 4.0.

Семинар

Контрольные вопросы

1. Какое ключевое слово используется в JavaScript для создания экземпляра объекта?
   a) object
   b) new
   c) instance
2. Каково назначение ключевого слова this?
   a) Определение параметров нового объекта
   b) Определение текущего сценария
   c) Нет однозначного назначения
3. Что позволяет сделать оператор prototype?
   a) Изменить синтаксис команд JavaScript
   b) Изменить встроенные объекты
   c) Изменить броузер пользователя, чтобы он мог использовать только пользовательские объекты

http://ITBookz.ru/

Ответы

1, b) Ключевое слово new позволяет создавать экземпляр объекта

2, a) Ключевое слово this соответствует текущему объекту

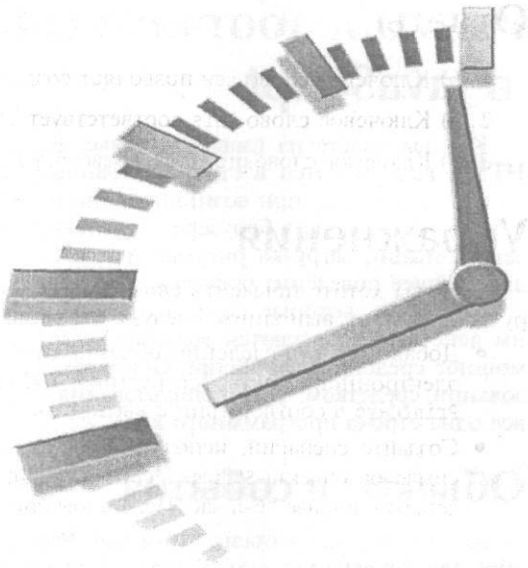
3, b) Ключевое слово prototype позволяет изменить определение встроенного объекта

Упражнения

Если вы хотите повысить свои знания об использовании создаваемых собственноручно объектов, выполните следующие упражнения.

- Добавьте в определение объекта Card новое свойство email, содержащее адрес электронной почты сотрудника. Измените также и определение функции PrintCard в соответствии с введенным в объект новшеством.
- Создайте сценарий, использующий метод first5, который возвращает первых пять символов объекта String. Используйте при его создании метод substring, описанный детально в главе "6-й час. Использование массивов и строковых данных"

http://ITBookz.ru/



12-й час

Обработка событий

Мои поздравления. Вы уже достигли середины книги. Это двенадцатый час из 24-х. Начиная с этого момента, вы должны знать все элементы, которые будут вам встречаться в листингах.

В этой главе вы познакомитесь с самыми различными обработчиками событий, поддерживаемыми в Java Script. Вместо того чтобы задавать сложные условия выполнения тех или иных действий, вы можете воспользоваться в своих сценариях специальными элементами (обработчиками событий), позволяющими тесно взаимодействовать программе с пользователем. Практически все следующие главы этой книги будут содержать в своих листингах программный код обработчиков событий.

В этом уроке рассмотрены следующие темы.

- **Роль обработчика событий в JavaScript**
- **Связь обработчика событий с объектом**
- **Создание обработчика событий**
- **Тестирование обработчика событий**
- **Выполнение действий с помощью мыши**
- **Выполнение действий с помощью клавиатуры**
- **События и специальные обработчики**
- **Добавление на Web-страницу описаний ссылок**

http://ITBookz.ru/

Роль обработчика событий в JavaScript

Как вы знаете из главы "4-й час. Выполнение программ JavaScript", программы HTML выполняются в строго указанном порядке. Порядок выполнения программы изменяется только при возникновении определенного *события*. События — это действия, происходящие в броузере — щелчки мышью, нажатия клавиш, перемещение указателя мыши, зафузка рисунка и т.п. Самые различные события могут повлиять на дальнейшее поведение броузера.

Сценарии, которые позволяют определять события и выполнять соответствующие им действия, называются *обработчиками событий*. Обработчик событий — это самое мощное средство в JavaScript. Одновременно он является и самым простым в использовании средством. Для использования обработчика событий, как правило, задается все одна строка программного кода.

Объекты и события

Как вы знаете из главы "10-й час. Работа с объектной моделью документа", JavaScript для сохранения самых разных типов данных использует объекты. Объекты используются и для управления различными элементами броузера: кнопками, ссылками, рисунками, окнами и т.п. Событие может произойти в любом из элементов броузера и документа HTML. Например, пользователь может щелкнуть на одной из нескольких ссылок на странице, связанных со своим объектом.

Каждое событие имеет собственное имя. Например, событие onMouseOver происходит при наведении указателя мыши на объект Web-страницы. При наведении указателя на соответствующую ссылку событие onMouseOver активизируется и запускается обработчик этого события.

Чтобы определить обработчик событий, в начало имени события добавляется ключевое слово on. Например, обработчик событий onMouseOver вызывается при наведении указателя мыши на ссылку. Для определения обработчика событий добавьте его код после кода ссылки, пометив дескриптором **<A>.**



Запомните регистр символов *в* обработчике событий onMouseOver. Это стандартный вид записи имени обработчика событий. Ключевое слово on всегда вводится строчными символами, а каждое слово в названии обработчика события начинается с прописной буквы.

Создание обработчика событий

Для определения события не нужно использовать дескриптор <SCRIPT>. Наоборот, обработчик событий вставляется в дескриптор HTML. Вот каким образом задается обработчик событий onMouseOver:

```
<AHREF="http://jsworkshop.com/"
  onMouseOver="window.alert('Вы указываете на ссылку');">
Щелкните здесь</A>
```

Обратите внимание на то, что один дескриптор **<A>** разбит на несколько строк. Это вызвано определением обработчика событий onMouseOver для ссылки. Этот оператор позволяет выводить сообщение на экран при наведении указателя мыши на ссылку.

http://ITBookz.ru/



В предыдущем примере для выделения текста сообщения использованы одинарные кавычки. Не думайте, что это ошибка. Одинарные кавычки используются потому, что двойными кавычками выделен непосредственно обработчик событий. (Можно поступить и наоборот, выделить двойными кавычками текст, а одинарными — обработчик событий.)

Подобный оператор используется для задания одного обработчика событий. Но если необходимо задать более чем один оператор, тогда лучше создать дополнительную функцию. Определите функцию в заголовке документа, а затем вызовите ее вместо обработчика событий:

```
<A HREF="#bottom" onMouseOver="Dolt();">Наведите указатель на ссылку</A>
```

В этом примере при наведении пользователем указателя на ссылку вместо обработчика событий вызывается функция Dolt(). Использование функции предпочтительнее, поскольку в ней вы можете задавать более сложные операторы, нежели в обработчике событий. Пример использования подобной функции приведен в разделе "Добавление описания ссылки" этой главы.



В простых обработчиках событий можно использовать два оператора, разделив их точкой с запятой. Тем не менее, в большинстве случаев проще использовать функцию.

Обработчики событий в JavaScript

Вместо того чтобы определять обработчик событий в документе HTML, используйте в качестве обработчика событий функцию JavaScript. Это позволит вам задавать обработчик событий при выполнении определенного условия, т.е. подключать и отключать его динамически.

Чтобы определить обработчик событий таким образом, сначала определите функцию, а затем назначьте ее в качестве обработчика событий. Обработчик событий может сохраняться в виде свойства объекта document или другого объекта, который поддерживает события. В приведенном ниже примере определяется функция mousealert, а затем назначается как обработчик событий onMouseDown:

```
function mousealert() {
  alert("Вы щелкнули кнопкой!");
}
document.onMouseDown = mousealert;
```

Использование объекта event

Объект event используется в JavaScript версии 1.2 и выше. Это специальный объект, который отправляется в обработчик событий при возникновении любого события. Обработчик события получает этот объект в виде параметра. Свойства объекта event содержат данные о событии, которое произошло. Ниже приведен список всех свойств объекта event.

- type. Это тип произошедшего события, например mouseover.
- target. Это целевой объект события (например, документ или ссылка).
- which. Это числовое значение, определяющее номер кнопки мыши или клавишу клавиатуры, которая была нажата.
- modifiers. Это список модифицирующих клавиш, используемых для возбуждения события (например <Alt>, <Shift> или <Ctrl>).

http://ITBookz.ru/

- data. Это список перемещаемых данных в событиях "перетащи и опусти".
- pageX и pageY. Это координаты указателя мыши вдоль оси X и Y соответственно. Начало координат находится в левом верхнем углу страницы.
- layerX и layerY. Это координаты указателя мыши вдоль оси X и Y соответственно. Начало координат находится в левом верхнем углу слоя. (Слои Web-страницы детально рассмотрены в главе "18-й час. Создание динамических страниц с помощью DOM".)
- screenX и screenY. Это координаты указателя мыши вдоль оси X и Y соответственно. Начало координат находится в левом верхнем углу экрана.

События, связанные с мышью

JavaScript содержит целый набор обработчиков событий, реагирующих на поведение мыши. Они позволяют определять координаты указателя мыши, а также нажатия и отпускания обеих ее кнопок.

В и Из

Вы уже познакомились с первым и, наверное, самым используемым обработчиком событий — onMouseOver. Этот обработчик событий вызывается при наведении указателя мыши на ссылку, рисунок или другой объект Web-страницы.

OnMouseOut — это противоположный обработчик. Он вызывается при удалении указателя мыши из необходимой области. При вызове этого обработчика событий всегда выполняются определенные действия (иначе зачем тогда его использовать).

Этот обработчик часто используется тогда, когда сценарий предусматривает добавление на Web-страницу ссылок и других объектов, на которые пользователь обязательно наводит указатель мыши. С помощью обработчика onMouseOver вы можете, при наведении указателя на объект, выполнять дополнительные действия — отображать сообщения в строке состояния или загружать рисунок. При необходимости отмените (с помощью обработчика onMouseOut) все проведенные вами действия после перемещения указателя с указанного объекта.

Оба обработчика событий onMouseOver и onMouseOut детально описаны в разделе "Добавление описания ссылки" этого урока.



Чаще всего обработчики событий onMouseOver и onMouseOut используются для создания изменяющихся при наведении на них указателей рисунков. Детально о них будет рассказано в главе "15-й час. Добавление рисунков и анимации".

Третий обработчик событий onMouseMove вызывается при произвольном перемещении указателя мыши. Поскольку пользователь постоянно перемещается указатель мыши, по умолчанию этот обработчик не активен. Чтобы активизировать его, используйте метод захвата события, описанный дальше в этой главе.

Щелчки и отпускания

Обработчики событий также позволяют определять нажатия и отпускания кнопок мыши. Основной обработчик событий, который используется для этого. — onclick. Этот обработчик вызывается после щелчка мышью при наведенном указателе мыши на определенный объект.

http://ITBookz.ru/



Объект в этом случае выступает в роли ссылки. Этот может быть и элемент формы. Подробно с формами вы познакомитесь в главе "14-й час. Формы введения данных".

Например, следующий обработчик событий используется для отображения предупреждения после щелчка на ссылке:

```
<A HREF="http://www.jsworkshop.com"
onClick="alert('Вы покидаете этот Web-узел');">Щелкните здесь</A>
```

В этом случае обработчик событий onclick запускается до отображения загружаемой страницы в окне броузера. Этот метод полезно использовать для отображения описания страницы перед ее отображением или задания условных ссылок.

Если обработчик событий onClick возвращает значение false, то страница, на которую указывает ссылка, загружаться не будет. Ниже приведен пример задания условной ссылки, которая позволяет загружать страницу только после выполнения дополнительных действий. Если вы щелкнете на кнопке Cancel (Отмена), страница загружена не будет; если щелкнуть на кнопке OK, страница загрузится так же, как и при использовании обычной ссылки:

```
<A HREF="http://www.jsworkshop.com"
onClick="return(window.confirm('Вы уверены?'));">Щелкните здесь</A>
```

В этом примере используется функция return, определяющая поведение обработчика событий. Она возвращает значение false после щелчка на кнопке Cancel. В этом случае документ ссылки загружен не будет.

Обработчик событий onDblClick выполняется подобным образом. Единственная разница состоит в том, что анализируется не одиночный щелчок на объекте, а двойной. Поскольку на ссылке обычно выполняется одинарный щелчок, вы можете использовать один объект для задания двух разных ссылок. Одна ссылка будет активизироваться при двойном щелчке на объекте, а вторая — при одинарном. (Честно говоря этот метод не эффективен из-за частых ошибок пользователей.) Двойной щелчок регистрируется не только на ссылке, но и на рисунке и другом объекте документа.

Для того чтобы повысить управляемость сценария с помощью мыши, в распоряжение разработчика предоставляется еще два, не рассмотренных выше, обработчика.

- onMouseDown. Используется для определения нажатия пользователем кнопки мыши.
- onMouseUp. Используется для определения отпускания нажатой кнопки мыши.

Эти два обработчика событий составляют один обработчик событий щелчка на объекте. Если вы хотите определить щелчок, то используйте onclick; если же только нажатие кнопки или ее отпускание — onMouseDown или onMouseUp.

Для определения используемой кнопки мыши объект событий имеет свойство which. Для левой кнопки мыши это свойство равно 1, а для правой — 3. Это свойство определяется как для onClick, onDblClick, так и для onMouseDown и onMouseUp.

В приведенном ниже примере создан обработчик событий onMouseDown, отображающий разные предупреждения при нажатии разных кнопок мыши:

```
function mousealert(e) {
whichone = (e.which == 1) ? "левой" : "правой";
message = "Вы щелкнули на "+whichone+" кнопке";
alert(message);
}
document.onmousedown = mousealert;
```


http://ITBookz.ru/

События, связанные с клавишами

До выпуска Netscape Navigator 4.0 язык JavaScript не имел средств определения нажатия клавиши на клавиатуре. Это не позволяло создавать много полезных программ. Например, сложно было создавать игры, не без возможности использовать клавиши со стрелками.

Именно поэтому последние версии JavaScript позволяют определять нажатие клавиш. Основной обработчик событий, который используется для этого, — onKeyPress. Он вызывается при нажатии и отпускании клавиши. Как и в случае с кнопками мыши, нажатие и отпускание клавиш мыши отдельно определяется обработчиками onKeyDown и onKeyUp.

Конечно, необходимо знать, какая клавиша нажимается. Это определяется с помощью объекта event. Свойство event.which точно указывает на нажимаемую клавишу — оно принимает значение кода ASCII этой клавиши.



ASCII (American Standard Code for Information Interchange — Американский стандартный код обмена информацией) — это стандартный числовой код, используемый для отображения символов клавиатуры. Этот код содержит значения в диапазоне 0-128, соответствующие всем символам клавиатуры. Например, буква A имеет код 65, а Z — 90.

Если вы предпочитаете использовать реальные символы, то преобразуйте код с помощью метода fromCharCode в реальные символы. Этот метод преобразует код ASCII в соответствующие символы клавиш. Например, следующий дескриптор <BODY> содержит обработчик событий, отображающий символ клавиши, нажимаемой пользователем:

```
<BODY onKeyPress="window.alert('Вы нажали: '+String.fromCharCode(event.which));">
```

Этот оператор использует метод String.fromCharCode для преобразования свойства event.which в строковое значение, которое отображается как часть сообщения.

Событие onLoad

Еще одно событие, на которое нельзя не обратить внимание, — это onLoad. Это событие происходит после окончания загрузки Web-страницы (включая и все рисунки) с сервера.

Событие onLoad связано с объектом document. Чтобы определить его соответствующий обработчик события задается в дескрипторе <BODY>. Следующий дескриптор <BODY> использует простой обработчик событий для отображения сообщения по окончании загрузки страницы:

```
<BODY onLoad="alert('Загрузка завершена');">
```



Поскольку событие onLoad происходит после загрузки и отображения документа HTML, в его обработчике нельзя использовать операторы document.write и document.open. Если их использовать, то текущий документ будет удален и замещен другим.

JavaScript имеет много других событий. Многие из них используются при создании форм ввода данных (детально о них мы поговорим в главе "14-й час. Формы введения данных"). Еще одно событие, которое вам встретится в этой книге, — это onError. Оно позволяет предотвратить отображение на экране сообщение об ошибке. Детально о нем будет рассказано в главе "21-й час. Отладка приложений JavaScript".

http://ITBookz.ru/

Добавление описания ссылки

Чаще всего обработчики событий используются для отображения в строке состояния описания ссылки при наведении на нее указателя мыши. Например, наведение указателя мыши на ссылку на форме заказа товаров может привести к отображению следующего сообщения "Закажите товар или проверьте его наличие".

Описания, отображаемые в строке состояния, обычно отображаются с помощью обработчика событий onMouseOver. Давайте создадим сценарий, отображающий подобные сообщения. Удаление сообщения будет проводиться при снятии указателя мыши со ссылки (используем для этого обработчик onMouseOut). Для упрощения сценария используем функцию.



При добавлении сообщений, созданных описанным выше образом, они замещают в строке состояния адрес URL. Поэтому убедитесь, что ваши сообщения больше полезны для пользователя, нежели адрес URL. Как правило, Web-дизайнеры используют этот метод для отображения многословных сообщений. Например, ссылка на форму заказа товаров может иметь сообщение "Переход к форме заказа необходимых товаров и услуг".

В самом начале сценария определим функцию, отображающую сообщение в строке состояния. Хотя функцию для этого использовать и не обязательно, она значительно упростит программный код. Например, ссылка с обработчиком событий, отображающим сообщение в строке состояния, выглядит следующим образом:

```
<A HREF="order.html"
  onMouseOver="window.status='Заказ товаров'; return true;">
Order Form</A>
```

В этом примере оператор return true используется для предотвращения замещения создаваемого сообщения адресом URL. Как вы видите, дескриптор <A> при таком задании сообщения достаточно сложен — ввести сообщение и то не хватает места.

Использование функции немного упрощает программный код. Но самое главное, это позволяет добавлять в будущем другие объекты (например, рисунки). В нашем примере функция будет вызываться для отображения сообщения describe (листинг 12.1).

Листинг 12.1. Определение функции describe

```
1:    <SCRIPT LANGUAGE="JavaScript">
2:    function describe(text) {
3:         window.status = text;
4:         return true;
5:    }
6:    </SCRIPT>
```

Эта функция имеет всего один параметр text. Содержимое переменной отображается в строке состояния. Поскольку функция возвращает значение true, строка состояния продолжает отображать сообщение вместо адреса URL. Чтобы удалить сообщение, необходимо создать простую функцию, вызываемую обработчиком событий onMouseOut, показанную в листинге 12.2.

Листинг 12.2. Функция очистки строки состояния

```
1:    function clearstatus() {
2:         window.status="";
3:    }
```


http://ITBookz.ru/

И конечно, документ HTML должен содержать ссылки с соответствующими обработчиками событий, позволяющими вызывать приведенные выше функции. Листинг 12.3 содержит программный код документа HTML с тремя типичными ссылками.

Листинг 12.3. Пример страницы с тремя ссылками

```
<HTML>
<HEAD>
<TITLE>Описание ссылок</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function describe(text) {
     window.status = text;
     return true;
}
function clearstatus() {
window.status="";
}
</SCRIPT>
</HEAD>
<BODY>
<H1>Описание ссылок</H1>
<P>Наведите указатель на ссылки, чтобы
отобразить их описания</P>
<UL>
<LI><A HREF="order.html"
     onMouseOver="describe('Закажите товар'); return true;"
     onMouseOut="clearstatus()";>
Заказ товаров</A>
<LI><A HREF="email.html"
     onMouseOver="describe('Отправь письмо'); return true;"
     onMouseOut="clearstatus()";>
Email</A>
<LI><A HREF="complain.html"
     onMouseOver="describe( 'Критикуйте все'); return true;"
     onMouseOut="clearstatus()" ;>
Отдел защиты прав потребителей</A>
</UL>
</BODY>
</HTML>
```

В этом примере функции создаются в заголовке документа. Каждая ссылка имеет обработчики событий onMouseOver и OnMouseOut, вызывающих две определенные раньше функции.

Чтобы протестировать сценарий, загрузите листинг 12.3 в броузер. При этом в его окне должен отобразиться документ, подобный изображенному на рис. 12.1.

http://ITBookz.ru/

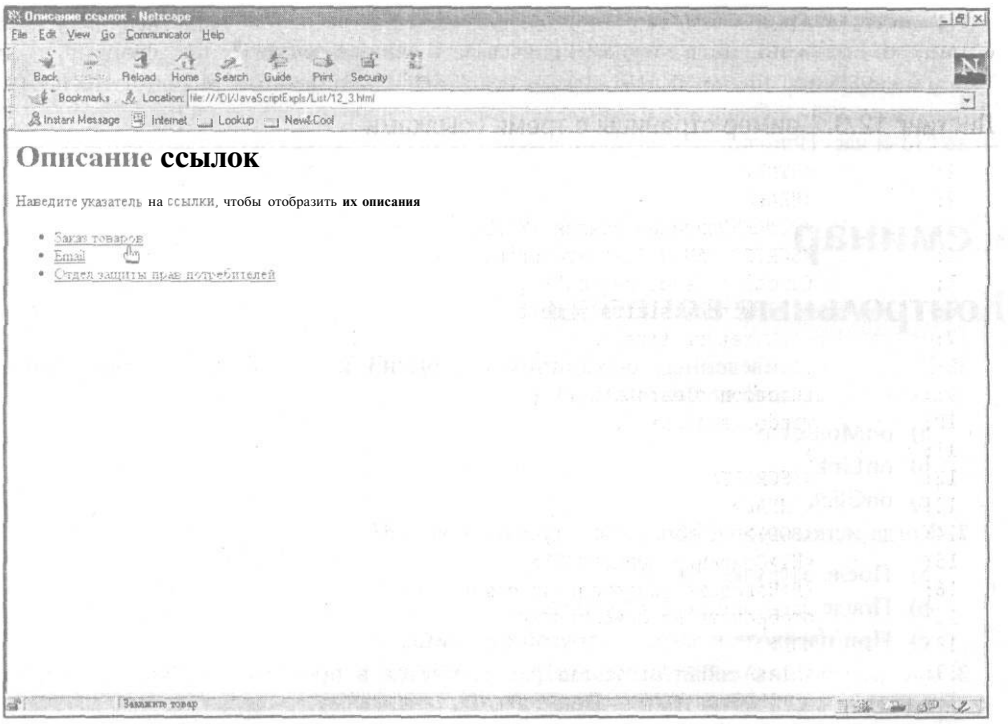


Рис. ***12.1. Пример добавления описания ссылок на Web-страницу***

Резюме

За этот час вы научились использовать события для определения поведения мыши, клавиатуры и других объектов. Теперь вы умеете использовать обработчики событий для выполнения самых разных операций при возникновении событий.

Примите мои поздравления. Вы уже изучили половину книги. В следующих нескольких уроках вы познакомитесь с некоторыми специальными методами улучшения Web-страниц. Вы узнаете, как создавать фреймы, формы, рисунки и т.п.

Вопросы и ответы

Я заметил, что дескриптор <IMG> не может содержать обработчики событий `onMouseOver` и onclick. Разве нельзя сделать так, чтобы сценарий реагировал на наведение указателя мыши на рисунок?

Самый простой способ выполнить это — сделать рисунок ссылкой, пометив его в коде дескриптором <A>. Чтобы предотвратить выделение рисунка синей рамкой, используйте атрибут `BORDER=0`. Детально о рисунках мы поговорим в главе 15.

Что произойдет, если определить оба обработчика событий `onKeyPress` и `onKeyDown`? Будут ли они вызываться оба при нажатии клавиши?

Первым вызывается onKeyDown. Если он возвращает значение true, то вызывается обработчик onKeyPress. В противном случае событие нажатия клавиши не генерируется.

http://ITBookz.ru/

При использовании события onLoad **обработчик событий иногда запускался до полной загрузки страницы, перед загрузкой рисунков. С чем это связано?**

Эту недоработку имеют старые версии JavaScript. Единственный выход — это переделать сценарий, добавив в него метод setTimeout. Подробно этот метод рассмотрен в главе "13-й час. Использование окон и фреймов".

Семинар

Контрольные вопросы

1. Какой из приведенных обработчиков событий используется для регистрации щелчков мышью на ссылке?
   a) onMouseUp
   b) onLink
   c) onClick
2. Когда используется обработчик событий onLoad?
   a) После загрузки рисунка
   b) После загрузки всей страницы
   c) При переходе к загрузке другой страницы
3. Где обработчик событий onLoad располагается в программном коде документа HTML?
   a) В теле сценария
   b) В теле документа HTML
   c) В самом конце документа

Ответы

1, c) Обработчик события щелчка мышью называется onClick

2, b) Обработчик onLoad выполняется после того, как вся страница полностью загружена, включая и рисунки

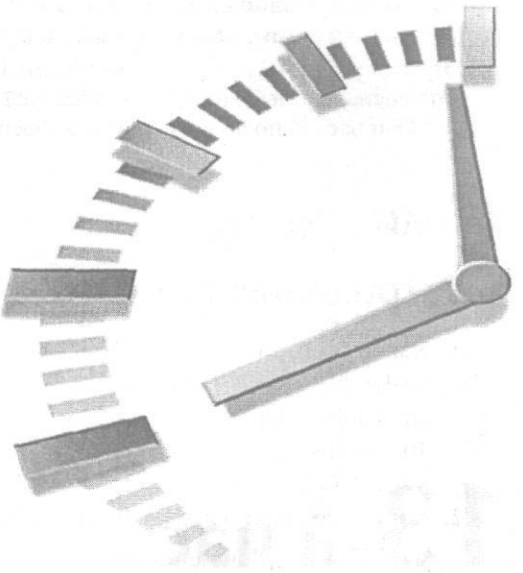
3, b) Обработчик onLoad располагается в теле кода документа HTML

Упражнения

Чтобы повысить свой опыт в использовании обработчиков событий, выполните следующие упражнения.

- Добавьте несколько ссылок в документ HTML листинга 12.3. Добавьте к ним обработчик событий, позволяющий отображать их описание.
- Измените код листинга 12.3 так, чтобы в строке состояния, в случае неотображения описания ссылки, отображалось приветственное сообщение. (Подсказка: добавьте оператор отображения этого сообщения при загрузке страницы. Для очистки строки состояния от других сообщений используйте функцию `clearstatus`.)

http://ITBookz.ru/



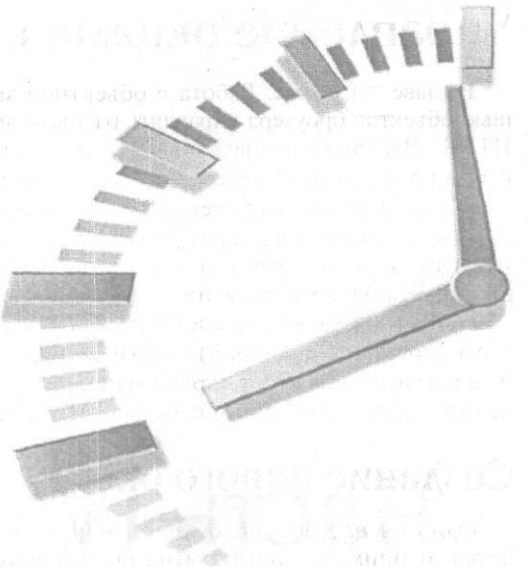
Часть IV

Управление Web-страницами

Темы занятий

13. Использование окон и фреймов
14. Формы введения данных
15. Добавление рисунков и анимации
16. Создание сценариев для разных броузеров

http://ITBookz.ru/



13-й час

Использование окон и фреймов

Рад приветствовать вас в части IV этой книги. Вы уже познакомились с половиной материала, изложенного в книге, и вашего багажа знаний достаточно для выполнения простых задач с помощью JavaScript. В течение следующих трех часов вы познакомитесь с некоторыми дополнительными методами управления элементами Web-страниц и броузеров с помощью JavaScript.

За этот час вы детальнее познакомитесь с окном броузера и фреймами. Вы также научитесь управлять ими в своих сценариях. В этой главе рассмотрены следующие темы.

- **Иерархия объектов объекта** window
- **Создание нового окна в JavaScript**
- **Выполнение операций с задержкой**
- **Отображение предупреждений, подтверждений и запросов**
- **Управление фреймами с помощью JavaScript**
- **Создание навигационного фрейма в JavaScript**

http://ITBookz.ru/

Управление окнами с помощью объектов

В главе "10-й час. Работа с объектной моделью документа" вы узнали, как с помощью объектов броузера управлять разными частями окна броузера и текущего документа HTML. Вы также познакомились с иерархией объектов броузера. Объект window выступает в роли родительского, а document, history и location — в роли дочерних.

На этом занятии вы детально познакомитесь с объектом window. Как вы уже догадались, я покажу вам, как с помощью этого объекта управлять окнами. Следует заметить, что этот же объект позволяет определять фреймы. Об этом также рассказывается во второй половине занятия.

Объект window всегда соответствует текущему открытому окну (окну документа, который содержит сценарий). Текущее окно также определяется ключевым словом self. В действительности на экране одновременно отображается несколько окон и каждое из них управляется своим собственным именем.

Создание нового окна

Одно из назначений объекта окна — это создание нового окна. Оно вам понадобится, например, для открытия нового документа или отображения инструкций к выполняемым действиям. Окна создаются и для выполнения специальных задач (например, навигационные окна).

Новое окно броузера создается с помощью метода window.open(). Типичный оператор создания нового окна имеет следующий вид:

```
WinObj=window.open("URL", "WindowName", "Feature List");
```

В записи этого оператора метода window.open() вам встречаются следующие элементы.

- Переменная WinObj сохраняет данные нового объекта window. Ее имя можно использовать с методами и свойствами созданного объекта window.
- Первый параметр метода window.method — это URL документа, загружаемого в окне. Если его оставить незаполненным, то окно останется пустым.
- Второй параметр определяет название окна (WindowName). Его значение приписывается свойству name нового объекта window.
- Третий параметр представляет список необязательных опций, разделенных запятой. С их помощью вы определяете вид нового окна: наличие в нем панелей инструментов, строки состояния и других элементов. Они позволяют вам создавать различные "плавающие" окна, отличающиеся от стандартных окон броузера.

В качестве примера опций третьего параметра метода window.open() можно привести width и height, определяющие размеры окна, а также toolbar, location, directories, status и menubar, принимающие одно из двух значений — да (1) или нет (0). Указывайте только те опции, значения которых необходимо указывать; неуказанные опции принимают значения по умолчанию. Ниже приведен пример создания нового окна, не содержащего панели инструментов и строки состояния:

```
SmallWin =
window.open("", "small", "width=100, heigth=120, toolbar=0,status=0");
```



Вы можете управлять текущим окном броузера с помощью готовых сценариев, которые свободного распространяются в Web. В Приложении А приведены Web-адреса ресурсов, содержащих сценарии, доступные для свободного использования, а также полезные программные коды.

http://ITBookz.ru/

Открытие и закрытие окон

Окно создается именно для того, чтобы его закрывали и открывали. Метод window.close() закрывает окно. Без пользовательских полномочий вы не можете закрыть основное окно Netscape Navigator. Главное предназначение этого метода — это закрытие созданных вами окон. Например, следующий оператор закрывает окно с именем updatewindow:

```
updatewindow.close();
```

В другом примере (листинг 13.1) представлен документ HTML, позволяющий открывать новое окно после щелчка на кнопке. (Появляется окно маленького размера. Это позволяет отличить его от основного окна броузера.) После щелчка на второй кнопке окно исчезает. Третья кнопка используется для закрытия текущего (основного) окна. Netscape Navigator позволяет это делать, но выводит на экран запрос на подтверждение операции закрытия окна.

Листинг 13.1. Документ HTML, позволяющий открывать и закрывать окна с помощью щелчков кнопками мыши

```
<HTML>
<HEAD><TITLE>Создание нового окна</TITLE>
</HEAD>
<BODY>
<H1>Создание нового окна</H1>
<HR>
Для открытия и закрытия окна используйте эти кнопки
<HR>
<FORM NAME="winform">
<INPUT TYPE="button" VALUE="Открыть новое окно"
onClick="NewWin=window.open('','NewWin',
'toolbar=no,status=no,width=100,height=200');">
<P><INPUT TYPE="button" VALUE="Закрыть новое окно"
onClick="NewWin.close();">
<P><INPUT TYPE="button" VALUE="Закрыть главное окно"
onClick="window.close();">
</FORM>
<HR>
</BODY>
</HTML>
```

В этой программе активно используются обработчики событий. После щелчка на любой кнопке запускается отдельный обработчик событий. На рис. 13.1 представлена программа из листинга 13.1 в действии. В главном окне есть три кнопки для выполнения описанных выше действий и открытия нового маленького окна.

Временные задержки

Иногда самое тяжелое — это заставить сценарий некоторое время не выполнять никаких действий. JavaScript содержит специальные функции, позволяющие задать временную задержку. Метод window.setTimeout определяет временную задержку перед выполнением указанной команды.

http://ITBookz.ru/



Временная задержка не прекращает выполнение сценария. Она только определяет время, через которое будет выполняться следующая после метода `setTimeout()` команда. Во время ожидания выполнения необходимой команды броузер продолжает совершать другие действия, для которых временная задержка не определена.

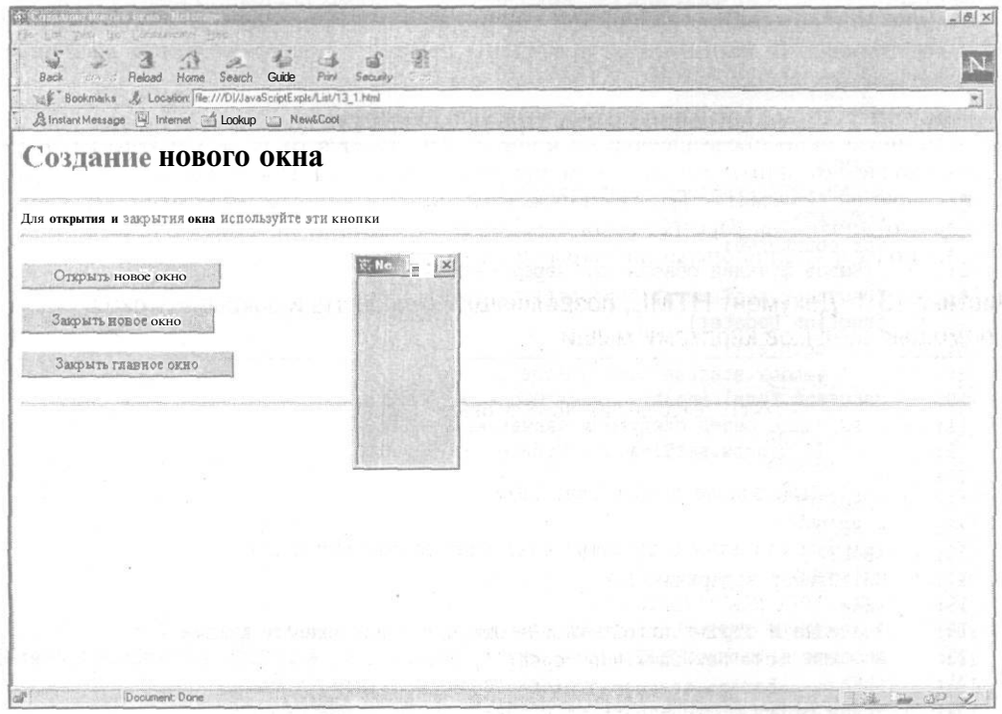


Рис. 13.1. Новое окно, открытое в Netscape Navigator с помощью JavaScript

Временная задержка начинается при вызове метода `setTimeout()`. Он имеет два параметра. Первый — это оператор JavaScript (или группа операторов), заключенный в скобки. Второй параметр — это время задержки в миллисекундах (тысячных долях секунды). В приведенном ниже примере показано, как задать команду отображения предупреждения с задержкой в 10 секунд:

```
ident=window.setTimeout("alert('Ваше время истекло')", 1000);
```



Подобно обработчикам событий, операторы временной задержки используют кавычки для выделения выполняемых операций. Поэтому, при введении текстовых значений в этих операторах, используйте одинарные кавычки (апострофы), как это показано в предыдущем примере.

Переменная (в примере ident) имеет собственное имя и представляет собой идентификатор временной задержки. Это позволяет в одном сценарии использовать несколько разных временных задержек. До истечения указанного времени вы можете с помощью метода `clearTimeout()` прервать задержку. Для этого в скобках укажите идентификатор:

```
window.clearTimeout(ident);
```


http://ITBookz.ru/

Обновление страницы с задержкой

Обычно операция с временной задержкой выполняется однократно, поскольку в операторе `setTimeout()` операторы действия указываются только один раз. Иногда необходимо выполнить одни и те же операции с временной задержкой несколько раз. Например, если ваша страница содержит часы или счетчик, значения которых постоянно изменяются.

Повторение операций с временной задержкой также выполняется с помощью метода `setTimeout()`. В листинге 13.2 приведен сценарий с повторяющимися временными задержками.

Листинг 13.2. Сценарий обновления страницы через каждые две секунды

```
<HTML>
<HEAD><TITLE>Пример задержки</TITLE>
<SCRIPT>
var counter=0;
//Вызов функции обновления через каждые 2 с
ID=window.setTimeout("Update();", 2000);
function Update()    {
     counter++;
     window.status="Отсчет"+counter;
document.form1.input1.value="Отсчет: "+counter;
//задержка перед следующим значением
     ID=window.setTimeout("Update();", 2000);
}
 </SCRIPT>
 </HEAD>
 <BODY>
<H1>Пример задержки</H1>
 <HR>
 Значение в строке состояния и на странице обновляется каждые 2 с
 Щелкните на кнопке RESET для запуска счетчика с нуля, а на STOP для остановки счета.
 <HR>
 <FORM NAME="form1">
<INPUT TYPE="text" NAME="input1" SIZE="40"><BR>
 <INPUT TYPE="button" VALUE="RESET" onClick="counter=0;"><BR>
 <INPUT TYPE="button" VALUE="STOP" onClick="window.clearTimeout(ID);"><BR>
 <HR>
 </BODY>
 </HTML>
```

Эта программа обновляет сообщение в строке состояния и текстовом поле через каждые две секунды. Сообщение содержит значение счетчика. Для повторного запуска счетчика используется кнопка Reset, а для прекращения счета — Stop.

В этом сценарии метод `setTimeout()` первый раз запускается при загрузке страницы. Последующие вызовы метода происходят при каждом обновлении страницы, т.е. через каждые две секунды. Обновление страницы проводится с помощью функции `Update()`.

С ее же помощью к значению счетчика добавляется единица. Кнопка Reset применяется для обнуления значения счетчика, а Stop — для вызова метода `clearTimeout()`. На рис. 13.2 показан результат выполнения кода листинга 13.2.



В этом и следующем примере для управления содержимым страницы используются кнопки. Они выступают в роли элементов форм HTML. Детально о формах мы поговорим *на* следующем занятии.

http://ITBookz.ru/

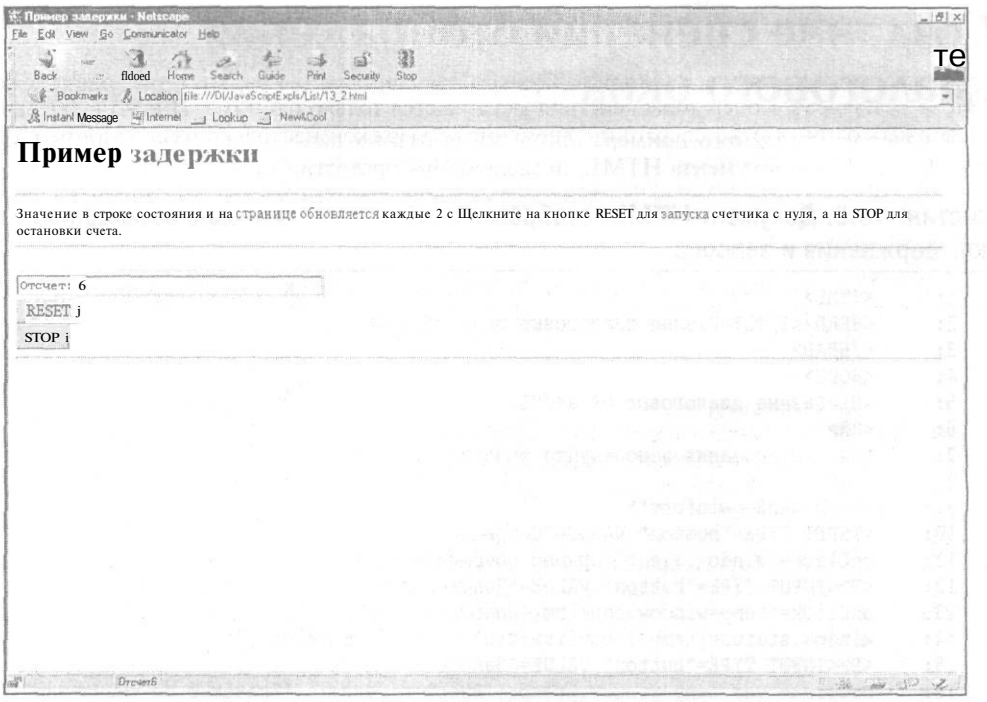


Рис. 13.2. ***Результат периодического обновления страницы, реализуемого с помощью временных задержек***

Отображение диалоговых окон

Объект window использует три метода для отображения сообщений, добавляя в сценарий элементы интерактивности. Вы уже использовали их в рассмотренных сценариях.

- Метод alert отображает диалоговое окно, подобное изображенному на рис. 13.3. В этом диалоговом окне отображается сообщение для пользователя.



Рис. 13.3. Диалоговое окно сообщения, созданное в JavaScript

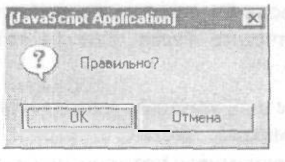


Рис. 13.4. В этом диалоговом окне необходимо подтвердить выполнение операции

- Метод confirm отображает диалоговое окно подтверждения выполнения операции. Оно содержит кнопки OK и Cancel (Отмена), позволяющие выбрать один из вариантов. Диалоговое окно подтверждения показано на рис. 13.4.
- Метод prompt запрашивает у пользователя необходимые для сценария данные. Он возвращает введенный пользователем текст.

http://ITBookz.ru/

Создание сценария отображения диалогового окна

В качестве наглядного примера применения разных диалоговых окон приведен листинг 13.3 с кодом документа HTML, позволяющим протестировать диалоговые окна.

Листинг 13.3. Документ HTML, отображающий диалоговые окна сообщения, подтверждения и запроса

```
<HTML>
<HEAD><TITLE>Разные диалоговые окна</TITLE>
</HEAD>
<BODY>
<H1>Разные диалоговые окна</H1>
<HR>
Для тестирования используйте кнопки
<HR>
<FORM NAME="winform">
<INPUT TYPE="button" VALUE="Сообщение"
onClick="window.alert('Пример сообщения');">
<P><INPUT TYPE="button" VALUE="Подтверждение"
onClick="temp=window.confirm('Правильно?');
window.status=(temp)?'confirm:true': 'confirm:false';">
<P><INPUT TYPE="button" VALUE="Запрос"
onClick="var temp=window.prompt('Введите текст:','Значение по умолчанию');"
window.status=temp;">
</FORM>
<BR>Просто супер!
<HR>
</BODY>
</HTML>
```

На Web-странице отображаются три кнопки, каждая из них запускает свой обработчик события. Давайте детально их рассмотрим.

- **Проще всего вызывается сообщение. Оно отображается после щелчка на кнопке.**
- **После щелчка на второй кнопке в строке состояния отображается возвращаемое значение** (true или false). **Возвращаемое значение присваивается переменой** temp.
- **Третья кнопка используется для вывода диалогового окна запроса данных. Заметьте, что метод** prompt **имеет и второй параметр, определяющий значение по умолчанию, которое указывается в поле. Если в этом диалоговом окне щелкнуть на кнопке Cancel (Отмена), то будет возвращено значение** null.

На рис. 13.5 показан результат выполнения сценария листинга 13.4. Диалоговое окно запроса содержит значение по умолчанию, а строка состояния содержит значение, возвращаемое при вызове предыдущего диалогового окна (подтверждения).

http://ITBookz.ru/

Управление фреймами

Некоторые броузеры (последние версии Netscape Navigator и Internet Explorer) поддерживают фреймы или наборы фреймов, разделяющих окно броузера на отдельные области. Каждый фрейм может иметь отдельный URL или управляющий сценарий.

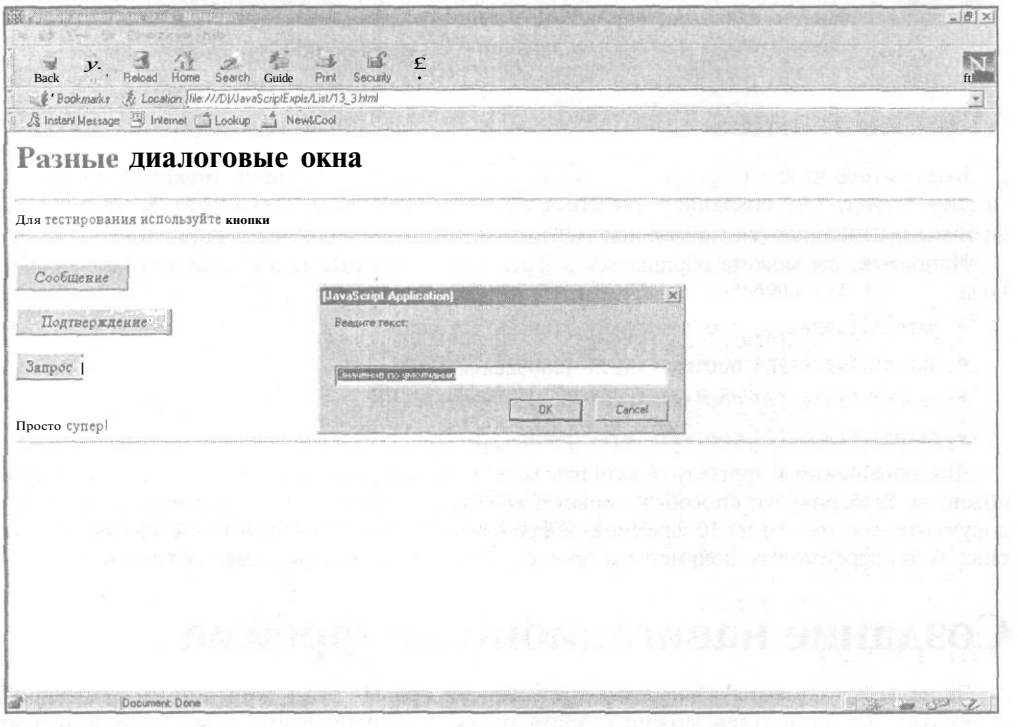


Рис. 13,5. Пример диалогового окна запроса данных

Использование объектов фреймов

При разделении окна на отдельные фреймы управление ими осуществляется с помощью объекта frame. Объект frame используется для управления только текущим фреймом. Этот объект синтаксически равноправен атрибуту NAME в дескрипторе <FRAME>.

Вы еще помните, что оба ключевых слова window и self соответствуют текущему окну? При использовании фреймов эти ключевые слова относятся к текущему фрейму. Еще одно ключевое слово parent позволяет обращаться к главному окну броузера.

Каждый объект frame выступает в качестве дочернего по отношению к родительскому объекту window. В листинге 13.4 приведен код HTML определения набора фреймов.

Листинг 13.4. Документ HTML с фреймами

```
<FRAMESET ROWS="*,*" COLS="*,*">
<FRAME NAME="topleft" SRC="topleft.htm">
<FRAME NAME="topright" SRC="topright.htm">
<FRAME NAME="bottomleft" SRC="bottomleft.htm">
<FRAME NAME="bottomright" SRC="bottomright.htm">
</FRAMESET>
```


http://ITBookz.ru/

Этот код разделяет документ HTML на равные четверти. Если вы имеет готовый файл topleft.htm, то он будет соответствовать объекту parent.topleft. В этом случае ключевые слова window и seft также будут соответствовать фрейму topleft.



Если вы используете закрепленные фреймы, то ситуация немного осложняется: объект window по-прежнему соответствует текущему фрейму, parent — определяет набор фреймов, в котором расположен текущий фрейм, а top — соответствует основному набору фреймов, содержащему остальные наборы.

Массив frames

Вместо того чтобы в программе обращаться к фрейму по имени, можно использовать массив frames. Он сохраняет сведения обо всех фреймах документа. В этом случае фреймы индексируются (начальное значение 0) и определяются в дескрипторе <FRAME>.

Например, вы можете обращаться к фреймам в листинге 13.5 с помощью следующего кода:

- parent.frames[0] соответствует фрейму topleft
- parent.frames[1] соответствует фрейму topright
- parent.frames[2] соответствует фрейму bottomleft
- parent.frames[3] соответствует фрейму bottomright

Для обращения к фрейму документа можно использовать и иерархическую структуру объектов. Выберите тот способ, который проще и удобнее для использования. Например, в документе, состоящем из 10 фреймов, к фреймам проще обращаться как к элементам массива. А двухфреймовым документом проще управлять с помощью имен объектов.

Создание навигационного фрейма

Чаще всего фреймы разделяют страницу на две области, верхнюю и нижнюю. С помощью объектов frame можно создать навигационный фрейм, определяющий отображаемые на экране данные, введенные во втором фрейме.

Сначала необходимо определить набор фреймов. Это проще простого. Листинг 13.6 показывает, как определить фреймы, разделяющие страницу на левую и правую области.

Листинг 13.5. Разделенный на левый и правый фреймы документ **HTML**

```
<HTML>
<HEAD>
<TITLE>Навигационный фрейм</TITLE>
</HEAD>
<FRAMESET COLS="*,*">
<FRAME NAME="left" SRC="left.htm">
<FRAME NAME="right" SRC="about:blank">
</FRAMESET>
</HTML>
```

Затем нам необходимо определить документ для левого фрейма, используемого для перемещения по документу другого фрейма. Он представлен в листинге 13.6.

http://ITBookz.ru/

Листинг 13.6. Документ HTML навигационного фрейма

```
<HTML>
<HEAD>
<TITLE>Навигационный фрейм</TITLE></HEAD>
<BODY>
<P>
Одна из следующих ссылок загружает новую
страницу в правой области
</P>
<UL>
<LI><A HREF="#"
onClick="parent.right.location='order.html';
         window.location='ordernav.html';">
Order Form<A>
<LI><A HREF="#"
onClick="parent.right.location='email.html';
         window.location='emailnav.html';">
Email</A>
<LI><A HREF="#"
onClick="parent.right.location='sales.html';
         window.location='salesnav.html';">
Sales</A>
<LI><A HREF="#"
onClick="parent.right.location='links.html';
         window.location='linksnav.html';">
Other Links</A>
</UL>
</BODY>
</HTML>
```

Этот программный код выглядит сложным из-за присутствия в нем двух операторов JavaScript, определяющих ссылки на странице. Эти операторы немного отличаются для разных ссылок. Вот пример первого по следованию оператора, задающего в программе HTML ссылку на ресурс:

```
onClick="parent.right.location='order.html';
       window.location='ordernav.html';"
```

Этот оператор представляет собой обработчик событий, загружающий необходимую Web-страницу в правом фрейме, а также новый документ в навигационном фрейме. Поскольку сценарий выполняется только в пределах фрейма, необходимо перед именами объектов другого фрейма использоваться ключевое слово parent.



Если вы хотите только загрузить указанный документ после щелчка на ссылке, используйте атрибут TARGET дескриптора <A>. Он позволяет избежать использования сценария JavaScript. Если вы добиваетесь одновременного обновления обоих кадров, используйте приведенный выше код.

Чтобы протестировать сценарий, загрузите его в броузере. Убедитесь, что после щелчка на ссылке в правой области загружается необходимый документ HTML. Сначала загрузите код листинга 13.5. (Все используемые в этом примере документы можно загрузить с Web-узла www.jsworkshop.com.) На рис. 13.6 показан результат выполнения созданного выше сценария.

http://ITBookz.ru/

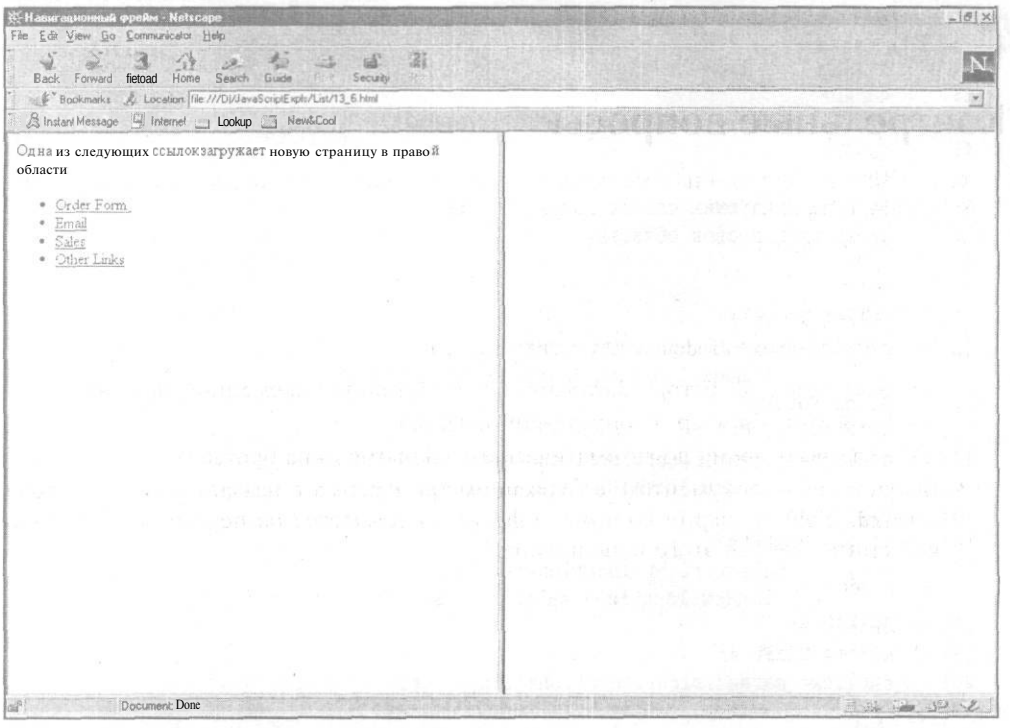


Рис. 13.6. ***Пример использования навигационного фрейма***

Резюме

На этом занятии вы познакомились с объектом window и научились использовать его для управления окном броузера. Вы узнали о временных задержках и фреймах. Кроме того, вы создали готовый документ HTML, разделенный на два фрейма.

На следующем занятии вы познакомитесь с еще одним полезнейшим элементом Web-страницы — формой введения данных. Вы научитесь создавать формы и управлять ими с помощью объектов form.

Вопросы и ответы

Как при запуске сценария в окне, созданном другим сценарием, перейти в главное окно?

В Netscape Navigator версии 3.0 и выше используйте свойство `window.opener`, позволяющее ссылаться на окно, в котором открыто текущее окно.

Я слышал, что слоями Web-страницы можно управлять так же, как и фреймами, но они доступны только в Netscape Navigator версии 4.0 и выше. Можно ли мне использовать слои в сценариях JavaScript?

Можно. Подобно окнам и фреймам каждый слой имеет свой объект window. О них мы поговорим в главе "18-й час. "Создание динамических страниц с помощью DOM".

Как обновить одновременно два фрейма после щелчка на одной ссылке?

Вы обновляете оба фрейма с помощью кода, приведенного в листинге 13.6. Для упрощения операции вы можете создать функцию, загружающую оба фрейма, а затем вызвать ее в обработчике события.

http://ITBookz.ru/

Семинар

Контрольные вопросы

1. Какой из приведенных методов отображает диалоговое окно с кнопками OK и Cancel (Отмена)?
   a) `window.alert`
   b) `window.confirm`
   c) `window.prompt`
2. Что делает метод window. `setTimeout`?
   a) Выполняет операторы JavaScript после указанной временной задержки
   b) Блокирует броузер на определенное время
   c) Указывает время до автоматического закрытия окна броузера
3. Вы управляете документом, содержащим три фрейма с названиями first, second и third. Если сценарий со вторым фреймом ссылается на первый фрейм, то какой синтаксис для этого используется?
   a) `window.first`
   b) `parent.first`
   c) `frames.first`

Ответы

1, b) Метод `window.confirm` отображает диалоговое окно подтверждения

2, a) Метод window. setTimeout позволяет выполнить операторы JavaScript после временной задержки

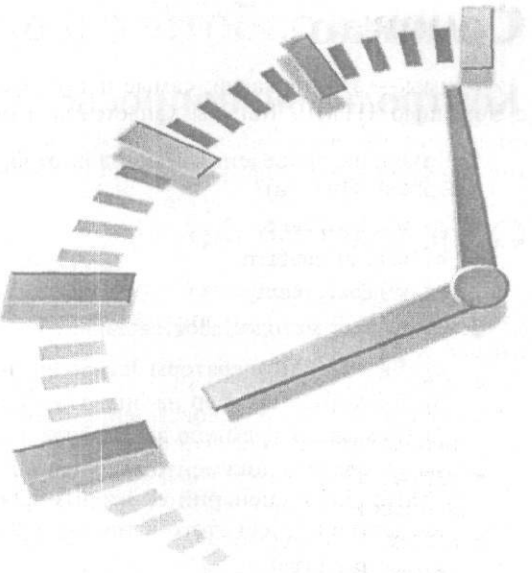
3, b) Сценарий со вторым фреймом должен использовать синтаксис `parent.first`

Упражнения

Если вы планируете активно использовать объект window в своих документах HTML, то для более глубокого изучения его свойств и методов выполните следующие задания.

- Перейдите к рассмотрению сценария определения времени (глава "2-й час. Создание простых сценариев"). Этот сценарий отображает новое значение только при ручном обновлении страницы. Создайте сценарий JavaScript, позволяющий автоматически (с помощью временных задержек) обновлять значение счетчика через одну или две секунды.
- Измените код листингов 13.5 и 13.6 таким образом, чтобы страница содержала три фрейма, а не два. Из навигационного фрейма загружайте документы в два других фрейма. Измените расположение навигационного фрейма.

http://ITBookz.ru/



14-й час

Формы введения данных

В течение этого часа вы узнаете об одном из самых главных применений JavaScript — создании форм. JavaScript позволяет делать интерактивные формы, проверять введенные пользователем данные и вводить данные на основе уже имеющихся данных.

В этой главе рассмотрены следующие темы.

- **Определение формы HTML**
- **Создание формы**
- **Управление формой с помощью объекта `form`**
- **Отображение элементов форм**
- **Получение данные, введенных на форме**
- **Отправка данных по электронной почте**
- **Управление формами CGI**
- **Проверка правильности введенных данных**

http://ITBookz.ru/

Основы работы с формами

Формы — это, пожалуй, самые используемые элементы Web-страниц, создаваемые с помощью HTML. Как вы узнаете на этом занятии, с помощью JavaScript формы становятся интерактивными и функциональными. Первый шаг в создании интерактивной формы заключается в создании обычной формы HTML.

Определение формы

Форма HTML начинается с дескриптора <FORM>. Этот дескриптор указывает на начало формы и позволяет определять элементы формы. Дескриптор <FORM> имеет три параметра.

- NAME. Имя формы. Вы можете использовать форму, не присваивая ей имени. Для дополнительного усовершенствования формы с помощью JavaScript ей необходимо дать название
- METHOD. Принимает значение GET или POST. Существует два способа отправки данных на сервер.
- ACTION. Это сценарий CGI, в соответствии с которым данные отправляются на сервер. Вы можете также использовать параметр mailto:. Он позволяет отправить результаты обработки введенных данных по указанному почтовому адресу.

Например, в следующем примере отображен дескриптор <FORM> для формы Order. Эта форма использует метод GET и отправляет данные в сценарий CGI с названием order.cgi, расположенном в том же каталоге, что и сама Web-страница:

```
<FORM NAME="Order" METHOD="GET" ACTION="order.cgi">
```

В форме, которая полностью создается с помощью JavaScript (например, калькулятор или интерактивная игра), параметры METHOD и ACTION не используются. В них используется упрощенный дескриптор <FORM>:

```
<FORM NAME="calcform">
```

После дескриптора <FORM> указывается несколько элементов формы. Среди них текстовые поля, флажки, кнопки и переключатели. В следующих разделах вы детально познакомитесь с каждым элементом формы.

Использование объекта form

Каждая форма на странице HTML представляется отдельным объектом form, имеющим то же название, что и атрибут NAME в дескрипторе <FORM>.

Для задания формы также можно использовать массив forms. Этот массив содержит столько элементов, сколько их есть на форме. Все элементы проиндексированы, начиная со значения 0. Например, к первой форме документа, имеющей название form1, можно обратить одним из двух способов:

```
document.form1
document.forms[0]
```


http://ITBookz.ru/

Свойства объекта form

Наряду с элементами, каждый объект form обладает целым набором свойств, которые в основном определяются в дескрипторе <FORM>. Всех их можно определить и *с* помощью операторов JavaScript. Ниже приведены все используемые свойства объекта form.

- action. Соответствует атрибуту формы ACTION или программе, в которую отправляются данные, введенные на форме.
- encoding. Определяет форму типа MIME. Определяется атрибутом ENCTYPE. В большинстве случаев не используется.
- length. Указывает количество элементов на форме. Значение этого свойства напрямую изменить нельзя.
- method. Определяет метод, используемый для отправки данных, — GET или POST.
- target. Определяет окно, в котором будут отображаться результаты обработки введенных на форме данных (в соответствии с сценарием CGI). В большинстве случаев используется главное окно — результат обработки данных замещает саму форму.

Отправка данных и очистка формы

Объект form имеет два метода: submit и reset. Первый используется для отправки данных на обработку, а второй — для автоматической очистки формы. Предназначение этих методов становится ясным, когда пользователь, забыв отправить форму, щелкает на рисунке Web-страницы или выполняет другое действие.



При использовании метода submit для отправки данных на сервер или с помощью электронной почты броузер выводит запрос на подтверждение отправки данных. Без подтверждения пользователя отправку данных провести нельзя.

Определение событий формы

Объект form имеет два обработчика событий: onSubmit и onReset. В эти обработчики событий, задаваемые в пределах дескриптора <FORM>, добавляется группа операторов JavaScript или функция, управляющие формой.

Если вы добавите оператор (или функцию) в обработчик onSubmit, то он (или она) вызывается до отправки данных в сценарий CGI. Для того чтобы отменить отправку данных на обработку сценарием CGI, обработчик событий onSubmit должен возвратить значение false. Если же он возвращает значение true, то данные отправляются на сервер. В некоторых случаях необходимо добавить в форму кнопку Reset, запускающую обработчик событий onReset.

Создание элементов форм

Самое важное свойство объекта form — это массив elements. Он содержит объекты каждого элемента формы. Вызывается каждый элемент по имени или индексу в массиве. Например, следующих два выражения соответствуют первому элементу формы заказа (текстовому полю name1):

```
document.order.elements[0]
document.order.name1
```


http://ITBookz.ru/



Как формы, так и элементы можно задавать присвоенными им именами или с помощью индексов соответствующих массивов. Во всех примерах в этой главе формы и их элементы определяются по именам, а не по индексам массива.

Обращаясь к элементам форм как к элементам массива, объект к свойствам формы добавляет свойство length, определяющее общее количество элементов массива.

Например, `document.forms.length` определяет количество форм в документе, а `document.form1.elements.length` — количество элементов в форме `form1`.

Текстовое поле

Чаще всего в формах ввода данных используются текстовые поля. С их помощью запрашиваются имена, адреса и другая текстовая информация. Используя JavaScript, вы можете ввести значение в текстовое поле автоматически. Ниже приведен пример простого текстового поля:

```
<INPUT TYPE="TEXT" NAME="text1" VALUE="Привет" SIZE="30">
```

В этом примере определено текстовое поле `text1`. Оно содержит текст Привет и имеет длину 30 символов. В JavaScript это поле рассматривается как объект text с именем text1.

Текстовые поля — это самые простые элементы формы. Каждое текстовое поле имеет следующие свойства.

- name. Имя, определяющее это поле. Оно используется и как имя объекта.
- `defaultValue`. Значение по умолчанию, определяемое в параметре `VALUE`. Это свойство имеет атрибут только для чтения.
- value. Текущее значение, введенное в поле. Это измененное пользователем или функцией значение по умолчанию.

При работе с текстовыми полями больше всего времени уходит на считывание значения, введенного пользователем, или на изменение значения по умолчанию программным способом. В следующем примере значение, введенное в текстовом поле `username`, изменяется на Джон К. Юзер:

```
document.order.username.value="Джон К. Юзер"
```

Текстовыепанели

Текстовые панели определяются с помощью собственного дескриптора `<TEXTAREA>` и представляются объектом textarea. Существует большая разница между текстовым полем и текстовой панелью. Текстовые панели позволяют вводить несколько строк данных. Ниже приведен пример задания текстовой панели в документе HTML:

```
<TEXTAREA NAME="text1" ROWS="2" COLS="70">
Содержимое текстовой панели
</TEXTAREA>
```

Таким образом определяется текстовая панель с названием text1. Она состоит из двух строк, каждая из которых имеет длину 70 символов. В JavaScript использовать эту текстовую панель вы сможете с помощью объекта text1, дочернего по отношению к объекту `form`.

Текст между открывающим и закрывающим дескриптором `<TEXTAREA>` используется в качестве значения по умолчанию. Вводя текст по умолчанию, вы можете использовать символы конца строки.

http://ITBookz.ru/

Управление текстом в формах

Объекты text и textarea имеют следующие дополнительные методы.

- focus(). Определяет расположение курсора в поле и выделяет текущее поле.
- blur(). Удаляет курсор из поля.
- select(). Выделяет поле подобно тому, как это делает пользователь мышью. Выделяет текст в поле. Не позволяет выделить часть текста.

Изменение значения поля также определяется с помощью обработчика события. Объекты text и textarea поддерживают такие обработчики событий.

- onFocus. Это событие происходит при определении расположения курсора в поле.
- onBlur. Происходит при удалении курсора из текстового поля.
- onChange. Происходит при изменении пользователем значения в поле.
- onSelect. Происходит при выделении пользователем части или всего текста в текстовом поле. Правда, нет возможности определить, какая часть текста выделена, а какая — нет. (При выделении текста с помощью метода Select(), описанного выше, этот обработчик событий не запускается.)

При использовании эти обработчики событий объявляются в дескрипторе <INPUT>. Следующий пример иллюстрирует применение в программе обработчика onChange для отображения сообщения:

```
<INPUT TYPE="TEXT" NAME="text1" onChange="window.alert('Изменено');">
```

Кнопка

После заполнения формы вы обязательно используете кнопки. Кнопки определяются также в дескрипторе <INPUT> и могут быть нескольких типов.

- type=SUBMIT. Это кнопка Submit. Она позволяет отправлять данные на сервер для обработки сценарием CGI.
- type=RESET. Кнопка Reset. Она возвращает форму в исходное состояние после ввода всех параметров по умолчанию.
- type=BUTTON. Произвольная кнопка. Она не вызывает никакого действия, но определяется одному из обработчиков событий JavaScript.

Все три типа кнопок имеют атрибут NAME для определения имени кнопки и атрибут VALUE для указания текста, отображаемого на кнопке. Некоторые кнопки уже рассмотрены нами в примерах главы "13-й час. Использование окон и фреймов". В приведенном ниже примере определяется кнопка Submit, имеющая название sub1 и значение Щелкните здесь:

```
<INPUT TYPE="SUBMIT" NAME="sub1" VALUE="Щелкните здесь">
```

После щелчка на кнопке Reset или Submit вызывается обработчик onReset или onSubmit. Для создаваемых произвольных кнопок запускается обработчик onclick.

http://ITBookz.ru/

Флажок

Флажок — это элемент формы, выглядящий как маленький пустой квадрат. Щелкнув на флажке, вы определяете одно из его состояний — выставлен или снят. Эти состояния соответствуют возвращаемым значениям true или false. Для определения флажка опции используется все тот же дескриптор <INPUT>. Ниже приведен пример его определения:

```
<INPUT TYPE="CHECKBOX" NAME="chek1" VALUE="Yes" CHECKED>
```

И опять элементу формы определяется собственное имя. Атрибут VALUE определяет текст опции флажка. Это утверждение считается справедливым при выставленном флажке. По умолчанию вводится значение on (Да). Атрибут CHECKED указывает на выставленное по умолчанию состояние флажка.

Использовать флажок очень просто. Он имеет только два состояния, но целых четыре свойства.

- name. Определяет имя объекта checkbox и флажка.
- value. Это действительное значение флажка — по умолчанию on. Это значение используется сервером для определения состояния флажка. В JavaScript для определения состояния используется свойство checked.
- defaultChecked. Это состояние по умолчанию флажка, определенное в атрибуте CHECKED.
- checked. Определяет текущее значение флажка. Это булево значение — true или false.

Для управления флажком или определения его значения используется последнее свойство. Следующий оператор выставляет флажок опции same в объекте формы order:

```
document.order.same.checked=true;
```

Объект флажка имеет всего один метод click(). Этот метод симулирует щелчок мышью на флажке. Для него также определен один обработчик события onclick, запускающийся при выставлении флажка. Поскольку он запускается не при любом щелчке на флажке, для его успешного использования необходимо точно знать значение свойства checked.

Переключатель

Еще один элемент, дающий пользователю право выбора, — это переключатель. Он определяется во все том же дескрипторе <INPUT> с помощью ключевого слова RADIO. Переключатели похожи на флажки, за тем исключением, что они задаются группами. Они позволяют сделать пользователю выбор в пользу одного из вариантов. Ниже приведен пример задания группы переключателей:

```
<INPUT TYPE="RADIO" NAME="radio1" VALUE="Option1" CHECKED> Option 1
<INPUT TYPE="RADIO" NAME="radio1" VALUE="Option2"> Option 2
<INPUT TYPE="RADIO" NAME="radio1" VALUE="Option3"> Option 3
```

Этот оператор определяет группу из трех переключателей. Атрибут NAME одинаков для всех трех переключателей (он и объединяет их в одну группу). Значение атрибута VALUE отправляется на сервер при выделении текущего переключателя. Удостоверьтесь, что каждому переключателю определено свое значение атрибута VALUE.



Переключатели приобрели свое название благодаря многопозиционным механическим переключателям, используемым в радиоэлектронике. Они работают таким образом, что в одном положении активна только одна электрическая цепь, а все остальные отключены.

http://ITBookz.ru/

С точки зрения сценария переключатель равносилен флажку, за исключением того, что в группе переключателей можно выставить одновременно только один, а не несколько или даже все. Объект radio имеет следующие свойства.

- name. Определяет имя группы переключателей.
- length. Это количество переключателей в группе.

Чтобы обратить к отдельному переключателю в группе, используется массив radio. Индексирование его элементов проводится с нуля. Каждый отдельно взятый переключатель имеет следующие свойства.

- value. Это значение, определенное переключателю. (Используется сервером.)
- defaultChecked. Определяет значение атрибута CHECKED и начальное положение переключателя.
- checked. Это текущее состояние переключателя.

Например, вы можете выставить первый переключатель в группе radiol формы form1 с помощью следующего оператора:

```
document.form1.radio1[0].checked=true;
```

Тем не менее при определении этого оператора убедитесь, что остальные переключатели группы имеют значения false. Автоматически эта операция не выполняется. Используйте метод click() для одновременного назначения остальным переключателям группы значений false.

Подобно флажку, переключатель имеет метод click() и обработчик событий onClick. Для каждого переключателя группы задается свой обработчик событий.

Раскрывающийся список

Последний элемент форм, используемый в JavaScript, — это *раскрывающийся список*. Этот элемент определяется дескриптором <SELECT>. Раскрывающийся список содержит набор текстовых значений или вариантов выбора. Ниже приведен пример определения в программе раскрывающего списка:

```
<SELECT NAME="select1" SIZE=40>
<OPTION VALUE="choice1" SELECTED> Первый вариант
<OPTION VALUE="choice2"> Второй вариант
<OPTION VALUE="choice3"> Третий вариант
</SELECT>
```

Каждый дескриптор <OPTION> определяет один элемент раскрывающего списка. Атрибут VALUE — это возвращаемое в программу значение, а текст вне дескрипторов <OPTION> отображается при открытии списка.

Дополнительный атрибут MULTIPLE позволяет выделять одновременно несколько элементов раскрывающегося списка. Броузеры, как правило, управляют одновременно только одним из элементов в раскрывающемся списке.

Управление раскрывающимся списком осуществляется с помощью объекта select. Он имеет следующие свойства.

- name. Имя объекта раскрывающегося списка.
- length. Число элементов в списке.
- options. Массив значений элементов списка. Каждый элемент массива соответствует отдельному элементу раскрывающего списка.
- selectedIndex. Возвращает значение индекса выделенного элемента списка. В списках, позволяющих выделять одновременно несколько элементов, оно определяет первый выделенный элемент.

http://ITBookz.ru/

Массив option имеет одно свойство length, указывающее количество элементов раскрывающегося списка. В дополнение каждый элемент массива имеет такие свойства.

- index. Индекс элемента в массиве.
- defaultSelected. Определяет значение атрибута SELECTED.
- selected. Текущее состояние элемента. Значение true определяет выделенный элемент. В списках с атрибутом MULTIPLE это значение может определяться нескольким свойствам selected элементов массива.
- name. Определяет значение атрибута NAME. Используется сервером.
- text. Текст, отображаемый в раскрывающемся списке на месте каждого элемента. В Netscape Navigator версии 3.0 и выше его значение можно изменить.

Объект select имеет два метода: blur() и focus(). Они выполняют те же операции, что и одноименные методы объекта text. Соответствующие обработчики событий onBlue, onFocus и onChange вызываются так же, как и в других объектах.



Вы можете изменять раскрывающиеся списки динамически. Например, выделение соответствующего элемента в одном из раскрывающихся списков определяет отображаемые элементы в другом раскрывающемся списке. При необходимости можно добавлять или удалять элементы списка.

Чтение значения выделенного элемента проводится в два этапа. Сначала определяется значение свойства selectedIndex и только затем — значение свойства value. Приведем пример:

```
ind=document.navform.choice.selectedIndex;
val=document.navform.choice.options[ind].value;
```

В этом примере переменная ind используется для сохранения текущего индекса. Используя значение переменной ind, дальше определяем значение свойства value и присваиваем переменной val. Этот метод активно используется в главе "22-й час. Улучшение Web-страниц".

Отображение данных на форме

В качестве небольшого примера использования форм приведем листинг 14.1, который содержит программу HTML-формы с полями ввода имени, адреса и телефонных номеров. Введенные данные выводятся на экран в виде отдельного окна.

Листинг 14.1. Форма с отдельным окном отображения введенных данных

```
1:    <HTML>
2:    <HEAD>
3:    <TITLE>Первая форма</TITLE>
4:    <SCRIPT LANGUAGE="JavaScript">
5:    function display() {
6:    DispWin=window.open('', 'NewWin', 'toolbar=no,status=no,width=300,
          height=200')
7:    message="<UL><LI><B>NAME:      </B>"+document.form1.yourname.value;
8:    DispWin.document.write(message);
9:    message="<LI><B>ADDRESS: </B>"+document.form1.address.value;
10:   DispWin.document.write(message);
11:   message="<LI><B>PHONE: </B>"+document.form1.phone.value+"</UL>";
12:   DispWin.document.write(message);
13:   }
```


http://ITBookz.ru/

```
14:    </SCRIPT>
15:    </HEAD>
16:    <BODY>
17:    <H1>Первая форма</H1>
18:    Введите все данные. При щелчке на кнопке Display данные
           отобразятся в отдельном окне
19:    <FORM name="form1">
20:    <B>Имя:</B><INPUT TYPE="TEXT" LENGTH="20" NAME="yourname">
21:    <P>
22:    <B>Адрес:</B><INPUT TYPE="TEXT" LENGTH="30" NAME="address">
23:    <P>
24:    <B>Тел.:</B><INPUT TYPE="TEXT" LENGTH="15" NAME="phone">
25:    <P><INPUT TYPE="BUTTON" VALUE="Display" onClick="display();">
26:    </FORM>
27:    </BODY>
28:    </HTML>
```

Ниже приведено краткое описание выполнения сценария приведенного документа HTML

- В строках 5—13 определена функция display, открывающая новое окно, как описано в главе "13-й час. Использование окон и фреймов", и отображающая данные из формы.
- Строка 19 содержит определение формы. Поскольку эта форма полностью управляется JavaScript, никакие дополнительные операторы не вводятся.
- Строки 20—26 определяют три текстовых поля формы: yourname, address и phone. Строка 25 содержит определение кнопки Display, запускающей функцию display.

На рис. 14.1 отображена созданная нами форма. На экране, в отдельном окне, отображается результат выполнения функции display.

Отправка данных формы в виде почтового сообщения

Один из самых простых способов использования формы — это отправка ее данных в виде почтового сообщения. Эта операция выполняется без привлечения средств JavaScript, хотя проверку правильности введенных данных провести стоило бы (о ней рассказывается в следующей главе).

Чтобы отправить данные с формы в виде почтового сообщения, добавьте в код атрибута ACTION формы ключевое слово mailto:. Листинг 14.2 содержит модифицированный код листинга 14.1, в котором изменена кнопка и данные отправляются в виде почтового сообщения.

Листинг 14.2. Отправка данных формы в виде почтового сообщения

```
1:    <HTML>
2:    <HEAD>
3:    <TITLE>Пример формы</TITLE>
4:    </HEAD>
5:    <BODY>
6:    <H1>Пример формы</H1>
7:    Введите все данные. При щелчке на кнопке
```


http://ITBookz.ru/

```
8:      Submit данные отправятся в виде сообщения
9:     <FORM name="form1" action="mailto:user@host.com" enctype="text/plain">
10:    <B>Имя:</B><INPUT TYPE="TEXT" LENGTH="20" NAME="yourname">
11:    <P>
12:    <B>Адрес:</B><INPUT TYPE="TEXT" LENGTH="30" NAME="address">
13:    <P>
14:    <B>Тел.:</B><INPUT TYPE="TEXT" LENGTH="15" NAME="phone">
15:    <P>
16:    <INPUT TYPE="SUBMIT" VALUE="Submit">
17:    </FORM>
18:    </BODY>
19;    </HTML>
```

Чтобы правильно выполнить эту форму, вместо user8host.com введите адрес своего почтового ящика. Обратите внимание на оператор `enctype="text/plain"`, приведенный в конце строки 9. Этот оператор определяет формат представления данных в почтовом сообщении.

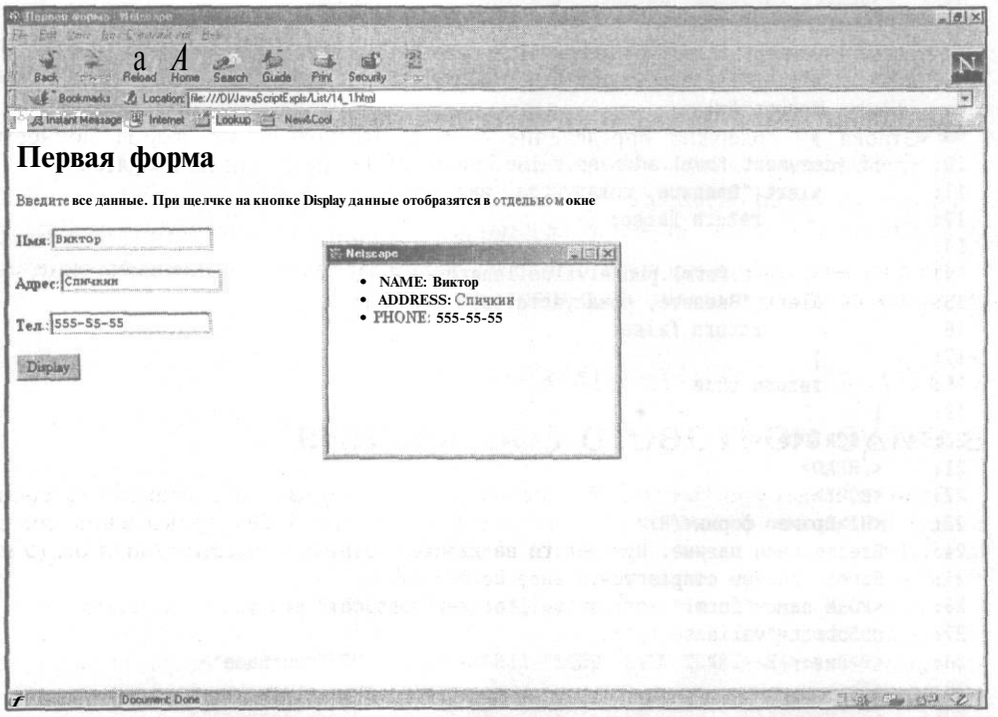


Рис. 14.1. Отображение данных, введенных на форме, в дополнительном окне

http://ITBookz.ru/

Проверка правильности заполнения формы

JavaScript очень часто используется для проверки правильности заполнения формы. Это означает, что операторы JavaScript позволяют определить, правильно ли введены все необходимые значения. Например, необходимо проверить, что нет пустых полей и данные введены в правильном формате.

Проверка введенных данных проводится при отправке их на сервер или в почтовый ящик пользователя. В листинге 14.3 представлена форма, содержащая только поля адреса и имени и проверяемая перед отправкой в виде почтового сообщения.

Листинг 14.3. Форма с проверкой правильности введения данных

```
<HTML>
<HEAD>
<TITLE>Форма с проверкой</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function validate() {
if (document.form1.yourname.value.length<1){
     alert("Введите, пожалуйста, имя");
     return false;
     }
if (document.form1.address.value.length<3) {
     alert("Введите, пожалуйста, ваш адрес");
          return false;
     }
if (document.form1.phone.value.length<3) {
     alert("Введите, пожалуйста, ваш тел-н");
          return false;
     }
     return true
}
</SCRIPT>
</HEAD>
<BODY>
<H1>Пример формы</H1>
Введите все данные. При щелчке на кнопке
Submit данные отправятся в виде сообщения
<FORM name="form1" action="mailto:user@host.com" enctype="text/plain"
onSubmit="validate();">
<B>Имя:</B><INPUT TYPE="TEXT" LENGTH="20" NAME="yourname">
<P>
<B>Адрес:</B><INPUT TYPE="TEXT" LENGTH="30" NAME="address">
<P>
<B>Тел.:</B><INPUT TYPE="TEXT" LENGTH="15" NAME="phone">
<P>
<INPUT TYPE="SUBMIT" VALUE="Submit">
</FORM>
</BODY>
</HTML>
```


http://ITBookz.ru/

В форме используется функция validate(), проверяющая данные в каждом поле формы. Проверка осуществляется по значению свойства length каждого поля. Если введенное значение достаточно длинное, чтобы быть правильным, данные с формы отправляются по указанному адресу. В противном случае отображается предупреждающее сообщение и отправка данных прерывается.

Дескриптор <FORM> в строке 26 содержит обработчик события onSubmit. Этот обработчик вызывает функцию validate(). Ключевое слово return определяет возвращаемое функцией значение. Это значение и задает отправляемость данных.



Для запуска процедуры проверки данных можно использовать и обработчик события onChange. Он позволяет провести проверку правильности введенных данных до щелчка на кнопке Submit.

На рис. 14.2 показан результат выполнения сценария. Как вы видите, форма заполнена наполовину — отсутствует полное имя в соответствующем поле, на что и указывает выведенное на экран сообщение.

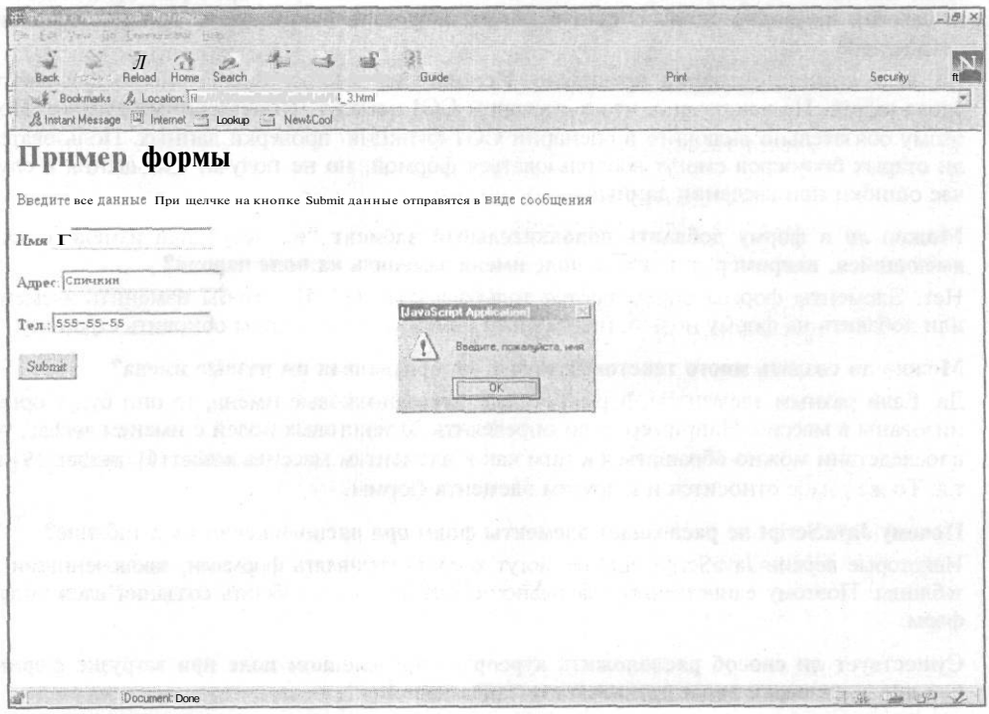


Рис. 14.2. Пример проверки правильности введения данных на форме

http://ITBookz.ru/

Резюме

За этот час вы познакомились с формами HTML и средствами управления ими в JavaScript. Вы научились управлять объектом form и объектами разных элементов форм. На основе полученных знаний вы создали несколько сценариев.

Вы также узнали, как данные с формы отправляются на сервер или в почтовый ящик в виде сообщения. Кроме того, вы теперь умеете проводить проверку правильности введенных данных.

Если формы — это сугубо деловое применение JavaScript, то динамические рисунки и анимационные изображения — это из области развлечений. О них мы и поговорим в следующей главе.

Вопросы и ответы

Если добавить в программу с помощью JavaScript сценарий проверки правильности введенных на форме данных, смогут ли его запустить броузеры, не поддерживающие JavaScript?

Да, если создать сценарий правильно. Используйте кнопку Submit вместо ключевого слова submit. Не исключено, что в сценарий CGI попадут непроверенные данные. Поэтому обязательно включите в сценарий CGI функцию проверки данных. Пользователи старых броузеров смогут воспользоваться формой, но не получат сообщения в случае ошибки при введении данных.

Можно ли в форму добавить дополнительный элемент "на лету" или изменить уже имеющийся, например, текстовое поле имени заменить на поле пароля?

Нет. Элементы формы определяются только в коде HTML. Чтобы изменить элемент или добавить на форму новый, необходимо изменить код, а затем обновить страницу.

Можно ли создать много текстовых полей, не присваивая им разные имена?

Да. Если разным элементам формы определить одинаковые имена, то они будут организованы в массив. Например, если определить 20 текстовых полей с именем member, то впоследствии можно обращаться к ним как к элементам массива `member[0]`, `member[19]` и т.д. То же самое относится и к другим элемента формы.

Почему JavaScript не распознает элементы форм при расположении их в таблице?

Некоторые версии JavaScript еще не могут хорошо управлять формами, заключенными в таблицы. Поэтому единственно правильное решение — это избегать создания вложенных форм.

Существует ли способ расположить курсор в определенном поле при загрузке формы или после проверки ее на правильность заполнения?

Да. Для этого используется метод `focus()` объекта поля. Для размещения курсора при загрузке страницы используйте в дескрипторе `<BODY>` метод `onLoad`. Правда, не существует метода для размещения курсора в необходимой позиции поля.

http://ITBookz.ru/

Семинар

Контрольные вопросы

1. Какой из атрибутов дескриптора <FORM> определяет пункт назначения данных, введенных на форме?
   a) ACTION
   b) METHOD
   c) NAME
2. Где вводится обработчик событий onSubmit для проверки правильности заполнения формы?
   a) В дескрипторе <BODY>
   b) В дескрипторе <FORM>
   c) В дескрипторе <INPUT>, в котором определена кнопка Submit
3. Что позволяет сделать с формой JavaScript, чего не позволяет сделать CGI?
   a) Устранить причины ошибок
   b) Установить обратную связь с ошибками
   c) Отправить данные на сервер

Ответы

1, a) Атрибут ACTION определяет пункт назначения данных формы

2, b) Обработчик событий onSubmit вводится в дескрипторе <FORM>

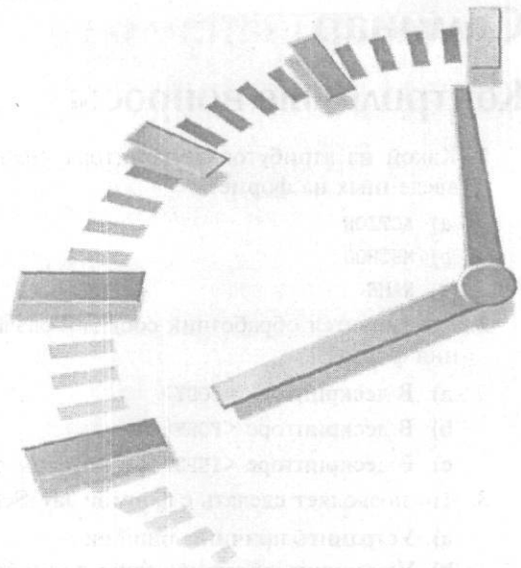
3, b) JavaScript позволяет обнаружить ошибку и известить о ней пользователя, не привлекая для этого ресурсы сервера.

Упражнения

Если вы хотите посвятить себя созданию форм, то обязательно выполните приведенные ниже задания.

- Измените функцию validate(), приведенную в листинге 14.3, так, чтобы после вывода сообщения о неправильности значения в поле курсор автоматически перемещался в это поле. (Используйте метод focus для соответствующего объекта элемента формы.)
- Добавьте в листинг 14.3 программный код текстового поля введения адреса электронной почты. Проверку правильности вводимого в него значения проводите по наличию в нем символа @.

http://ITBookz.ru/



15-й час

Добавление рисунков и анимации

Одна из самых примечательных и сложных возможностей языков программирования — это создание приложений с рисунками и игр. За этот час вы познакомитесь с некоторыми методами добавления на Web-страницы рисунков, без которых ни один документа HTML не будет выглядеть красиво.

Конечно, JavaScript — это сравнительно простой язык, не позволяющий создавать такие сложные игры, как Quake. В этой книге, которая также относительно проста, мы рассмотрим методы управления в JavaScript рисунками и простыми анимационными изображениями.

В этой главе рассмотрены следующие темы.

- **Использование разделенного рисунка**
- **Применение объектов рисунков**
- **Создание изменяющихся рисунков**
- **Загрузка рисунков в кэш**
- **Создание простых анимационных изображений**

http://ITBookz.ru/

Использование разделенного рисунка

Разделенные рисунки — это один самых распространенных способов перемещения по страницам Web-узла. Они представляют собой рисунки, разделенные на несколько областей, каждая из которых выступает в роли отдельной ссылки. Щелчок на каждой из областей одного рисунка приводит к выполнению разных команд сценария.



Существует два типа разделенных рисунков — клиентные и серверные. Клиентные разделенные рисунки определяют области ссылок на изображении в документе HTML и создаются с помощью HTML. Серверные разделенные рисунки требуют создания специального файла определения карты разделения рисунка, которым управляет Web-сервер.

В качестве примера клиентского разделенного рисунка давайте создадим с помощью JavaScript простое графическое меню для какой-нибудь компании. (Название компании роли не играет, но это точно не компания, занимающаяся дизайном Web-страниц.)

Чтобы создать клиентский разделенный рисунок, сделайте следующее.

- В вашем графическом приложении создайте рисунки в формате GIF и JPEG.
- Определите разбиение основного рисунка, задающее разделение рисунка на отдельные области.
- Включите разделенный рисунок в документ с помощью атрибута USEMAP.

Листинг 15.1 содержит полный программный код подобного документа HTML.

Листинг 15.1. Использование клиентского разделенного рисунка

```
<HTML>
<HEAD>
<TITLE>Пример разделенного рисунка</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function update(t) {
    document.form1.text1.value=t;
}
</SCRIPT>
</HEAD>
<BODY>
<MAP NAME="map1">
<AREA SHAPE=RECT COORDS="14,15,151,87"
HREF="javascript:update('Сервис');"
onMouseOver="window.status='Сервисный отдел'; return true;">
<AREA SHAPE=RECT COORDS="162,16,283,85"
HREF="javascript:update('Магазин');"
onMouseOver="window.status='Отдел продаж'; return true;">
<AREA SHAPE=RECT COORDS="294,15,388,87"
HREF="javascript:update('О нас');"
onMouseOver="window.status='О нас'; return true;">
<AREA SHAPE=RECT COORDS="13,98,79,178"
HREF="javascript:update('Email');"
onMouseOver="window.status='Свяжитесь с нами'; return true;">
<AREA SHAPE=RECT COORDS="92,97,223,117"
HREF="javascript:update('Товары');"
onMouseOver="window.status='Товары и услуги'; return true;">
<AREA SHAPE=RECT COORDS="235,98,388,177"
HREF="javascript:update('История');"
onMouseOver="window.status='История'; return true;">
<AREA SHAPE=default
HREF="javascript:update('Раздел не указан')";
onMouseOver="window.status='Укажите раздел'; return true;">
</MAP>
```


http://ITBookz.ru/

```
27:    <H1>Пример клиентского разделенного рисунка</H1>
28:     <HR>
29:    Приведенный разделенный рисунок создан с помощью
30:     JavaScript. Наведение указателя на одну из
31:     его областей позволяет отобразить описание
32:     раздела в строке состояния. Щелкнув на
33:     выбранной области, вы добавите название
34:     раздела в приведенном ниже текстовом поле.
35:     <HR>
36:     <IMG SRC="imagemap.gif" USEMAP="#map1">
37:     <HR>
38:     <FORM NAME="form1">
39:     <B>Выбран раздел:</B>
40:     <INPUT TYPE="text" NAME="text1"
        VALUE="Выберите, пожалуйста, раздел">
41:     </FORM>
42:     <HR>
43:     </BODY>
44:     </HTML>
```

Этот сценарий для выполнения необходимых операций использует и обработчики событий и ссылки. Вместо того чтобы загрузить новую страницу после щелчка на ссылке, ее название отображается в текстовом поле, приведенном на рис. 15.1. Кроме того, обработчик событий onMouseOver отображает в строке состояния описание ссылки.

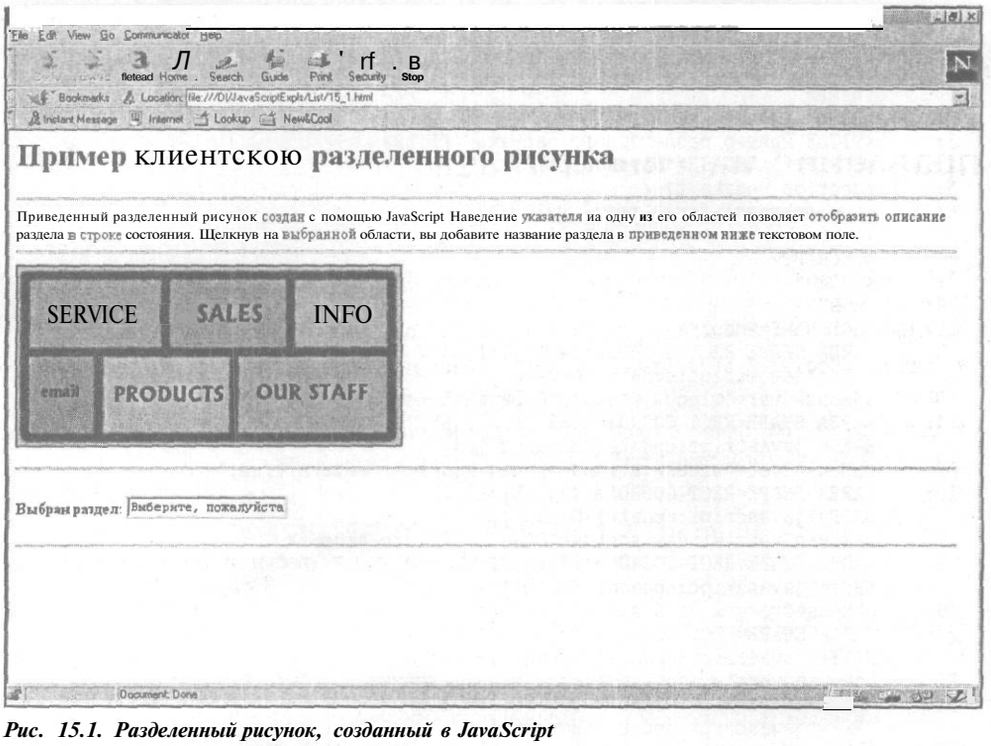


Рис. 15.1. Разделенный рисунок, созданный в JavaScript

В программе HTML используется функция JavaScript с именем `update()`. Эта функция размещает текстовое значение в поле формы form1.text1. Текст в поле формы соответствует названию ссылки.

http://ITBookz.ru/

Динамические рисунки

Динамические рисунки — вот что делает Web-страницы по-настоящему привлекательными. Динамические рисунки представляют собой изображения, которые постоянно и произвольно изменяются. Динамические рисунки нашли широкое применение — при создании изображения часов, изменяющихся рисунков (изображений, которые изменяются при наведении на них указателя мыши) и даже простых анимаций.

Рисунки на Web-страницах представлены в виде массива, подобно элементам формы. Изменяя свойства элементов массива, можно добиться замены одного рисунка другим. Это и позволяет создавать динамические картинки, не обновляя содержимое Web-страницы целиком. Этот метод также не требует использования слоев и фреймов.

Следует заметить, что описанный метод создания динамических рисунков далеко не совершенный. Перед тем как им воспользоваться, примите к вниманию следующие замечания.

- Вы можете только изменять рисунок на Web-странице. Добавить новый или удалить существующий нельзя.
- Вы можете заменить рисунок большим или меньшим, но это не очень красиво выглядит на экране.
- Все отображаемые рисунки загружаются с сервера. Поэтому невозможно создать большие динамические рисунки.



Поддержка динамических рисунков впервые была реализована в JavaScript 1.1, т.е. в Netscape Navigator 3.0 и Internet Explorer 4.0. Старшие версии броузеров также поддерживают динамические изображения.

Управление массивом images

Рисунки изменяются динамически с помощью массива images. Этот массив состоит из элементов, каждый из которых соответствует отдельному изображению, сохраненному на сервере. Каждый элемент массива имеет свое имя. В иерархической структуре объект image выступает дочерним по отношению к объекту document.

Каждый объект image имеет следующие свойства.

- border. Соответствует атрибуту BORDER дескриптора <IMG>. Определяет границы рисунка.
- complete. Этот флаг определяет степень загруженности рисунка. Принимает булевы значения (true или false).
- height и width. Задают размеры рисунка. Это полностью информационные свойства. Изменить их при создании динамических рисунков нельзя.
- hspace и vspace. Определяют место расположения рисунка на странице. Эти свойства также имеют значения с атрибутом только для чтения.
- name. Имя рисунка определяется атрибутом NAME при определении рисунка.
- lowsrc. Принимает значение атрибута LOWSRC. Это специальный атрибут, используемый броузером, который определяет загрузку рисунка в низком разрешении перед загрузкой основного изображения.
- src. Источник рисунка, определяемый адресом URL. При создании динамического рисунка это значение обязательно изменяется.

http://ITBookz.ru/

В большинстве случаев используется только свойство src. Тем не менее, иногда требуется изменить значение свойства lowsrc. Это позволяет загрузить рисунок в низком разрешении перед загрузкой основного изображения.

Объект image не обладает ни одним методом, но позволяет применять три обработчика событий.

- onLoad. Запускается после окончания загрузки рисунка. (Лучше использовать обработчик onLoad для объекта image, а не для всего документа, поскольку последний запускается после загрузки всех рисунков.)
- onAbort. Запускается при отмене пользователем загрузки страницы, на которой еще не отображены рисунки.
- onError. Запускается, если файл рисунка поврежден или не найден.

Предварительная загрузка рисунка

Хотя и нельзя динамически добавить на страницу рисунок, вы можете создать независимый объект image. Этот объект позволяет определить изображение, которое будет загружено в кэш-память, но не отображено на экране.

Это звучит глупо, но этот метод часто используется в современных технологиях. После предварительной загрузки изображения вы можете заменить им рисунок на странице. Поскольку изображение содержится в кэше, замена осуществляется практически мгновенно.

Кэширование рисунка проводится после создания нового объекта image с помощью ключевого слова new. Вот как это делается:

```
image2=new Image();
image2.src="arrow1.gif"
```



Детально создание нового объекта и объектно-ориентированное программирование рассмотрены в главе "11-й час. Создание пользовательских объектов".

Создание изменяющихся рисунков

Чаще всего динамические рисунки применяются для создания *изменяющихся рисунков,* которые меняют свой вид при наведении на них указателя мыши.

Изменяющиеся рисунки, как правило, используются в качестве ссылок на Web-странице. Используя их, вы выделяете ссылку. При наведении указателя мыши изменяется весь рисунок или только определенная его часть (например, добавляется рамка вокруг уже существующего текста).

Преобразовать изображение в изменяющийся рисунок можно с помощью обработчика событий onMouseOver, который позволяет заменить один рисунок другим (немного отличающимся от исходного). В листинге 15.2 приведен пример кода документа HTML с изменяющимся рисунком.

Листинг 15.2. Пример использования изменяющего рисунка

```
1:          <HTML>
2:          <HEAD>
3:          <TITLE>Пример динамического рисунка</TITLE>
4:          </HEAD>
5:          <BODY>
6:          <H1>Пример изменяющегося рисунка</H1>
```


http://ITBookz.ru/

```
7:        <HR>
8:       Рисунок изменяется при наведении на него указателя
9:        <P>
10:       <A HREF="home.html"
11:       onMouseOver="document.images[0].src='home1.gif';"
12:       onMouseOut="document.images[0].src='home.gif';">
13:      <IMG src="home.gif" width=192 height=47 alt="" border="0">
14:      </A>
15:       <BR>
16:       <A HREF="links.html"
17:       onMouseOver="document.images[1].src='link1.gif';"
18:      onMouseOut="document.images[1].src='link.gif' ;">
19:       <IMG src="links.gif" width=93 height=42 alt=""
    border="0">
20:      </A>
21:       <BR>
22:       <A HREF="guest.html"
23:       onMouseOver="document.images[2].src='guest1.gif';"
24:       onMouseOut="document.images[2].src='guest.gif';">
25:       <IMG src="guest.gif" width=195 height=42 alt=""
    border="0">
26:      </A>
27:       <BR>
28:      <A HREF="email.html"
29:       onMouseOver="document.images[3].src='email1.gif';"
30:       onMouseOut="document.images[3].src='email.gif';">
31:       <IMG src="email.gif" width=185 height=42 alt=""
    border="0">
32:      </A>
33:       </BODY>
34:       </HTML>
```

В этом примере используются две версии каждого рисунка. Например, guest.gif и guestl.gif **отличаются только рамкой вокруг текста. На первом рисунке ее нет. Каждая ссылка содержит обработчики событий** OnMouseOver и onMouseOut. **Именно они и определяют, когда изменять рисунок. Рис. 15.2 иллюстрирует правильное выполнение сценария.**



Выделить рисунок можно и другим способом, например изменить цвет фона. Еще один интересный метод выделения ссылки — разместить возле текста небольшой пустой рисунок, который преобразуется в стрелку или в другой, бросающийся в глаза, символ при наведении на него указателя мыши.

http://ITBookz.ru/

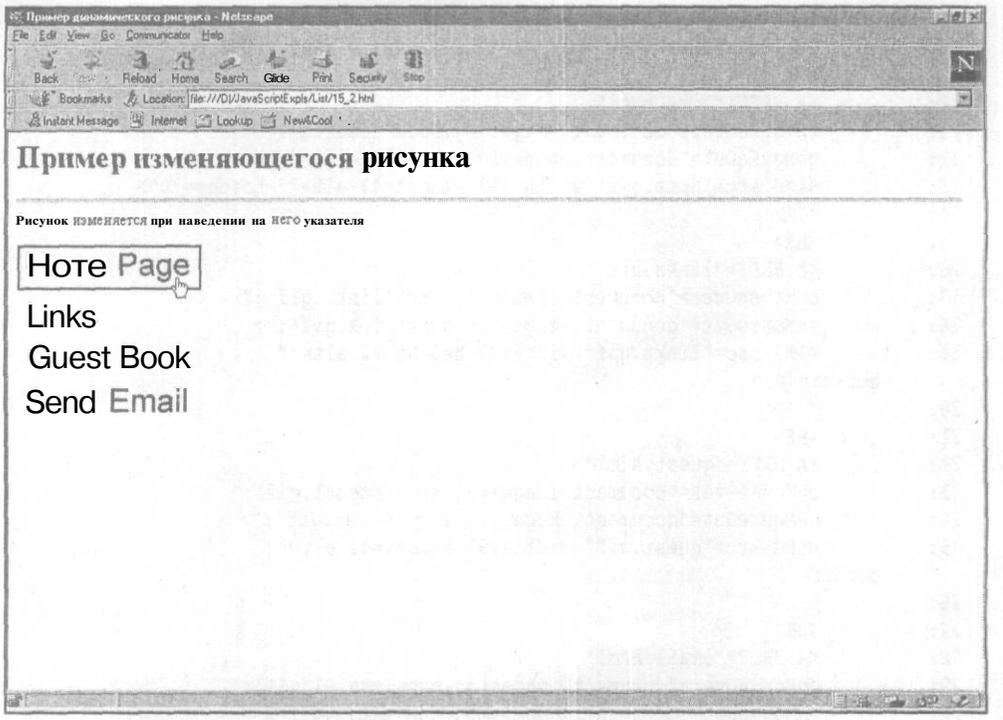


Рис. 15.2. Использование JavaScript для создания динамических рисунков

Создание простой анимации

Изменяющиеся рисунки — это, несомненно, важные элементы современных Web-страниц. Еще больше привлекательности документам HTML придают анимационные изображения. Следует сразу заметить, что JavaScript — не самое мощное средство для создания анимации. Но этот язык часто используется для создания анимации тогда, когда необходимо управлять ею в сценарии.



Если вы хотите создать периодически повторяющееся изображение, то лучше динамического рисунка в формате GIF вам просто не найти. Существует огромное количество программных продуктов, позволяющих создавать динамические рисунки в формате GIF.

В качестве примера анимационного изображения создадим сценарий перемещения рисованного объекта вдоль страницы.

Создание рисунков

Первый этап при создании анимационного изображения состоит в создании набора рисунков. Например, давайте создадим движущуюся по экрану мышь. (Вы можете рисовать все, что вам вздумается, но мышь — это предел моих художественных способностей.)

Для анимационного изображения нам потребуется набор изображений, показанных на рис. 15.3. Чтобы создать эти изображения, создайте квадрат размером 100x100 и перемещайте его влево и вправо относительно анимируемого объекта, создавая разные "кадры".

http://ITBookz.ru/

Назовите "кадры" именами mouse1.gif-mouse8.gif. Вам также понадобится пустой рисунок, отображаемый при отсутствии указателя над изображением. Назовем его mouse0.gif.

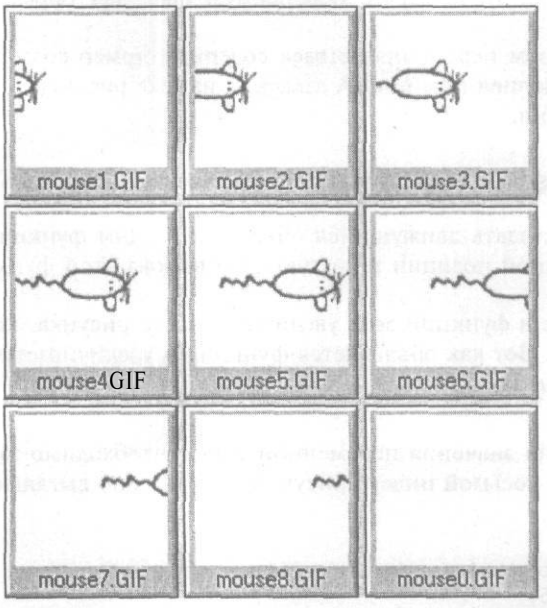


Рис. 15.3. Восемь рисунков, составляющих основу анимационного изображения

Создание документа HTML

Теперь необходимо создать документ HTML, задающий анимационную картинку. Пусть наша мышь перемешается в пяти позициях страницы, от одной ее границы к другой. Вот как задаются в коде HTML начальные рисунки:

```
<H1>Анимация в JavaScript</H1>
<HR>
<CENTER>
<IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
<IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
<IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
<IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
<IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
</CENTER>
```

Таким образом, все пять позиций отображают пустой рисунок mouse0.gif, т.е. остаются невидимыми. Поскольку используется пять исходных анимационных позиций, в сценарии они будут представлены массивом image[0-4].

Определение переменных

В самом начале сценария необходимо определить некоторые глобальные переменные:

```
var cbox=0;
var nbox=1;
var cimage=0;
var nimage=0;
```


http://ITBookz.ru/

Переменная cbox принимает значения 0-4, определяя позицию, в которой анимируется изображение в текущий момент. Часть изображение мыши будет "высовываться" в следующую позицию. Используйте переменную nbox для определения расположения правой части мыши.

Подобным образом переменная cimage содержит номер созданного рисунка (1–8), отображаемого в текущей позиции. А nimage — индекс рисунка, который отображается в следующей позиции.

Выполнение анимации

Настало время создать движущийся объект. Создадим функцию next, перемещающую рисунок из одной позиции в другую. Для вызова этой функции используем метод setTimeout.

На первом этапе в функции next увеличим индекс рисунка, который отображается в текущей позиции. Вот как объявляется функция и увеличивается индекс:

```
function next() {
cimage+=1;
```

После увеличения значения переменной cimage необходимо задать условие, позволяющее определить восьмой индекс рисунка. Вот как оно выглядит в JavaScript:

```
if (cimage>8) {
  cimage=4;
  document.images[cbox].src="mouse0.gif";
  cbox=(cbox+1)%5;
  nbox=(cbox+1)%5;
```

Если текущий индекс cbox меньше 8, то его значение увеличивается на единицу и переменной nbox определяется новое значение. Оба выражения используют оператор модуля (%), не позволяющего переменным принимать значения больше 4. Если это происходит, наша мышь начинает "двигаться" заново от левой границы текущей позиции.

Предыдущий код также содержит оператор присваивания переменной cimage значения 4. (В этой позиции, при отображении в предыдущей рисунков 5, 6, 7, 8, уже отображались рисунки 1, 2, 3.)

Затем необходимо рассчитать индекс рисунка, отображаемого в следующей позиции:

```
nimage = cimage - 5;
if (nimage <=0) nimage = 0;
```

Первое выражение определяет значение переменной nimage как cimage - 5, В этом нет ничего удивительного. Если взглянуть на рис 15.3, то несложно обнаружить, что рисунки 1, 2 и 3 при сопоставлении с рисунками 6, 7 и 8 соответственно дают полное изображение мыши. Если в результате выполнения этой операции возвращается отрицательное значение, переменная nimage принимает значение 0.

И в конце функции задаются операторы вывода на экран рисунков, определенных переменными nimage и cimage в указанных позициях:

```
document.images[cbox].src="mouse"+cimage+".gif";
document.images[nbox].src="mouse"+nimage+".gif";
window.setTimeout("next();",100);
}
```

В последнем операторе функции задается временная задержка перед следующим вызовом функции (0,1 с). Для увеличения или уменьшения скорости движения мыши уменьшайте или увеличивайте соответственно это значение.

http://ITBookz.ru/

Компоновка сценария

Теперь вы имеете все необходимые элементы для создания готового сценария анимированного изображения. В листинге 15.3 представлен код документа HTML с таким сценарием. Дескриптор <BODY> содержит обработчик события onLoad, вызывающий функцию next после указанной временной задержки.

Листинг 15.3. Пример анимированного изображения

```
<HTML>
<HEAD>
<TITLE>Анимация с помощью JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var cbox=0;
var nbox=1;
var cimage=0;
var nimage=0;

function preload() {
a1 = new Image();
a1.src = "mouse1.gif";
a2 = new Image();
a2.src = "mouse2.gif";
a3 = new Image();
a3.src = "mouse3.gif";
a4 = new Image();
a4.src = "mouse4.gif";
a5 = new Image();
a5.src = "mouse5.gif";
a6 = new Image();
a6.src = "mouse6.gif";
a7 = new Image();
a7.src = "mouse7.gif";
a8 = new Image();
a8.src = "mouse8.gif";
window.setTimeout("next();",500);
}
function next() {
     cimage+=1
if (cimage>8) {
     cimage=4;
     document.images[cbox] .src="mouse0.gif";
     cbox=(cbox+1)%5;
     nbox=(cbox+1)%5;
     }
nimage = cimage - 5;
if (nimage <=0) nimage = 0;
document.images[cbox].src="mouse"+cimage+".gif";
document.images[nbox].src="mouse"+nimage+".gif";
window.setTimeout("next();",100);
}
</SCRIPT>
</HEAD>
```


http://ITBookz.ru/

```
45:         <BODY onLoad="preload();">
46:         <H1>Анимация в JavaScript</H1>
47:         <HR>
48:         <CENTER>
49:         <IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
50:         <IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
51:         <IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
52:         <IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
53:         <IMG src="mouse0.gif" width=100 height=100 alt="" border="0">
54:         </CENTER>
55:         <HR>
56:         </BODY>
57:         </HTML>
```

Чтобы протестировать сценарий, загрузите документ HTML в броузер. Убедитесь в том, что файл .html расположен в той же папке, что и файлы рисунков mouse0.gif-mouse8.gif. При правильном выполнении сценария вы увидите на экране перемещающуюся от левой границы страницы до правой маленькую мышку (рис. 15.4).

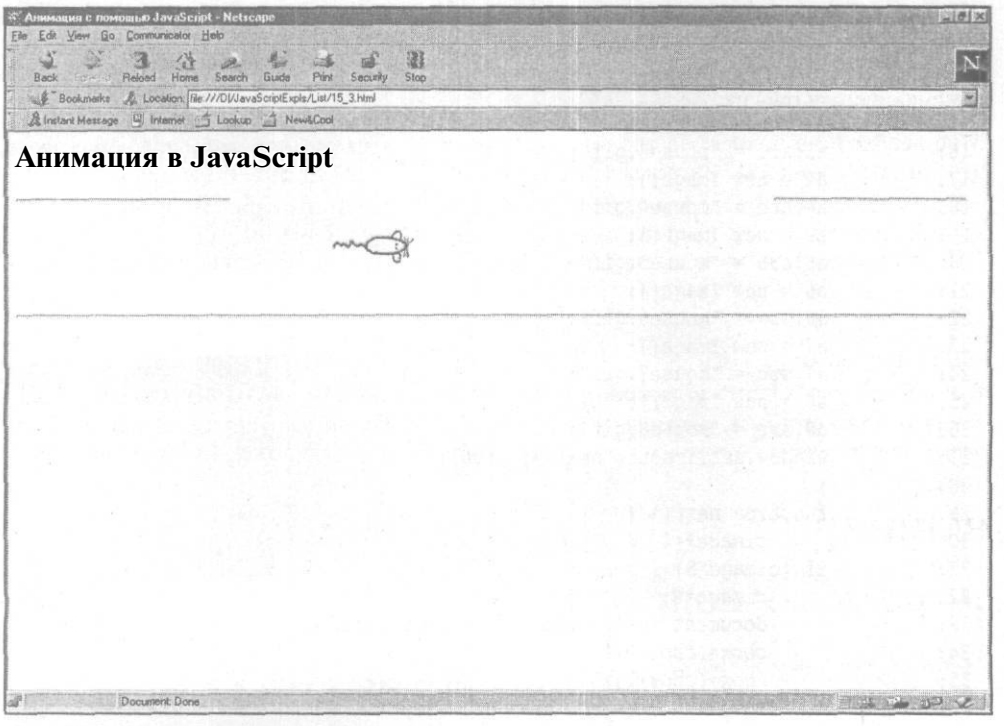


Рис. 15.4. Вот она — бегущая мышь



При детальном рассмотрении вы будете наблюдать небольшие скачки при перемещении мыши по экрану. Чтобы устранить их, удалите пробелы и символы перехода к новой строке в дескрипторах <IMG> строк 49-53.

http://ITBookz.ru/

Резюме

На этом занятии вы узнали о том, как в JavaScript можно управлять рисунками. Вы также познакомились с разделенными рисунками и создали пример изменяющегося рисунка.

В последней главе части IV вы узнаете о некоторых специальных методах создания сценариев — научитесь определять версию броузера и создавать сценарии, запускаемые в самых разных броузерах.

Вопросы и ответы

Я определил обработчик событий onMouseOver, **используемый в изменяющихся рисунках. Он запускается в Internet Explorer, но не запускается в** Netscape Navigator. Почему?

Это происходит потому, что вы объявили этот обработчик событий в дескрипторе <IMG>. Этот дескриптор используется только в Internet Explorer. В Netscape Navigator для этих целей применяется дескриптор <A>. (Он поддерживается и в Internet Explorer.)

Можно ли использовать изменяющиеся рисунки на разделенных рисунках?

Нет. Для этого вам придется полностью заменить разделенный рисунок. Проще и эффективнее разделить рисунок на области и использовать каждую из них как элемент изменяющегося рисунка.

Мне необходимо полностью удалить рисунок, а не изменить его источник. Можно ли это сделать с помощью JavaScript?

Нет. Вы не можете удалить рисунок. Но вы можете заменить его пустым рисунком такого же размера, чем добьетесь того же результата — одноцветный рисунок в формате GIF занимает очень мало места на жестком диске. Помните, что вы не можете изменить текстовый слой страницы.

Я создал программу JavaScript документа HTML с рисунками, но обработчики событий не запускаются. Почему?

В JavaScript необходимо в дескрипторах <IMG> определить атрибуты HIEGHT и WIDTH. Добавив их, можете быть уверены, что обработчики событий запустятся. Детально о отладке программ мы поговорим в главе "21-й час. Отладка приложений JavaScript".

Семинар

Контрольные вопросы

1. Каких два обработчика событий используют в изменяющихся рисунках?
   a) onMouseOver и onClick
   b) onMouseOver и onMouseUnder
   c) onMouseOver и onMouseOut
2. Какой тип разделенных рисунков используется в JavaScript?
   a) Серверные разделенные рисунки
   b) Клиентские разделенные рисунки
   c) **Оба**

http://ITBookz.ru/

3. Какой объект JavaScript используется для управления вторым рисунком сценария?

   a) `image[2]`
   b) `images[2]`
   c) `images[1]`

Ответы

1, c) Изменяющиеся рисунки создаются с помощью обработчиков `onMouseOver` и `onMouseOut`

2, b) В JavaScript используются клиентские разделенные рисунки

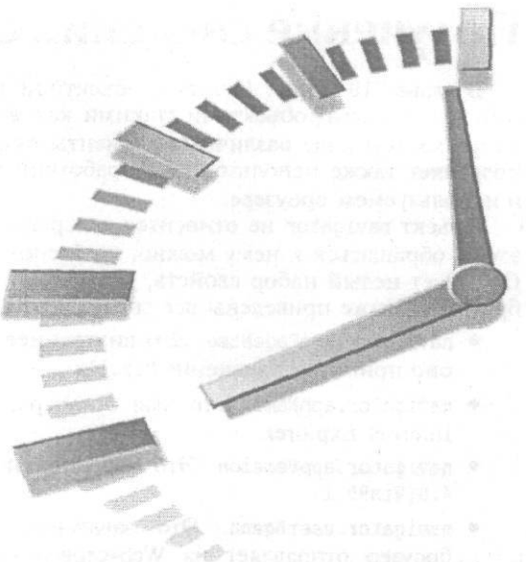
3, c) Второй рисунок страницы определяется объектом `images[1]`

Упражнения

Для углубления познаний о использовании в документах HTML рисунков выполните следующие задания.

- Измените листинг 15.1 так, чтобы после щелчка на разных областях разделенного рисунка отображались не названия разделов, а открывались соответствующие страницы.
- В листинге 15.2 добавьте третий рисунок, который отображается после щелчка мышью на исходном рисунке. Используйте для этого обработчик событий `onMouseDown`.

http://ITBookz.ru/



16-й час

Создание сценариев для разных броузеров

Вот вы и добрались к концу части IV. До этого момента создаваемые нами сценарии в основном были рассчитаны на последние версии популярных броузеров: Netscape Navigator и Internet Explorer. Этих два броузера использует подавляющее большинство пользователей.

Сценарии, приведенные в следующей части этой книги, не будут так же универсальны, как созданные раньше. Последние усовершенствования языков подготовки сценариев, такие как слои и каналы, выполняются по-разному в разных броузерах, а некоторые средства JavaScript вообще поддерживаются в одних броузерах и не поддерживаются в других.

Новый стандарт DOM обуславливает переход к совершенно новым методам построения универсальных документов HTML. Но до его полной поддержки вам придется с помощью JavaScript создавать сценарии специально "под броузер", используя в них только те средства, которые поддерживаются необходимым броузером.

В этой главе рассмотрены следующие темы.

- **Получение сведений о броузере**
- **Отображение сведений о броузере**
- **Создание страниц для разных броузеров**
- **Управление броузерами, не поддерживающими JavaScript**
- **Создание универсальных сценариев, запускающихся в любых броузерах**

http://ITBookz.ru/

Получение сведений о броузере

В главе "10-й час. Работа с объектной моделью документа" вы познакомились с самыми разными объектами (такими как window и document), которые представляют в программном коде различные элементы окна броузера и документа HTML. JavaScript позволяет также использовать разработчикам объект navigator, содержащий сведения о используемом броузере.

Объект navigator не относится к иерархической структуре объектов JavaScript. Поэтому обращаться к нему можно, не беспокоясь о соблюдении иерархических правил. Он имеет целый набор свойств, каждое из которых содержит те или иные сведения о броузере. Ниже приведены все свойства этого объекта.

- `navigator.appCodeName`. Это внутреннее кодовое название броузера. Как правило, оно принимает значение `Mozilla`.
- `navigator.appName`. Это имя броузера, обычно Netscape Navigator или Microsoft Internet Explorer.
- `navigator.appVersion`. Это версия используемого броузера Netscape. Например, `4.0(Win95;I)`
- `navigator.userAgent`. Это заголовок агента пользователя — записи, которую броузер отправляет на Web-сервер при загрузке Web-страницы. Он содержит полные сведения о версии броузера, например `Mozilla/4.0(Win95;I)`
- `navigator.language`. Это язык общения броузера с пользователем (английский или русский). Это свойство содержит двухзначный код языка, например *en для* английского. Если вы создаете страницу на разных языках, то используете это свойство для определения языка общения броузера с пользователем.
- `navigator.platform`. Это платформа, на которой установлен броузер. Это свойство содержит сокращенное значение. Например `Win16`, Win32 или MacPPC. Это свойство позволяет подключать специальные средства платформы (например, ActiveX в Windows).

Все эти свойства обычно используются вместе с условным оператором if. Например, следующий оператор загружает другую Web-страницу, если используется броузер, отличный от Netscape Navigator 4.x:

```
if (navigator.userAgent.indexOf("Mozilla/4")==-1)
  window.location="non_netscape.html";
```



Как вы уже заметили, объект navigator получил свое название от имени броузера Netscape Navigator, который первым поддерживал JavaScript. К счастью, этот язык также поддерживается и в Internet Explorer.

Отображение сведений о броузере

В листинге 16.1 приведен пример кода сценария считывания сведений о броузере. Этот сценарий при выполнении отображает на экране список значений свойств объекта используемого броузера.

Листинг 16.1. Отображение сведений о броузере

```
1:          <HTML>
2:          <HEAD>
3:          <TITLE>Сведения о броузере</TITLE>
4:          </HEAD>
5:          <BODY>
```


http://ITBookz.ru/

```
6:          <H1>Сведения о броузере</H1>
7:          <HR>
8:          <P>
9:          Объект <B>navigator</B> содержит следующие сведения
10:         о броузере
11:         </P>
12:         <UL>
13:         <SCRIPT LANGUAGE="JavaScript">
14:         document.write("<LI><B>Кодовое название:</B>" +
        navigator.appCodeName);
15:         document.write("<LI><B>Имя программы:</B>" +
        navigator.appName);
16:         document.write("<LI><B>Версия программы:</B>" +
        navigator.appVersion);
17:         document.write("<LI><B>Агент пользователя:</B>" +
        navigator.userAgent);
18:         document.write("<LI><B>Язык:</B>" +
        navigator.language);
19:         document.write("<LI><B>Платформа:</B>" +
        navigator.platform);
20:         </SCRIPT>
21:         </UL>
22:         <HR>
23:         </BODY>
24:         </HTML>
```

В этом листинге представлен обычный документ HTML. Сценарий добавлен в теле кода документа HTML (строки 13–20). Эти строки и отвечают за отображение на экране значений свойств объекта navigator, т.е. сведений о броузере. Для отображения данных используется хорошо известный вам оператор `document.write`.

Чтобы протестировать этот сценарий, загрузите его в броузере. Если у вас установлено несколько броузеров, загрузите сценарий во всех броузерах и сравните полученные результаты. Netscape Navigator отображает документ HTML, показанный на рис. 16.1.

Броузеры независимых производителей

ЕСЛИ запустить программу HTML листинга 16.1 в Internet Explorer, то можно получить достаточно неожиданный результат. На рис. 16.2 показан результат запуска документа HTML в Internet Explorer.

Присмотревшись внимательно, вы обнаружите две странности. Первая — не определен язык общения с пользователем. На самом деле, ничего удивительного в этом нет, поскольку свойство `navigator.language` в Internet Explorer не поддерживается.

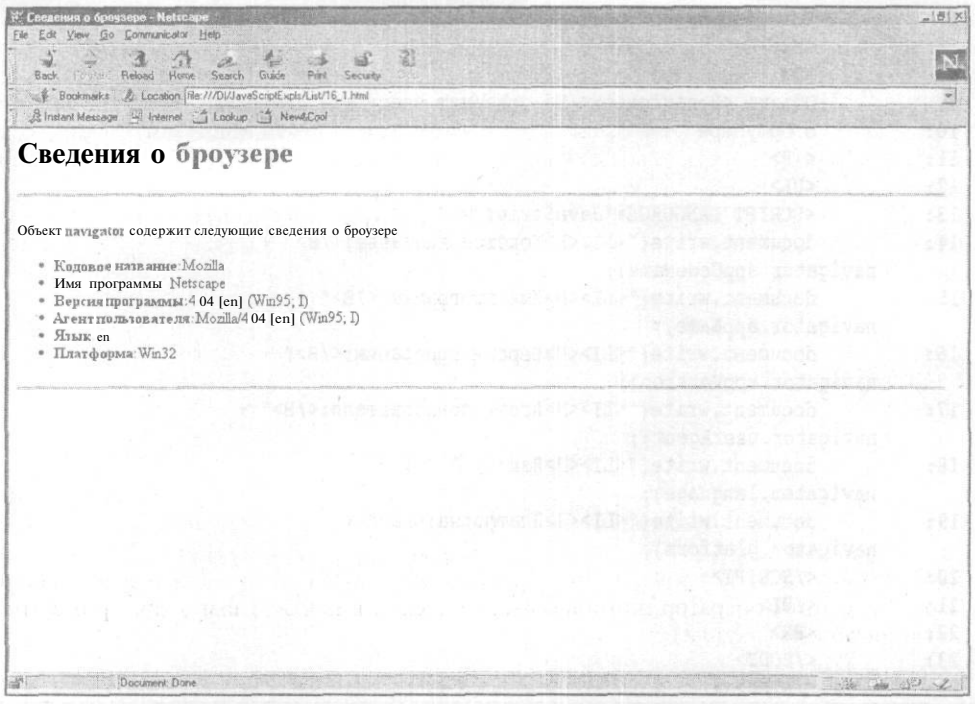
Еще важнее то, что в качестве кодового названия используется Mozilla. Агент пользователя также принимает соответствующее значение:

```
Mozilla/4.0(compatible; MSIE 5.5; Windows 98; Win 9x4.90)
```

Что это? Доказательство тайного сговора *Microsoft* и *Netscape*? Или мифического монстра Mozilla выкрали и скрывают в тайных подземельях *Microsoft*?

Все не так сложно, как этого хотелось бы некоторым из вас. В "войне" броузеров главным оружием разработчиков стали Web-страницы, отображаемые только одним броузером.

http://ITBookz.ru/



Рис, 16.1. Информация о броузере в окне Netscape Navigator

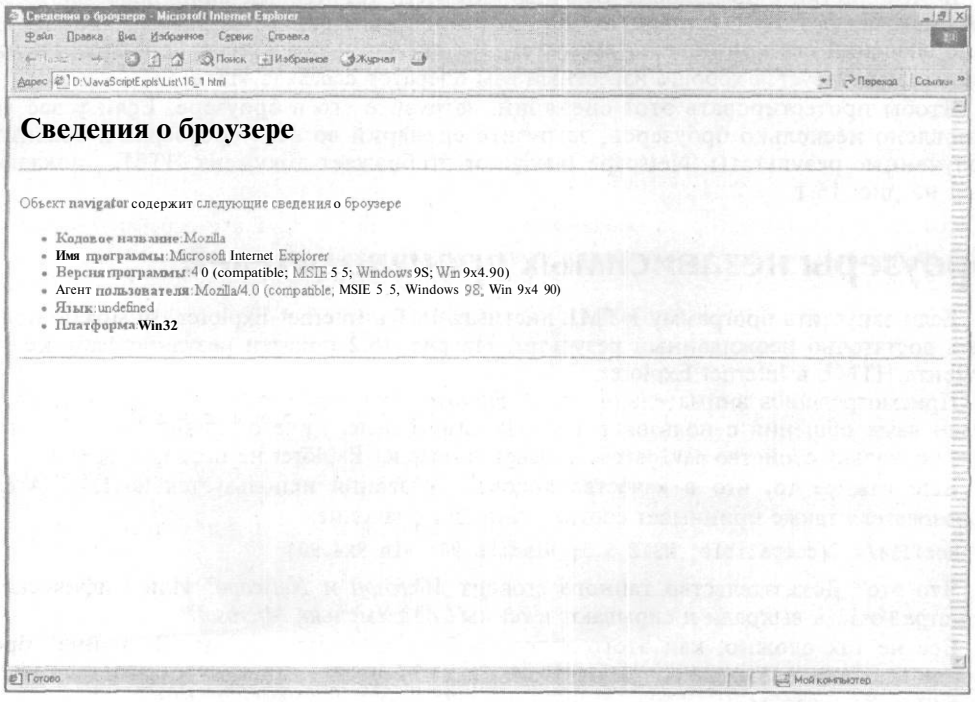


Рис. 16.2. Информация о броузере в окне Internet Explorer

http://ITBookz.ru/

Netscape, не отставая от *Microsoft,* также поспешила воспользоваться этим незамысловатым приемом (это было в эпоху господствования Internet Explorer 3.0 и Netscape Navigator 3.0). Некоторые Web-дизайнеры, до смерти верные *Netscape,* поспешили создать множество сценариев JavaScript, не запускаемых в броузерах с агентом пользователя Mozilla. Единственным правильным решением со стороны *Microsoft* стало присвоение своему броузеру кодового название Mozilla. Вот так злорадство одних вызывает искренне недоумение других.

Следует заметить, чтобы отличить агент Internet Explorer от агента Netscape, в имени первого в скобках приводится ключевое слово compatible (совместимый). Это позволяет Internet Explorer не только отображать все Web-страницы, но и использовать только его специальные средства.



Много Web-серверов сохраняют статистику о количестве посетителей, воспользовавшихся для отображения его Web-страниц тем или иным броузером. После изменения названия агента пользователя *Microsoft* сделала "медвежью услугу" *Netscape.* При определении броузера пользователя Internet Explorer стал определяться как программный продукт *Netscape.* В результате статистические показатели *Netscape* повысились, а *Microsoft* — снизились.

Хотя эта история о методах ведения войны броузеров занимательна, вас, наверное, больше интересует, как это отразится на создании сценариев JavaScript. Действительно, необходимо быть осторожным при использовании в сценарии значения версии броузера. Вспомним оператор, который был приведен в начале главы, как пример использования объекта navigator:

```
if (navigator.userAgent.indexOf("Mozilla/4")==-1)
  window.location="non_netscape.html";
```

Как вам теперь стало понятно, этот оператор задает условие по версии броузера Mozilla/4, что соответствует Netscape Navigator 4.x или Internet Explorer 4.x. Невелика ошибка — подумаете вы. А что делать, если ваш сценарий полностью поддерживается только в Netscape Navigator 4.0? В этом случае необходимо немного изменить условие:

```
if (navigator.appName.indexOf("Netscape")==-1)
  && (navigator.appVersion.indexOf("4.0")==-1)
  window.location="non_netscape.html";
```

Этот длинный оператор if сначала проверяет свойство navigator.appName, которое точно указывает на производителя *(Netscape* или *Microsoft).* И только после этого проверяется свойство navigator.appVertion. Если версия броузера не 4.0 или он произведен не *Netscape,* то на рассмотрение пользователя открывается другая страница.



В этом примере использован метод indexOf, позволяющий искать строковое значение внутри другого строкового значения. Детально эта функция рассмотрена в главе "6-й час. Использование массивов и строковых данных".

В заключение хотелось бы предупредить вас. Не доверяйте свойствам броузера, а всегда проверяйте создаваемые вами сценарии во всех доступных вам броузерах. В следующих разделах вы узнаете, как определить броузеры версий 5.0 и выше.

Поддержка JavaScript броузером

Если вы в сценарии используете средства, которые поддерживаются только в определенных броузерах или определенных их версиях, воспользуйтесь одним из описанных ниже методов правильного создания сценария.

http://ITBookz.ru/

Создание сценария для определенного броузера

Самое простое решение — и наиболее неприемлемое с точки зрения пользователя — это создать сценарий, поддерживаемый только в необходимом броузере. В остальных броузерах его запустить нельзя. В верхней части тела программы HTML введите код сценария, подобный приведенному в листинге 16.2.

Листинг 16.2. Создание сценария для определенного броузера

```
<SCRIPT LANGUAGE="JavaScript">
if (navigator.appName.indexOf("Netscape")==-1)
||(navigator.appVersion.indexOf("5.0")==-1)
window.alert("Загружается Web-узел Netscape")
window.location="http://www.netscape.com//";
</SCRIPT>
```

Если вы используете броузер стороннего производителя (не *Microsoft)* или броузер Netscape версии ниже 5.0, то на экране появится указанное сообщение и в окне будет загружена начальная страница Web-узла *Netscape.*



Задавая *в* сценарии условие использования определенных броузеров для загрузки Web-страницы, имейте на это веские основания. Не рекомендуется делать это "ради интереса" или "из вредности". Большая часть пользователей не захочет приобретать новый броузер только из-за желания посетить ваш узел. Вы потеряете огромную часть аудитории.

Создание разных страниц

В большинстве случаев реализовать поддержку разных броузеров не так уж и сложно. Если вы хотите, чтобы ваша страница прекрасно выглядела и в Netscape Navigator, и в Internet Explorer, создайте разные версии страницы и загружайте их в разных броузерах.

Листинг 16.3 содержит код, который размещается в начале документа HTML. Определив версию броузера, этот сценарий отправляет пользователя к необходимой ему версии страницы.

Листинг 16.3. Загрузка разных версий страницы

```
<SCRIPT LANGUAGE="JavaScript">
if (navigator.appName.indexOf("Netscape")>-1)
&&(navigator.appVersion.indexOf("5")>-1)
window.location="netscape.html";
if (navigator.appName.indexOf("Microsoft")>-1)
S&(navigator.appVersion.indexOf("4")>-1)
window.location="ie.html";
window.location="default.html";
</SCRIPT>
```

Этот сценарий позволяет пользователю отобразить ту страницу, которая создана специально для его броузера: netscape.html для Netscape Navigator 5.x или 6.x, ie.html для Internet Explorer 4.x или 5.x и `default` для остальных броузеров.

http://ITBookz.ru/



Вы, наверное, заметили, что в сценарии задано только два оператора if. Третий оператор if использовать нет необходимости. Если не выполняется ни одно из двух условий, то по умолчанию загружается страница `default.html`. Если же одно из условий выполняется, то загружается указанная страница, а оператор отображения страницы `default.html` не выполняется вообще.

Создание универсальной страницы

Третий метод заключается в создании универсальной страницы, которая загружается во всех распространенных броузерах. В подобных сценариях вы также можете определить используемые броузеры и для каждого из них выполнить специфические операции — и все это в пределах одного документа HTML.

Этот метод подразумевает создание только одного документа, как правило, длинного и громоздкого. Иногда подобные проекты заканчиваются трагически — страница прекращает свое существование еще на стадии планирования. Пример простой страницы, созданной по этому методу, приведен в следующем разделе.



Некоторые дополнительные способы создания сценариев для разных броузеров представлены в главе "19-й час. Дополнительные средства DOM".

Сценарии в броузерах, не поддерживающих JavaScript

Как поступить с броузерами, которые вообще не поддерживают JavaScript? Вы не можете определить их, поскольку сценарий JavaScript в них попросту *выполняться не будет*. Как ни странно, но существует методика определения и таких броузеров.

Практически все пользователи (около 99%) для просмотра Web-страниц используют Netscape Navigator или Internet Explorer. К остальным относятся старые броузеры, например Lynx, который не поддерживает сценарии. Существуют также пользователи, которые из соображений безопасности отключили в Netscape Navigator или Internet Explorer поддержку сценариев.

За последние несколько лет на рынке программных продуктов появилось огромное количество броузеров сторонних производителей. Некоторые платформы, например PalmPilot или Windows CE, позволяют соединяться с Internet, но поддерживают только специальные броузеры. Как видно из перечисленного выше, один процент пользователей может вызвать бурю в стакане воды, если их интересы не будут удовлетворены разработчиками сценариев.

Один из способов избежать ошибок в документах HTML, содержащих сценарии, — это использовать дескриптор <NOSCRIPT>. Этот дескриптор поддерживается Netscape Navigator версии 3.0 и выше; он отображает сообщение при загрузке его в броузере, не поддерживающем JavaScript. Броузеры, поддерживающие JavaScript, игнорируют строку с дескриптором <NOSCRIPT>. Ниже приведен пример использования этого дескриптора:

```
<NOSCRIPT>
Эта страница загружается только в броузерах, поддерживающих JavaScript. Загрузите
копию Netscape Navigator с узла <A HREF="http://www.netscape.com/">; перейдите на
страницу <A HREF="nojs.html">Без сценариев</A>
</NOSCRIPT>
```


http://ITBookz.ru/

Главная проблема при использовании дескриптора <NOSCRIPT> заключается в том, что Netscape Navigator 2.0 отображает сообщение даже в том случае, если сценарий им поддерживается. Существует метод устранения и этого недоразумения. Листинг 16.4 содержит код, позволяющий определить броузер, не поддерживающий JavaScript, и отобразить сообщение.

Листинг 16.4. Определение броузеров, не поддерживающих JavaScript

```
1:      <SCRIPT LANGUAGE='JavaScript'>
2:          <!-- //скрыть от старых броузеров
3:            ...команды...
4:           /*(начало комментария JavaScript, конец
        комментария HTML)
5:           -->
6:        Вы используетет броузер, не поддерживающий
    JavaScript. Как вам не стыдно?
7:      <!-- */ // -->
8:      </SCRIPT>
```

Чтобы скрыть команды сценария от старого броузера, используется дескриптор комментария HTML (<!-- и -->). Дескриптор комментария JavaScript в броузерах, не поддерживающих JavaScript, не отображается.

Самый простой вариант — это загрузить в броузере, поддерживающем JavaScript, отдельную страницу. Для этого используется следующий оператор JavaScript:

```
<SCRIPT LANGUAGE="JavaScript">
window.location="JavaScript.html";
</SCRIPT>
```

Если броузер не поддерживает JavaScript, то этот сценарий просто не будет выполняться.

Последний способ — это сделать ваш сценарий ненавязчивым. Детально о комментариях, позволяющих скрыть сценарий JavaScript от старых броузеров, рассказано в главе "4-й час. Выполнение программ JavaScript".

Сценарий для разных броузеров

В качестве примера описанных в этой главе методов давайте создадим сценарий определения типа и версии броузера, соответствующий следующим категориям:

- Internet Explorer или Netscape Navigator версии 5.0 и выше
- Netscape 4.x
- Internet Explorer 4.x
- Остальные броузеры

Этот сценарий часто используется многими дизайнерами, поскольку броузеры версии выше 5.0 поддерживают DOM в DHTML. Броузеры же версии 4.x также поддерживают динамические средства HTML, но они несовместимы между собой. Остальные броузеры поддерживают только ограниченный набор инструкций JavaScript.

В нашем примере определяется тип броузера и переменной browser определяется соответствующее значение. В листинге 16.5 приведен полный код сценария.

http://ITBookz.ru/

Листинг 16.5. Определение категории броузера

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Определение броузера</TITLE>
4:  <SCRIPT LANGUAGE="JavaScript">
5:  //Броузеры версии 5.0 и выше
6:  if (parseInt(navigator.appVersion)>=5
7:       || navigator.appVersion.indexOf("MSIE 5") !=-1) {
8:       browser="DOM";
9:  } else if (navigator.userAgent.indexOf("Mozilla/4")!=-1)
10: {
11:      if (navigator.appName.indexOf("Netscape")!=-1)
12:           browser="NS4";
13:       if (navigator.appVersion.indexOf("MSIE 4") !=-1)
14:           browser="IE4";
15:       } else browser="Другой";
16: </SCRIPT>
17: </HEAD>
18: <BODY>
19: <H1>Пример определения категории броузера<-H1>
20: <SCRIPT LANGUAGE="JavaScript">
21: document.write("Броузер: " + browser + "<BR>");
16: </SCRIPT>
17: </BODY>
18: </HTML>
```

Сценарий в заголовке страницы (строки 4–16) определяют для переменной browser значение DOM (для броузеров версии выше 5.0), NS4 (броузеры Netscape 4.x), IE4 (броузеры Internet Explorer 4.x). В теле этого документа приведен всего один оператор отображения значения переменной browser, а сценарий использует его для изменения отображаемой на экране величины.

Резюме

За этот час вы познакомились с разными способами отображения страниц со сценариями JavaScript в разных броузерах.

Вот вы и изучили часть IV книги. Вы уже эксперт в технологиях JavaScript. В следующей части вы познакомитесь с некоторыми специфическими средствами броузеров, которые используются в JavaScript.

Вопросы и ответы

Требует ли мой броузер установки встроенной утилиты? Можно ли это определить с помощью JavaScript?

Да. Объект navigator имеет свойство, позволяющее это сделать. Детально о нем будет рассказано в главе "20-й час. Использование мультимедиа и встроенных утилит". Следует заметить, что это свойство в Internet Explorer не поддерживается.

Можно ли с помощью объекта navigator **определить почтовый адрес пользователя? А с помощью других способов?**

http://ITBookz.ru/

Нет. Определить таким образом почтовый адрес пользователя нельзя. (Если бы это можно было бы сделать, то вы получали бы каждый день сотни рекламных сообщений от компаний, Web-узлы которых вы посещали.) Для получения адреса пользователя используются личные сценарии, доступ к которым защищен.

Нужно ли беспокоиться, если используется броузер, созданный не Netscape или Microsoft и не поддерживающий JavaScript?

В Windows можно установить броузер Opera, который поддерживает сценарии JavaScript. По возможности установите несколько броузеров.

Семинар

Контрольные вопросы

1. Какое из следующих свойств объекта navigator имеет одинаковое значение в Internet Explorer и в Netscape Navigator?
   a) `navigator.appCodeName`
   b) `navigator.appName`
   c) `navigator.appVersion`
2. Что из указанного ниже вы *не можете* сделать с помощью JavaScript?
   a) Открыть в окне Netscape Navigator другую страницу
   b) Открыть в окне Internet Explorer другую страницу
   c) Открыть в окне броузера, не поддерживающего JavaScript, другую страницу
3. Что делает дескриптор <NOSCRIPT>?
   a) Скрывает текст от старых броузеров
   b) Предотвращает выполнение сценария
   c) Отображает код сценария

Ответы

1, a) Свойство `navigator.appCodeName` имеет одинаковое (Mozilla) значение в обоих броузерах

2, c) Вы не можете с помощью JavaScript открыть другую страницу в броузерах, не поддерживающих JavaScript

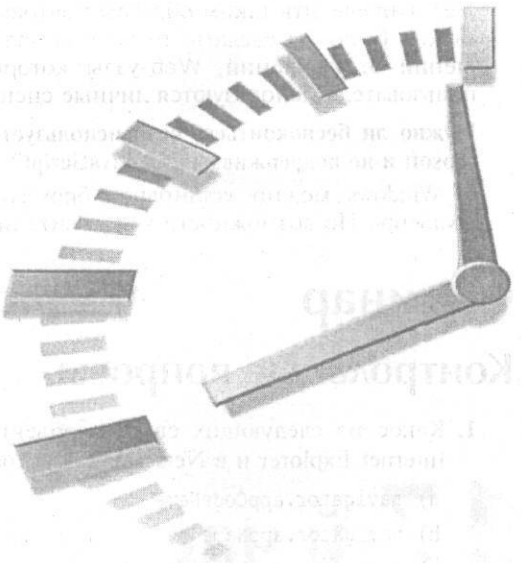
3, a) Дескриптор <NOSCRIPT> скрывает текст от старых броузеров

Упражнения

Для повышения своих знаний о использовании сценариев JavaScript в разных броузерах выполните следующие задания

- Создайте сценарий (подобный приведенному в листинге 16.1), открывающий в Internet Explorer 4.0 и Netscape Navigator 4.0 одну страницу, а в Internet Explorer 3.0 и Netscape Navigator 3.0 — другую.
- В код листинга 16.5 добавьте операторы определения броузеров Netscape 3.x и Internet Explorer 3.x.

http://ITBookz.ru/

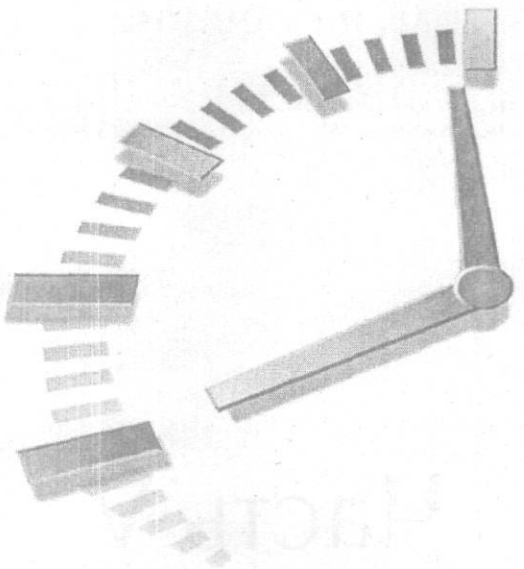


Часть V
Дополнительные средства JavaScript

Темы занятий

17. Использование таблиц стилей
18. Создание динамических страниц с помощью DOM
19. Дополнительные средства DOM
20. Использование мультимедиа и встроенных утилит

http://ITBookz.ru/



17-й час

Использование таблиц стилей

Примите мои поздравления! Вы переходите к изучению части V этой книги. За следующих четыре часа вы познакомитесь с некоторыми полезными дополнениями к JavaScript, позволяющими улучшить ваши сценарии.

В этой главе я познакомлю вас с таблицами стилей, позволяющими улучшить внешний вид создаваемых вами документов HTML. Таблицы стилей также используются для динамического изменения элементов Web-страницы.

В этой главе мы рассмотрим следующие темы.

- **Предназначение таблиц стилей**
- **Определение каскадных таблиц стилей**
- **Использование таблиц стилей на практике**
- **Использование файлов таблиц стилей**
- **Динамическое изменение стилей с помощью JavaScript**

http://ITBookz.ru/

Стили и внешний вид

При создании действительно привлекательной Web-страницы вы столкнетесь с целым рядом проблем. Первая — HTML не позволяет проводить сложное форматирование страницы. Например, вы не можете изменить интервал между символами, вы даже не можете вставить два пробела между словами — и сразу же преобразуют в один.

Вторая — даже если вам с помощью HTML и удастся создать по-настоящему красивый документ, вы обнаружите, что он неодинаково отображается в разных броузерах или в одинаковых броузерах, запущенных на компьютерах с разной конфигурацией.

Причина ваших проблем очень проста. HTML не предназначен для управления такими средствами документов, как слои, инструменты выравнивания и определения интервалов. HTML управляет *структурой* документа. Другими словами, он определяет параметры абзацев, заголовков, списков и других больших структурных элементов.

Но не стоит создавать из этого проблему. HTML содержит основные средства форматирования документа. Задав структуру документа HTML, вы позволяете отображаться ему по-разному в разных броузерах. Основные элементы при этом остаются неизменными. Например, в Netscape Navigator и Internet Explorer по-разному отображается расстояние между абзацами, размер шрифтов заголовков, полужирный текст и т.п.

Поскольку в HTML определена структура документа, текстовая его часть может отображаться в любых текстовых броузерах, например Lynx. В этом случае привычные вам элементы будут отображаться по-другому, но структура документа и его содержимое остаются неизменными.



Броузеры, отображающие только текст, не единственная альтернатива отображения документов HTML. Броузеры для слепых используют голосовые синтезаторы, позволяющие разное форматирование озвучивать разными голосами в разных тональностях.

Как вы уже поняли, HTML делает то, ради чего он был создан, — задает структуру документа. Не удивительно, что заставить его определять внешний вид отдельных элементов практически бессмысленно.

Но вот настало время, когда разработчики Web-страниц захотели определять детальное форматирование документов HTML. Это привело к возникновению технологии *каскадных таблиц стилей* (Cascade Style Sheet — CSS).

CSS добавляет в HTML целый набор дополнительных средств, позволяющих управлять стилями и определять внешний вид документа. Самое важное, эти средства не имеют никакого отношения к созданию структуры документа. Если с помощью таблиц стилей вам удастся создать документ HTML, одинаково отображаемый во всех броузерах и платформах, считайте, что вы правильно поняли суть CSS.

Приведу пример из жизни. При просмотре документа HTML, созданного с помощью CSS, в броузере, поддерживающем CSS, он будет выглядеть так же, как и в броузере разработчика. Отключив в броузере поддержку CSS, вы сможете увидеть документ HTML в другом, более простом, виде.

Определение и использование стилей CSS

Для того чтобы использовать стили при создании страницы, необходимо сначала их определить. Эта задача выполняется с помощью дескрипторов <STYLE>, определяющих стили самых различных элементов.

Основной дескриптор HTML, использующийся для определения стилей, задает одновременно несколько стилей. Открывающий дескриптор <STYLE> определяет все типы стилей, а закрывающий </STYLE> — обозначает конец их определения. В примере ниже приведено определение стиля CSS:

```
<STYLE TYPE="text/css">
```


http://ITBookz.ru/

Вместо того чтобы использовать стиль, определенный в заголовке документа, вы определяете стиль текущего элемента. Например, следующий оператор определяет синий цвет для шрифта заголовка первого уровня:

```
<H1 STYLE="color: blue">Это синий заголовок</H1>
```

Вы можете разместить дескриптор <STYLE> в теле заголовка документа HTML. В следующих разделах вы познакомитесь с правилами создания стилей и параметрами настройки CSS.

Создание правил

Каждая строка, содержащая дескриптор <STYLE>, называется *правилом*. Чтобы создать правило, необходимо сначала указать элемент HTML, к которому оно относится, а также все свойства и значения, определяющие его вид. Об этих свойствах мы поговорим в следующих разделах.

Приведу простой пример. Следующая таблица стилей имеет только одно правило — все заголовки первого уровня отображаются синим шрифтом:

```
<STYLE TYPE="text/css">
H1 {color: blue}
</STYLE>
```

При определении правил вы можете задавать несколько записей, а также несколько стилей. Например, следующее правило определяет всем заголовкам всех уровней синий шрифт, курсивное начертание и выравнивание по центру:

```
<STYLE TYPE="text/css">
H1,H2,H3,H4,H5,H6 {color: blue;
                   font-style: italic;
                   text-align: center}
</STYLE>
```

В одном правиле можно указывать отдельные элементы HTML или их набор. Если вы задаете правило в дескрипторе <BODY> документа HTML, то стили будут применяться ко всему документу. Правило становится правилом по умолчанию для документа. В этом случае изменить форматирование элемента можно, только указав для него отдельное правило в теле документа.

Выравнивание текста

Одно из часто используемых назначений таблиц стилей — это изменение интервалов между символами и выравнивание текста. Описанные средства недоступны в стандартном HTML. Чтобы изменить выравнивание текста и интервалы, используйте следующие свойства.

- letter-spacing. Определяет расстояние между символами (только в Internet Explorer).
- text-decoration. Позволяет создавать линии над, под и поверх текста и создавать мерцающий текст.
- vertical-align. Позволяет перемещать элемент вверх и вниз, выравнивая его относительно других элементов по вертикали.
- text-align. Определяет выравнивание текста: по правому краю, по левому краю, по центру или по ширине.

http://ITBookz.ru/

- text-transform. Изменяет регистр символов. Значение capitalize делает первые символы каждого слова прописными, uppercase — все символы строчными, а lowercase — все символы прописными.
- text-indent. Определяет отступ абзацев и других элементов.
- line-height. Определяет расстояние между строками текста.

Изменение цвета и рисунка фона

Таблицы стилей, кроме всего прочего, используются еще и для определения цвета и рисунка фона Web-страницы. Для этого в CSS определены следующие свойства.

- color. Определяет цвет текста элемента. Оно используется для привлечения внимания к тексту или при определении для документа цветовой схемы.
- background-color. Определяет фоновый цвет элемента. Определив значение этого параметра, вы добавите привлекательности абзацам, таблицам и другим элементам Web-страницы.
- background-image. Позволяет использовать рисунок в формате GIF в качестве фонового изображения элемента.
- background-repeat. Определяет тип заполнения элемента рисунком фона. Рисунок может периодически повторяться по вертикали или по горизонтали (только в Internet Explorer 4.0 и выше).
- background-attachment. Определяет статус рисунка фона при прокручивании содержимого элемента. Значение fixed определяет закрепленный элемент, который не прокручивается при прокручивании элемента. Значение scroll определяет прокручиваемый рисунок фона (как правило, это значение используется в большинстве Web-страниц). Это свойство применяется только в Internet Explorer 4.0 и выше.
- background-position. Позволяет указать расположение фонового рисунка (только в Internet Explorer).
- background. Позволяет быстро указать все описанные выше параметры в одном списке. Все атрибуты указываются в одном составном правиле.

Управление шрифтами

Таблицы стилей позволяют также управлять шрифтами, используемыми в Web-документах. Для управления шрифтами используются следующие свойства.

- font-family. Определяет название шрифта, например arial или helvetica, используемого в элементе. Поскольку не во всех компьютерах установлены все шрифты, вы можете приводить несколько шрифтов. CSS позволяет указывать характерные шрифты, которые гарантированно устанавливаются во всех компьютерах: serif, san-serif, cursive, fantasy и monospace.
- font-style. Определяет стиль шрифта: нормальный, курсивный, полужирный.
- font-variant. Значение normal соответствует нормальному тексту, а small-caps — отображению прописных символов малыми символами.
- font-weight. Определяет полужирное начертание. Для указания степени полужирности шрифта используются разные числовые значения.
- font-size. Размер символов шрифта.
- font. Позволяет в одном правиле указать все перечисленные выше параметры. Все значения вводятся в одном составном списке.

http://ITBookz.ru/

Границы и поля

И наконец, CSS позволяет управлять форматированием страницы документа. Приведенные ниже свойства определяют параметры полей, границ, а также размеров элементов на Web-странице.

- margin-top, margin-bottom, margin-left, margin-right. Определяют поля элемента. Поля указываются числовыми значениями или в процентах от общих размеров страницы.
- margin. Позволяет указывать все поля в виде одного правила.
- width. Определяет ширину элемента, например изображения.
- height. Определяет высоту элемента.
- float. Позволяет создавать обтекающий элемент текст. Это свойство часто используется при добавлении на страницу рисунков и таблиц.
- clear. Определяет конец обтекания текстом элемента.



Наряду с этими возможностями, CSS позволяет создавать разделы документа, имеющие собственные размеры и поля. Детально об этом мы поговорим в главе "18-й час. Создание динамических страниц с помощью DOM".

Создание простой таблицы стилей

В качестве примера CSS давайте создадим Web-страницу, использующую различные стили. Пусть наш документ HTML подчиняется следующим правилам.

- Во всем документе текст синий.
- Абзацы выровнены по центру и с большими полями с обоих краев.
- Заголовки первого, второго и третьего уровней отображены красным шрифтом.
- Маркированные списки отображаются полужирным зеленым шрифтом.

Листинг 17.1 показывает, как необходимо правильно определить описанные выше правила в синтаксисе HTML.

Листинг 17.1. Простая таблица стилей

```
1:    <STYLE>
2:    BODY {color: blue}
3:    P {text-align: center;
4:         margin-left:20%;
5:         margin-right:20%}
6:  H1, H2, H3 {color: red}
7:    UL {color: green;
8:         font-weight: bold}
9:    </STYLE>
```

Вот по каким принципам построена эта таблица стилей.

- Строки 1 и 9 содержат открывающий и закрывающий дескрипторы <STYLE>.
- Строка 2 определяет цвет по умолчанию для шрифтов как синий.
- Строки 3, 4 и 5 определяют форматирование абзаца.
- Строка 6 определяет цвет шрифтов заголовков первых трех уровней.
- Строки 7 и 8 определяют форматирование маркированных списков.

http://ITBookz.ru/

Для иллюстрации работы кода таблицы стилей включим его в простой документ HTML. Приведем пример применения стилей элементов поверх общих стилей. На рис. 17.1 приведен результат выполнения кода листинга 17.2.

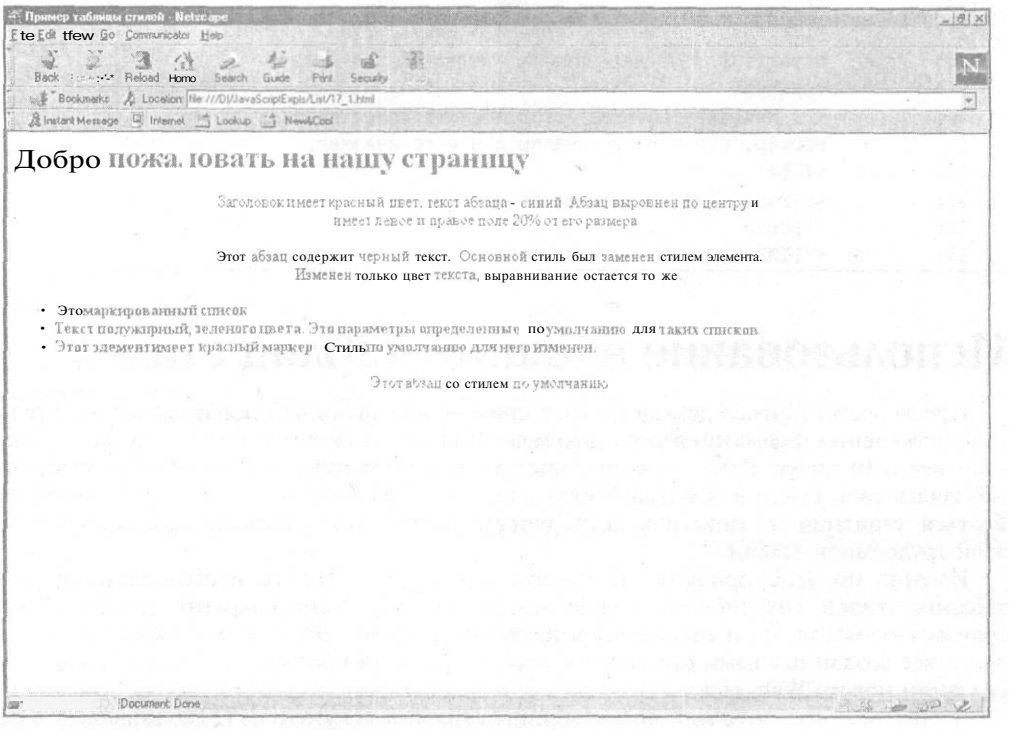


Рис. 17.1. Пример документа HTML с таблицей стилей

Листинг 17.2. Пример документа HTML с таблицей стилей

```
<HTML>
<HEAD><TITLE>Пример таблицы стилей</TITLE>
<STYLE>
BODY {color: blue}
P {text-align: center;
     margin-left:20%;
     margin-right:20%}
H1, H2, H3 {color: red}
UL {color: green;
font-weight: bold}
</STYLE>
</HEAD>
<BODY>
<H1>Добро пожаловать на нашу страницу</H1>
<P>Заголовок имеет синий цвет, текст
абзаца - красный. Абзац выровнен по центру
и имеет левое и правое поле 20% от его размера.
</P>
<P STYLE="color:black">Этот абзац содержит черный
текст. Основной стиль был заменен стилем элемента.
```


http://ITBookz.ru/

```
21:          Изменен только цвет текста, выравнивание остается
22:          то же.</P>
23:          <UL>
24:         <LI>Это маркированный список
25:          <LI>Текст полужирный, зеленого цвета. Это
26:          параметры, определенные по умолчанию для таких
27:          списков.
28:          <LI STYLE="color:red">Этот элемент имеет красный
29:          маркер. Стиль по умолчанию для него изменен.
30:          </UL>
31:          <P>Этот абзац со стилем по умолчанию</P>
32:          </BODY>
33:         </HTML>
```

Использование внешних таблиц стилей

Предыдущий пример демонстрирует применение таблицы стилей для незначительного изменения форматирования документа HTML. Таблица стилей в его коде занимает всего 10 строк. Если вы хотите сделать вашу страницу красочной и собираетесь назначать свои стили для большинства параметров форматирования, то вам стоит набраться терпения и запастись достаточным количеством времени для выполнения этой трудоемкой задачи.

Именно по этой причине вы можете в документе HTML использовать готовые таблицы стилей, сохраненные в виде отдельного файла. Это сократит не только программу документа, но и сэкономит ваше рабочее время. Это также позволит вам привести все созданные вами страницы к одному форматированию, что используется при создании нового Web-узла.

Чтобы добавить внешний файл таблицы стилей в документ HTML, включите в дескриптор <STYLE> специальную команду его подсоединения. Эта задача выполняется с помощью дескриптора <LINK>, размещаемого в заголовке документа:

```
<LINK REL=STYLESHEET TYPE="text/css" HREF="style.css">
```

Этот дескриптор позволяет ссылаться на внешний файл таблицы стилей style.css.

Управление таблицами стилей в JavaScript

Новая стандартизированная модель DOM упрощает процедуру управления таблицами стилей на странице. Не зависимо от того, используете вы таблицы стилей в сценарии или нет, JavaScript позволяет изменить стиль любого элемента страницы.

Как вы узнали в главе "10-й час. Работа с объектной моделью документа", модель DOM используется для управления любым элементом документа HTML. Управление элементами документа HTML осуществляется с помощью объектов. Чтобы изменить любой стиль объекта достаточно модифицировать его значения свойств в элементе style.

В области элемента style вы можете указывать свойства и значения, описанные в этой главе. Например, для изменения цвета элемента достаточно задать значение параметра style.color:

```
element.style.color="blue"
```

В данном случае element — это объект элемента. Существует огромное количество способов определения объекта броузеру. Детально об этом рассказано в следующей главе.

http://ITBookz.ru/

В основном для задания элементу объекта достаточно создать его идентификатор. В следующем примере элементу <H1> определен идентификатор head1:

```
<H1 ID="head1">Это заголовок</H1>
```

После определения идентификатора вы можете найти соответствующий элементу объект с помощью методы GetElementById:

```
element.getElementById("head1");
```

Используя такую методику, можно немного автоматизировать форматирование объекта следующим образом:

```
document.getElementById("head1").style.color="blue";
```

Этот оператор определяет синий цвет элементу head1 страницы. В следующем разделе применена похожая методика для задания динамических стилей.

Создание динамических стилей

С помощью DOM вы можете создавать страницы со средствами управления цветами страницы. Для начала создадим форму, позволяющую выбирать цвета на странице. В листинге 17.3 приведен классический пример задания раскрывающихся списков с помощью дескриптора <select>, определяющих цвета текста заголовка и тела документа.

Листинг 17.3. Форма изменения цветов элементов

```
<FORM NAME="form1">
<B>Цвет текста абзаца:</B>
<SELECT name="body" onChange="changebody();">
     <option value="red">Красный</option>
     <option value="blue">Синий</option>
     <option value="green">Зеленый</option>
     <option value="yellow">Желтый</option>
     <option value="black">Черный</option>
</SELECT>
<BR>
<B>Цвет текста заголовка:</B>
<SELECT name="heading" onChange="changehead();">
     <option value="red">Красный</option>
     <option value="blue">Синий</option>
     <option value="green">Зеленый</option>
     <option value="yellow">Желтый</option>
     <option value="black">Черный</option>
</SELECT>
</FORM>
```



Если вы не понимаете полностью синтаксис приведенного кода, то обратитесь к главе "14-й час. Формы введения данных".

Заметьте, что в этом коде определены операторы вызова двух функций – changehead и changebody. Эти функции проводят изменение цвета соответствующих элементов.

В листинге 17.4 приведен код обеих функций в том виде, в каком он используется.

http://ITBookz.ru/

Листинг 17.4. Сценарий задания функций изменения цветов элементов

```
<SCRIPT LANGUAGE="JavaScript">
function changehead() {
      i=document.form1.heading.selectedIndex;
      headcolor=document.form1.heading.options[i].value;
     document.getElementById("head1").style.color=headcolor;
}
function changebody() {
      i=document.form1.body.selectedIndex;
   doccolor=document.form1.body.options[i].value;
      document.getElementById("p1").style.color=doccolor;
}
</SCRIPT>
```

В строках 2–6 задана функция changehead. В ней сначала считывается индекс выбранного цвета, который затем сопоставляется с одним из цветов. В строке 5 используется метод getElementByld, описанный в предыдущей главе. Функция changebody задана в 7–11 строках листинга.

Нам осталось только добавить текст заголовка и тела страницы, цвет которых будет впоследствии изменяться. Листинг 17.5 содержит полный код документа.

Листинг 17.5. Пример использования динамических стилей

```
<HTML>
<HEAD>
<TITLE>Применение динамических стилей</TITLE>
 <SCRIPT LANGUAGE="JavaScript">
function changehead() {
      i=document.form1.heading.selectedIndex;
      headcolor=document.form1.heading.options[i].value;
     document.getElementById("head1") .style.color=headcolor;
}
 function changebody() {
     i=document.form1.body.selectedIndex;
   doccolor=document.form1.body.options[i].value;
      document.getElementById("p1").style.color=doccolor;
}
</SCRIPT>
</HEAD>
<BODY>
<H1 ID="head1">
 Цвет заголовка задается динамическим стилем</H1>
<HR>
<p ID="p1">
 Выберите цвет заголовка и абзаца в приведенной ниже форме. Все
указанные цвета изменяются автоматически после выбора нового значения
 в списке.</p>
<FORM NAME="form1">
<B>Цвет текста абзаца:</B>
<SELECT name="body" onChange="changebody();">
      <option value="red">Красный</option>
      <option value="blue">Синий</option>
```


http://ITBookz.ru/

```
30:              <option value="green">Зеленый</option>
31:               <option value="yellow">Желтый</option>
32:              <option value="black">Черный</option>
33:          </SELECT>
34:          <BR>
35:          <B>Цвет текста заголовка:</B>
36:          <SELECT name="heading" onChange="changehead();">
37:              <option value="red">Красный</option>
38:              <option value="blue">Синий</option>
39:              <option value="green">Зеленый</option>
40:              <option value="yellow">Желтый</option>
41:              <option value="black">Черный</option>
42:          </SELECT>
43:          </FORM>
44:          </BODY>
45:         </HTML>
```

Примите к сведению, что дескриптор <H1> в строке 18 содержит параметр ID со значением head1. В строке 21 этот параметр дескриптора <p> имеет значение p1. Эти значения используются в методе getElementById в строках 8 и 13.

На рис. 17.2 показан результат выполнения последнего кода. Загрузите код в браузере, выберите цвет абзаца и заголовка. Удостоверьтесь, что сценарий выполняется правильно.

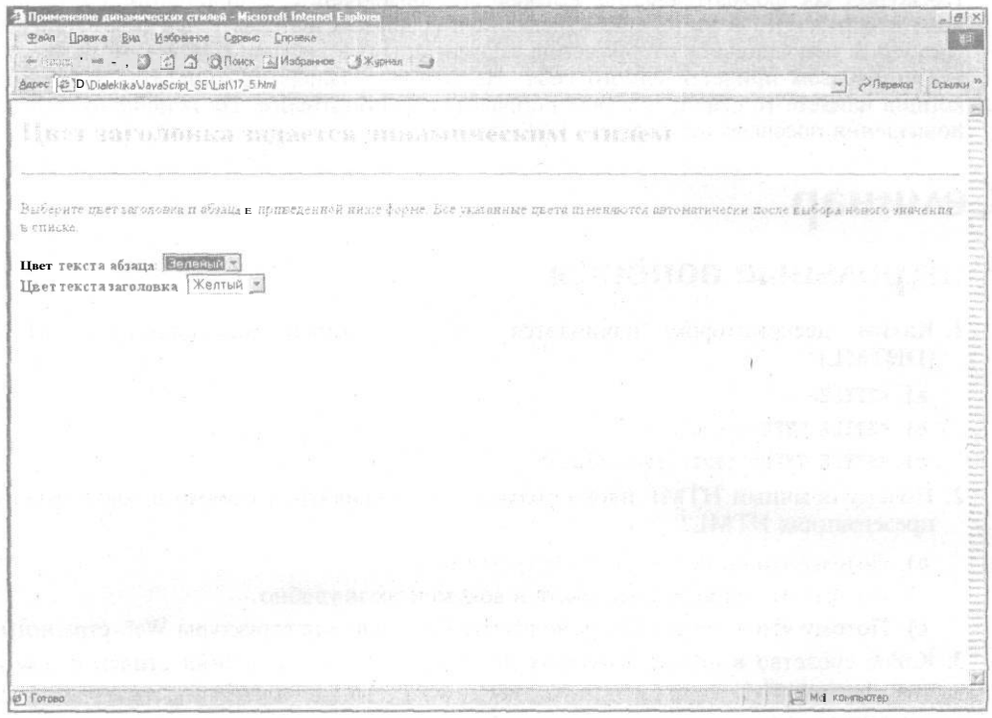


Рис. *17.2. Применение динамических стилей*

http://ITBookz.ru/

Резюме

На этом занятии вы познакомились с таблицами стилей, которые позволяют красивее оформить Web-документ. Вы узнали о CSS и синтаксисе таблиц стилей в JavaScript и научились применять полученные знания на практике.

В следующей главе я расскажу о слоях — еще одном средстве стандарта DOM.

Вопросы и ответы

Какая разница между применением для форматирования документов традиционных дескрипторов, таких как <B> и <I>, и таблиц стилей?

С формальной точки зрения разницы нет. Если вы хотите сделать текст полужирным или курсивным, то лучше не применять таблицы стилей. Если же вы хотите применить более сложное форматирование, то без таблиц стилей вам просто не обойтись.

Что произойдет, если к одному и тому же тексту применить две разные таблицы стилей?

Технология CSS позволяет применять к одному элементу несколько разных таблиц стилей. Если операции форматирования обеих таблиц стилей не перекрываются, то текст приобретает форматирование обоих таблиц. При перекрывании операций форматирования текст приобретает форматирование той таблицы, которая применялась к нему последней.

Что делать, если пользователям не нравится форматирование созданными мною стилями?

Поскольку все форматирование, которое вы применяете с помощью стилей, одинаково отображается в разных броузерах, нельзя винить используемый пользователем броузер за неправильное отображение тех или иных элементов. Как это ни печально, но винить вам придется самого себя. Не бойтесь экспериментировать, и в конце концов найдете те стили, которые понравятся пользователям. Их и используйте для повышения посещаемости вашего Web-узла.

Семинар

Контрольные вопросы

1. Каким дескриптором начинается таблица стилей динамического HTML (DHTML)?
   a) <STYLE>
   b) <STYLE TYPE="text/css">
   c) <STYLE TYPE="text/javascript">
2. Почему обычный HTML плохо подходит для управления слоями в документах и презентациях HTML?
   a) Потому что он разработан программистами
   b) Потому что слои использовать в документах неудобно
   c) Потому что язык HTML используется для задания структуры Web-страницы
3. Какое средство в новых броузерах используется для изменения стилей с помощью JavaScript?
   a) HTML 4.0
   b) DOM
   c) CSS 2.0

http://ITBookz.ru/

Ответы

1, b) Таблица стилей CSS начинается с дескриптора `<STYLE TYPE="text/css">`

2, c) HTML разрабатывался как язык задания структуры документа

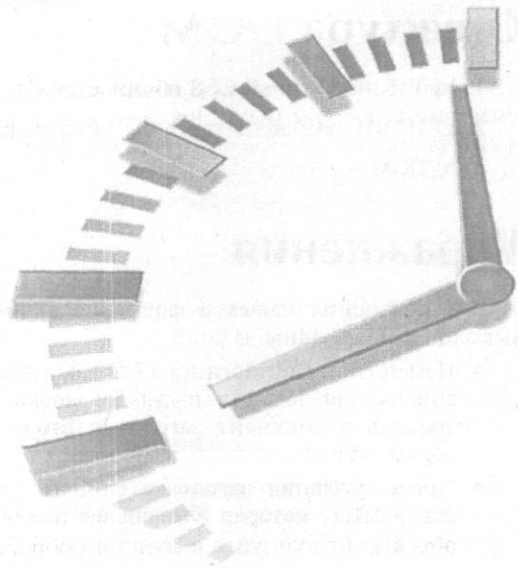
3, b) DOM

Упражнения

Для получения навыков форматирования документов с помощью таблиц стилей выполните следующие задания.

- Измените код листинга 17.5 так, чтобы создать отдельный раскрывающийся список для задания цвета заголовка второго уровня. Измените листинг 17.5, добавив в документ заголовок второго уровня, и протестируйте измененный документ с двумя фреймами.
- После создания заголовка второго уровня добавьте в листинг 17.5 функцию `ChangeColor`, которая возвращает идентификатор соответствующего изменяемого объекта; используйте в функции обработчик событий onChange.

http://ITBookz.ru/



18-й час

Создание динамических страниц с помощью DOM

В предыдущих уроках постоянно вспоминалась спецификация DOM (объектная модель документа), С ее помощью JavaScript осуществляет управление объектами документа и броузера. В предыдущем уроке вы узнали, как с помощью объектов динамически изменяются стили.

В настоящем уроке вы познакомитесь с другими средствами DOM, которые используются для управления документом Web и научитесь перемещать объекты на странице. В этой главе рассмотрены следующие темы.

- **Структура объектов DOM**
- **Узлы, родительские и дочерние объекты, а также дерево объектов**
- **Создание перемещаемых слоев**
- **Управление слоями с помощью JavaScript**
- **Слои в старых броузерах**
- **Создание анимации с помощью DOM**

http://ITBookz.ru/

Структура DOM

В главе "10-й час. Работа с объектной моделью документа" вы уже познакомились с элементами структуры DOM. Документ window включает в себя объект document и т.д. Эти два объекта понимаются и в старых броузерах. Этого не скажешь о новых объектах DOM, которые на сегодня сопоставляются практически всем элементам документа HTML.

Чтобы разобраться во всем многообразии структуры DOM, давайте рассмотрим документ HTML, приведенный в листинге 18.1. Этот документ имеет область заголовка и область тела (в качестве содержимого введен один абзац текста и заголовок к нему).

Листинг 18.1. Простой документ HTML

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Простой документ HTML</TITLE>
4:  </HEAD>
5:  <BODY>
6:  <H1>Это заголовок</H1>
7:  <P>Это абзац обычного текста</P>
8:  </BODY>
9:  </HTML>
```

Подобно всем документам HTML этот составлен из разных *контейнеров* и их содержимого. Дескрипторы <HTML>—</HTML> — это контейнер всего документа, дескрипторы <BODY>—</BODY> представляют собой контейнер для тела документа и т.д.

В DOM каждый контейнер страницы и его содержимое представляется отдельным объектом. Объекты организованы в древовидную структуру с корневым объектом document. Ветви дерева структуры составляют тело и заголовок документа. На рис. 18.1 показана схема объектной структуры DOM.

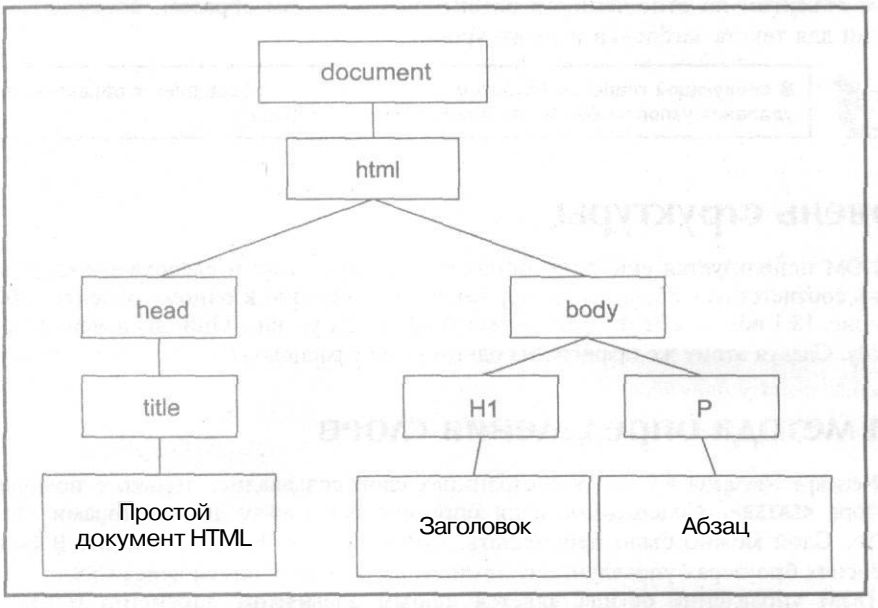


Рис. 18.1. Документ HTML в представлении DOM

http://ITBookz.ru/

В следующих разделах вы познакомитесь со структурой DOM детальнее.



Не теряйтесь, если структура объектов пугает вас своей громоздкостью. В любой момент вы можете определить элементу идентификатор (ID) и задать в коде ссылку на него. Этот метод активно использовался в примерах предыдущей главы. С ним вы также встретитесь и в этой главе при создании анимации. В следующей главе вы создадите сложный пример управления объектами. Чтобы понять его, вам придется досконально изучить структуру DOM.

Элемент

Каждый контейнер или объект в документе в DOM называется *элементом* (node). На рис. 18.1 каждый объект схемы, помещенный в прямоугольник, — это элемент. Линии представляют соотношения отдельных объектов.

Очень часто приходится в сценариях ссылаться на отдельные узлы. Для этого определяются идентификаторы (ID) или название объекта задается соответственно ее иерархическому расположению в структуре.

Родительские и дочерние объекты

Из предыдущих глав книги вы уже узнали, что каждый объект JavaScript может выступать в роли *родительского* по отношению к другим объектам. Объект, содержащийся в родительском, называется дочерним (естественно, по отношению к родительскому). В DOM используется та же терминология.

На рис. 18.1 объект document является родительским по отношению к остальным объектам документа. Объект html родительский для объектов body и head, а h1 и p — дочерние объекты объекта body.

Текстовые узлы представляются немного по-другому. Текст абзаца выступает дочерним объектом по отношению в объекту p. Подобным образом, объект h1 — родительский для текста заголовка первого уровня.



В следующей главе вы познакомитесь с методами обращения к объектам, а также удаления узлов из документа и добавления к документу.

Уровень структуры

В DOM используется еще одно понятие — *уровень*. Как и следовало ожидать, один уровень соответствует объектам, дочерним по отношению к одному объекту дерева.

На рис. 18.1 объекты h1 и p расположены на одном уровне. Они оба дочерние для объекта body. Следуя этому же правилу, на одном уровне расположены объекты body и head.

Два метода определения слоев

В Netscape Navigator 4.0 на Web-страницах слои создавались только с помощью дескриптора <LAYER>. Содержимое слоя определялось между дескрипторами <LAYER> и </LAYER>. Слой можно было перемещать, скрывать и отображать заново. В более новых версиях броузерах управление слоями стало осуществляться через CSS.

В DOM управление осуществляется любым элементом документа HTML. Слой, группа слоев или других элементов в DOM управляется с помощью объекта .

http://ITBookz.ru/



Дескриптор <DIV> впервые стали использовать в HTML 3.0. Он определяет произвольную часть документа HTML и сам по себе не задает форматирование содержащегося *в* нем текста. Именно поэтому он идеален для определения слоев и таблиц стилей.

Чтобы создать слой с помощью дескриптора <DIV>, заключите его содержимое в пару дескрипторов <DIV> и определите свойства после атрибута STYLE. Вот как это необходимо сделать:

```
<DIV ID="layer1" STYLE="position:absolute; left:100; top=100">
Это содержимое слоя
</DIV>
```

Этот код определяет слой layerl. Это перемешаемый слой, смещенный относительно левого верхнего угла окна броузера на 100 по вертикали и 100 пикселей по горизонтали. Детально со свойствами слоев вы познакомитесь в следующем разделе.

Определение свойств слоев

В атрибуте STYLE дескриптора <DIV> можно указывать самые различные свойства: расположение слоя, его состояния отображения и другие параметры. Ниже приведены все свойства слоев.

- position. Это главный параметр, определяющий расположение слоя в окне. Он принимает три значения.
  - static. Определяет неперемещаемые слои, размещенные в области документа HTML. Это значение по умолчанию.
  - absolute. Определяет координаты расположения слоя.
  - relative. Определяет слой, сдвинутый относительно нормального расположения в документе HTML, указанного значением static.
- left и top. Определяют сдвиг элемента. Для значения absolute отсчет ведется относительно окна броузера, а для relative — относительно нормального расположения, задаваемого значением static.
- width и height. Подобны стандартным атрибутам WIDTH и HEIGHT языка HTML. Определяют размер слоя.
- clip. Определяет границу на слое. Отображается только та часть слоя, которая расположена внутри границы.
- overflow. Указывает на усечение границей слоя, а также использование полосы прокрутки для отображения остального изображения слоя. Принимает значения none, clip и scroll.
- z-index. Определяет расположение слоя. Самый нижний индекс имеет значение этого индекса 1. Слой, расположенный над ним, имеет индекс 2 и т.д. Указывая индексы, вы определяете порядок "наслоения" изображения. Это свойство используется в JavaScript для расширенного управления слоями.
- visibility. Определяет отображаемость слоя. Принимает значения visible (по умолчанию), hidden и inherit. Значение inherit используется для отображения слоя только при сосуществовании с ним другого слоя.
- background color. Определяет цвет фона только текста на слое.
- layer-background-color. Определяет цвет фона всего слоя (независимо от того, содержит тот текст или нет).
  - background-image. Определяет рисунок, который используется в качестве фонового только для текста.
  - layer-background-image. Определяет рисунок фона всего слоя (независимо от того, содержит тот текст или нет).

http://ITBookz.ru/

Расположение слоя

Как вы уже знаете из предыдущих глав, параметры стиля объекта определяются дочерним объектом style. Все описанные в предыдущем разделе свойства задаются для слоя так же, как и для остальных объектов.

Предположим, вы создали слой с помощью дескриптора .

```
<DIV ID="layer1" STYLE="position:absolute; left:100; top=100">
<p>Это содержимое слоя</p>
</DIV>
```

Для перемещения слоя вниз или вверх на странице измените значение оператора style.top. Чтобы сместить объект, например на 200 пикселей, используйте следующий код:

```
var obj=document.getElementById("layer1");
obj.style.top=200;
```

Метод document.getElementByID возвращает объект, определенный дескриптором . Второй оператор определяет смещение объекта на 200 пикселей. Как вы знаете из предыдущей главы, этих два оператора можно объединить в один:

```
document.getElementById("layer1").style.top=200;
```

Именно этот прием используется в при создании анимационного изображения.

Управление старыми броузерами

Несмотря на то, что новая спецификация DOM позволяет управлять слоями, стилями или целыми документами, вам необходимо побеспокоиться о поддержке новых средств в старых броузерах. Все новые стандарты поддерживаются только в Internet Explorer версии 5.0 или выше и Netscape версии 6.0. и выше.

Более того, поскольку DOM относительно недавно представлена на рассмотрение разработчиков Web-страниц, часто возникают ситуации, в которых те или иные средства не поддерживаются относительно новыми броузерами. Поэтому при создании сценариев с использованием DOM обязательно необходимо тестировать их в как можно большем количестве броузеров (как новых, так и старых).

Также вы должны учитывать, что огромное количество пользователей используют старые броузеры и не собираются переходить к их новым версиям. Поскольку Netscape Navigator и Internet Explorer версии 4.0 поддерживают подавляющее большинство средств, представленных в DOM, но управляют ими по-своему, вам обязательно необходимо добавлять в документ HTML код определения типа и версии броузера.



Детально об определении типа и версии броузера рассказано в главе "16-й час. Создание сценариев для разных броузеров".

В частности, методика размещения слоя, изученная в этой главе, применима и в Internet Explorer, и Netscape Navigator версии 4.0. Оба броузера поддерживают использование CSS и дескриптор . Слой для этих броузеров задается так же, как и для броузеров, поддерживающих DOM.

```
<DIV ID="layer1" STYLE="position:absolute; left:100; top=100">
<p>Это содержимое слоя</p>
</DIV>
```

Разница ощущается только при задании расположения броузера. Вот как задается смешение слоя в DOM (вы уже это знаете):

```
var obj=document.getElementById("layer1");
obj.style.top=200;
```


http://ITBookz.ru/

В Internet Explorer 4.0 используется подобный синтаксис, но все объекты сохраняются в объекте document.all. Вот как это выглядит:

```
var obj=document.all.layer1;
obj.style.top=200;
```

В Netscape 4.0 для смещения слоя используется массив layers. Вот как с помощью массива можно изменить расположение слоя:

```
var obj=document.layers["layer1"]
obj.top=200;
```

Как вы видите, новые средства DOM действительно решают проблему совместимости документа HTML с новыми броузерами. Но оставляют открытой проблему совместимости со старыми броузерами.

Создание анимации с помощью слоев

Создание анимации — это одно из интереснейших занятий при программировании Web-страниц. В главе "15-й час. Добавление рисунков и анимации" вы использовали средства JavaScript для создания простого анимационного изображения. С помощью слоев вы сможете создать движущуюся мышь другим способом.

Давайте немного усложним задачу и (поскольку я не умею рисовать ничего другого, кроме мыши) создадим три движущиеся мыши, используя для этого DHTML. Использование слоев для решения этой задачи позволит не только упростить ее, но и создать более плавную картину перемещения мыши. Листинг 18.2 содержит программный код анимации требуемого изображения.

Листинг 18.2. Сценарий анимационного изображения

```
1:  <HTML>
2:  <HEAD>
3: <TITLE>Создание анимации с помощью DHTML</TITLE>
4:  <SCRIPT LANGUAGE="JavaScript">
5:  var pos1=-95;
6:  var pos2=-95;
7:  var pos3=-95;
8:  var speed1=Math.floor(Math.random()*10)+2;
9: var speed2=Math.floor(Math.random()*10)+2;
10: var speed3=Math.floor(Math.random()*10)+2;
11: function next(){
12:      pos1+=speed1;
13:      pos2+=speed2;
14:      pos3+=speed3;
.15:     if (pos1>795) pos1=-95;
16:     if (pos2>795) pos2=-95;
17:     if (pos3>795) pos3=-95;
18:     document.getElementById("mouse1").style.left=pos1;
19:     document. getElementById("mouse2").style.left=pos2;
20:     document. getElementById("mouse3").style.left=pos3;
21:     window.setTimeout("next();",10);
22: }
23: </SCRIPT>
24: </HEAD>
25: <BODY onLoad="next();">
26: <H1>Создание анимации с помощью DHTML</H1>
```


http://ITBookz.ru/

```
27:         <HR>
28:         <DIV ID="mouse1" STYLE="position:absolute; left=0;
28:        top=100; width=100; height=100; visibility:show">
30:        <IMG src="mouse5.gif" width=100 height=100 alt="" border="0">
31:         </DIV>
32:         <DIV ID="mouse2" STYLE="position:absolute; left=0;
33:        top=200; width=100; height=100; visibility:show">
34:        <IMG src="mouse5.gif" width=100 height=100 alt="" border="0">
35:         </DIV>
36:        <DIV ID="mouse3" STYLE="position:absolute; left=0;
37:        top=300; width=100; height=100; visibility:show">
38:        <IMG src="mouse5.gif" width=100 height=100 alt="" border="0">
39:         </DIV>
40:         </BODY>
41:         </HTML>
```

Загрузите полученный программный код в броузер. Помните, что, поскольку в этом примере использованы средства DOM, для его выполнения необходим Internet Explorer версии 5.0 и выше или Netscape Navigator версии 6.0 и выше. На рис. 18.2 показана полученная Web-страница в Internet Explorer 5.0.

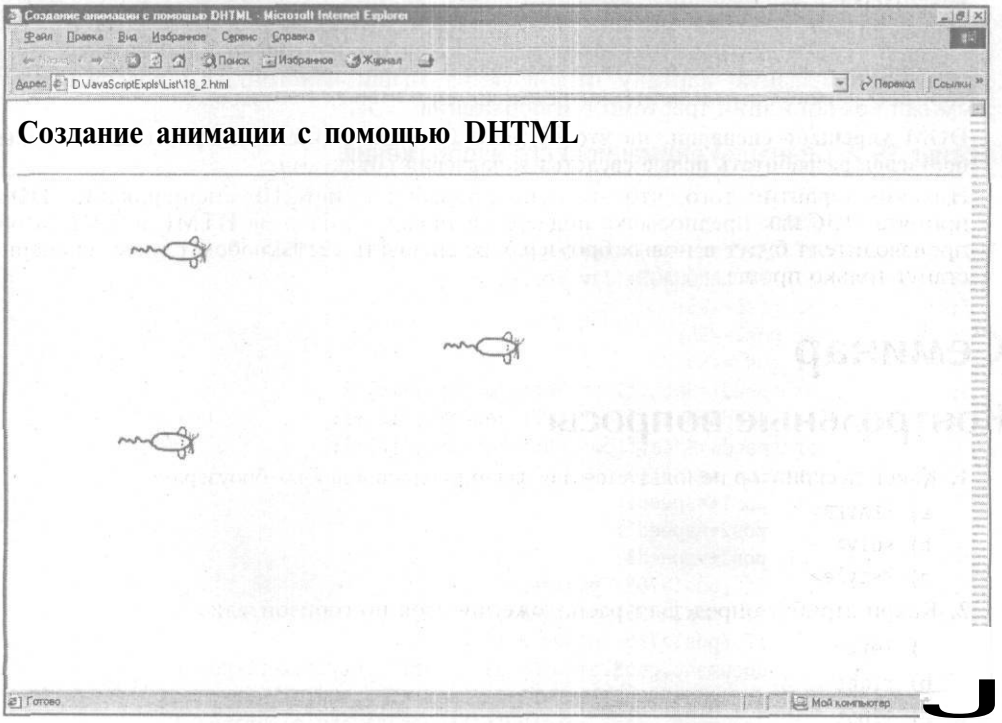


Рис. 18.2. Пример анимации в Internet Explorer 5.0.

http://ITBookz.ru/

Резюме

В этом занятии вы узнали о управлении объектами с помощью DOM, а также научились изменять расположение объектов на странице.

В следующей главе вы познакомитесь с другими средствами DOM, среди которых свойства и методы, необходимые для перемещения по дереву DOM, добавления элементов и изменения теста на странице.

Вопросы и ответы

Что произойдет, если моя Web-страница состоит из нескольких документов HTML (фреймов)?

В этом случае каждое окно или кадр имеет собственный объект document, в котором сохранены все его элементы.

Я создал страницу, использующую DOM, но она не выполняется даже в Internet Explorer 5.0. Чтобы это значило?

По непонятным причинам номер версии Internet Explorer 5.0 определяется как 4.0. Это и вызывает неправильное определение версии броузера.

Если DOM позволяет динамически изменять объект, зачем тогда использовать дескриптор ?

Вы правы, можно вместо него использовать объект <img>. Я использовал дескриптор , поскольку он легче поддерживается в старых броузерах.

DOM упрощает сценарии, но что не дает *Netscape* и *Microsoft* при создании новых броузеров разработать новые средства управления объектами?

Никаких гарантий того, что этого не произойдет, нет. Но спецификация DOM принята *W3C* как предпосылка поддержки новых стандартов HTML и XML, а оба производителя будут в новых броузерах реализовать ее. В любом случае, сценарии станут только проще.

Семинар

Контрольные вопросы

1. Какой дескриптор используется для задания слоев в любых броузерах?
   a) <LAYER>
   b)
   c) <style>
2. Какой атрибут определяет расположения слоя по горизонтали?
   a) Left
   b) right
   c) lrpos
1. Какой объект поддерживается в Internet Explorer версии 4.0 и выше, но не входит в стандарт DOM?
   a) document
   b) layer
   c) document.all

http://ITBookz.ru/

Ответы

1, b) Дескриптор используется для создания броузеров и в старых броузерах.

2, a) Атрибут left определяет расположение слоев по горизонтали

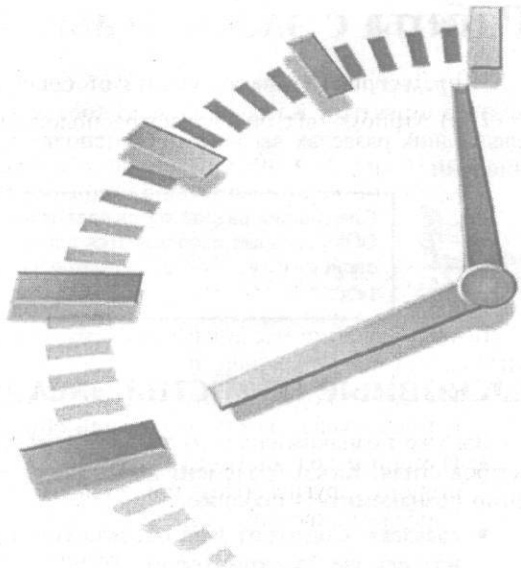
3, b) Только дескриптор document.all поддерживается в Internet Explorer версии 4.0 и выше, но не описывается в стандарте DOM.

Упражнения

Если вы планируете посвятить себя созданию многослойных рисунков, то выполните следующие упражнения.

- Измените код листинга 18.2 так, чтобы в нем определялась версия броузера и использовался массив layers или объект document.all.
- Измените код листинга 18.2 так, чтобы одна мышь бежала в противоположную сторону другим (вам придется нарисовать новую мышь, иначе она у вас "побежит" хвостом вперед)

http://ITBookz.ru/



19-й час

Дополнительные средства DOM

В течение предыдущего часа вы изучили древовидную структуру DOM и узнали, как элементы документа HTML представляются в ней. В качестве примера использования DOM вы создали анимированный объект.

В настоящей главе я подробнее остановлюсь на описании DOM, ее свойствах и методах. В ней вы познакомитесь с несколькими примерами динамических Web-страниц, в которых используется DOM. В этой главе рассмотрены следующие вопросы.

- **Использование свойств элементов DOM**
- **Методы управления элементами DOM**
- **Скрытие и отображение объектов на странице**
- **Изменение текста на странице**
- **Добавление текста на страницу**
- **Создание бегущих сообщений**

http://ITBookz.ru/

Работа с элементами DOM

В предыдущей главе вы узнали об организации объектов на Web-странице в древовидную структуру. Каждый элемент (объект) дерева управляется средствами JavaScript. В следующих разделах вы научитесь использовать свойства и методы элементов управления ими.



Следующий раздел описывает только самые важные методы и свойства элементов DOM, которые используются последними версиями броузеров. Полный список всех свойств и методов вы найдете на узле *W3C* по адресу `http://www.w3.org/TR/DOM-Level-2/`.

Основные свойства элемента

Вы уже познакомились со свойством style, позволяющим изменять значения параметров стиля. Каждый элемент имеет также и другие свойства, с которыми вам необходимо познакомиться поближе.

- `nodeName`. Содержит имя объекта (не идентификатор). Для элементов, созданных на основе дескриптором HTML, в качестве названия используется имя дескриптора. Например, `BODY`, `HEAD` или `P`. Для элементов документа используются специальные названия. Например, `#document` или `#text`.
- nodeType. Целочисленное значение, описывающее тип элемента. Для обычных дескрипторов HTML определено значение 1, для текстовых элементов — 3 и 9 - для остальных элементов документа.
- `nodeValue`. Это текст, сохраняемый в текстовом элементе.



Свойство nodeValue для нетекстовых элементов недоступно. Не существует более простого способа определить содержимое элемента HTML, чем с помощью дескрипторов `BODY` и `DIV`.

Свойства связей элементов

В дополнение к основным свойствам, описанным выше, каждый элемент обладает набором свойств, описывающих его взаимодействие с другими элементами. Среди них следующие.

- firstChild. Первый дочерний объект текущего элемента. Для текстовых элементов, например `h1` и `p`, — это сам текст.
- lastChild. Последний дочерний объект элемента.
- childNodes. Массив всех дочерних элементов текущего элемента. Для управления этими элементами удобно использовать цикл.
- previousSibling. Это предыдущий элемент того же уровня, что и текущий элемент.
- nextsibling. Это следующий элемент того же уровня, что и текущий элемент.



Помните, что, подобно всем объектам и свойствам JavaScript, свойства и функции элемента, описанные в этой главе, чувствительны к регистру символов. Поэтому внимательно определяйте их в будущем.

http://ITBookz.ru/

Методы документа

Элемент document имеет несколько часто используемых методов, с которыми вам не стоит познакомиться. Вы уже встречались с методом getElementById. Он позволяет обращаться к объектам DOM по идентификатору. Среди методов элемента такие:

- getElementById(*идентификатор*). Возвращает элемент по указанному идентификатору.
- getElementByTagName(*дескриптор*). Возвращает массив элементов с определенным названием дескриптора. В качестве общего символа используйте символ звездочки (*).
- createTextNode(*текст*). Создает новый элемент, содержащий введенный в скобках текст.

Методы элемента

Каждый элемент страницы имеет собственные методы. Доступность методов определяется от расположения элемента и его связей с остальными элементами.

- appendChild(*новый объект*). Добавляет новый элемент после всех дочерних элементов текущего элемента.
- insertBefore(*новый, старый*). Добавляет новый дочерний объект перед указанным дочерним объектом, который создан раньше.
- replaceChild(*новый, старый*). Заменяет указанный старый элемент новым.
- removeChild(*старый*). Удаляет указанный элемент из набора дочерних элементов.
- hasChildNodes(). Возвращает булево значение: true соответствует существованию у текущего элемента дочерних, а false возвращается в случае их отсутствия.

Скрытие и отображение объектов

Давайте теперь перейдем к рассмотрению реальных примеров использования объектов DOM для управления Web-страницами. В качестве простого примера мы создадим сценарий скрытия и отображения объектов на странице.

Как вы уже знаете из главы "17-й час. Использование таблиц стилей", объекты имеют свойство visibility, определяющее вид объекта на странице:

```
object.style.visibility="hidden";//объект скрыт
object.style.visibility="visible";//объект отображен
```

В сценарии очень просто изменять значения этого свойства и добиваться скрытия и отображения элемента. Тем не менее, описанный метод применим только в броузерах Internet Explorer версии 5.0 и выше. В JavaScript это свойство не используется. Поэтому в них следует использовать альтернативный метод:

```
object.style.display="none";//объект скрыт
object.style.display="";//объект отображен
```

Помня об этом, давайте создадим сценарий скрытия объектов в обоих типах броузеров. Листинг 19.1 содержит код документа HTML скрытия и отображения двух заголовков.

http://ITBookz.ru/

Листинг 19.1. Скрытие и отображение объектов

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Скрытие и отображение заголовков</TITLE>
4:  <SCRIPT LANGUAGE="JavaScript">
5:  function ShowHide(){
6:       var head1=document.getElementById("head1");
7:       var head2=document.getElementById("head2");
8:       var showhead1=document.form1.head1.checked;
9:       var showhead2=document.form1.head2.checked;
10:      if (navigator.userAgent.indexOf("Netscape6")!=-1){
11:           head1.style.visibility=(showhead1)?"visible":"hidden";
12:           head2.style.visibility=(showhead2)?"visible":"hidden";
13:      }else{
14:           head1.style.display=(showhead1)?"":"none";
15:           head2.style.display=(showhead2)?"":"none";
16:      }
17: }
18: </SCRIPT>
19: </HEAD>
20: <BODY>
21: <h1 ID="head1">Это первый заголовок</h1>
22: <h1 ID="head2">Это второй заголовок</h1>
23: <p>С помощью DOM вы можете выбирать вид заголовка,
24: используя флажки опций.</p>
25: <FORM NAME="form1">
26: <INPUT TYPE="checkbox" name="head1"
27:      checked onClick="ShowHide();">
28: <B>Отобразить первый заголовок</B><BR>
29: <INPUT TYPE="checkbox" name="head2"
30:      checked onClick="ShowHide();">
31: <B>Отобразить второй заголовок</B><BR>
32: </FORM>
33: </BODY>
34: </HTML>
```



Помните, что примеры этой главы используют DOM, и поэтому требуют для выполнения Netscape 6.0 или Internet Explorer 5.0 и выше. В этих броузерах все примеры этой главы выполняются.

В строках 21-22 этого кода задаются заголовки с идентификаторами head1 и head2. Строки 26–33 определяют форму, содержащую два флажка опций. Каждый флажок соответствует своему заголовку. Если выставить флажок опции, соответствующий заголовок появится в окне броузера (обработчик onclick вызывает функцию ShowHide).

Эта функция определена в дескрипторе <SCRIPT> в строках 4-18. Строки 6-7 задают переменные head1 и head2 с помощью метода getElementById. В строках 8–9 переменные showhead1 и showhead2 определяют опции флажков.

Наконец, строки 10-16 непосредственно задают операции скрытия и отображения заголовков. Строка 10 проверяет броузер на принадлежность к семейству Netscape 6.0, а строки 11–12 скрывают и отображают. В остальных броузерах заголовки скрываются и отображаются строками 14–15.

http://ITBookz.ru/



Строки 10–11 и 14-15 выглядят немного странно. Символы ? и : задают условные выражения в операторе if. Детально условные выражения рассмотрены в главе "7-й час. Тестирование и сравнение значений".

Рис. 19.1 демонстрирует созданный нами пример в действии. На рисунке отображен только один заголовок, поскольку один флажок опции уже выставлен.

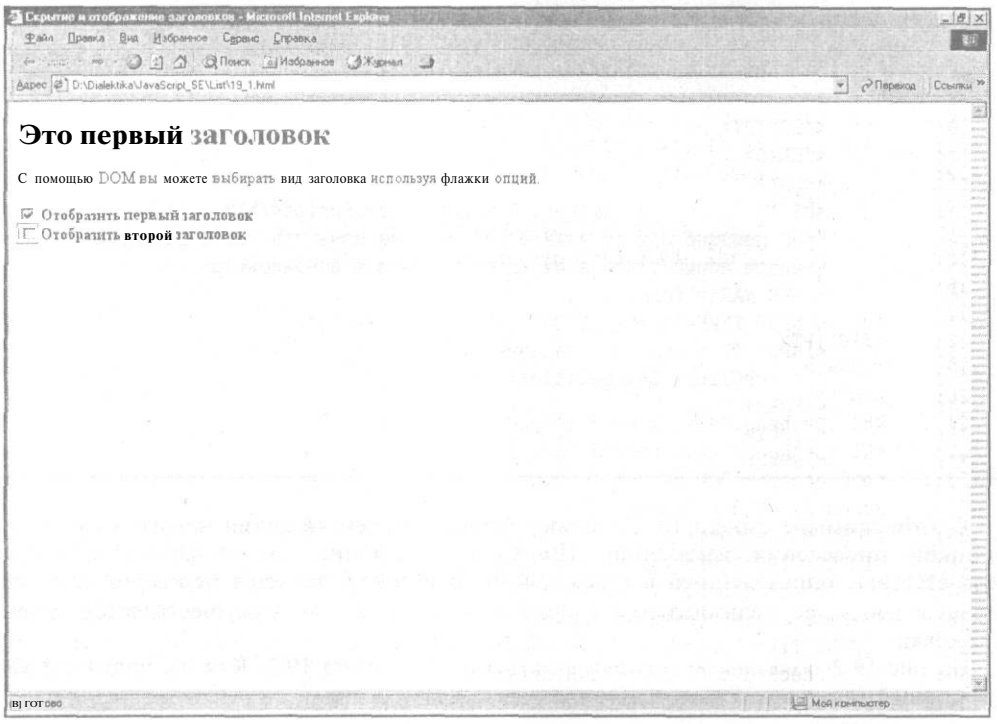


Рис. 19.1. Пример страницы, на которой скрывается и отображается текст с помощью элементов управления формы

Изменение текста на странице

Теперь давайте создадим сценарий, позволяющий изменять содержимое заголовка непосредственно на Web-странице. Как вы знаете из предыдущего урока, свойство nodeValue содержит текст элемента (объекта), а сам текст является дочерним элементом по отношению к дескриптору заголовка. Поэтому код изменения текста заголовка с идентификатором head1 выглядит следующим образом:

```
var head1=document.getElementById("head1");
head1.firstChild.nodeValue="Новый текст";
```

В этом листинге переменной head1 определяется объект заголовка. Свойство firstChild возвращает текст элемента, дочернего по отношению к самому заголовку, а свойство nodeValue содержит этот текст.

Воспользовавшись этим приемом, можно просто создать Web-страницу, позволяющую динамически изменять заголовок. В листинге 19.2 приведен листинг кода этого документа HTML.

http://ITBookz.ru/

Листинг 19.2. Пример изменения текста на Web-странице

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Динамическое изменение текста</TITLE>
4:  <SCRIPT LANGUAGE="JavaScript">
5:  function ChangeTitle() {
6:       var newtitle=document.form1.newtitle.value;
7:       var head1=document.getElementById("head1");
8:       head1.firstChild.nodeValue=newtitle;
9:  }
10: </SCRIPT>
11: </HEAD>
12: <BODY>
13:  <hl ID="head1">Динамический текст в JavaScript</hl>
14: <p>С помощью DOM вы можете динамически изменить текст заголовка.
15: Введите новый текст и щелкните на кнопке Изменить<Ур>
16:  <FORM NAME="form1">
17: <INPUT TYPE="text" NAME="newtitle" SIZE="25">
18: <INPUT TYPE="button" value="Изменить"
19:       onClick="ChangeTitle();">
20: </FORM>
21: </BODY>
22: </HTML>
```

В этом примере строки 18–22 задают форму с полем введения нового заголовка и кнопкой проведения изменения. Щелкнув на кнопке, вы вызываете функцию ChangeTitle(), определенную в строках 4–10. В строке 6 задается переменная, сохраняющая вводимое пользователем значение. В строках 7 и 8 осуществляется замена заголовка.

На рис 19.2 показана выполненная страница листинга 19.2. Как вы видите, я уже ввел новый заголовок и заменил старый.

Добавление текста на страницу

Следующим этапом изучения DOM будет добавление на Web-страницу текста. Чтобы выполнить эту задачу, необходимо сначала создать новый текстовый элемент. Оператор создания нового текстового элемента выглядит следующим образом:

```
var node=document.createTextNode("это тест");
```

Теперь вставим этот элемент в документ HTML. Эта задача выполняется с помощью метода appendChild. Текст можно вставить в любой элемент, уже содержащий текст. Пусть нам необходимо вставить текст в абзац готового текста. Приведенный ниже код демонстрирует, как новый текст вставляется в конец абзаца p1:

```
document.getElementById("p1").appendChild(node);
```

В листинге 19.3 показан документ HTML, в котором реализован создаваемый нами пример добавления текста на Web-страницу.

http://ITBookz.ru/

Листинг 19.3. Добавление текста на страницу

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Добавление текста</TITLE>
4:  <SCRIPT LANGUAGE="JavaScript">
5:  function AddText() {
6:       var sentence=document.form1.sentence.value;
7:       var node=document.createTextNode("" + sentence);
8:       document.getElementById("p1").appendChild(node);
9:  }
10: </SCRIPT>
11: </HEAD>
12: <BODY>
13: <H1>Введите новое предложение</H1>
14: <p ID="p1">С помощью DOM вы динамически вставляете новое предложение
15: в готовый абзац текста. Введите предложение и щелкните на кнопке
16: Добавить.</p>
17: <FORM NAME="form1">
18: <INPUT TYPE="text" NAME="sentence" SIZE="65">
19: <INPUT TYPE="button" VALUE="Добавить" onClick="AddText();">
20: </FORM>
21: </BODY>
22: </HTML>
```

В этом примере в строках 14–16 определен абзац, к которому добавляется текст. Строки 17–20 задают форму с текстовым полем, в котором вводится новое предложение, и кнопкой запуска функции добавления текста. Функция AddText определена в строках 4–10.

В строке 6 определена переменная sentence. Ей присваивается текстовое значение вводимого на форме предложения. Следующая строка создает на основе нового текста новый элемент DOM. В строке 8 осуществляется добавление нового элемента к уже существующему абзацу.

Загрузите документ в броузер и протестируйте его. Добавьте несколько предложений. На рис. 19.3 показано, как документ выполняется в Internet Explorer 5.5.

Великолепное бегущее сообщение

В главе "6-й час. Использование массивов и строковых данных" вы создали сценарий Web-страницы с бегущим сообщением в строке состояния броузера. С помощью DOM вы можете создавать более эффектные бегущие сообщения — непосредственно в области страницы.

Чтобы выполнить эту задачу, давайте частично воспользуемся кодом, приведенным в шестом уроке. Вместо того чтобы определить сообщению переменную `window.status`, создадим для него новый элемент страницы. В листинге 19.4 приведен полный код измененного документа HTML с бегущей строкой.

http://ITBookz.ru/

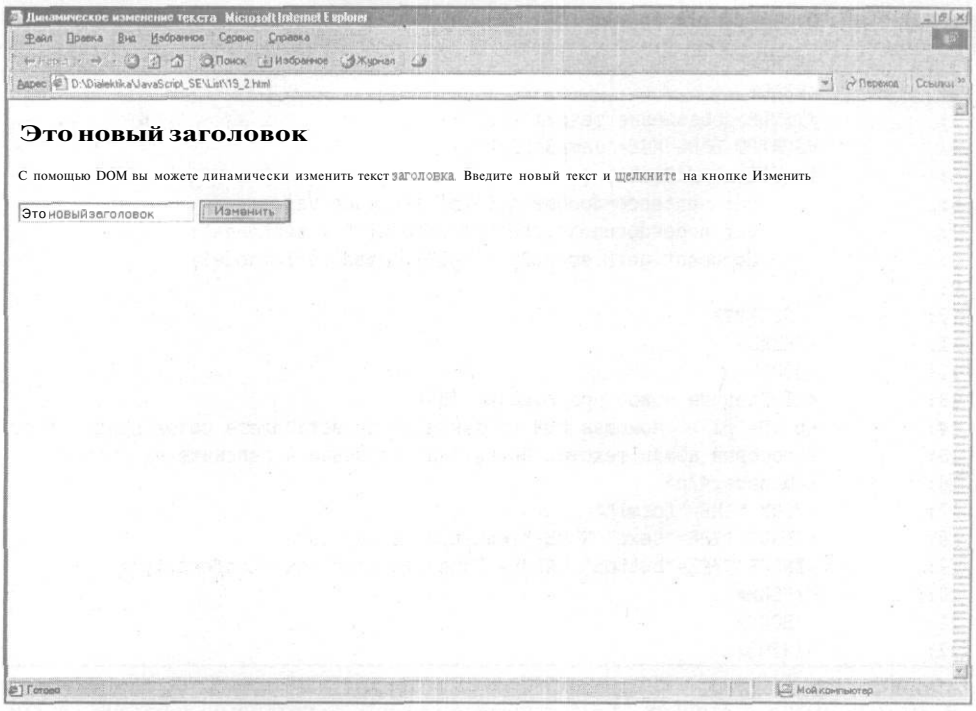


Рис. 19.2. Web-страница с измененным заголовком

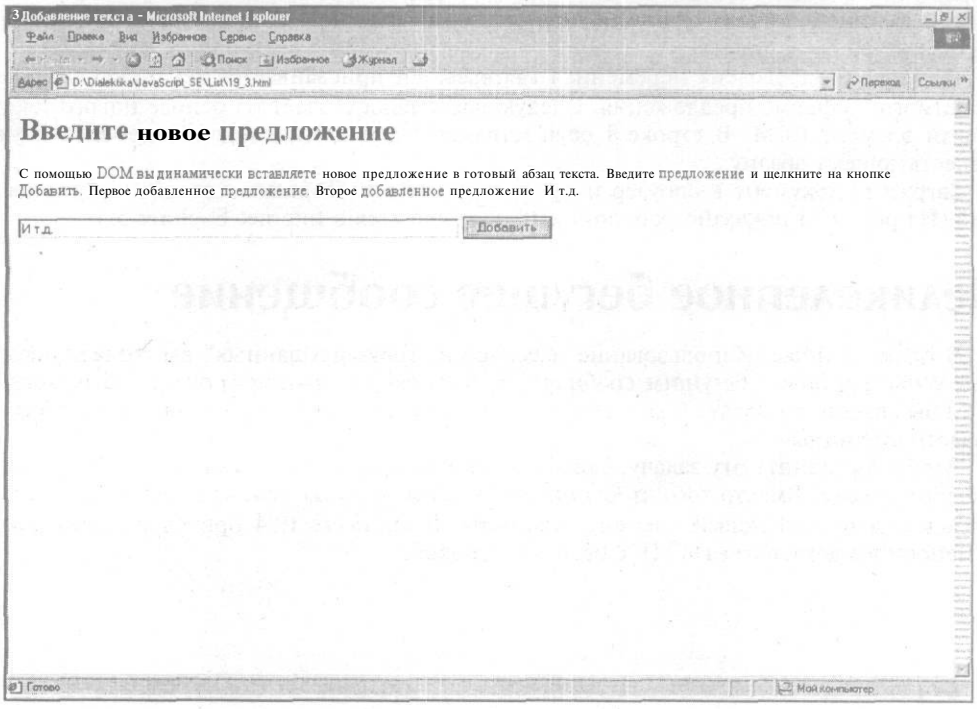


Рис. 19.3. Добавление текста в окне броузера

http://ITBookz.ru/

Листинг 19.4. Обновленное бегущее сообщение

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Пример бегущего сообщения</TITLE>
4:  <SCRIPT LANGUAGE="JavaScript">
5:  msg="Это пример бегущего сообщения.  ";
6:  msg+="Оно отображается в теле страницы,";
7:  msg+=" а не строке состояния";
8:  spacer="...                    ...";
9:  pos=0;
10: function ScrollMessage() {
11:      var newtext=msg.substring(pos, msg.length)+
12:           spacer+msg.substring(0,pos);
13:      var td=document.getElementById("scroll");
14:      td.firstChild.nodeValue=newtext;
15:      pos++;
16:      if (pos > msg.length) pos=0;
17:      window.setTimeout("ScrollMessage()",200);
18: }
19: </SCRIPT>
20: </HEAD>
21: <BODY onLoad="ScrollMessage();">
22: <h1>Улучшенное бегущее сообщение</h1>
23: <p>Вместо строки состояния теперь бегущее сообщение
24: отображается непосредственно на странице</p>
25: <TABLE width="90%" border>
26: <tr>
27: <td ID="scroll" width="90%">Бегущее сообщение</td>
28: </tr>
29: </TABLE>
30: </BODY>
31: </HTML>
```

В этом примере строки 25–29 задают простую таблицу, содержащую бегущее сообщение. Дескриптор <td> в строке 27 — это идентификатор элемента scroll **и контейнер текста бегущего сообщения.**

В строках 5–9 объявляются переменные сценария; функция ScrollMessage **задана в строках 10–18. Эта функция вызывается обработчиком** onLoad.

Строки 11-12 создают бегущее сообщение на основе строки newtext. **В строке 13 ячейке таблицы, в которой** "бежит" **сообщение, определяется переменная, а строка 14 необходима для задания элементу ячейки текста сообщения.**

Наконец, строки 15-16 определяют значение счетчика pos. В строке 17 реализуется вызов функции ScrollMessage **после небольшой задержки. На рис. 19.4 показан выполненный в окне броузера код листинга** 19.4.



В этом примере вокруг бегущего сообщения создается рамка. В случае необходимости вы можете создавать бегущее сообщение не только в ячейке таблицы, но и в заголовке, абзаце и т.д.

http://ITBookz.ru/

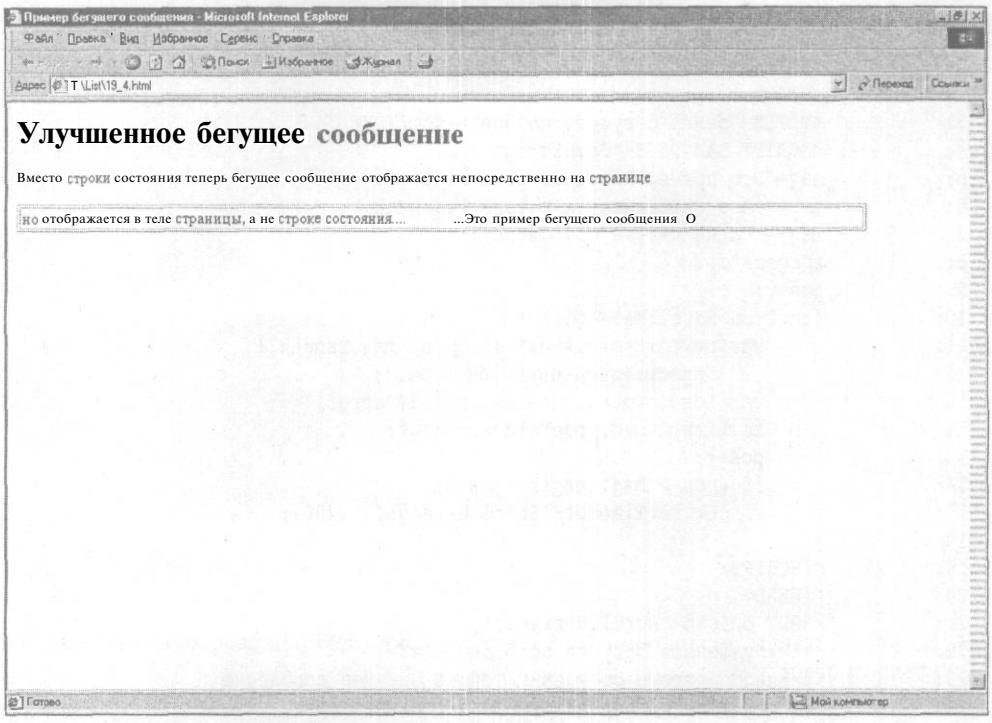


Рис. 19.4. Обновленное бегущее сообщение на странице

Резюме

За этот час вы познакомились с основными средствами DOM. Вы изучили ее методы и свойства и научились управлять объектами DOM. Вы также создали несколько простых примеров использования DOM на Web-странице. В конце урока создали пример бегущего сообщения.

В следующем уроке вы познакомитесь еще с одним аспектом использования JavaScript — добавлением мультимедийных данных и встраиваемых модулей. Эти элементы использовать намного проще, нежели объекты DOM. В то же время они позволяют добиться ошеломляющих эффектов.

Вопросы и ответы

Можно ли определить идентификатор ID любому объекту DOM, используемому в сценарии?

Конечно, да. В сценариях этой главы все основные объекты DOM определены идентификатором ID. Кроме того, вы можете определять объект по названиям свойств. Например, `firstChild` или nextsibling.

Может ли текст, определенный как элемент DOM, содержать дескриптор HTML, например `<b>`?

Нет. Содержимое текстовых элементов ограничено только текстом. Чтобы создать более сложный элемент, вам необходимо удалить текущий и заменить его новым.

http://ITBookz.ru/

Существует ли полное руководство по методам и свойствам DOM для каждой версии броузеров?

Да. Несколько Web-узлов постоянно обновляют списки свойств и методов для разных типов броузеров. Некоторые из них описаны в Приложении А.

Семинар

Контрольные вопросы

1. ЕСЛИ para1 — **это объект** DOM для **абзаца, то какой правильный синтаксис для изменения его текста на** New Text?

   a) `para1.value="New Text";`

   b) `para1.firstChild.nodeValue="New Text";`

   c) `para1.nodeValue="New Texy";`

2. **Какой из** следующих **объектов DOM является родительским?**

   a) body

   b) div

   c) document

3. **Какой синтаксис правильно определяет объект DOM для заголовка с идентификатором** head1?

   a) `document.getElementById("head1")`

   b) `document.GetElementById("head1")`

   c) `document.getElementByID("head1")`

Ответы

1, b) Реальный текст представляется элементом nodeValue, **дочерним по отношению к элементу абзаца**

2, c) Объект `document` — **это родительский объект по отношению ко всем остальным в дереве DOM.**

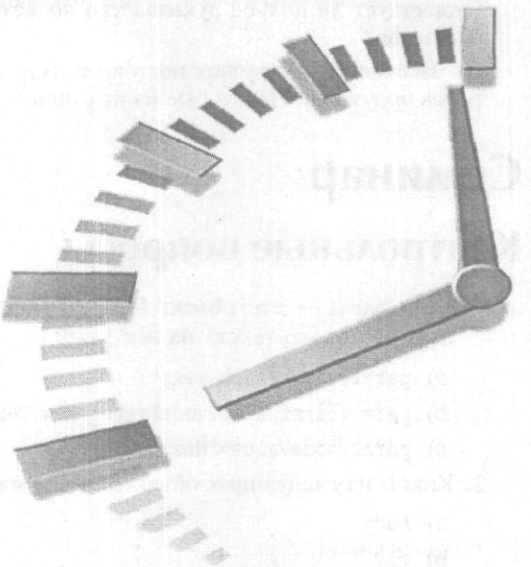
3, a) Второй вариант отличается написанием G вместо д, а третий D вместо d.

Упражнения

Чтобы закрепить свои познания об элементах DOM и их использовании, выполните следующие упражнения.

- Добавьте третий флажок в листинг 19.1 для текста абзаца. Добавьте идентификатор к дескриптору <p>.
- Измените листинг 19.3 так, чтобы вводимый текст добавлялся в начале абзаца, а не в конце. Замените метод appendChild методом `insertBef`ore.
- Измените листинг 19.4 таким образом, чтобы сообщение "бежало" в заголовке, а не ячейке таблицы.

http://ITBookz.ru/



20-й час

Использование мультимедиа и встроенных утилит

Это последняя глава в части V. В предыдущих трех главах вы познакомились с некоторыми дополнительными средствами JavaScript: таблицами стилей, слоями и дескрипторами DHTML. На этом занятии вы будете изучать встроенные утилиты.

Встроенные утилиты — это внедренные в броузер надстройки, позволяющие использовать в документах HTML всевозможные дополнительные данные — звуковые и видеоклипы и т.п. С помощью JavaScript вы можете управлять надстройками, добавляя в документ HTML разные эффекты.

В этой главе рассмотрены следующие темы.

- **Взаимосвязь между LiveConnect, JavaScript, Java и утилитами**
- **Определение утилит в JavaScript**
- **Проверка доступных типов MIME**
- **Использование объектов утилит**
- **Создание приложения, проигрывающего музыку**

http://ITBookz.ru/

Что такое LiveConnect

Встроенные утилиты впервые появились при выпуске Netscape Navigator 4.0. Вместо того чтобы встроить в броузер поддержку средств добавления дополнительных типов данных (например, форматированного текста, музыки или видеоклипов), *Netscape* создала модульную архитектуру, позволяющую разработчикам создавать собственные надстройки в соответствии с решаемыми ими задачами.

Создано уже несколько сотен встроенных утилит. Ниже приведены наиболее популярные из них.

- ShockWave и Flash фирмы *Macromedia* поддерживают анимационные изображения и видеоклипы.
- Adobe Acrobat позволяет использовать на любых платформах форматированный текст.
- RealPlayer поддерживает потоковый звук и видеоизображения.
- Beatnik компании *Headspace* позволяет использовать музыку.

Эти надстройки можно загрузить с Web-узлов разработчиков. Netscape Navigator содержит несколько основных встроенных утилит.

- QuickTime для воспроизведения видеоклипов.
- LiveAudio для воспроизведения музыки в формате MIDI.
- Типичные пакеты RealPlayer, ShockWave и Headspace.

Поскольку все они включены в состав пакета Netscape, можете быть уверены, что по меньшей мере половина пользователей ими пользуются. В конце этой главы приведен пример использования LiveAudio для воспроизведения музыкального клипа.

поддерживает надстройки, но, как правило, других версий. Кроме того, некоторые надстройки запускаются только на определенных платформах (например, Windows или Macintosh). Все примеры, приведенные в этой главе, универсальны и запускаются на любых платформах, только в Netscape Navigator версии 3.0 и выше.



Все примеры этой главы используют модули, поддерживаемые в обеих платформах. Один только пример определения поддерживаемых утилит требует использования Netscape Navigator версии 3.0 и выше.

Файл, который запускается встроенной утилитой, подсоединяется на Web-страницу с помощью дескриптора `<EMBED>`. Этот дескриптор содержит название файла и необходимые параметры, которые определяют способ добавления в документ HTML внешнего файла.

Типы MIME

Многоцелевые расширения электронной почты в сети Internet (Multipurpose Internet Mail Extensions — MIME) — это стандарт классификации разных типов файлов и способов передачи их с помощью Internet. Различные типы файлов имеют разные *типы MIME.*

Вы уже использовали некоторые типы MIME: HTML (тип text/html), текст (тип text/plain) и GIF (тип image/gif). Хотя Web-броузеры по умолчанию не поддерживают много типов MIME, их поддержки можно добиться с помощью встроенных утилит и надстроек.

Отправляя документ HTML в броузер, Web-сервер включает его тип MIME в заголовок документа. Если броузер поддерживает указанный тип MIME, то файл отображается в окне броузера. Если нет, то на экране появится вопрос о назначении этого файла (подобный выводимому после щелчка на архивном или выполняемом файле при его загрузке).

http://ITBookz.ru/

Использование LiveConnect

Вы можете часто использовать в своих сценариях дополнительные модули, но по умолчанию они не выполняются. Для их выполнения применяется утилита LiveConnect. LiveConnect — это стандарт *Netscape,* позволяющий JavaScript, Java и дополнительным модулям взаимодействовать между собой.

LiveConnect позволяет управлять параметрами как объектами. Эти объекты сохраняются в иерархической структуре объектов броузера. Их впоследствии можно использовать в программах Java и JavaScript.



LiveConnect также используется для добавления команд Java в сценарии JavaScript и для установки взаимосвязи между Java и JavaScript. Детальную информацию об этом вы найдете на Web-узле *Netscape:*

`http://www.netscape.com/docs/manuals/liveconnect.html`

Управление объектами утилит

Объект navigator, с которым вы познакомились в главе "10-й час. Работа с объектной моделью документа" содержит дочерний объект `plugins`. Этот объект представляет собой массив, элементы которого представляют установленные в броузере дополнительные утилиты.



К сожалению, объект `navigator.plugins` поддерживается только в Netscape Navigator версии 3.0 и выше.

Каждый элемент массива имеет следующие свойства:

- name. Имя установленной надстройки;
- filename. Имя выполняемого файла утилиты;
- description. Описание утилиты, предоставленное разработчиком;
- mimeTypes. Массив типов MIME, поддерживаемых установленной надстройкой.

Эти свойства используются в сценариях управления надстройками и их файлами. Детально об их использовании рассказано в следующих разделах.



Объект navigator также выступает в роли дочернего по отношению к объекту mimeTypes. Напомню, что объект mimeTypes — это массив типов MIME, использующихся установленной в броузере утилитой.

Проверка утилит

Что же вам делать, если необходимо воспользоваться новой встроенной утилитой, но вы не уверены в том, что она установлена в вашем броузере? Тут вам опять пригодится JavaScript. С его помощью вы сможете узнать, установлена ли необходимая вам утилита, перед загрузкой страницы.

Листинг 20.1 содержит код простого сценария проверки наличия в броузере утилиты QuickTime (или другой надстройки, воспроизводящей клипы QuickTime). Если эта утилита находится в броузере, то автоматически загружается дескриптор `<EMBED>`, позволяющий воспроизвести клип. В противном случае отображается неподвижный кадр.

http://ITBookz.ru/

Листинг 20.1. Определение утилиты в броузере

```
1:          test=navigator.mimeTypes["video/quicktime"];
2:          if (test)
3:               document.writeln("<EMBED SRC='quick.mov' HEIGHT=100 WIDTH=100>");
4:          else
5:               document.writeln("<IMG SRC='quick.gif' HEIGHT=100 WIDTH=100>");
```

Если сценарий не находит необходимой надстройки, то можете добавить к неподвижному рисунку ссылку на Web-узел, с которого ее можно загрузить, или альтернативную страницу, которая не содержит мультимедийных данных.

Перечень утилит

Массив plugins можно использовать еще с одной полезной целью — отобразить список всех установленных в броузере утилит. Листинг 20.2 содержит код документа HTML, с перечнем установленных в броузере надстроек, оформленный в виде таблицы, в которой заданы также имена выполняемых файлов и описания утилит.

Листинг 20.2. Список установленных в броузере утилит

```
1:          <HTML>
2:          <HEAD>
3:          <TITLE>Список утилит</TITLE>
4:          </HEAD>
5:          <BODY>
6:          <H1>Список утилит</H1>
7:          <HR>
8:          Следующий список содержит названия установленных в
9:          броузере утилит. Он создан с помощью объекта
10:         navigator.plugins.
11:         <HR>
12:         <TABLE BORDER>
13:         <TR><TH>Название утилиты</TH>
14:         <TH>Имя файла</TH>
15:         <TH>Описание</TH>
16:         </TR>
17:         <SCRIPT LANGUAGE="JavaScript">
18:         for (i=0; i<navigator.plugins.length; i++) {
19:         document.write ("<TR><TD>");
20:         document.write(navigator.plugins[i].name);
21:         document.write("</TD><TD>");
22:         document .write(navigator.plugins [ i ].filename);
23:         document.write ("</TD><TD>");
24:         document. write(navigator.plugins[i].description);
25:         document.write("</TD></TR>");
26:         }
27:         </SCRIPT>
28:         </TABLE>
29:         </BODY>
30:         </HTML>
```

Все выполняемые сценарием действия задаются в строках 18–26. Строка 18 открывает тело цикла. Цикл выполняется столько раз, сколько элементов имеет массив plugins. Таким образом отображаются названия и описания всех утилит (элементов массива).

http://ITBookz.ru/

Операторы document.write, приведенные в строках 19–25, отображают свойства объекта plugins, оформленные в виде таблицы. Результат выполнения сценария показан на рис. 20.1.

Использование в надстройках объектов

Теперь вы знаете, как с помошью объекта navigator определить наличие в броузере той или иной утилиты. Благодаря LiveConnect вы также можете для управления утилитами использовать их объекты.

Каждая установленная в броузере утилита представлена отдельным элементом в массиве embeds, дочернем по отношению к массиву document. Например, если документ содержит всего один встроенный объект, то обратиться к нему можно следующим образом:

```
document.embeds[0]
```



В то время как приведенный выше пример выполняется только в Netscape, массив embeds и пример простого пианино, создаваемого в следующем разделе, выполняются и в Internet Explorer (версии 4.0 и выше).

Свойства и методы объекта embed могут быть самыми разными, в зависимости от установленной надстройки. Например, LiveAudio имеет методы play и stop, позволяющие управлять воспроизведением музыкального файла. Некоторыми утилитами управлять в сценариях вообще нельзя.

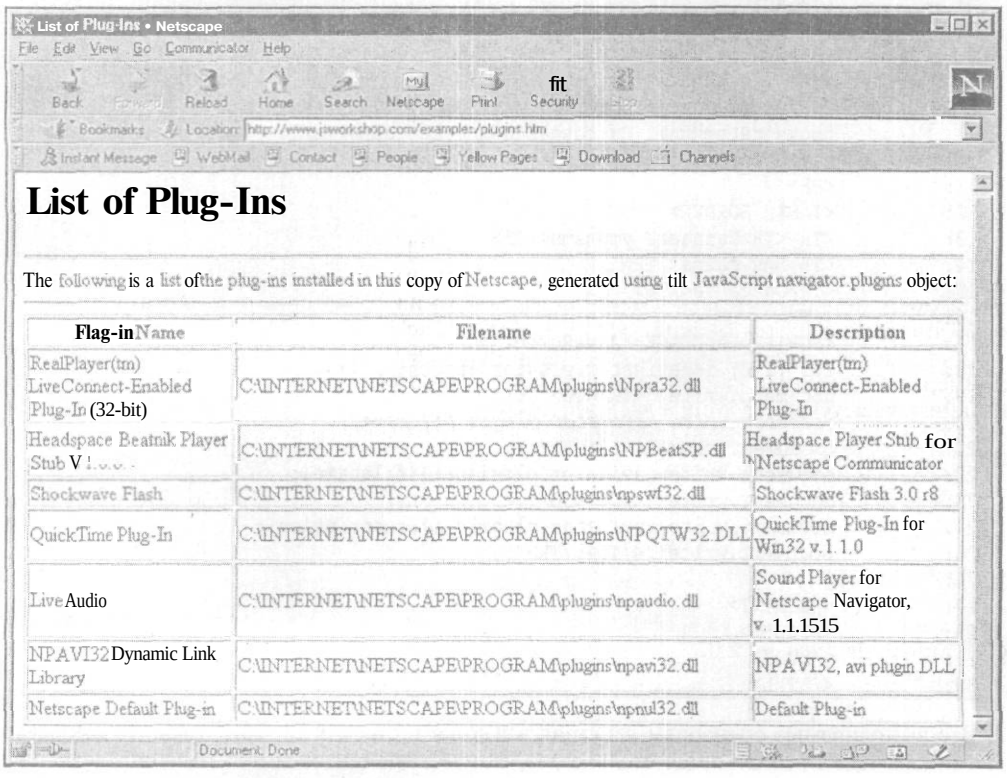


Рис. 2ft Л *Список надстроек, установленных в Netscape Navigator*

http://ITBookz.ru/

Воспроизведение музыки с помощью мыши

В качестве примера управления встроенной утилитой с помощью JavaScript давайте создадим сценарий воспроизведения музыки. Я предварительно создал рисунок клавиатуры пианино и записал отдельные ноты в виде файлов. Сценарий будет определять щелчок мышью на "клавише" и воспроизводить соответствующую ей ноту.



Вы можете загрузить звуковые файлы и рисунки этого примера с узла http://www.jsworkshop.com.

Вставка звуковых файлов

Для нашего примера были специально созданы звуковые файлы в формате WAV для каждой из 13 клавиш пианино. Каждый файл имеет название, соответствующее воспроизводимой им ноте. В документ HTML эти файлы вставляются с помощью дескриптора <EMBED>.

Обычно при вставке звуковых файлов они воспроизводятся во время загрузки страницы. Вам не нужно, чтобы одновременно воспроизводились все 13 нот (уши этого не выдержат). Поэтому в сценарий вводится параметр AUTOSTART=false, предотвращающий воспроизведение звука при загрузке страницы.

В программный код также включен параметр HIDDEN=true, не позволяющий отображать для каждого файла свою клавиатуру. В результате дескриптор <EMBED> выглядит так:

```
<EMBED SRC="C0.wav" HIDDEN=true AUTOSTART=false>
```

Поскольку с помощью этого дескриптора воспроизводится каждый файл, все они будут загружены одновременно. Это позволит предотвратить загрузку каждого отдельного файла при щелчке на его "клавише".

Отображение клавиатуры

Для управления воспроизведением звуковых файлов создан рисунок клавиатуры пианино в формате GIF. Он состоит из последовательных белых и черных клавиш, задаваемых отдельными файлами. Для отображения 13 клавиш последовательно и поочередно размещаются рисунки whitekey.gif и blackkey.gif.

Для пущего эффекта можно использовать разделенный рисунок. В сценарии этого примера разделенный рисунок не используется из соображений простоты. Отдельная клавиша задается следующим дескриптором:

```
<A HREF="#" onClick="playnote(0);">
  <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
```

Таким образом отображается белая клавиша. Она управляется обработчиком событий onclick, запускающим функцию воспроизведения файла.

Воспроизведение звуков

Функция playnote() отвечает за воспроизведение звуков после щелчка на клавише. Она запрашивается переменной note (принимает значения 0–11, в зависимости от указанной ноты) и запускает для необходимого файла метод play. Вот как она выглядит:

```
function playnote(note) {
document.embeds[note].play();
}
```


http://ITBookz.ru/

Составление программы

Листинг 20.3 содержит готовый программный код документа HTML, имитирующего пианино. Этот листинг может показаться вам громоздким, но большинство строк в нем повторяются 13 раз: 13 раз определяется дескриптор <EHBED> ссылки. Каждая ссылка имеет собственный параметр, определяющий свой файл воспроизводимой ноты.

Листинг 20.3. Документ HTML, имитирующий пианино

```
<HTML>
<HEAD>
 <TITLE>Piano</TITLE>
 <SCRIPT LANGUAGE="JavaScript">
function playnote(note) {
 document.embeds[note].play();
 }
 </SCRIPT>
</HEAD>
<BODY>
<EMBED SRC="C0.wav" HIDDEN=true AUTOSTART=false>
<EMBED SRC="cs0.wav" HIDDEN=true AUTOSTART=false>
 <EMBED SRC="d0.wav" HIDDEN=true AUTOSTART=false>
<EMBED SRC="ds0.wav" HIDDEN=true AUTOSTART=false>
<EMBED SRC="e0.wav" HIDDEN=true AUTOSTART=false>
 <EMBED SRC="f0.wav" HIDDEN=true AUTOSTART=false>
 <EMBED SRC="fs0.wav" HIDDEN=true AUTOSTART=false>
 <EMBED SRC="g0.wav" HIDDEN=true AUTOSTART=false>
 <EMBED SRC="gs0.wav" HIDDEN=true AUTOSTART=false>
<E«BED SRC="a0.wav" HIDDEN=true AUTOSTART=false>
 <EMBED SRC="as0.wav" HIDDEN=true AUTOSTART=false>
 <EMBED SRC="b0.wav" HIDDEN=true AUTOSTART=false>
<EMBED SRC="c1.wav" HIDDEN=true AUTOSTART=false>
 <H1>Пианино JavaScript</H1>
 <HR>
<p>Щелкните на клавише пианино и услышите ноту</p>
 <HR>
 <A HREF="#" onClick="playnote(0); ">
      <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
 <A HREF="#" onClick="playnote(1); ">
      <IMG border=0 SRC="blackkey.gif" ALIGN=TOP></A>
 <A HREF="#" onClick="playnote(2);">
      <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
 <A HREF="#" onClick="playnote(3); ">
      <IMG border=0 SRC="blackkey.gif" ALIGN=TOP></A>
 <A HREF="#" onClick="playnote(4);">
      <IMG border=0 SRC="whitekey.gif" ALIGN=TOPx/A>
 <A HREF="#" onClick="playnote(5); ">
      <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
 <A HREF="#" onClick="playnote(6); ">
      <IMG border=0 SRC="blackkey.gif" ALIGN=TOP></A>
 <A HREF="#" onClick="playnote(7); ">
     <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
 <A HREF="#" onClick="playnote(8); ">
```


http://ITBookz.ru/

```
45:             <IMG border=0 SRC="blackkey.gif" ALIGN=TOP></A>
46:         <A HREF="#" onClick="playnote(9);">
47:             <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
48:         <A HREF="#" onClick="playnote(10);">
49:             <IMG border=0 SRC="blackkey.gif" ALIGN=TOP></A>
50:         <A HREF="#" onClick="playnote(11);">
51:             <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
52:         <A HREF="#" onClick="playnote(12);">
53:             <IMG border=0 SRC="whitekey.gif" ALIGN=TOP></A>
54:         <HR>
55:         </BODY>
56:         </HTML>
```

Разместите файл HTML в том же каталоге, что и звуковые файлы и файлы используемых в нем рисунков (в вашем компьютере или на Web-сервере). Загрузите файл и протестируйте пианино. (Аккорды воспроизводить нельзя, но все равно весело.)

На рис. 20.2 представлено окно броузера с загруженным в него документом HTML. Книга не позволяет отобразить воспроизводимые звуки, поэтому постарайтесь протестировать их самостоятельно.

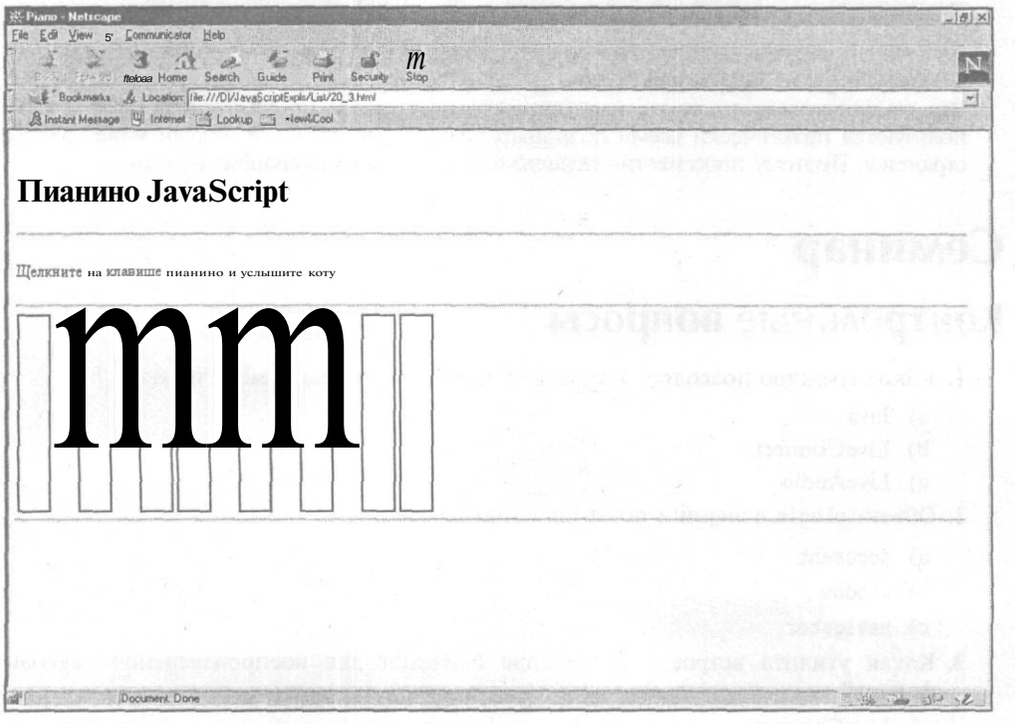


Рис. 20.2. Клавиатура пианино готова к действию

http://ITBookz.ru/

Резюме

За этот час вы познакомились с дополнительными утилитами JavaScript, позволяющими украсить документы HTML звуками, видеоклипами и рисунками. Вы узнали, как управлять утилитами с помощью их объектов. В конце занятия вы создали сценарий воспроизведения нот щелчками на "клавишах", нарисованных на Web-странице.

Предпоследняя часть уже позади. Недалек и конец книги. На последних четырех занятиях вы узнаете, как правильно проводить отладку программ, и познакомитесь с интересными примерами использования уже изученных вами средств JavaScript.

Вопросы и ответы

Можно ли загружать надстройку автоматически при возникновении в ней необходимости?

Нет. Хотя будущие версии Navigator и будут обладать подобными средствами. Для обновления списка надстроек (после установки еще одной) используется метод `navigator.plugins.reload`.

Где можно познакомиться с полным списком свойств и методов указанной утилиты?

Обратитесь за подобными сведениями на Web-узел производителя надстройки. Утилиты, поставляемые вместе с Navigator, также описаны на Web-узле *Netscape*.

Как сильно распространены надстройки? Не будет ли ущемлена часть аудитории невозможностью их использовать.

Все основные модули, такие как QuickTime и звуковые проигрыватели свободно используются практически всеми пользователями. Более сложные модули менее распространены. Поэтому повсеместно использовать их в своих сценариях не стоит.

Семинар

Контрольные вопросы

1. Какое средство позволяет JavaScript управлять встроенными утилитами?
   a) Java
   b) LiveConnect
   c) LiveAudio
2. Объект plugin дочерний по отношению к объекту:
   a) `document`
   b) `window`
   c) `navigator`
3. Какая утилита встроена в Netscape Navigator для воспроизведения звуковых файлов?
   a) LiveConnect
   b) LiveAudio
   c) Не включена ни одна

http://ITBookz.ru/

Ответы

1, b) LiveConnect позволяет JavaScript управлять надстройками

2, c) Объект `plugins` дочерний по отношению к объекту navigator

3, b) Надстройка LiveAudio используется для воспроизведения звука в Netscape Navigator

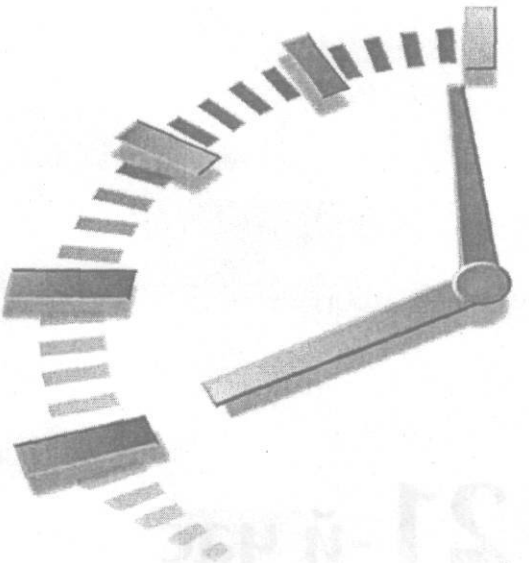
Упражнения

Для повышения своей квалификации в управлении встраиваемыми утилитами выполните следующие упражнения.

- Измените листинг 20.1 таким образом, чтобы после определения утилиты на экран выводилось сообщение со ссылкой на Web-узел ее производителя.
- Добавьте на "пианино" листинга 20.3 еще одну октаву. (Звуковые файлы остальных нот вы найдете на том же Web-узле, что и первой октавы.)

http://ITBookz.ru/

http://ITBookz.ru/



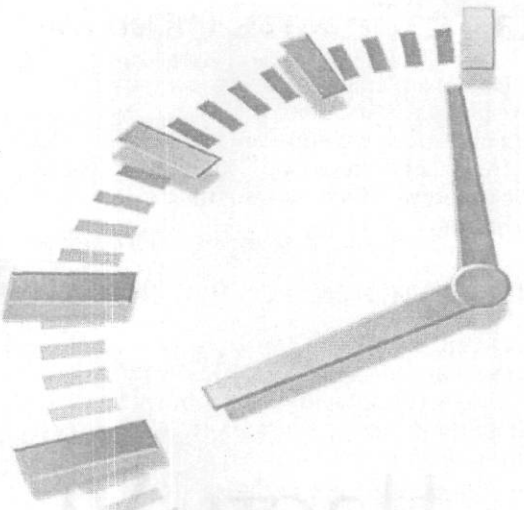
Часть VI

Сложные приложения JavaScript

Темы занятий

21. Отладка приложений JavaScript
22. Улучшение Web-страниц
23. Создание сценария игры
24. Тенденции развития технологий Web

http://ITBookz.ru/



21-й час

Отладка приложений JavaScript

Добро пожаловать в последнюю часть книги! На следующих занятиях вы используете все изученные методы и средства для создания готовых приложений JavaScript. Перед тем как перейти к непосредственным действиям, хорошо бы узнать, как устранять возникающие в процессе создания приложений ошибки.

За этот час вы познакомитесь с некоторыми способами создания безошибочных сценариев. Вы также познакомитесь со средствами и инструментами отыскания и устранения ошибок. В этой главе рассмотрены следующие темы.

- **Использование правильной техники программирования**
- **Советы по отладке сценариев с помощью консоли JavaScript**
- **Использование сообщений для отладки сценариев**
- **Использование отладчика JavaScript**
- **Отладка готового приложения**

http://ITBookz.ru/

Как избежать ошибок

Ошибка в сценарии возникает при неправильном выполнении одной или нескольких команд. Собственноручно создавая сценарий (особенно сложный), вы обязательно встретитесь с ошибками — независимо от того, насколько внимательны вы были.

Не следует теряться, обнаружив ошибки при создании сложного приложения. Большинства из них можно избежать при внимательном введении программного кода сценария.

Правильная техника программирования

История не знает ни одного программиста, который при создании своего первого приложения не сделал хотя бы одной ошибки. Опытные программисты, чтобы избежать случайных ошибок, следуют нескольким неписаным правилам. Ниже приведены основные правила, следуя которым, вы значительно уменьшите количество неточностей и ошибок в сценариях.

- Выделяйте в сценариях отдельные блоки программы. Старайтесь поддерживать ваш сценарий удобным для просмотра. Используйте описательные имена переменных, которые объясняют их назначение. Очень трудно определить использование кода, если вы не можете объяснить, для чего вводилась та или иная строка.
- Не бойтесь добавлять в сценарии частые комментарии. Они помогут вам разобраться в деталях программы, которые со временем вы наверняка забудете. Чем больше комментариев, тем выше вероятность избежать ошибок, вызванных недопониманием назначения строк программы.
- Заканчивайте операторы JavaScript точкой с запятой. Хотя это и необязательно, но ваш сценарий будет выглядеть более читабельным. Это также позволяет броузеру выводить сообщения об ошибках более содержательного характера.
- Объявляйте все переменные с помощью оператора var. Это, в большинстве случаев, необязательно, но позволяет избежать проблем, вызванных неправильным определением переменных.
- Разделяйте сложный сценарий на несколько разделов. Это облегчает его анализ и позволяет проще выявить ошибку.
- Создавайте сложный сценарий в несколько этапов и тестируйте отдельно каждый его раздел. Таким образом легче выявить простые ошибки, не относящиеся к взаимодействия составных частей сценария.

Как избежать простых ошибок

Следуя приведенным ниже неписаным правилам, вы избежите многих ошибок, но полностью избавиться от них не сможете. Разные люди делают разные ошибки при создании сценарии. Но существуют ошибки, которые делают большинство пользователей.

Ошибки синтаксиса

Ошибки синтаксиса вызваны неправильным введением ключевых слов, операторов, символов пунктуации и других элементов. Как правило, это просто опечатки

Операторы, содержащие подобные ошибки, неправильно загружаются или не загружаются вообще в броузере. Поэтому сценарий интерпретируется неправильно. Эти ошибки вызывают появления на экране сообщений об ошибках. Устранять их проще других.

http://ITBookz.ru/

Присвоение значений и операторы равенства

Еще одна из распространенных синтаксических ошибок, которую делают начинающие пользователи, заключается в использовании оператора равенства (==) вместо оператора присвоения значения (=). Эту ошибку определить сложно, поскольку она не вызывает появление на экране сообщения об ошибке.

Если вы не уверены в том, какой оператор использовать, то запомните одно простое правило: оператор "=" используется для изменения значения переменной, а "==" — для сравнения двух значений. Вот как выглядит неправильно использованный оператор равенства:

```
if (a=5) alert("Пятерка найдена");
```

Этот условный оператор выглядит введенным правильно. Проблема состоит в том, что команда a = 5 указывает присвоить переменной a значение 5, а не сравнить ее значение с числом 5. В подобных случаях броузер определяет причину ошибки и выводит сообщение в окне консоли JavaScript. А вот использование оператора "==" вместо "=" появления сообщения об ошибке не вызывает.

Локальные и глобальные переменные

Часто пользователи делают ошибки, вызванные неправильным использованием глобальных и локальных переменных. Самая распространенная ошибка — это использование переменной, которая объявлена в функции, в остальной части программы. Если вам необходимо использовать подобную переменную, либо объявите ее как глобальную, либо возвратите функцией локальную переменную в программу.



В главе "5-й час. Использование переменных и функций" детально описано использование глобальных и локальных переменных.

Правильное использование объектов

Еще одна частая ошибка встречается при задании неправильной ссылки на объект. Помните о существовании иерархической структуры объектов и задавайте в операторах имена переменных только в соответствии с ней.

Например, вы привыкли вместо метода window.alert использовать просто alert. Тем не менее, в некоторых случаях (например, при задании обработчика события) необходимо использовать полное имя объекта. Если alert или другой метод или свойство нельзя определить броузером, то это указывает на необходимость указания объекта, к которому он относится.

Вторая часто допускаемая ошибка — пропуск ключевого слова document в названии объекта. Например, write вместо document.write.

Ошибки в коде HTML

И наконец, не забывайте о том, что JavaScript — это не единственный язык, на котором создается Web-страница. Очень легко создать ошибки в коде HTML, например пропустить один из закрывающих дескрипторов.

Правильное введение кода HTML не входит в число тем, рассматриваемых в этой книге. Поэтому будьте особенно внимательны при его введении. Дважды проверьте правильность кода HTML, а только затем приступайте к коду JavaScript.



Для проверки кода HTML создано огромное количество специальных программ. В Приложении Б приведен список некоторых из них.

http://ITBookz.ru/

Основные средства отладки

Определив в сценарии обычные ошибки и несогласованности, самое время приступить к отладке программы. Отладка — это процесс отыскания в профамме ошибок и их устранение. Некоторые основные средства отладки сценариев содержатся в JavaScript и будут описаны в следующих разделах.

Консоль JavaScript

Первым делом вам необходимо проверить сценарий на наличие ошибок, для которых отображаются сообщения на консоли JavaScript. В Netscape Navigator 4.5 сообщения об ошибках по умолчанию не отображаются, но фиксируются средствами консоли JavaScript.

Чтобы запустить консоль, введите в строке адреса Netscape Navigator следующее: javascript:. На экране появится окно консоли JavaScript, в котором будут отображены последних несколько сообщений об ошибках (рис. 21.1).

Кроме определения ошибок, вы можете использовать консоль JavaScript для выполнения любых команд JavaScript. Это применяется, если необходимо проверить правильность выполнения определенного оператора в сценарии.

Автоматическое отображение консоли

После устранения всех простых ошибок вы будете пользоваться для отладки приложения исключительно консолью JavaScript. Постоянный запуск ее с помощью команды javascript: может порядком утомить даже самого терпеливого разработчика. Чтобы облегчить задачу создателям сценариев, в Netscape Navigator версии 4.5 и выше добавлена команда автоматического запуска консоли при возникновении ошибки.

Настройку этой опции нельзя выполнить в диалоговом окне свойств броузера. Чтобы изменить эту установку, вам придется исправить файл настроек броузера. Этот файл prefs.js расположен в каталоге файлов настроек пользователя (обычно названный именем пользователя) корневого каталога Netscape.

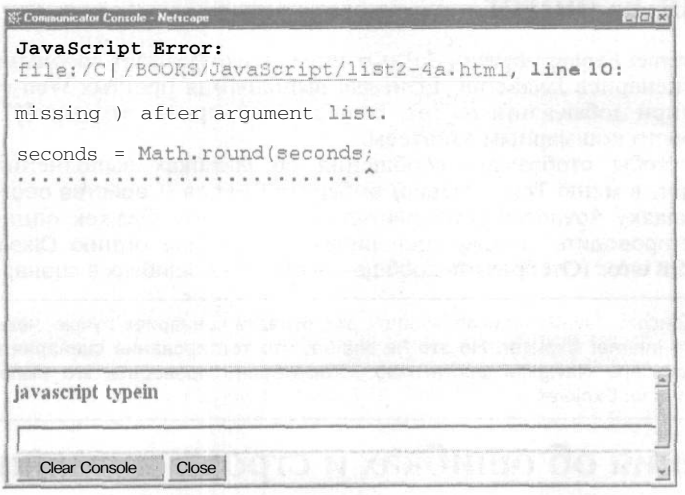


Рис. 21.1. На консоли JavaScript отображаются автоматически обнаруженные ошибки в сценарии

http://ITBookz.ru/

Чтобы изменить файл, откройте его в текстовом процессоре. Этот файл содержит перечень команд JavaScript, определяющих параметры броузера. Чтобы настроить консоль JavaScript на автоматическую загрузку, при нахождении ошибки, добавьте в конец файла следующую строку:

```
users_pref("javascript.console.open_on_error", true);
```

После изменения файла сохраните его и загрузите Netscape Navigator. Теперь при нахождении ошибки в сценарии будет автоматически загружаться консоль JavaScript.



Содержимое файла prefs.js обновляется при запуске броузера. Поэтому вы не сможете изменить его при запущенном Netscape Navigator. Перед изменением файла закройте все запущенные окна броузера.

Сообщения в строке состояния

Если вам повезет, сообщения об ошибках на консоли JavaScript помогут правильно отладить сценарий. В большинстве же случае консоль JavaScript не позволяет определить и устранить все ошибки — сценарий все еще не хочет выполняться правильно. С этого момента вы приступаете к главной части отладки программы.

Один из наиболее важных методов отладки приложения — это добавление в сценарий временных операторов, позволяющих поэтапно проверять его работу. Например, вы можете добавлять в сценарий операторы отображения сообщения в строке состояния, которые содержат значения текущих переменных. По значениям переменных вы можете узнать, правильно ли на данном этапе выполняются программы.



Наряду с сообщениями, в строке состояния вы можете отображать сведения о работе приложения в отдельных окнах броузера. Для этого используйте метод document.write. Этот метод, правда, выполняется только после полной загрузки приложения.

Отображение сообщений об ошибках в Internet Explorer

Microsoft Internet Explorer версии 4.0 и ранних не отображает сообщений об ошибках выполнения сценариев JavaScript. Если для выполнения простых Web-страниц это некритично, то при добавлении на них больших сценариев, отладка HTML-документа становится просто кошмарным занятием.

Для того чтобы отображать сообщения об ошибках выполнения сценариев в Internet Explorer, в меню Tool (Сервис) выберите Options (Свойства обозревателя). Перейдите на вкладку Advanced (Дополнительно). Снимите флажок опции Disable script debugging (Не проводить отладку сценариев) и выставьте опцию Display a notification about every script error (Отображать сообщения обо всех ошибках в сценариях).



Консоль JavaScript использовать для отладки сценариев лучше, чем диалоговые окна Internet Explorer. Но это не значит, что тестирования сценария в одном только Netscape Navigator достаточно. Обязательно проверьте его выполняемость и в Internet Explorer.

Сообщения об ошибках и строка состояния

Если вам повезет, сообщение на консоли правильно укажет причину ошибки сценария. Тем не менее, средство отладки может и не определить ошибку в сценарии. Он все еще будет выполняться неправильно.

http://ITBookz.ru/

Среди методик отладки сценариев используются неписаные методы, которые очень просты в реализации. Например, вы можете выводить на экран временные сообщения, в которых указывается значение той или иной переменной. Правильно оценив переменную на каждом этапе выполнения сценария, вы сможет отладить сценарий даже быстрее, нежели с помощью консоли JavaScript.



Вы также можете отображать значение переменной и в строке состояния или отдельном окне броузера. Это как вам удобнее. В некоторых случаях удобно использовать метод `document.write`, но он применяется только до момента загрузки сценария.

Отладчик JavaScript

Хотя с помощью сообщений в строке состояния и можно быстро определить ошибки в простых сценариях, громоздкие приложения отлаживаются по-другому. Для проверки сложных сценариев *Netscape* снабдила свои броузеры самым эффективным изо всех используемых средств — отладчиком сценариев.

Установка отладчика

Отладчик JavaScript — это программа, запускаемая в броузере Netscape. Вы можете загрузить последнюю версию отладчика с Web-узла *Netscape* по адресу:

`http://developer.netscape.com/software/jsdebugg.html`



Отладчик устанавливается с помощью средства SmartUpdate броузера Netscape Navigator. Поэтому для выполнения загрузки и установки отладчика вам сначала необходимо инсталлировать последнюю версию Nescape Navigator.

Детальные инструкции по загрузке и установке отладчика приведены на Web-узле *Netscape*. Большинство операций во время установки отладчика выполняются автоматически.

Окно отладчика

После установки отладчика можно смело запускать его. Для этого в поле адреса броузера введите `jsdebugger.html`. Этой файл сохранен в корневом каталоге каталога *Netscape*. При первом запуске отладчика вам необходимо определить несколько параметров защиты данных броузера. Выполнить все необходимые операции вы можете, только имея определенные полномочия.

После этого на экране отображается окно отладчика. При запуске сценария JavaScript это окно не закрывается, а остается открытым на экране.

Отладчик сохраняет сведения обо всех сценария, выполненных в броузере, после его запуска. Для того чтобы отладить сценарий, загрузите документ HTML, содержащий его, в броузере. Далее щелкните на кнопке Open (Открыть) на панели инструментов отладчика. При этом будет отображен список все открытых в броузере документов. Выделите необходимую Web-страницу. Сценарий отобразится в окне отладчика, как это показано на рис. 21.2.

После отображения сценария в окне отладчика можете непосредственно приступать к отладке сценария. В следующих разделах описаны основные средства отладчика. Детально о выполняемых им операциях вы можете узнать на Web-узле *Netscape*.

http://ITBookz.ru/

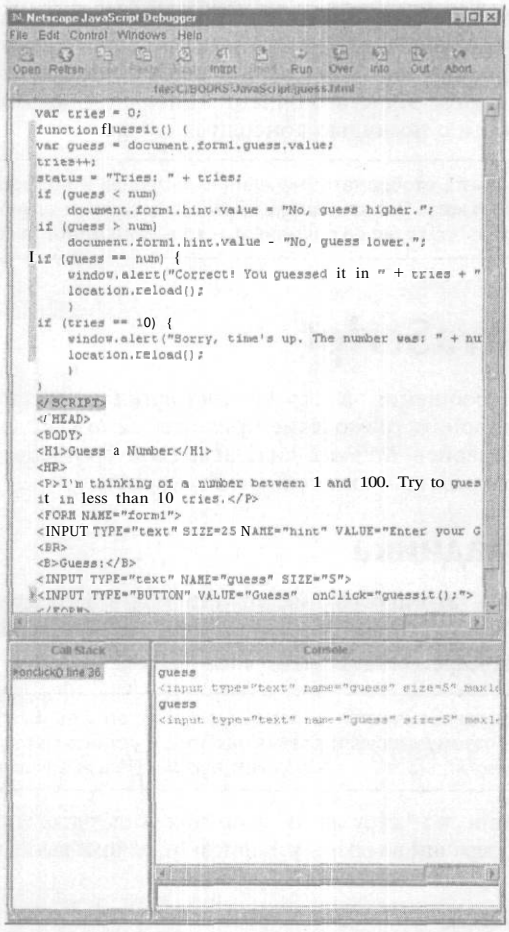


Рис. 21.2. Окно отладчика JavaScript, содержащее сценарий документа HTML

Вставка разрывов и прерываний

После загрузки отладчика проводится фоновая проверка загруженного в броузере сценария. Отладчик позволяет вам прерывать выполнение сценария на определенном этапе. Оператор прерывания выполнения сценария называется *разрывом*.

Чтобы вставить разрыв, щелкните слева от необходимого оператора в окне отладчика. Появится красная точка, символизирующая разрыв. Вы можете добавлять в сценарий сколько угодно точек разрыва. Достигнув разрыва, выполнение сценария прекращается и начинается отладка текущей части сценария.

Если вы хотите остановить выполнение сценария в текущий момент, воспользуйтесь кнопкой Interrupt (Прервать), расположенной на панели инструментов отладчика. При этом выполнение сценария будет прервано (текущий оператор будет выполнен до конца) и управление им передается отладчику.



При нахождении отладчиком ошибки выполнение сценария также прерывается и управление им передается отладчику.

http://ITBookz.ru/

Проверка переменных

Главная причина добавления в сценарий разрывов и прерываний — это определение текущих значений тех или иных переменных. Отслеживать значения переменных можно, добавив их в список отслеживания отладчика.

Чтобы добавить переменную в список отслеживания отладчика, выделите ее в окне отладчика и в меню Edit (Правка) выберите команду Copy to Watch (Добавить в список отслеживания). Значения всех переменных, добавленных в список отслеживания, отображаются в правом нижнем углу окна отладчика.

Выполнение сценария

После прерывания выполнения сценария и устранения всех найденных вами ошибок необходимо продолжить его выполнение. Самый простой способ продолжить выполнение сценария заключается в использовании кнопки Run (Выполнить), расположенной на панели инструментов отладчика. Она позволяет продолжить выполнение сценария, начиная со строки, на которой оно было прервано.

При необходимости вы можете выполнить сценарий поэтапно. Для этого в меню Control (Действия) выберите команду Step Into (Выполнить оператор). При этом будет выполнен только один оператор.



За один раз вы можете выполнить всю функцию. Для этого воспользуйтесь командой Step Over (Выполнить блок). Детально об этих параметрах вы можете узнать в документации к отладчику *Netscape*.

Отладка сценария

Теперь, после детального рассмотрения методов отладки приложений JavaScript, самое время применить их на практике. Зная все предназначенные для этого инструменты броузера и следуя моим инструкциям, вы без труда с этим справитесь.

В листинге 21.1 приведен сценарий классической игры "угадай число". Сценарий запрашивает число в диапазоне 1 — 100 и позволяет угадать его с десяти попыток. Если вы угадали неправильно, сценарий выводит сообщение, в котором указана близость указанного вами числа к угадываемому (больше или меньше).

Этот сценарий простой, но содержит по меньшей мере одну серьезную ошибку, которая не позволяет выполняться ему при загрузке в броузере.

Листинг 21.1. Сценарий угадывания числа, содержащий ошибки

```
1:  <HTML>
2:  <HEAD>
3:  <TITLE>Угадай число</TITLE>
4:  <SCRIPT LANGUAGE="JavaScript">
5:  var num=Math.random()*100+1;
6:  var tries=0;
7:  function guess() {
8:  var guess=document.form1.guess1.value;
9:   tries++;
10: status=tries+" попытка";
11: if (guess<num)
12:       document.form1.hint.value="Больше";
13: if (guess>num)
14:      document.form1.hint.value="Меньше";
15: if (guess==num) {
16:       window.alert("Правильно! Вы угадали с "+tries+"
          попыток");
17:       location.reload();
```


http://ITBookz.ru/

```
18:              }
19:          if (tries==10) {
20:              window.alert("Очень жаль. Вы не угадали. Правильное
                  число: "+num);
21:              location.reload();
22:              }
23:          }
24:          </SCRIPT>
25:          </HEAD>
26:          <BODY>
27:         <H1>Угадай число</H1>
28:          <HR>
29:          <P>Я загадал число от 1 до 100. Попробуй его
30:         угадать с 10 попыток</P>
31:          <FORM NAME="form1">
32:         <INPUT TYPE="text" SIZE=25 NAME="hint" VALUE="Ваше число">
33:          <BR>
34:          <B>Правильное число:</B>
35:         <INPUT TYPE="text" NAME="guess1" SIZE="5">
36:         <INPUT TYPE="BUTTON" VALUE="Угадать" onClick="guess();">
37:          </FORM>
38:          </BODY>
39:          </HTML>
```

Вот какие действия выполняются в этом сценарии.

- Строка 5 определяет случайное число и назначение его в качестве значения переменной num.
- Функция guess, определенная в строках 7–23, вызывается каждый раз при введении пользователем числа.
- Проверка введенного числа на равенство с угадываемым проводится с помощью оператора if. Если введенное число не совпадает с угадываемым, в верхнем поле отображается его отношение к угадываемому (больше или меньше). Если число угадано правильно, то выводится поздравление победителю.

Тестирование программы

Чтобы протестировать эту программу, сначала загрузите ее в броузере. В самом начале она загрузится правильно и не будет выдавать никаких сообщений об ошибках. Тем не менее, после введения числа и щелчка на кнопке Угадать выполнение сценария будет прервано ошибкой.

На консоли JavaScript появится следующее сообщение об ошибке:

```
guess is not a function (guess - это не функция)
```

Исправление ошибки

Это сообщение об ошибке указывает, что неправильно вызвана функция guess. Это происходит в том случае, если определенное в сценарии имя функции и название вызываемой функции не совпадают.

При детальном рассмотрении программного кода несложно заметить, что загадка кроется в первых двух строках определения функции:

```
function quess() {
var quess=document.form1.guess1.value;
```


http://ITBookz.ru/

Хотя такая запись и выглядит корректно, ошибка в ней все же есть. Переменная, определенная в функции, имеет то же имя, что и сама функция. В результате определение переменной перекрывает определение функции. При попытке вызова обработчиком событий функции вызывается переменная guess. Конечно, обработчик событий тут же определяет, что это не функция.

Эту ошибку легко устранить. Просто измените имя функции с guess на guessit. Измените ее имя также и в строке 36 вызова функции обработчиком событий.

Повторная проверка сценария

После устранения ошибки давайте запустим сценарий еще один раз. Он опять загружается без ошибок. Даже сообщения о величине введенного вами числа по отношению к угадываемому выводятся правильно. Тем не менее, вам необходимо протестировать выполнение сценария до конца. И тут вы обнаружите еще одну недоработку. Выиграть у компьютера невозможно, как бы вы ни старались.

После десятого угадывания появится сообщение о вашем проигрыше. Будет также выведено правильное число. И тут вы поймете всю тщетность ваших стараний. На рис. 21.3 показано окно броузера, отображаемое после окончания игры.

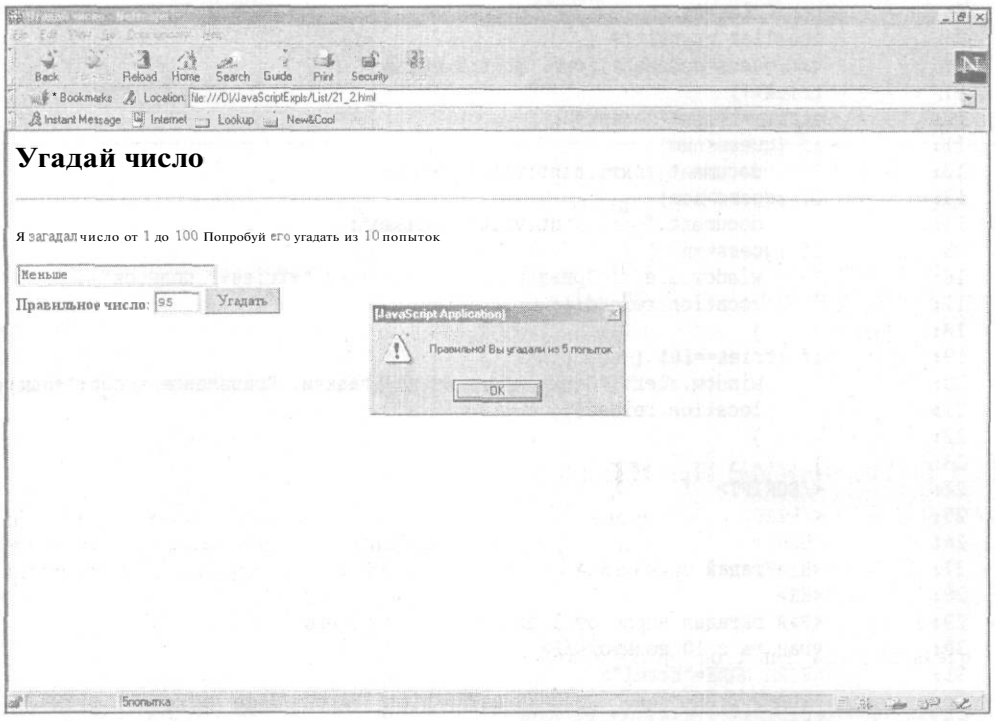


Рис. 21.3. Правильное число выводится после десятого угадывания

Как вы видите, в том, что вы проиграли, нет ничего удивительного. Произвольно заданное число имеет 10 разрядов после запятой, а вы угадывали целое число. Вы можете угадывать и десятичное число, но для угадывания его вам вряд ли будет достаточно 10 попыток. Игра, кажется, утратила свою привлекательность и простоту.

Чтобы устранить эту недоработку, давайте взглянем на определение произвольного угадываемого числа:

```
5:              var num=Math.random()*100+1;
```


http://ITBookz.ru/

В нем используется метод Math.random, который возвращает произвольное число в диапазоне 0—1. Затем полученное значение умножается на 100 и увеличивается на 1. В результате вы получаете произвольное (десятичное) число в диапазоне 1—100.

Этот оператор позволяет получить произвольное число в указанном диапазоне, но оно не целое. Давайте сделаем из него целое, добавив метод Math.floor, округляющий десятичные числа до наименьшего целого:

```
var num=Math.floor(Math.random()*100)+1;
```

После исправления сценария протестируйте его еще раз. Теперь он должен выполняться правильно — победа станет вопросом времени. Листинг 21.2 содержит правильный вариант кода сценария.

Листинг 21.2. Правильный код сценария угадывания числа

```
<HTML>
<HEAD>
<TITLE>Угадай число</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var num=Math.floor(Math.random()*100)+1;
var tries=0;
function guessit() {
var guess=document.form1.guess1.value;
tries++;
status=tries+"попытка";
if (guess<num)
     document.form1.hint.value="Больше";
if (guess>num)
     document.form1.hint.value="Меньше";
if (guess==num) {
    window.alert("Правильно! Вы угадали с "+tries+" попыток");
     location.reload();
     }
if (tries==10) {
     window.alert("Очень жаль. Вы не угадали. Правильное число:"+num);
     location.reload();
     }
}
</SCRIPT>
</HEAD>
<BODY>
<H1>Угадай число</H1>
<HR>
<P>Я загадал число от 1 до 100. Попробуй его
угадать с 10 попыток</P>
<FORM NAME="form1">
<INPUT TYPE="text" SIZE=25 NAME="hint" VALUE="Ваше число">
<BR>
<B>Правильное число:</B>
<INPUT TYPE="text" NAME="guess1" SIZE="5">
<INPUT TYPE="BUTTON" VALUE="Угадать" onClick="guessit();">
</FORM>
</BODY>
</HTML>
```


http://ITBookz.ru/

Резюме

За этот час вы познакомились со средствами отладки программ JavaScript. Вы узнали об основных методах выявления и устранения ошибок, а также о способах предупреждения их появления при введении сценариев. В конце занятия вы отладили сценарий простой игры.

Примите мои поздравления. За этот час вы закончили изучать последнюю главу с новым материалом. В последних трех занятиях вы используете изученный вами материал на практике — создадите несколько простых приложений

Вопросы и ответы

Почему одни ошибки появляются только после выполнения сценария, а другие — сразу же после загрузки документа HTML?

Интерпретатор JavaScript просматривает сценарии (например, проверяет определения функции), добавленные в документ HTML при загрузке страницы. Обработчики событий при этом не проверяются. Операторы могут быть введены правильно (что не вызывает ошибок при загрузке документа), а выполняться некорректно (что вызывает ошибки при выполнении сценария).

Какое предназначение оператора `location.reload` в примере сценария угадывания чисел?

Этот метод позволяет запустить новую игру, поскольку при обновлении страницы все переменные принимают исходные значения. В результате задается новое угадываемое число.

Существует ли в Internet Explorer эквивалент консоли JavaScript, представленной в Netscape Navigator?

Да, но не встроенный. Я создал консоль, которая выполняется и в Internet Explorer. Она не умеет отображать ошибки так, как это делает консоль Netscape, но очень полезна при поиске ошибок. Вы можете загрузить ее с узла `http://www.jsworkshop.com/`.

Семинар

Контрольные вопросы

1. ЕСЛИ вы неправильно ввели ключевое слово, какая это будет ошибка?
   a) Синтаксическая
   b) Ошибка определения функции
   c) Ошибка чтения
2. Процедура отыскания ошибок в сценариях и программах называется:
   a) определением ошибок
   b) сбоем
   c) отладкой
3. Какое средство отладчика JavaScript позволяет остановить выполнение сценария по достижении определенного сценария?
   a) Интерпретатор
   b) Средство отслеживания
   c) Разрыв

http://ITBookz.ru/

Ответы

1, a) Синтаксическая ошибка вызывается неправильным введением оператора JavaScript

2, c) Отладка — это процесс обнаружения и устранения ошибок в программах

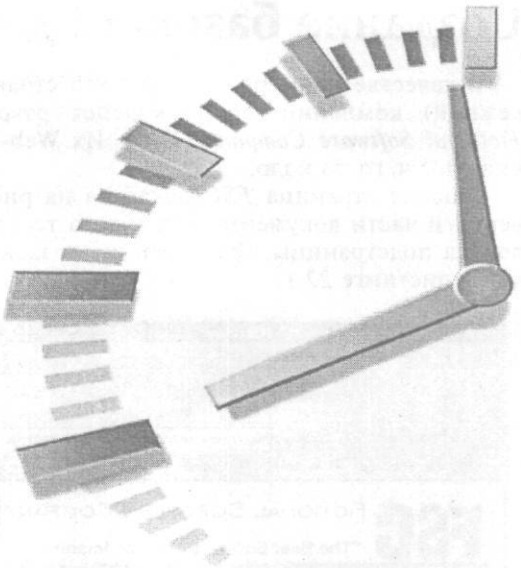
3, c) Разрыв позволяет остановить выполнение сценария в отладчике

Упражнения

Для повышения вашей квалификации отладчика выполните следующие упражнения.

- Хотя код сценария угадывания числа уже не содержит ошибок, он все еще чувствителен к неправильному введению пользователем значений. Добавьте в сценарий листинга 21.2 операторы проверки вводимого пользователями значения.
- Загрузите сценарий листинга 21.1 в отладчик. С помощью разрывов и прерываний определите в нем все существующие ошибки.

http://ITBookz.ru/



22-й час

Улучшение Web-страниц

Теперь вы знаете не только о методах создания Web-страниц, но и о способах их отладки. На последних двух занятия в этой книге вы примените полученные вами знания для создания нескольких полных приложений.

Начнем с создания простого приложения — обычной Web-страницы с несколькими элементами JavaScript, упрощающими перемещение по ней. В этой главе рассмотрены следующие темы.

- **Создание базового документа HTML**
- **Добавление раскрывающихся списков**
- **Добавление описаний в строке состояния**
- **Добавление ссылок на рисунки и изменяющиеся изображения**
- **Создание Web-страницы, содержащей все перечисленные элементы**

http://ITBookz.ru/

Создание базового документа HTML

В качестве примера возьмем Web-страницу придуманной небольшой (и безнадежной) компании, занимающейся разработкой программного обеспечения - *Fictional Software Company (FSC)*. Их Web-страница не так уж и красочна, но начинать с чего-то надо.

Главная страница *FSC* показана на рис. 22.1. Она содержит логотип фирмы в верхней части документа, три абзаца текста, простой маркированный список ссылок на подстраницы. Эта страница создана полностью в HTML. Ее код представлен в листинге 22.1.

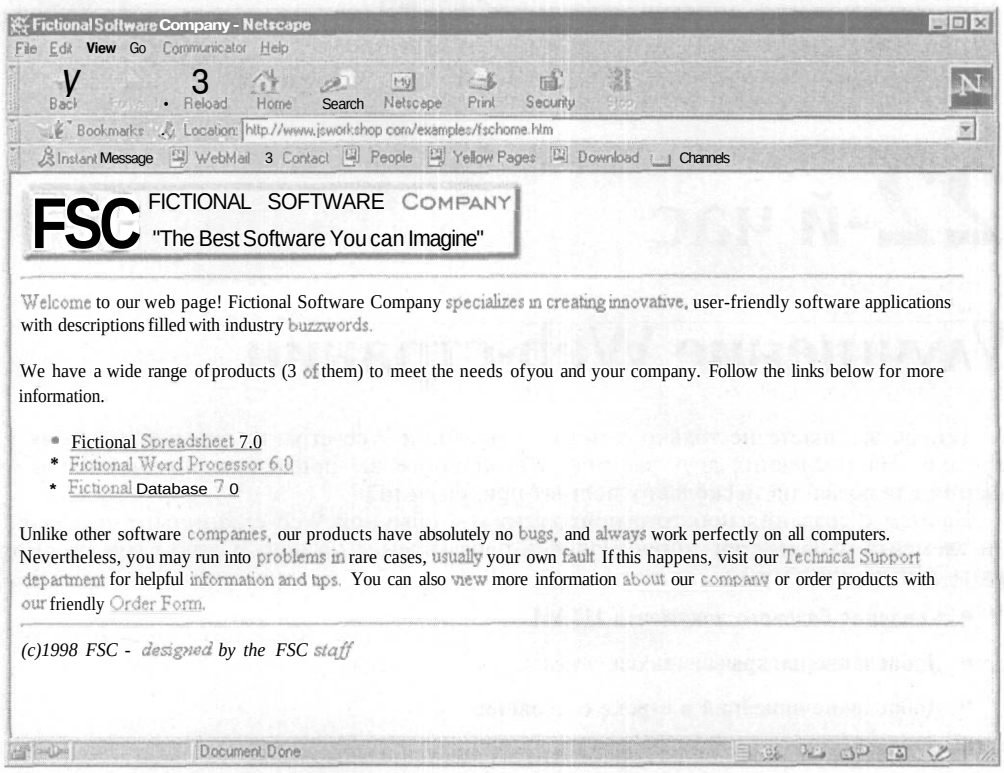


Рис. 22.1. Простая Web-страница, созданная с помощью HTML

Листинг 22.1. Документ HTML начальной страницы компании FSC

```
<HTML>
<HEAD><TITLE>Fictional Software Company</TITLE></HEAD>
<BODY>
<IMG SRC="fsclogo.gif" alt="Fictional Software Company">
<HR>
 Welcome to our page! Fictional Software Company
specializes in creating innovative, user-friendly software
applications with descriptions filled with industry
buzzwords.
```


http://ITBookz.ru/

```
10:    <P>We have a wide range of products (3 of them) to meet
11:    the needs of you and your company/ Follow the links
12:    below for more information.
13:    <P>
14:    <UL>
15:    <LI><A HREF="spread.html">Fictional Spreadsheet 7.0</A>
16:    <LI><A HREF="word.html">Fictional Word Processor 6.0</A>
17:    <LI><A HREF="data.html">Fictional Database 7.0</A>
18:    </UL>
19:    <P>
20:    Unlike other software companies, our product have
21:    absolutely no bugs, and always work properly on all
22:    computers. Nevertheless, you may run into problem in
23:    rare cases, usually your own fault. If this happens,
24:    visit our <A HREF="support.html">Technical Support</A>
25:    department for helpful information and tips. You can
26:    also view more information <A HREF="company.html">about
27:    our company</A> or order products with our friendly
28:    <A HREF="order.html">Order Form</A>
29:    <HR>
30:    <I>(c) 1998 FSC - designed by the FSC staff</I>
31:    </BODY>
32;    </HTML>
```

Все ссылки на странице позволяют отображать другие страницы. Одна из них содержит описания продуктов компании, другая — сведения о компании, а остальные предоставляют техническую информацию.

Использование раскрывающихся списков

По мере развития компании на ее Web-страницы добавляются все новые и новые сведения. Начальная страница остается неизменной, изменяются подстраницы, которые содержат описания производимых ею продуктов.

При возрастании числа производимых программ начальная страница становится громоздкой и неудобной для управления. Например, если вам необходимо узнать системные требования для установки определенной программы, то вы должны выполнить следующие действия: выбрать на начальной странице ссылку на страницу, содержащую описание этого продукта, а затем на ней щелкнуть на ссылке System Requirements.

С помощью JavaScript вы можете создать простой интерфейс, позволяющий быстро отображать необходимую информацию. Давайте используем один раскрывающийся список для указания программы, а второй — для типа информации, которую необходимо о нем отобразить.

Имена страниц

При написании программы создание программного кода — это не самая тяжелая часть. Вначале вы должны правильно определить все необходимые операции и только затем на их основе создавать самый простой программный код.

Для того чтобы упростить программирование панели перемещения по Web-узлу, придумайте подстраницам простые описательные названия. Пусть имена будут частично или полностью совпадать с соответствующими им элементами раскрывающего-

http://ITBookz.ru/

ся списка. Давайте определим каждой из программ свою кодовую букву: *w* — для текстового процессора, *s — для* программы создания электронных таблиц, а *d* — для приложения управления базами данных. Используя символ подчеркивания, вы можете определять программе разные категории.

Предположим, существует несколько категорий сведений, которые необходимо донести до пользователей:

- tech. Информация о технической поддержке;
- sales. Данные о наличии программных продуктов и их продаже;
- feat. Перечень свойств и возможностей программы;
- price. Сведения о цене на товары;
- tips. Советы по повышению производительности программ.

Например, страница s_feat.html будет содержать список свойств и возможностей программы управления электронными таблицами. Описательные имена делают процедуру создания программы HTML проще.



Для загрузки этого примера вам потребуются готовые файлы подстраниц. Вы можете загрузить их с Web-узла.

Создание структуры данных и документа HTML

Перед созданием функции, позволяющей перемещаться между страницами, вам необходимо сохранить используемые в сценарии данные. Для этого вам потребуется три переменные для сохранения кодов созданных программ и пять переменных для сохранения кодов типов страниц. Для каждого вида переменных можно задать массив, но это необязательно.

Вместо создания массива давайте используем средства HTML сохранения значений. Сохраним необходимые величины в виде значений свойств объекта form. Создадим на форме два раскрывающихся списка, содержащих три и пять элементов, принимающих значения указанных кодов. Для загрузки страницы, которая удовлетворяет выбранным элементам раскрывающихся списков, добавим На форму кнопку Go. После щелчка на ней будет загружаться страница, соответствующая значениям указанных свойств объекта form.

Листинг 22.2 содержит код, добавляемый к коду начальной страницы. Этот код самодостаточен, и вставить его можно в любую часть кода. Как правило, подобные программы добавляются в конец кода HTML страницы.

Листинг 22.2. Определение отображаемой страницы

```
<FORM name="navform">
<SELECT name="program">
<OPTION VALUE="x" SELECTED>Select Product
<OPTION VALUE="w">Fictional Word Processor
<OPTION VALUE="s">Fictional Spreadsheet
<OPTION VALUE="d">Fictional Database
</SELECT>
<SELECT name="category">
<OPTION VALUE="x" SELECTED>Select a Category
<OPTION VALUE="tech">Technical Support
<OPTION VALUE="sales">Sales and availability
<OPTION VALUE="feat">List of Features
```


http://ITBookz.ru/

```
13:          <OPTION VALUE="price">Pricing Information
14:          <OPTION VALUE="tips">Tips and Techniques
15:          </SELECT>
16:         <INPUT TYPE="button" NAME="go" VALUE="Go to Page">
17:          onClick="Navigate();"
18:          </FORM>
```

В дополнение ко всем описанным выше категориям в каждый раскрывающийся список добавлен еще один элемент — x. Этот элемент отображается по умолчанию при загрузке страницы. Щелчок на кнопке Go при хотя бы одном выбранном элементе x к загрузке новой страницы не приводит.

Создание функции панели перемещения

Для кнопки Go определен обработчик событий onclick, вызывающий функцию Navigate(). Вам предстоит создать эту функцию. Она запрашивает текущие значения элементов обоих раскрывающихся списков, создает на их основе необходимое имя файла и загружает соответствующий ему файл в броузере.

Листинг 22.3 содержит код этой функции. Давайте взглянем на него повнимательнее.

Листинг 22.3. Функция перемещения по подстраницам узла

```
1:    function Navigate() {
2:         prod=document.navform.program.selectedIndex;
3:          cat=document.navform.category.selectedIndex;
4:        prodval=document.navform.program.options[prod].value;
5:        catval=document.navform.category.options[cat].value;
6:       if (prodval=="x" || catval=="x") return;
7:         window.location=prodval+"_"+catval+".html";
8:    }
```

В самом начале этой функции определяются две переменные, prod и cat, содержащие индекс выделенных в раскрывающихся списках элементов. Переменные prodval и catval принимают значения, равные значениям свойств объектов этих элементов.

Оператор if используется для проверки выделения в каждом списке элемента x. Если ни в одном из списков этот элемент не выбран, то оператор прекращает свою работу, не возвращая никакого результата.

В конце функции создается имя файла отображаемой страницы, Между значениями свойств элементов вставляется символ подчеркивания, а в конце добавляется расширение .html.

Добавление описания ссылок

Некоторые пользователи по-прежнему предпочитают непосредственные ссылки, добавленные на страницу, новомодным раскрывающимся спискам. Чтобы сделать обычные ссылки более привлекательными, вы можете добавить в строку состояния их описания. Пусть описания ссылок отображаются при наведении на них указателя мыши.

Эта задача выполняется с помощью обработчика событий onMouseOver. При наведении указателя на ссылку этот обработчик вызывает функцию отображения соответствующего сообщения в строке состояния. Следующий программный код определяет ссылку и ее описание в строке состояния:

http://ITBookz.ru/

```
<A HREF="order.html"
onMouseOver="window.status='Allow you to order products';return true;"
onMouseOut="window.status='';">
Order form</A>
```

Свойство window.status принимает значение отображаемого в строке состояния значения. Кроме того, возвращается значение true. Это позволяет отобразить текущее сообщение вместо сообщения по умолчанию (URL страницы). Обработчик событий onMouseOut используется для очистки содержимого строки состояния при удалении указателя мыши со ссылки.



Детально с объектами окна вы можете познакомиться в главе "10-й час. Работа с объектной моделью документа". Обработчики событий рассмотрены в главе "12-й час. Обработка событий".

Листинг 22.4 отображает результат добавления обработчика onMouseOver в код начальной страницы *FSC*. Сама страница показана на рис. 22.2. Указатель мыши находится на ссылке Order Form.

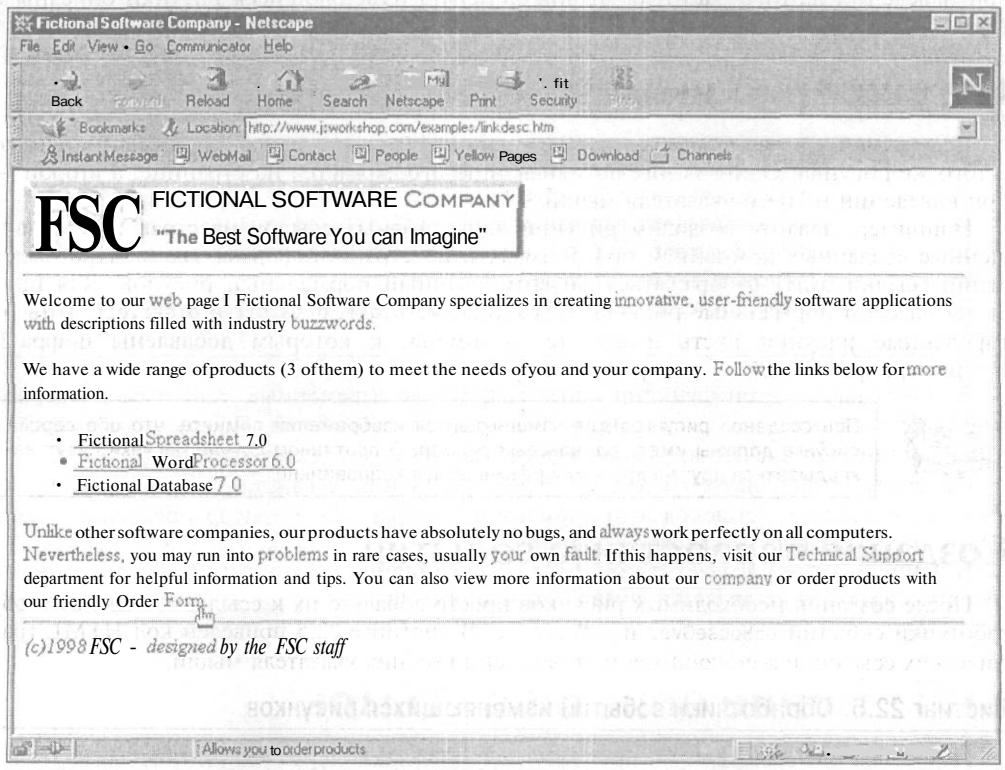


Рис. 22.2. Ссылки в документе HTML имеют свои описания

Листинг 22.4. Добавление описаний к ссылкам JavaScript

```
1:    <UL>
2:    <LI><A HREF="spread.html"
3:    onMouseOver="window.status='Spreadsheet Info'; return true;"
4:    onMouseOut="window.status='';">
```


http://ITBookz.ru/

```
5:     Fictional Spreadsheet 7.0</A>
6:     <LI><A HREF="word.html"
7:     onMouseOver="window.status='Word Processor Info'; return true;"
8:     onMouseOut="window.status='';">
9:      Fictional Word Processor 6.0</A>
10:    <LI><A HREF="data.html"
11:    onMouseOver="window.status='Database Info'; return true;"
12:    onMouseOut="window.status='';">
13:    Fictional Database 7.0</A>
14;    </UL>
```

Добавление графических ссылок

В качестве альтернативы раскрывающимся спискам для перемещения по страницам Web-узла можно использовать графические ссылки. Для большей красочности в качестве ссылок можно использовать изменяющиеся рисунки, трансформирующиеся при наведении на них указателя мыши. Детально изменяющиеся рисунки описаны в главе "15-й час. Добавление рисунков и анимации".

Создание рисунков

Для создания изменяющегося изображения необходимо иметь два варианта одного и того же рисунка. Один из них по умолчанию отображается на странице, а второй – при наведении на него указателя мыши.

Например, давайте создадим рисунки для основных ссылок на страницы, посвященные созданных компанией программам, и на странице формы заказа. При выделении ссылки будет отображаться инвертированный нормальный рисунок. Для простоты назовем нормальные рисунки spread.gif, word.gif, data.gif и order.gif. Инвертированные рисунки пусть имеют те же имена, к которым добавлены цифра 2. Например, spread2.gif.



При создании рисунков для изменяющихся изображений помните, что обе версии рисунка должны иметь одинаковый размер. В противном случае рисунки будут накладываться друг на друга или размещаться неправильно.

Создание обработчика события

После создания необходимых рисунков просто добавьте их к ссылкам, используя обработчики событий onMouseOver и onMouseOut. **В листинге 22.5 приведен код HTML графических ссылок, изменяющихся при** наведении **на них указателя мыши.**

Листинг 22.5. Обработчики событий изменяющихся рисунков

```
1:     <A HREF="spread.html"
2:          onMouseOver="document.images[1].src='spread2.gif';"
3:          onMouseOut="document.images[1].src='spread.gif';">
4:     <IMG BORDER=0 SRC="spread.gif" height=28 width=173></A>
5:     <A HREF="word.html"
6:          onMouseOver="document.images[2].src='word2.gif';"
7:          onMouseOut="document.images[2].src='word.gif';">
8:     <IMG BORDER=0 SRC="word.gif" height=28 width=225></A>
```


http://ITBookz.ru/

```
9:      <A HREF="data.html"
10:          onMouseOver="document.images[3].src='data2.gif';"
11:          onMouseOut="document.images[3].src='data.gif';">
12:     <IMG BORDER=0 SRC="data.gif" height=28 width=121></A>
13:     <A HREF="order.html"
14:          onMouseOver="document.images[4].src='order2.gif';"
15:          onMouseOut="document.images[4].src='order.gif';">
16:   <IMG BORDER=0 SRC="order.gif" height=28 width=152></A>
```

В нашем примере весь код изменяющегося рисунка включен в оператор обработчика событий, а не в отдельную функцию. Обработчик события каждой ссылки при его запуске использует свой файл рисунка для выделения ссылки.



Чтобы сделать вашу программу эффективнее и быстрее, необходимо предварительно загрузить все используемые в документе рисунки, даже если они в текущий момент и не отображаются. Детально о предварительной загрузке рисунков рассказано в главе 15. Полный сценарий улучшенной Web-страницы, приведенный в следующем разделе, содержит код этой операции.

Составление полного программного кода

В течение этого занятия отдельно рассмотрены три разных элемента, позволяющих улучшить внешний вид начальной страницы компании *FSC*. раскрывающийся список, описания в строке состояния и изменяющийся рисунок, используемый в качестве ссылки.

Каждый из описанных элементов имеет свои недостатки и преимущества. Чтобы добиться максимального эффекта, необходимо добавить на Web-страницу их все. В листинге 22.6 приведен полный код измененного документа HTML, содержащего все перечисленные выше средства.

Листинг 22.6. Полный документ HTML

```
1:      <HTML>
2:      <HEAD>
3:      <TITLE>Fictional Software Company</TITLE></HEAD>
4:      <SCRIPT LANGUAGE="JavaScript">
5:      o2=new Image();
6:      o2.src="oreder2.gif"
7:      d2=new Image();
8:      d2.src="data2.gif";
9:      w2=new Image();
10:     w2.src="word2.gif";
11:     s2=new Image();
12:     s2.src="spread2.gif";
13:      function Navigate() {
14:     var prod=document.navform.program.selectedIndex;
15:     var cat=document.navform.category.selectedIndex;
16:     var prodval=document.navform.program.options[prod].value;
17:     var catval=document.navform.category.options[cat] .value;
18:         if (prodval=="x" ! ! catval=="x") return;
19:          window.location=prodval+"_"+catval+".html";
20:     }
21:     </SCRIPT>
23:     </HEAD>
```


http://ITBookz.ru/

```
24:    <BODY>
25:    <IMG SRC="fsclogo.gif" alt="Fictional Software Company">
26:    <HR>
27:    <A HREF="spread.html"
28:         onMouseOver="document.images[1].src='spread2.gif';"
29:         onMouseOut="document.images[1].src='spread.gif';">
30:    <IMG BORDER=0 SRC="spread.gif" height=28 width=173></A>
31:    <A HREF="word.html"
32:         onMouseOver="document.images[2].src='word2.gif';"
33:         onMouseOut="document.images[2].src='word.gif';">
34:    <IMG BORDER=0 SRC="word.gif" height=28 width=225></A>
35:    <A HREF="data.html"
36:         onMouseOver="document.images[3].src='data2.gif';"
37:         onMouseOut="document.images[3].src='data.gif';">
38:    <IMG BORDER=0 SRC="data.gif" height=28 width=121x/A>
39:    <A HREF="order.html"
40:         onMouseOver="document.images[4].src='order2.gif';"
41:         onMouseOut="document.images[4].src='order.gif';">
42:    <IMG BORDER=0 SRC="order.gif" height=28 width=152></A>
43:    <P>Welcome to our page! Fictional Software Company
44:    specializes in creating innovative, user-friendly software
45:    applications with descriptions filled with industry
46:    buzzwords.
47:    We have a wide range of products
48:    </P>
49:    <UL>
50:    <LI><A HREF="spread.html"
51:    onMouseOver="window.status='Spreadsheet Info'; return true;"
52:    onMouseOut="window.status='';">
53:    Fictional Spreadsheet 7.0<A>
54:    <LIXA HREF="word.html"
55:    onMouseOver="window.status='Word Processor Info'; return true;"
56:    onMouseOut="window.status='';">
57:    Fictional Word Processor 6.0<A>
58:    <LIXA HREF="data.html"
59:    onMouseOver="window.status='Database Info'; return true;"
60:    onMouseOut="window.status='';">
61:    Fictional Database 7.0<A>
62:    </UL>
63:    <P>
64:    Unlike other software companies, our product have
65:    absolutely no bugs, and always work properly on all
66:    computers. Nevertheless, you may run into problem in
67:    rare cases, usually your own fault. If this happens,
68:    visit our <A HREF="support.html"
69:    onMouseOver="window.status='Technical Support';return true;"
70:         onMouseOut="window.status='';"">
71:    Technical Support</A>
72:    department for helpful information and tips. You can
73:    also view more information <A HREF="company.html"
74:    onMouseOver="window.status='FSC Software Co'; return true;"
75:         onMouseOut="window.status='';">
76:    about our company</A> or order products with our friendly
77:    <A HREF="order.html"
```


http://ITBookz.ru/

```
78:     onMouseOver="window.status='Order Products';return true;"
79:          onMouseOut="window.status='';">
80:    Order Form</A>
81:          <FORM name="navform">
82:          <SELECT name="program">
83:         <OPTION VALUE="x"  SELECTED>Select Product
84:         <OPTION VALUE="w">Fictional Word Processor
85:         <OPTION VALUE="s">Fictional Spreadsheet
86:         <OPTION VALUE="d">Fictional Database
87:          </SELECT>
88:          <SELECT name="category">
89:           <OPTION VALUE="x"  SELECTED>Select a Category
90:         <OPTION VALUE="tech">Technical Support
91:         <OPTION VALUE="sales">Sales and availability
92:          <OPTION VALUE="feat">List of Features
93:         <OPTION VALUE="price">Pricing Information
94:         <OPTION VALUE="tech">Tips and Techniques
95:          </SELECT>
96:         <INPHT TYPE="button" NAME="go" VALUE="Go to Page"
97:          onClick="Navigate();">
98:          </FORM>
99:    <HR>
100:   <I>(c) 1998 FSC - designed by the FSC staff</I>
101:   </BODY>
102:   </HTML>
```

На рис. 22.3 показана усовершенствованная начальная страница в окне Netscape Navigator. Конечно, страница с небольшим количеством содержимого со всеми добавленным на нее дополнительными элементами выглядит громоздко. На других документах HTML они выглядят совсем по-другому.

Резюме

За этот час вы использовали разные средства JavaScript, изученные на предыдущих занятиях, для улучшения готового документа HTML и повышения его привлекательности.

В следующей главе вы создадите более сложное приложение — сценарий компьютерной игры.

Вопросы и ответы

Существует ли способ добавления на Web-страницу всплывающих подсказок, вместо добавления описания в строке состояния?

С помощью JavaScript всплывающие подсказки не создаются. Подобного эффекта можно добиться с помощью динамического HTML.

Можно ли загрузить усовершенствованную на этом занятии страницу в других броузерах?

Да. Вы также можете изменить ее, чтобы отправлять значения раскрывающихся списков в сценарий CGI. Это медленнее, но тоже выполняется.

Можно ли добавить в строку состояния описания графических ссылок?

Да. Поскольку в этом случае код обработчика событий будет выглядеть громоздко, лучше создать специальную функцию.

http://ITBookz.ru/

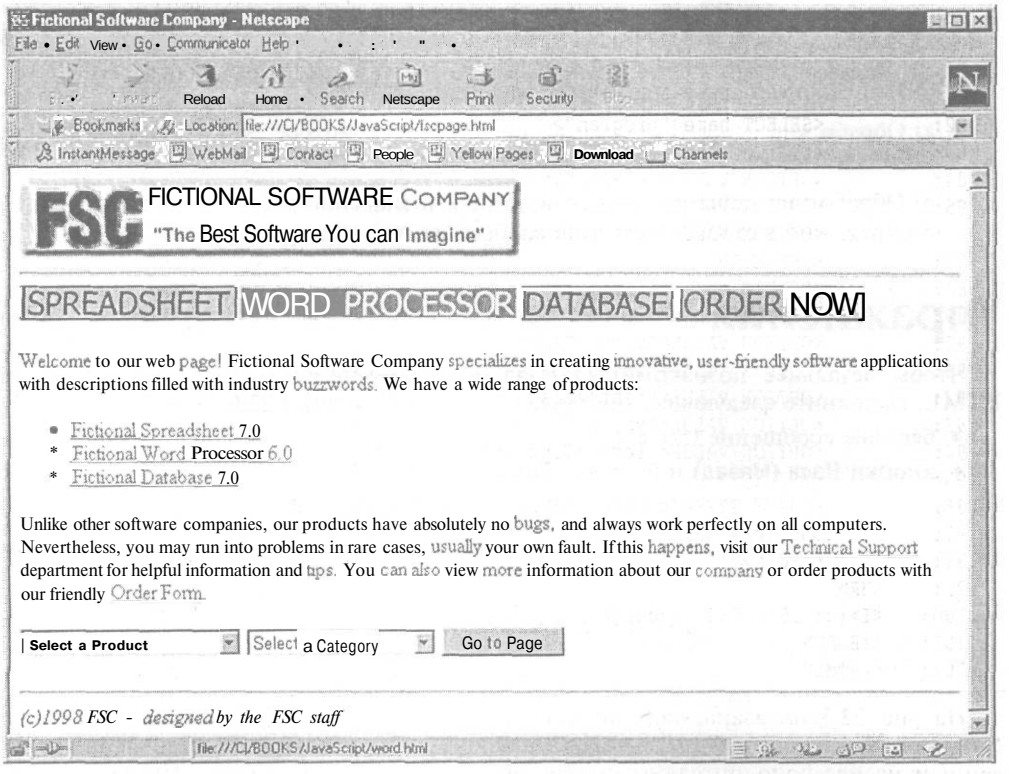


Рис. 22.3. Усовершенствованный документ HTML, отображенный в окне Netscape Navigator

Семинар

Контрольные вопросы

1. К чему в созданном сценарии относится свойство `document.navform.category.selectedIndex`?
   a) Номеру выделенного элемента
   b) К тексту элемента раскрывающего списка
   c) *К* индексу другого элемента
2. Какой из дескрипторов используется для добавления обработчика событий `onMouseOver` при создании изменяющихся рисунков?
   a) **<A>**
   b) `<IMG>`
   C) `<MOOSE>`
3. Что неправильно в коде обработчика `onMouseOver="window.status='test';"`?
   a) Он содержит слишком краткое описание ссылки
   b) Он не возвращает значение true
   c) После него не указан обработчик `onMouseOut`

http://ITBookz.ru/

Ответы

1, a) Это свойство определяет номер выделенного элеме

2, a) Обработчик событий определяется в дескрипторе <A>. Internet Explorer позволяет также определять его и в дескрипторе <IMG>

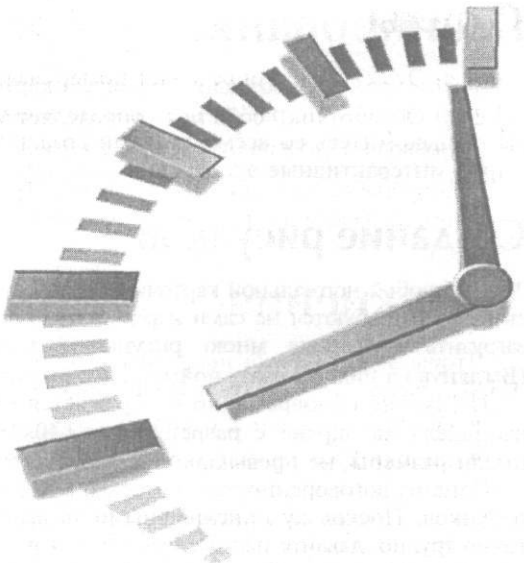
3, b) Обработчик события должен возвращать значение true, чтобы предотвратить отображения в строке состояния сообщения по умолчанию.

Упражнения

Чтобы детальнее познакомиться с элементами усовершенствования документов HTML, выполните следующее. Добавьте в документ листинга 22.6:

- бегущее сообщение (глава 6)
- кнопки Back (Назад) и Forward (Вперед) (глава 10)

http://ITBookz.ru/



23-й час

Создание сценария игры

Всего два часа вам осталось провести с этой книгой. Сейчас речь пойдет о создании сложного комплексного приложения JavaScript. Чтобы немного заинтересовать вас, давайте создадим компьютерную игру "Покер", которую вы встретите в любом казино мира.

В этой главе рассмотрены следующие темы.

- **Создание рисунков для сценария**
- **Выбор переменных для сохранения данных**
- **Создание структуры сценария**
- **Написание программного кода**

http://ITBookz.ru/

Планирование программы

В этом уроке мы создадим сценарий карточной игры. Это будет известная вам игра "Покер". Чтобы сыграть в нее, вам не понадобятся деньги. В процессе ее написания вы познакомитесь со всеми этапами создания сложных приложений, содержащих рисунки, интерактивные элементы и математические расчеты.

Создание рисунков

Для любой нормальной карточной игры, пусть даже и компьютерной, необходимы карты. Вам потребуются не сами карты, а их изображения. Если вы не художник, то можете загрузить созданные мною рисунки 52 карт с Web-узла http://www.jsworkshop.com. (Взглянув на них, вы сразу поймете, что я тоже не художник.)

На экране одновременно отображается пять карт. Для того чтобы игра нормально выглядела на экране с разрешением 640x480 пикселей, необходимо, чтобы рисунки имели размеры, не превышающие 136×106 пикселей.

Одна из договоренностей, которым мы будем следовать, относится к именам файлов рисунков. Поскольку описательными названиями карт оперировать в JavaScript достаточно трудно, давайте использовать в их качестве числовые значения от 1 до 13 (туз — 1, король — 13, дама — 12, валет — 11 и т.д.). Масть будем определять одной первой буквой *(c, s, h* и *d*). Например, файл рисунка валета треф будет иметь название 11c.gif.

Кроме рисунков карт, я создал рисунки оформления Web-страницы и рисунки кнопок Hold (Сдать), Deal (Раздать) и Draw (Взять).

Выбор переменных

Следующий этап в создании нашего приложения — мы выбираем все необходимые переменные. Можно, конечно, добавить переменные и по ходу создания сценария, но это не позволяет правильно спланировать структуру программы.

Листинг 23.1 содержит определения глобальных переменных нашего сценария. В нем определены следующие переменные.

- score. Целочисленная переменная, сохраняет текущий счет игрока. Начальное значение 100. Одно очко прибавляется при каждой раздаче карт. Счет игрока определяется раскладом карт, оставшихся на руках у игрока.
- dealt. Флаг, который используется для определения момента раздачи карт. Начиная с этого момента, игрок может сдавать или брать карты.
- hand. Массив из пяти карт, содержащий значения карт, находящихся на руках у игрока.
- held. Массив флагов, определяющий статус каждой карты (оставленная или сброшенная).
- deck. Массив, сохраняющий значения всех карт колоды (52 элемента).

Листинг 23.1. Глобальные переменные сценария

```
1:          var score=100;
2:          var dealt=false;
3:          var hand=new Array(6);
4:          var held=new Array(6);
5:          bar deck=new Array(53);
```


http://ITBookz.ru/

Для упрощения сохранения данных создадим объект, который будет представлять карту. Объект будет содержать номер карты и функцию расчета имени файла ее рисунка (листинг 23.2).

Листинг **23.2.** Определение объекта карты

```
1:        //Создание имени файла рисунка карты объекта Card
2:        function fname() {
3:              return this.num+this.suit+".gif";
4:        }
5:        //Создание объекта Card
6:        function Card(num,suit) {
7:              this.num=num;
8:              this.suit=suit;
9:              this.fname=fname;
10:       }
```

Строки 2–4 определяют функцию, которая вычисляет имя файла рисунка карты, а строки 6–10 определяют объект Card.

Создание документа HTML

Перед написанием сложного сценария необходимо выполнить еще одно обязательное задание: создать структуру документа HTML, в которую он будет вставляться. Листинг 23.3 представляет базовый документ HTML для нашего сценария. В нем для выравнивания кнопок и рисунков карт используется таблица HTML.

Листинг **23.3.** Документ HTML для игры

```
1:        <HTML>
2:        <HEAD>
3:        <TITLE>Draw Poker</TITLE>
4:        </HEAD>
5:        <BODY>
6:        <IMG SRC="title.gif" width=381 height=81>
7:        <HR>
8:        <FORM NAME="form1">
9:        <TABLE>
10:       <TR>
11:             <TD><IMG border=0 src="blank.gif" height=136 width=106>
12:           <TD><IMG border=0 src="blank.gif" height=136 width=106>
13:            <TD><IMG border=0 src="blank.gif" height=136 width=106>
14:           <TD><IMG border=0 src="blank.gif" height=136 width=106>
15:           <TD><IMG border=0 src="blank.gif" height=136 width=106>
16:            <TD></TD>
17:       </TR>
18:       <TR>
19:       <TD> <A HREF="#" onClick="Hold(1);">
20:             <IMG border=0 src="hold.gif" height=50 width=106x/A>
21:       <TD> <A HREF="#" onClick="Hold(2);">
22:            <IMG border=0 src="hold.gif" height=50 width=106></A>
23:       <TD> <A HREF="#" onClick="Hold(3);">
24:             <IMG border=0 src="hold.gif" height=50 width=106></A>
25:       <TD> <A HREF="#" onClick="Hold(4);">
26:            <IMG border=0 src="hold.gif" height=50 width=106x/A>
27:       <TD> <A HREF="#" onClick="Hold(5);">
28:             <IMG border=0 src="hold.gif" height=50 width=106></A>
29:       </TR>
30:       <TR>
31:            <TD> <B>Total<BR>Score:</B>
```


http://ITBookz.ru/

```
32: '                    <INPUT TYPE="text" SIZE=6 NAME="total" VALUE="100"></TD>
33:               <TD colspan=2> <B>Current<BS>Hand:</B>
34:                    <INPUT TYPE="text" SIZE=20 NAME="message"
35:                    VALUE="Press DEAL to begin.">
36:               <TD>
37:               <TD> <A HREF="#" onClick="DealDraw();">
38:                    <IMG BORDER=0 SRC="deal.gif" HEIGHT=50 WIDTH=106></A>
39:          </TR>
40:          </TABLE>
41:          </FORM>
42:          </BODY>
43:          </HTML>
```

Вот как выполняется этот документ HTML.

- **Строка 9 открывает код таблицы HTML, позволяющей выровнять на странице все элементы.**
- **Строки** 10–17 **определяют строку таблицы, содержащую пять карт. Пустой рисунок (blank.gif) используется потому, что карты еще не розданы.**
- **Строки** 18–29 **определяют вторую строку таблицы. Эта строка содержит кнопку Hold для каждой карты.**
- **Строки** 30–39 **определяют третью строку таблицы. Эта строка содержит текстовые поля отображения** общего **счета и статуса, а также кнопку Deal.**

На рис. 23.1 **показан этот документ HTML, зафуженный в броузере перед началом ифы.**

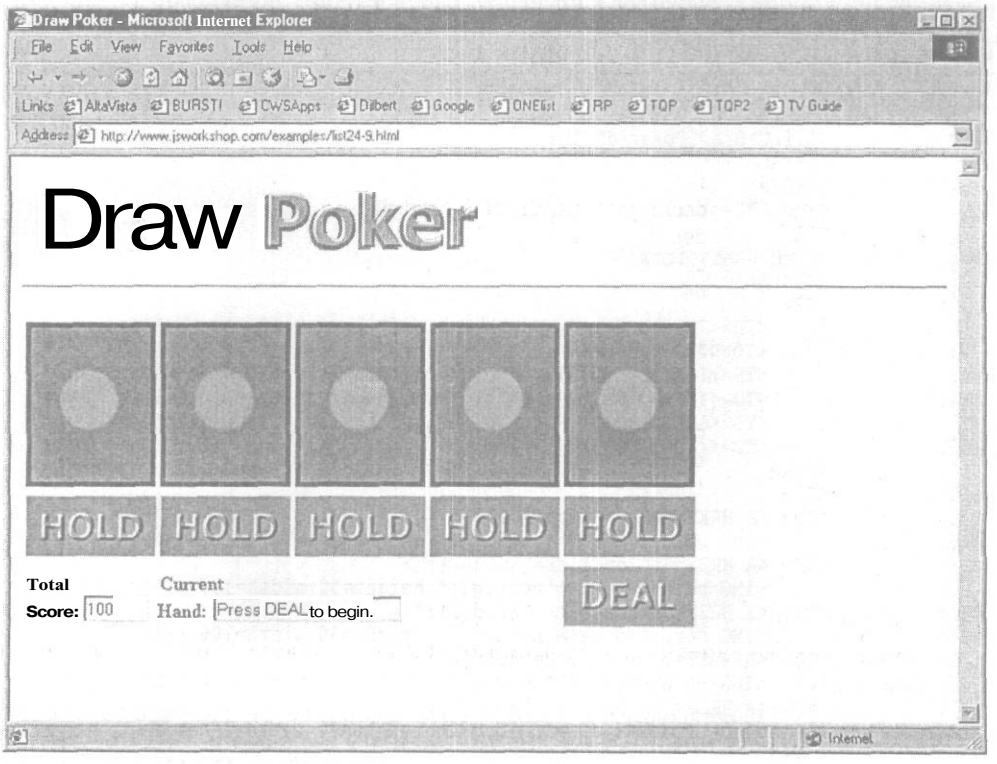


Рис. 23.1. Полный документ HTML до вставки сценария

http://ITBookz.ru/

Составление сценария

Теперь вы можете приступать к созданию сценария игры. Поскольку создается сценарий игры, управление им должен полностью выполнять пользователь. Каждая создаваемая функция будет вызываться обработчиком событий в документе HTML.

Управление кнопками Draw или Deal

Поскольку кнопки Draw и Deal отображаются на одном и том же месте страницы, для их управления достаточно определить одну функцию DealDraw:

```
function DealDraw() {
  if (deal==true) Draw();
  else Deal();
}
```

Эта функция выполняет проверку статуса карт (розданы или нет). Если да, то вызывается функция Draw, иначе — Deal.

Тасование колоды

Функция Deal раздает игроку пять карт, которые располагаются в пяти определенных ячейках таблицы HTML. Но перед раздачей необходимо перетасовать колоду. В листинге 23.4 представлена первая половина функции Deal, позволяющая заполнить массив deck картами и перетасовать их.

Листинг 23.4. Перетасовка карт в колоде

```
1:          function Deal() {
2:          //Заполнение колоды картами
3:               for (i=1; i<14; i++) {
4:                    deck[i] = new Card(i,"c");
5:                    deck[i+13] = new Card(i,"h");
6:                    deck[i+26] = new Card(i,"s");
7:                    deck[i+39] = new Card(i,"d");
8:               }
9:          //Перетасовка колоды
10:              var n = Math.floor(400 * Math.random() + 500);
11:              for (i=1; i<n; i++) {
12:                   card1 = Math.floor(52*Math.random() + 1);
13:                   card2 = Math.floor(52*Math.random() + 1);
14:                   temp = deck[card2];
15:                   deck[card2] = deck[card1];
16:                   deck[card1] = temp;
17:              }
```

Вот как выполняется этот код.

- Строки 3–8 заполняют колоду картами. Сначала колода заполняется картами одной масти, затем — второй и т.д.
- Строки 9–17 перетасовывают колоду. Это выполняется следующим образом. Выбираем произвольное число (строка 10) и на его основе создаем цикл. За каждую итерацию цикла выбираются две карты (строки 12, 13) и переразмещаются в колоде (строки 14–16).

http://ITBookz.ru/

Раздача карт

Вторая половина функции Deal относится к механизму раздачи карт. Листинг 23.5 представляет ее код.

Листинг 23.5. Раздача карт

```
1:          //Раздача и отображение карт
2:            for (i=1; i<6; i++) {
3:                  hand[i] = deck[i];
4:                   document.images[i].src = hand[i].fname();
5:                   document.images[i+5].src = "hold.gif";
6:                  held[i] = false;
7:              }
8:              dealt = true;
9:              score = score - 1;
10:             document.form1.total.value = score;
11:             document.images[11].src="draw.gif";
12:             Addscore();
13:         }
```

Эта половина функции выполняет следующие действия.

- Строка 2 начинает цикл в пять итераций.
- Строка 3 выбирает следующую карту из колоды.
- Строка 4 отображает розданную карту.
- Строка 5 изменяет вид кнопки на Hold (второй ее вариант отображается после щелчка на ней).
- Строка 6 обнуляет флаг held.
- Строка 7 завершает итерацию цикла.
- Строки 8, 9 определяют значение флага dealt.
- Строка 10 отображает счет в текстовом поле.
- Строка 11 изменяет кнопку Deal на Draw.
- Строка 12 вызывает функцию Addscore вычисления счета для розданных карт. Детально эта функция рассмотрена ниже.

Сдача карт

Вам необходима функция, которая будет вызываться после щелчка на кнопке Hold. **В листинге 23.6 представлен** полный ее код.

Листинг 23.6. Функция Hold

```
1:          //Сдача карт
2:          function Hold(num) {
3:               if ( !dealt) return;
4:               if ( !held[num]) {
5:                    held[num]=true;
6:                    document.images[5+num].src="hold2.gif";
7:               }
8:               else {
```


http://ITBookz.ru/

```
9:                    held[num]=false;
10:                   document.images[5+num].src="hold.gif";
11:             }
12:       }
```

Эта функция использует один параметр, определяющий сдаваемую карту. Строка 3 определяет закрытие функции, если карту не сдают. В строках 3-12 определяются флаг held, вид кнопки Hold и статус карт на руках у пользователя.

Добор новой карты

Функция Draw вызывается после щелчка на кнопке Draw. Она позволяет добрать вместо сдаваемых карт новые из колоды (с помощью кнопки Hold). В листинге 23.7 представлен полный ее код.

Листинг 23.7. Код отображения новых карт

```
1:          //Добор новых карт
2:          function Draw() {
3:               var curcard = 6;
4:             for (i=1; i<6; i++) {
5:                    if (!held[i]) {
6:                    hand[i] = deck[curcard++];
7:                     document.images[i].src = hand[i].fname();
8:                     }
9:               }
10:              dealt = false;
11:              document.images[11].src="deal.gif";
12:              score += Addscore();
13:             document.form1.total.value = score;
14:       }
```

Вот что делает эта функция.

- Строка 3 определяет локальную переменную curcard, используемую для указания следующей отображаемой карты. Поскольку всего на руках у игрока только пять карт, начальное ее значение 6.
- Строки 4-9 задают цикл проверки статуса карты по значению массива held. Если карта сдается, то она заменяется следующей картой и изображение страницы обновляется.
- Поскольку перебираются все карты, находящиеся на руках у пользователя, в строках 10 и 11 переменной dealt определяется значение false и кнопка Deal заменяет кнопку Draw.
- Строки 12 и 13 вызывают функцию определения счета. Детально она рассмотрена в следующем разделе.

Определение счета игрока

Последняя рассматриваемая нами функция — это Addscore, позволяющая определить очки пользователя. Очки определяются по наличию у игрока следующих комбинаций карт.

http://ITBookz.ru/

- Одна пара (валетов и выше) — 1 очко
- Две пары — 2 очка
- Три короля — 3 очка
- — 4 очка
- Флэш — 5 очков
- Фул хаус — 10 очков
- Четыре короля — 25 очков
- Прямой флэш — 50 очков
- Королевский флэш: 100 очков

В листинге 23.8 представлен код этой функции.

Листинг 23.8. Подсчет очков игрока

```
1:          //Вычисление счета
2:          function Addscore() {
3:               var straight = false;
4:               var flush = false;
5:               var pairs = 0;
6:               var three = false;
7:               var tally = new Array(14);
8:          //Сортировка по категориям
9:               var nums = new Array(5);
10:           for (i=0; i<5; i++) {
11:                nums[i] = hand[i+1].num;
12:           }
13:           nums.sort(Numsort);
14:      //Флэш
15:           if (hand[1].suit == hand[2].suit &s
16:                hand[2].suit == hand[3].suit &&
17:                hand[3].suit == hand[4].suit s&
18:                hand[4].suit == hand[5].suit) flush = true;
19:      //Стрит (малый)
20:           if (nums[0] == nums[1] - 1 SS
21:                nums[1] == nums[2] - 1 &S
22:                nums[2] == nums[3] - 1 S&
23:                nums[3] == nums[4] - 1) straight = true;
24:      //Стрит (большой)
25:           if (nums[0] == 1 && nums[1] == 10 SS nums[2] == 11
26:                SS nums[3] == 12 SS nums[4] == 13)
27:                straight = true;
28:      //Королевский флэш, прямой флэш, стрит, флэш
29:           if (straight SS flush SS nums[4] == 13 SS nums[0] == 1) {
30:                document.form1.message.value="Royal Flush";
31:                return 100;
32:           }
33:           if (straight SS flush) {
34:                document.form1.message.value="Straight Flush";
35:                return 50;
36:           }
37:           if (straight) {
```


http://ITBookz.ru/

```
38:                 document.form1.message.value="Straight";
39:                  return 4;
40:             }
41:             if (flush) {
42:                  document.form1.message.value="Flush";
43:                  return 5;
44:             }
45:          //Счет по отдельным картам
46:           for (i=1; i<14; i++) {
47:                 tally[i] = 0;
48:             }
49:           for (i=0; i<5; i++) {
50:                 tally[nums[i]] += 1;
51:             }
52:            for (i=1; i<14; i++) {
53:                 if (tally[i] == 4) {
54:                      document.form1.message.value = "Four of a Kind";
55:                       return 25;
56:                  }
57:                  if (tally[i] == 3) three = true;
58:                  if (tally[i] == 2) pairs += 1;
59:             }
60:             if (three && pairs == 1) {
61:                  document.form1.message.value="Full House";
62:                  return 10;
63:             }
64:             if (pairs == 2) {
65:                  document.form1.message.value="Two Pair";
66:                  return 2;
67:             }
68:             if (three) {
69:                  document.form1.message.value="Three of a Kind";
70:                  return 3;
71:             }
72:              if (pairs == 1) {
73:                   if (tally[1] == 2 ¦ ¦ tally[11]==2
74:                    i 1 tally[12] == 2 ]  ¦  tally[13]==2) {
75:                        document.form1.message.value="Jacks or Better";
76:                        return 1;
77:                   }
78:             }
79:             document.form1.message.value="No Score";
80:          return 0;
81:          }
```

Эта функция проводит некоторые сложные вычисления. Понять их не так уж и сложно.

- Строки 3–7 объявляют переменные и флаги для разных раскладов карт игрока.
- Строки 8–13 создают массив nums, определяющий разные категории наборов карт. Он позволяет упростить процедуру определения счета.
- Строки 14–18 определяют флэш.

http://ITBookz.ru/

- Строки 19-27 определяют стрит (последовательный набор карт). Поскольку туз может иметь индекс либо 1, либо 13, стрит с ним определяется отдельно.
- Строки 28-32 определяют королевский флэш (10, валет, дама, король, туз).
- Строки 33–44 возвращают значения описанных выше категорий.
- Строки 45–51 создают массив tally. Он позволяет рассчитать очки отдельных карт в раскладе игрока.
- Строки 52–71 определяют допустимые комбинации отдельных карт, за которые насчитываются очки.
- Если в предыдущем коде определены пары, то строки 72–78 определяют наличие пар валетов и возвращают соответствующий счет.
- Если комбинации карт, за которые насчитываются очки, не найдены, то в строках 79 и 80 выводится сообщение о добавлении нуля очков.

Готовый документ со сценарием

Листинг 23.9 содержит готовый программный код документа со сценарием игры. Если вы не хотите вводить его самостоятельно, то загрузите с Web-узла http://www.jsworkshop.com/.

Листинг 23.9. Полный код сценария игры "Покер"

```
<HTML>
<HEAD>
<TITLE>Draw Poker</TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
var score = 100;
var dealt = false;
var hand = new Array(6);
var held = new Array(6);
var deck = new Array(53);
function DealDraw() {
    if (dealt == true) Draw();
 else Deal();
 }
function Deal() {
 //Заполнение колоды картами
   for (i=1; i<14; i++) {
          deck[i] = new Card(i,"c");
          deck[i+13] = new Card(i,"h");
          deck[i+26] = new Card(i,"s");
          deck[i+39] = new Card(i,"d");
     }
 //Перетасовка колоды
    var n = Math.floor(400 * Math.random() + 500);
 for (i=1; i<n; i++) {
          card1 = Math.floor(52*Math.random() + 1);
          card2 = Math.floor(52*Math.random() + 1);
          temp = deck[card2];
          deck[card2] = deck[card1];
          deck[card1] = temp;
```


http://ITBookz.ru/

```
30:          }
31:        //Раздача и отображение карт
32:          for (i=1; i<6; i++) {
33:               hand[i] = deck[i];
34:               document.images[i].src = hand[i].fname();
35:               document.images[i+5].src = "hold.gif";
36:               held[i] = false;
37:          }
38:          dealt = true;
39:          score = score - 1;
40:          document.form1.total.value = score;
41:          document.images[11].src="draw.gif";
42:          Addscore();
43:       }
44:       //Сдача карт
45:       function Hold(num) {
46:          if (!dealt) return;
47:          if (!held[num]) {
48:               held[num]=true;
49:               document.images[5+num].src="hold2.gif";
50:          }
51:          else {
52:               held[num]=false;
53:               document.images[5+num].src="hold.gif";
54:          }
55:       }
56:       //Добор новых карт
57:       function Draw() {
58:          var curcard = 6;
59:          for (i=1; i<6; i++) {
60:               if (!held[i]) {
61:               hand[i] = deck[curcard++];
62:               document.images[i].src = hand[i].fname();
63:               }
64:          }
65:          dealt = false;
66:          document.images[11].src="deal.gif";
67:          score += Addscore();
68:          document.form1.total.value = score;
69:       }
70:       //Создание имени файла рисунка карты объекта Card
71:       function fname() {
72:          return this.num + this.suit + ".gif";
73:       }
74:       //Создание объекта Card
75:       function Card(num,suit) {
76:            this.num = num;
77:          this.suit = suit;
78:          this.fname = fname;
79:       }
80:       //Функция сортировки
81:       function Numsort(a,b) { return a - b; }
```


http://ITBookz.ru/

```
82:          //Вычисление счета
83:          function Addscore() {
84:               var straight = false;
85:               var flush = false;
86:               var pairs = 0;
87:               var three = false;
88:               var tally = new Array(14);
89:          //Сортировка по категориям
90:               var nums = new Array(5);
91:               for (i=0; i<5; i++) {
92:                    nums[i] = hand[i+1].num;
93:               }
94:               nums.sort(Numsort);
95:          //Флэш
96:               if (hand[1].suit == hand[2].suit &&
97:                    hand[2].suit == hand[3].suit &&
98:                    hand[3].suit == hand[4].suit &&
99:                    hand[4].suit == hand[5].suit) flush = true;
100:         //Стрит (малый)
101:              if (nums[0] == nums[1] - 1 S&
102:                   nums[1] == nums[2] - 1 &&
103:                   nums[2] == nums[3] - 1 &&
104:                   nums[3] == nums[4] - 1) straight = true;
105:         //Стрит (большой)
106:              if (nums[0] == 1 && nums[1] == 10 && nums[2] == 11
107:                   && nums[3] == 12 && nums[4] == 13)
108:                   straight = true;
109:         //Королевский флэш, прямой флэш, стрит, флэш
110:              if (straight && flush && nums[4] == 13 && nums[0] == 1) {
111:                   document.form1.message.value="Royal Flush";
112:                   return 100;
113:              }
114:              if (straight && flush) {
115:                   document.form1.message.value="Straight Flush";
116:                   return 50;
117:              }
118:              if (straight) {
119:                   document.form1.message.value="Straight";
120:                   return 4;
121:              }
122:              if (flush) {
123:                   document.form1.message.value="Flush";
124:                   return 5;
125:              }
126:         //Счет по отдельным картам
127:              for (i=1; i<14; i++) {
128:                   tally[i] = 0;
129:              }
130:              for (i=0; i<5; i++) {
131:                   tally[nums[i]] += 1;
132:              }
133:              for (i=1; i<14; i++) {
```


http://ITBookz.ru/

```
134:                if (tally[i] == 4) {
135:                     document.form1.message.value = "Four of a Kind";
136:                      return 25;
137:                }
13S:               if (tally[i] == 3) three = true;
139:               if (tally[i] == 2) pairs += 1;
140:           }
141:           if (three && pairs == 1) {
142:                document.form1.message.value="Full House";
143:                return 10;
144:           }
145:           if (pairs == 2) {
146:                document.form1.message.value="Two Pair";
147:                return 2;
148:           }
149:           if (three) {
150:                document.form1.message.value="Three of a Kind";
151:                return 3;
152:           }
153:           if (pairs == 1) {
154:              if (tally[l] == 2 ]  ¦  tally[ll]==2
155:                  | ¦ tally[12] == 2 ¦ j tally[13]==2) {
156:                   document.form1.message.value="Jacks or Better";
157:                   return 1;
158:               }
159:           }
160:           document.form1.message.value="No Score";
161:      return 0;
162:      }
163:      </SCRIPT>
164:      </HEAD>
165:      <BODY>
166:     <img src="title.gif" width=381 height=81>
167:      <HR>
168:      <FORM NAME="form1">
169:      <TABLE>
170:      <tr>
171:           <td> <img border=0 src="blank.gif" height=136 width=106>
172:          <td> <img border=0 src="blank.gif" height=136 width=106>
173:          <td> <img border=0 src="blank.gif" height=136 width=106>
174:          <td> <img border=0 src="blank.gif" height=136 width=106>
175:          <td> <img border=0 src="blank.gif" height=136 width=106>
176:           <td> </td>
177:      </tr>
178:      <tr>
179:           <td> <a href="#" onClick="Hold(1);">
180:                <img border=0 src="hold.gif" height=50 width=106></a>
181:          <td> <a href="#" onClick="Hold(2);">
182:               <img border=0 src="hold.gif" height=50 width=106></a>
183:          <td> <a href="#" onClick="Hold(3);">
184:               <img border=0 src="hold.gif" height=50 width=106></a>
185:          <td> <a href="#" onClick="Hold(4);">
```


3, c) Вероятность получения такого расклада приблизительно один к полумиллиону.

http://ITBookz.ru/

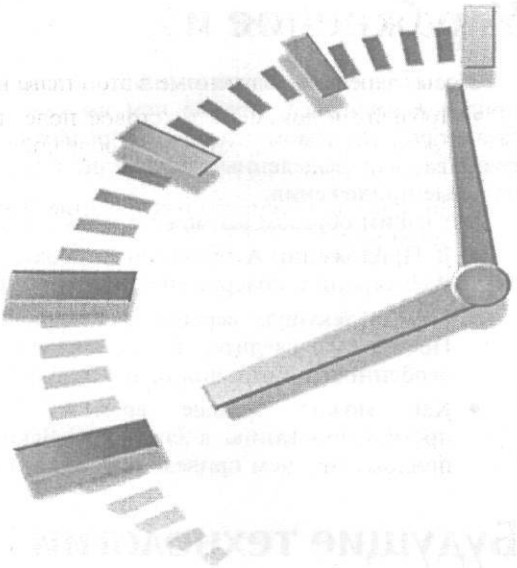
```
156:                 <img border=0 src="hold.gif" height=50 width=106></a>
157:            <td> <a href="#" onClick="Hold(5);">
188:                 <img border=0 src="hold.gif" height=50 width=106></a>
```

Упражнения

Чтобы закрепить полученные в этой главе навыки, выполните следующие упражнения.

- Добавьте в документ текстовое поле, в котором будет отображаться количество побед (раскладов, которые добавляют счет) и проигрышей (раскладов, которые не добавляют счет).
- Добавьте к документу озвучивание. Используйте для этого дескриптор <EMBED> и метод play.

http://ITBookz.ru/



24-й час

Тенденции развития технологий Web

Теперь вы не только имеете представление о программировании на JavaScript, но и умеете создавать простые сценарии и сложные приложения. Самое время рассмотреть тенденции развития JavaScript и Web, оценить их будущее взаимодействие и выбрать другие источники обучения этому языку подготовки сценариев.

В главе 24 вы узнаете, что лежит в основе JavaScript и Web, а также познакомитесь с технологиями, которые непосредственно взаимодействуют с JavaScript. В этой главе рассмотрены следующие темы.

- **Дополнительные источники обучения JavaScript**
- **Будущие версии JavaScript и их влияние на структуру сценариев**
- **Основы XML**
- **Технологии будущего**
- **Последние советы и замечания**

http://ITBookz.ru/

Углубленное изучение JavaScript

В этой книге вы познакомились с основами языка JavaScript. Но вам еще многому придется научиться, прежде чем вы сможете гордо называть себя "программистом JavaScript". На самом деле JavaScript находит применение не только в Web. Такие его средства, как выделенные сценарии, позволяют создавать мощные, многофункциональныеприложения.

Вот каким образом вы можете повысить свои познания JavaScript.

- В Приложении А приведен список других книг о JavaScript, а также перечень Web-страниц, содержащих важную информацию для вас.
- Изучив текущую версию спецификации, не останавливайтесь на достигнутом. Постоянно следите за обновлениями языка и тщательно изучайте его особенности и отличия от предыдущих версий.
- Как можно больше времени тратьте на практические занятия по программированию в JavaScript. Вскоре вы научитесь создавать более мощные приложения, чем приведенные в этой книге.

Будущие технологии Web

Web, как общедоступный источник информации, появилась не так давно и продолжает активно изменяться. В следующих разделах мы рассмотрим некоторые из технологий будущего (они уже разработаны, но еще не используются), которые непосредственно повлияют на вид и структуру сценариев.

Будущие версии JavaScript

Поскольку эта книга посвящена версии 1.5, вам придется смириться с фактом, что предыдущие несколько версий языка JavaScript оказались вне предмета рассмотрения. К счастью, ядро языка практически не изменяется от версии к версии, поэтому все старые версии сценариев прекрасно выполняются и в новой версии языка.

Следующая версия языка, скорее всего, будет 2.0. Точная дата ее выхода в свет еще не известна, но команда разработчиков *Netscape* и *ECMA* уже активно работает над ее спецификацией. В версии 2.0 точно будет реализована полная поддержка объектно-ориентированных средств программирования, таких как классы и механизм наследования.

Как и предыдущие версии, JavaScript 2.0 будет полностью совместима со старыми версиями языка. Чтобы быть уверенным в том, что ваши сценарии будут выполняться и в JavaScript 2.0, создавайте их только с помощью стандартных средств языка, не прибегая к незадокументированным, определенным только в специфичном броузере, средствам.

Будущее DOM

В настоящее время уровень стандарта приобрела только модель DOM первого уровня. DOM второго уровня сегодня рассматривается как рекомендация. Такие средства DOM второго уровня, как управление событиями, усовершенствованная поддержка таблиц стилей, частично реализованы в Netscape Navigator 6.0 и Internet Explorer 5.0 и более высоких версий.

http://ITBookz.ru/

В будущем ожидается более тесная интеграция DOM с основными броузерами, что приведет к снижению уровня их несовместимости. Вне всяких сомнений, будущие версии DOM будут содержать новые средства и спецификации. Пока, правда, никакими сведениями о них я не располагаю.

XML

HTML изначально создавался, как язык программирования документов Web. При разработке он базировался на спецификации SGML (Standart Generalized Markup Language — Стандартный обобщенный язык разметки документов). HTML — это, по сути, упрощенная версия SGML, которая используется для создания Web-страниц.

В последнее время на сцене появился относительно новый язык, который претендует на всеобщее внимание Web-дизайнеров. XML (Extensible Markup Language — Расширенный язык разметки документов) токе является упрощенной версией SGML, но он не настолько прост, как HTML. Если HTML имеет строго определенное назначение, то XML используется для выполнения самых разных задач.



XML разработал консорциум *W3C* и опубликовал спецификацию, которая стандартизирует этот язык.

Честно говоря, XML — это не совсем язык программирования. Не существует строго определенного перечня дескрипторов XML, поскольку он просто не определен. XML позволяет создавать собственные языки, в зависимости от выполняемых задач.

Как же использовать этот язык без строго определенного списка операторов? В этом-то и прелесть XML. Для каждой решаемой вами задачи XML позволяет создавать свой набор команд. Например, если вы создаете базу данных рецептов, то можете создать дескрипторы для ингредиентов рецептов, количества ингредиентов и инструкций по изготовлению.

В XML используется метод DTD (Document Type Definition — Определение типа документа). С его помощью определяются дескрипторы, которые будут использовать в конкретном документе. Никаким другим способом нельзя создать перечень используемых в документе дескрипторов, их атрибутов и значений параметров.

Как вы уже, наверное, знаете, XML и сейчас активно используется Web-дизайнерами. Для использования XML в Web необходимо преобразовать каждый его элемент в представление HTML. Эта операция выполняется относительно просто. Изменив правила в программе синтаксического анализа, вы можете изменить формат всех элементов. Автоматически выполнять эту задачу просто, а вот вручную не только сложно, но и долго.

XSL

Содержимое документов XML вводится внутри дескрипторов. XML не позволяет форматировать документы. Внешний вид содержимого документов XML определяется таблицами стилей XSL (Extensible Stylesheet Language — Расширенный язык таблицы стилей).

XSL основывается на XML, но фокусирует свое внимание на внешнем виде элементов документа. С его помощью задаются такие параметры, как размер шрифта, поля, форматирование таблицы документа XML. При использовании XML для создания документа HTML форматирование в нем задается с помощью XSL.



Документы XSL — это те же самые документы XML, в которых используется DTD, задающий дескрипторы таблиц стилей. XSL — это самая молодая спецификация *W3C*.

http://ITBookz.ru/

Планирование стандартов

В истории JavaScript еще не было ни одной такой версии, которая не позволяла бы выполнять большинство сценариев, созданных в старых версиях языка. Тем не менее, определенное число старых сценариев все же не выполняется в новых версиях JavaScript. К ним в большинстве своем относятся сценарии, в которых используются специфические для броузеров средства.

В следующих разделах вы найдете инструкции по написанию универсальных сценариев, которые будут выполняться и в будущих версиях JavaScript. По крайней мере вероятность их невыполнения очень мала.

Совместимость сценариев

Динамический HTML (DHTML) уже используется для создания Web-страниц. Новый стандарт DOM, принятый *W3C*, стандартизирует все объекты, используемые в DHTML. Но не всегда при создании Web-страниц этот стандарт поддерживается. Именно для таких страниц нельзя гарантировать выполняемость в будущих версиях броузеров. Чтобы избежать проблем несовместимости динамических средств с разными броузерами, примите к сведению следующие замечания.

- Определение версии броузера помогает правильно задать код, выполняемый отдельно в каждом типе броузеров.
- Сценарии часто используют специфические для броузера средства, недоступные в других броузерах.
- Процедура создания сценария чаще требует применения метода "проб и ошибок", чем стандартизации кода.

В то время как *W3C* активно старается стандартизировать все новые средства Web-программирования, постоянно рождаются нестандартизованные элементы, которые не входят в состав DOM и тем самым делают проблему несовместимости почти вечной.

Нет ничего плохого в использовании нестандартизованных средств. Но будьте особенно внимательны. Перед публикацией обязательно проверьте выполняемость сценариев в большинстве используемых современных броузерах. В дополнение к этому, лучше создать еще одну копию сценария, которая использует только средства, описанные в стандартах.

Совместимость с HTML

Разработчики броузеров делают все возможное для того, чтобы новые версии программ были максимально совместимы с стандартами *W3C*. По меньшей мере они всегда основываются на этих стандартах при разработке новых приложений. Это автоматически означает, что все Web-страницы, в которых соблюдены стандарты *W3C*, будут прекрасно выполняться и в будущих броузерах. Проблемы же будут возникать только с теми страницами, в которых используются специфические элементы.



В частности, первые выпуски Netscape Navigator 6.0 показали его несовместимость со "старыми" страницами. Старые броузеры "справлялись" с выполнением подобных страниц в большинстве случаев.

Чтобы избежать подобных проблем, старайтесь при написании кода использовать лишь стандартные элементы HTML. К ним относятся не только стандартизированные дескрипторы и их параметры, но и определенный набор правил. Например, всегда

http://ITBookz.ru/

используйте и открывающий и закрывающий дескриптор <p>; размеры таблицы и другие значения всегда берите в кавычки.

Чтобы убедиться в правильности вашего кода, используйте программы проверки документов HTML, описанные в Приложении Б. Эти программы помогут вам разобраться в стандартах и правильно создать совместимый документ HTML.

Понимание документа

Наконец последнее, но не менее важное замечание. Удостоверьтесь, что вы понимаете все операции, которые выполняет сценарий. Документируйте ваши сценарии комментариями. По меньшей мере снабжайте комментариями все длинные и сложные операторы сценария.

При правильном использовании комментариев вам будет намного проще изменять сценарий в будущем, вставлять его в новый документ HTML или переделывать его под использование в другом броузере.

Последние советы

Вот и достигли последнего раздела последнего урока. Перед созданием собственных сценариев примите к сведению последние советы и замечания относительно JavaScript.

Первое, избегайте чрезмерного использования JavaScript и его средств. Всегда оценивайте необходимость добавление того или иного элемента на страницу, а только затем приступайте к программированию.

Второе, не позволяйте сценарию преобладать над средствами броузера по управлению документами Web. Например, очень плохо, если при наведении указателя на ссылку в строке состояния выводится описание ссылки (задается в сценарии), но не выводится URL страницы (задается броузером).

Еще один пример неправильного использования сценария: создание нового окна специального размера. Это нарушает нормальное функционирование кнопок Back (Назад) и Forward (Вперед), а также строки состояния. Хуже того, страницы в подобных окнах выглядят неестественно маленькими даже (особенно) при рассмотрении на больших мониторах.

Семь раз подумайте перед тем, как использовать в сценарии нестандартизованное средство, как удобно бы оно ни было и какие эффекты не добавляло бы на страницу. Помните, что такой сценарий смогут выполнить не все пользователи, а только обладатели броузера, который поддерживает специфическое средство, Новые же броузеры и стандарты в большинстве случаев не поддерживают старые нестандартные средства. Если вы все же включили подобные средства на Web-страницу, то убедитесь, что она открывается в большинстве популярных броузеров.

Наконец тестируйте страницу всеми мыслимыми и немыслимыми способами. Убедитесь, что она выполняется во всех броузерах, которые вам только посчастливилось достать. Детально проверяйте все инструменты и средства Web-страницы. Тестируйте все ссылки и элементы управления. Только проверив "все и вся" можете позволить себе спокойно вздохнуть и перейти к выполнению нового проекта.



Особое внимание обратите на текстовый броузер Lynx. Детально онем вы можете узнать на Web-узле http://www.trill-horae.com/lynx.html.

http://ITBookz.ru/

Резюме

В этом уроке вы о узнали будущем Web-технологий, которое уже сейчас влияет на создаваемые сценарии JavaScript и документы HTML. Вы познакомились с некоторыми спецификациями, которые, несомненно, изменят в скором будущем вид Web.

Вот и все. Вы достигли конца последней главы книги. Искренне надеюсь, что она вам понравилась и вы продолжите изучать JavaScript. Детально об источниках углубленного изучения JavaScript вы узнаете в Приложении А.

Вопросы и ответы

Кроме преобразования в документы HTML, какое еще применение есть у документов XML?

XML прекрасно приспособлен для сохранения любого текста. Разработчики многих приложений, включая текстовые процессоры, часто используют файлы XML.

Зачем беспокоиться о выполнении сценариев в текстовых броузерах, в частности Lynx?

Lynx используется многими пользователями (он чрезвычайно производителен). Это вызвано тем, что огромное количество пользователей подключаются к Internet с помощью таких устройств, как карманные компьютеры и электронные телефонные книжки, для которых Lynx является единственным приемлемым решением.

Что делать, если на какой-то вопрос я не нашел ответа в этой книге?

Изучите источники, описанные в Приложении А. В них вы наверняка найдете ответы на большинство своих вопросов.

Семинар

Контрольные вопросы

1. Какая из моделей DOM рекомендуется *W3C*?
   a) DOM 1.5
   b) DOM первого уровня
   c) DOM второго уровня
2. Когда необходимо использовать новые средства JavaScript?
   a) Сразу после их разработки
   b) Как только они станут поддерживаться броузерами
   c) Только тогда, когда они войдут в один из стандартов
3. Какой из методов гарантирует выполняемость сценарияв в новых броузерах?
   a) Поддержка стандартов HTML, JavaScript и DOM
   b) Тестирование сценариев во всех доступных броузерах
   c) Ожидание выпуска новых броузеров и тестирование сценариев в них

http://ITBookz.ru/

Ответы

1, c) DOM второго уровня

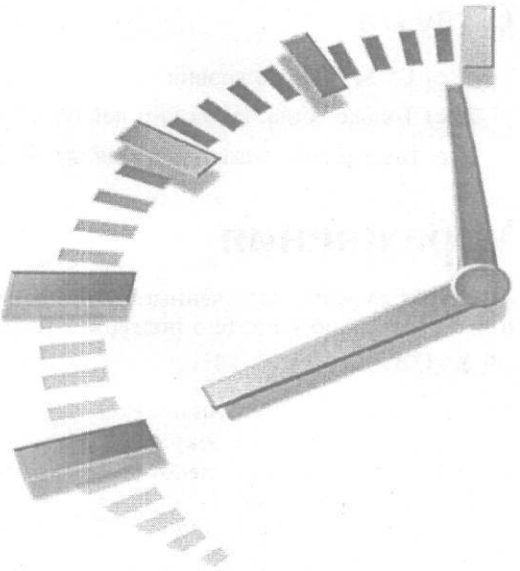
2, c) Только тогда, когда они войдут в один из стандартов

3, a) Поддержка стандартов, принятых *W3C*

Упражнения

Чтобы закрепить полученные в этой главе навыки, исследуйте ресурсы в Приложении A и детально узнайте о поддержке новых средств JavaScript.

http://ITBookz.ru/



Приложение А

Ресурсы JavaScipt

После изучения этой книги вам обязательно понадобятся дополнительные источники для повышения своей квалификации. Если вы хотите продолжить свое обучение, используйте приведенные в этом приложении ресурсы.

Книги

Следующие книги (на английском языке) содержат более полное описание возможностей JavaScript.

- Arman Danesh, *Sams Teach Yourself JavaScript in 24 Days.* ISBN 1-57521-195-5.
- Richard Wagnerl et al., *JavaScript Unleashed.* ISBN 1-57521-306-0

Web-узелы, посвященные JavaScript

Следующие Web-узлы помогут вам при изучении JavaScript.

- Центральный узел разработчиков приложений на JavaScript. Представляет собой подузел узла DevEdge фирмы *Netscape.* Содержит ссылки на самые различные ресурсы JavaScript.
- http: //www.developer.netscape.com/tech/javascript/index.html.
- Полное собрание ссылок на ресурсы по JavaScript содержит узел *Netscape* по адресу http://www.developer.netscape.com/docs /manuals/js/client/jsref/index.htm.
- Web-узел Website Abstraction содержит готовые сценарии и обучающие занятия по JavaScript, в которых описаны и новые объекты DOM.
- http://wsabstract.com/.

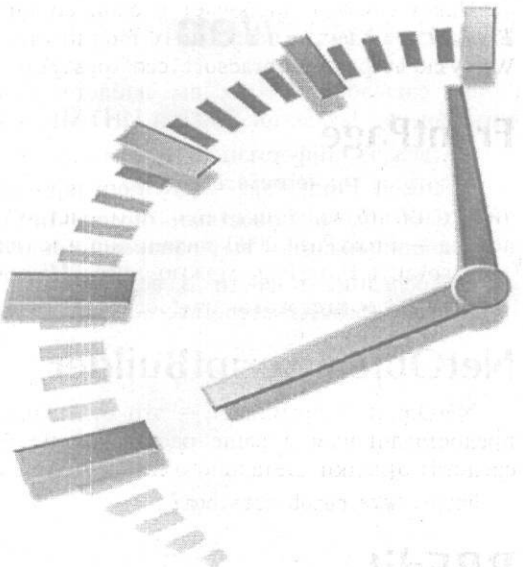
http://ITBookz.ru/

Развитие Web

На следующих узлах вы найдете самые свежие новости и сведения о Web-технологиях: JavaScript, XML и DHTML, а также базовом HTML.

- Свежую информацию и заметки о будущих технологиях вы найдете на узле http://www.webreference.com/.
- С обзорами, новостями, обучающими занятиями по HTML, JavaScript и другим спецификациям вы познакомитесь на узле http://www.htmlcenter.com/.
- Последние новости о встраиваемых в броузеры модулях доступны на узле http://browserwatch.internet.com/.

http://ITBookz.ru/



Приложение Б

Средства разработчика сценариев JavaScript

Одно из преимуществ JavaScript заключается в том, что он не требует никаких специальных средств. Все, что вам потребуется, — это броузер и простой текстовый процессор. При создании сценариев, правда, можно использовать и другие средства. Некоторые из них рассмотрены в этом приложении.

Редакторы HTML и текстовые процессоры

Если вы серьезно настроены создавать сценарии для документов HTML, то вам лучше использовать один из специальных текстовых процессоров. Они позволяют автоматизировать процесс введения операторов и дескрипторов, а также помогают при отладке приложений.

HomeSite

HomeSite создан компанией *Allaire* и представляет собой полноценный редактор HTML. Он очень похож на текстовый процессор, но содержит средства автоматического введения дескрипторов и сложных элементов кода HTML.

Последние его версии позволяют автоматизировать введение операторов JavaScript и выделять разные блоки сценария цветами.

Демо-версия этого редактора загружается с Web-узла компании *Allaire*:

```
http://www.allaire.com/
```


http://ITBookz.ru/

Пакет HomeSite включает в свой состав базовую версию утилиты TopStyte фирмы *Bradsoft*— редактор каскадных таблиц стилей. Детально этот редактор рассмотрен на Web-узле http://www.bradsoft.com/topstyle/.

FrontPage

Microsoft Front Page — это популярный редактор WYSIWYG (как пишется, так и читается!), позволяющий просто и быстро создавать документы HTML. Последняя его версия — FrontPage 2000 рассчитана и на введение операторов сценариев.

Загрузить FrontPage можно с узла *Microsoft*

http://www.microsoft.com/frontpage/

NetObject ScriptBuilder

NetObject ScriptBuilder — это отдельная среда разработки приложений JavaScript, предоставляющая в ваше распоряжение мощный редактор кода и дополнительные средства отладки. Детально о ней вы можете узнать на Web-узле

http://www.netobjects.com/

BBEdit

BBEdit рассчитан на пользователей платформ Macintosh, которые занимаются разработкой документов HTML. Вы можете загрузить BBEdit Lite или приобрести полную его версию на Web-узле фирмы *Bare Bones Software*:

http://www.bbedit.com/

Текстовые редакторы

Любой простой текстовый редактор удовлетворит ваши потребности. Ниже приведены некоторые часто используемые средства.

- TextPad можно использовать вместо встроенного в Windows редактора NotePad. Он очень быстрый, имеет дополнительные средства и простой интерфейс, позволяющий работать с дескрипторами HTML. Вы можете загрузить его с узла http://www.textpad.com/.
- UltratEdit-32 фирмы *IDM Computer Solutions* — это еще один текстовый редактор для Windows, поддерживающий редактирование шестнадцатеричного кода битовых файлов. Загрузить его вы можете с узла http://www.ultraedit.com/.

Проверка кода HTML

Создание Web-страниц, содержащих не только код HTML, но и сценарии, требует проверки правильности как сценариев, так и самих дескрипторов HTML. Не стоит забывать также и о совместимости документов HTML с разными броузерами. Ниже приведены программы, помогающие проверить документ HTML на правильность и выполняемость.

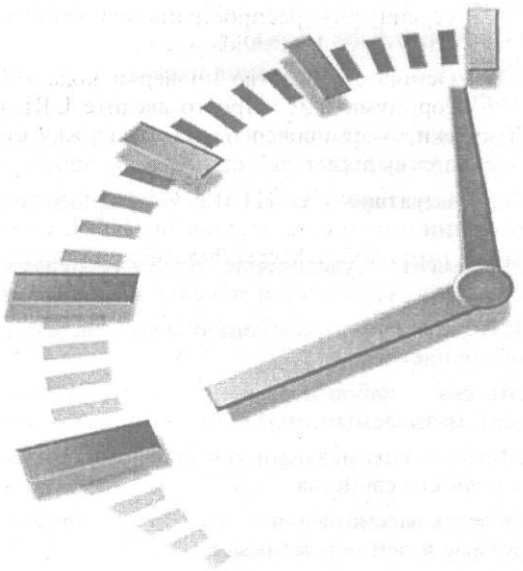
- CSE HTML Validator фирмы *AI Internet Solutions*. Прекрасное решение для платформ Windows. Проверяет документ HTML на соответствие указанной версии HTML. При желании интегрируется в HomeSite, TextPad и некоторые другие редакторы HTML. В то время, как версия Pro коммерческая (не бесплатная),

http://ITBookz.ru/

версия Lite распространяется бесплатно. Для ее загрузки посетите Web-узел http://www.htmlvalidator.com/.

- Основное средство проверки кода HTML — это HTML Validation Service консорциума *W3C*. Просто введите URL необходимой страницы и она автоматически будет проверена на поддержку стандартов HTML. Эта служба доступна по адресу http://validator.com/.
- Служба WDG HTML Validator позволяет, как и предыдущая служба, оценить совместимость документа HTML со стандартом HTML. Доступ к ней вы получите на узле http://htmlhelp.com/tools/validator/.

http://ITBookz.ru/



Приложение В

Словарь терминов

ActiveX — технология, разработанная *Microsoft для* создания программных элементов управления в компьютерах, как правило, с операционной системой Windows. Элементы управления ActiveX внедряются и на Web-страницы.

HTML (Hypertext Markup Language) — гипертекстовый язык разметки документа. Язык, используемый для создания Web-документов. Операторы JavaScript не принадлежат HTML, но их можно вставить в документ HTML.

Java — объектно-ориентированный язык программирования, разработанный фирмой *Sun Microsystems.* Аплеты Java также внедряются в Web-документы. Операторы JavaScript синтаксически похожи на операторы Java, но функционально совершенно другие.

JavaScript — язык создания сценариев для Web-документов, основанный на Java и разработанный фирмой *Netscape.* JavaScript поддерживается большинством современных броузеров.

Navigator — броузер, разработанный *Netscape,* который первым стал поддерживать JavaScript.

VBScript — язык создания сценариев, разработанный фирмой *Microsoft,* синтаксис которого напоминает Visual Basic. VBScript поддерживается только броузером Internet Explorer.

XML — настраиваемый язык программирования, разработанный *W3C,* позволяющий создавать стандартизованные HTML-подобные языки с помощью DTD (Document Type Definition — Определение типа документа).

Аплет — программа, написанная на языке Java, внедряемая на Web-страницу.

Аргумент — параметр, который подставляется в вызываемую функцию. При введении функции аргументы отображаются в скобках.

http://ITBookz.ru/

Булев — тип переменной, который может принимать только два значения – True (Истина) и False (Ложь).

Выражение — комбинация переменных, констант и операторов, совмещенных в одной записи.

Декремент — уменьшение значения переменной на единицу. В JavaScript подобная операция выполняется только с помощью оператора отрицательного приращения.

Знак оператора — символ, разделяющий переменные и константы, записанные в выражении

Инкремент — увеличение значения переменной на единицу. В JavaScript подобная операция выполняется только с помощью оператора положительного приращения.

Интерпретатор — компонент броузера, который интерпретирует операторы JavaScript и выполняет их.

Массив — набор переменных, которые обозначаются одним и тем же именем и числом, называемым индексом.

Метод — специальный тип функции, сохраняемый вместе с объектом и выступающий в роли его свойства.

Область рассмотрения — часть программы JavaScript, в которой объявлены все используемые в ней переменные.

Объект — тип переменной, имеющей несколько значений, называемых свойствами, и функций, называемых методами.

Объектная модель документа (DOM — Document Object Model) — перечень объектов, которые JavaScript использует для обращения к окну броузера и отдельным частям кода HTML. Разработанная консорциумом *W3C*, модель DOM стандартизирует объекты, используемые во всех броузерах, и позволяет получить доступ к любому элементу и ресурсу Web.

Оператор — строка программы или сценария.

Отладка — отыскание и устранение ошибок, неточностей и конфликтов в программе или сценарии.

Параметр — переменная, получаемая функцией при ее вызове. То же самое, что и аргумент.

Переменная — контейнер, имеющий отдельное имя, может принимать числовые, строковые и объектные значения.

Свойство — переменная, которая сохраняется вместе с объектом. Каждый объект может иметь сколько угодно свойств.

Связывание — соединение двух строк в одну длинную строку.

Событие — условие, обычно получаемое в результате выполнения пользователем операций, распознаваемых сценарием.

Строка — набор текстовых символов, объединенных одной переменной.

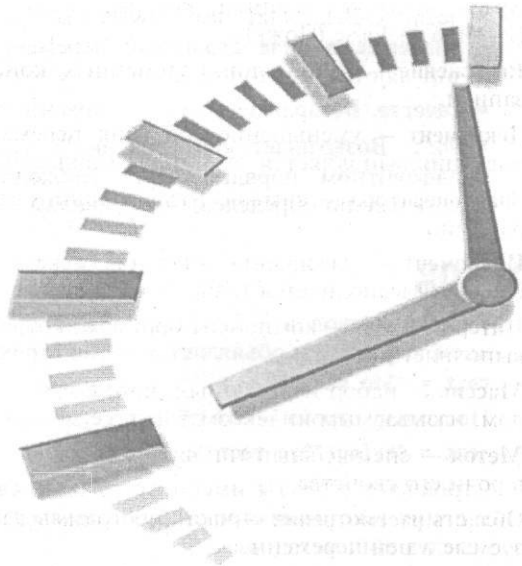
Условный оператор — оператор JavaScript, который разрешает выполнять операцию только при соблюдении указанного условия.

Функция — группа операторов JavaScript, объединенных одним общим именем.

Цикл — набор операторов JavaScript, которые выполняются необходимое количество раз или до выполнения указанного условия.

Элемент — отдельный член массива, имеющий отдельный индекс.

http://ITBookz.ru/



Приложение Г

Краткий справочник по JavaScript

Это приложение представляет собой краткий обзор языка JavaScript. В него включены встроенные объекты и объекты основной иерархической структуры, операторы JavaScript и встроенные функции.

Встроенные объекты

Следующие объекты встроены в JavaScript. Некоторые из них используются для создания пользовательских объектов. Остальные используются в явном виде.

Array

Для определения массива данных — нумерованного списка переменных — вы создаете новый объект массива. В отличие от обычных переменных, массив необходимо обязательно объявить. Для определения массива используется ключевое слово new. Например:

```
students = new Array(30);
```

Элементы массива индексируются, начиная с нуля. Ссылка на элемент массива задается по индексу, указываемому в скобках:

```
fifth = students[4];
```

Массив имеет всего одно свойство, length, определяющее текущее число элементов в нем. Но управляется объект массива тремя методами.

http://ITBookz.ru/

- join. Объединяет все элементы массива в один. Возвращает полученное значение в виде строковой переменной. Элементы в строковой переменной отделяются запятой или другим дополнительно указанным разделителем.
- reverse. Возвращает зарезервированную версию массива.
- sort. Возвращает сортированный массив. Обычно сортировка проводится в алфавитном порядке. При необходимости можно провести сортировку по специально определенным правилам.

String

Строка символов в JavaScript также представляется отдельным объектов. В следующем выражении объявляется новая строковая переменная.

text = "Это тест";

Поскольку строка — это объект, ее можно создать и с помощью ключевого слова new:

```
text = String("Это тест");
```

Строковые объекты имеют всего одно свойство, length, которое определяет текущую длину текстовой строки. Для управления строковыми данными используется набор следующих методов.

- substring. Возвращает часть строковой переменной.
- toUpperCase. Преобразует все символы строки в прописные.
- toLOwerCase. Преобразует все символы строки в строчные.
- indexOf. Ищет необходимый текст в строке.
- lastIndexOf. Ищет необходимый текст в строке, начиная с ее конца.
- link. Создает ссылку на основе текста строкового объекта.
- anchor. Создает анкер в пределах текущей страницы.

Существуют и другие методы, позволяющие изменить вид строки при отображении ее в документе HTML.

- string.big. Отображает большой текст с помощью дескриптора <big> в HTML 3.0.
- string.blink. Отображает мерцающий текст с помощью дескриптора <blink> в Netscape.
- string.bold. Отображает текст в полужирном начертании с помощью дескриптора <b>.
- string.fixed. Отображает текст строго определенного размера с помощью дескриптора <tt>.
- string.fontcolor. Отображает текст цветным шрифтом, подобно дескриптору <fontcolor> в Netscape.
- string.fontsize. Изменяет размер шрифта с помощью дескриптора <fontsize> в Netscape.
- string. italics. Отображает текст курсивом с помощью дескриптора <i>.
- string.small. Отображает текст маленькими символами с помощью дескриптора <small> в HTML 3.0.
- string.strike. Перечеркивает текст строки с помощью дескриптора <strike>.
- string.sub. Отображает текст в нижнем индексе, подобно дескриптору <sub> в HTML 3.0.
- string.sup. Отображает текст в верхнем индексе, подобно дескриптору <sup> в HTML 3.0.

http://ITBookz.ru/

Math

Объект Math — это не "настоящий" объект. С его помощью нельзя создавать собственные объекты. В качестве свойств объекта Math задается огромное количество всевозможных математических констант.

- Math.E. Это основа натурального логарифма (приблизительно 2,718).
- Math.LN2. Натуральный логарифм двух (приблизительно 0,693).
- Math.LN10. Натуральный логарифм десяти (приблизительно 2,302).
- Math.LOG2E. Логарифм е с с основой два (приблизительно 1,442).
- Math.LOG10E. Логарифм е с основой десять (приблизительно 0,434).
- Math.PI. Число я, отношение длины круга к его диаметру (приблизительно 3,14159).
- Math.SQRT1_2. Квадратный корень 0,5 (приблизительно 0,707).
- Math.SQRT2. Квадратный корень 2 (приблизительно 2,7178).

Методы объекта Math представляют собой математические функции. Ниже приведены методы этого объекта, упорядоченные по категориям

Тригонометрические функции

- Math.acos. Вычисляет арккосинус числа. Возвращает значение в радианах.
- Math.asin. Вычисляет арксинус числа. Возвращает значение в радианах.
- Math.atan. Вычисляет арктангенс числа. Возвращает значение в радианах.
- Math.cos. Вычисляет косинус числа.
- Math.sin. Вычисляет синус числа.
- Math.tan. Вычисляет тангенс числа.

Статистические и логарифмические функции

- Math.exp. Возвращает число е (основа натурального логарифма), возведенное в степень.
- Math. log. Возвращает натуральный логарифм числа.
- Math.max. Возвращает максимальное из двух чисел.
- Math.min. Возвращает минимальное из двух чисел.

Основные математические функции

- Math.abs. Вычисляет модуль числа.
- Math.ceil. Округляет число до ближайшего большего целого.
- Math.floor. Округляет число до ближайшего меньшего целого
- Math.pow. Возводит одно число в степень второго.
- Math.round. Округляет число до ближайшего целого.
- Math.sqrt. Вычисляет квадратный корень числа.

Функция случайного числа

- Math.random. Возвращает произвольное число в диапазоне 0–1.

http://ITBookz.ru/

Date

Объект Date — это встроенный в JavaScript объект, позволяющий управлять датами и временем. Объект даты создается в любой момент, когда необходимо сохранить время или дату. Для управления датами и временем в JavaScript предусмотрены следующие методы.

- setDate. Задает число месяца.
- setMonth. Задает месяц. Отсчет начинается с 0 и заканчивается 11.
- setYear. Задает год. setFullYear — определяет год в виде четырехзначного числа. Последний формат не вызывает "проблемы 2000".
- setTime. Задает время (и дату) в миллисекундах, отсчитанных от 1 января 1970 года.
- setHours, setMinutes и setSeconds. Задает время в часах, минутах и секундах.
- getDate. Определяет текущую дату (число месяца).
- getMonth. Определяет текущий месяц.
- getYear. Определяет текущий год.
- getTime. Определяет текущее время (и дату) в миллисекундах, отсчитанных от 1 января 1970 года.
- getHours, getMinutes и getSeconds. Определяет текущее время в часах, минутах и секундах.
- getTimeZoneOffset. Возвращает локальное время зоны в стандарте всеобщего скоординированного времени (по Гринвичу).
- toGMTString. Преобразует объект даты в текстовое значение времени в формате всеобщего скоординированного времени.
- toLocatString. Преобразует объект даты в строковую переменную местного времени.
- Date.parse. Преобразует дату, сохраненную в виде строковой переменной в объект даты (например "June 20, 2003" в число миллисекунд до этой даты от 1 января 1970 года).
- Date.UTC. Преобразует объект даты (в миллисекундах) в формат всеобщего скоординированного времени.

Основы модели DOM

Модель DOM включает в себя объекты, представляющие окно броузера, окно текущего документа и их элементы. Ниже (и в уроке 10) описаны основные объекты модели. Более сложные объекты приведены в уроках 17-19.

Window

Объект window представляет текущее окно броузера. Если на экране открыто несколько окон броузера или в одном окне задано несколько фреймов, то в сценарии определяет несколько объектов window. Вот каким образом они управляются.

- self. Этим ключевым словом представлено текущее окно, содержащее страницу со сценарием JavaScript.

http://ITBookz.ru/

- top. Это окно, открытое в настоящее время поверх остальных.
- parent. Это окно, содержащее несколько фреймов. Каждый фрейм также представляется отдельным объектом, дочерним по отношению к parent.
- Массив frames содержит объекты фреймов окна. К фрейму обращаются как к объекту массива, дочернему по отношению к объекту parent. Например `parent.frames[0]`. Или по именам фреймов (если они определены) `parent.docframe`.

Объект window имеет три дочерних объекта.

- location. Содержит URL документа, отображенного в окне.
- document. Содержит сам документ HTML, отображенный в окне.
- history. Содержит список узлов, посещенных до и после текущего узла, отображенного в окне.

Location

Объект location содержит сведения о URL страницы, открытой в текущем окне броузера. Он имеет большой набор свойств, которыми управляются отдельные части URL.

- `location.protocol`. Определяет протокол (или метод).
- `location.hostname`. Определяет имя узла.
- `location.port`. Определяет порт соединения.
- `location.host`. Комбинация двух предыдущих свойств.
- `location.pathname`. Каталог расположения документа на узле и имя файла.
- `location.hash`. Название анкера в документе, если такой определен.
- `location.target`. Атрибут TARGET ссылки, которая привела к открытию текущего документа.
- `location.query`. Определяет строку запроса.
- `location.href`. Определяет полный URL.

History

Объект history содержит данные об адресах, которые посещались до и после документа, открытого в текущем окне. Для него определены методы перехода к этим страницам.

- `history.back`. Переход к предыдущей странице.
- `history.foward`. Переход к следующей странице.
- `history.go`. Переход к определенной странице.

Document

Объект document представляет текущий, открытый в окне броузера документ HTML. Он имеет следующие дочерние объекты.

- `document.forms`. Массив объектов всех форм объекта. К форме можно обращаться не только как к элементу массива, но и по имени. Например `document.regform`. Элементы формы представляются дочерними объектами объекта формы.

http://ITBookz.ru/

- document.links. Массив всех ссылок, содержащихся в документе.
- document.archors. Массив всех анкеров документа.
- document.images. Массив всех рисунков документа.
- document.applets. Массив ссылок на встроенные аплеты Java в документе.

Navigator

Объект navigator содержит информацию о версии используемого броузера.

- appCodeName. Кодовое имя броузера, например Mazzila.
- appName. Полное имя броузера.
- appVersion. Номер версии броузера.
- userAgent. Заголовок агента пользователя, отправляемый на узел при запросе Web-страницы. Он содержит полное имя броузера, например Mozilla/4.5(Win95;I).
- plugIns. Массив сведений обо всех установленных в броузере встраиваемых моделях.
- mimeType. Массив элементов, представляющих все типы MIME.

Создание и настройка объектов

Ниже приведены краткие инструкции по созданию объектов и их настройке. Подобные инструкции вы также найдете в уроке 11 этой книги.

Создание объектов

Для создания объектов используется три ключевых слова.

- new. Создание нового объекта.
- this. Обращение к текущему объекту. Используется в функции конструктора объекта или в обработчике событий.
- with. Определяет объект по умолчанию в группе операторов. Все свойства, не относящиеся к другим объектам, приписываются объекту по умолчанию.

Чтобы создать объект, необходимо построить функцию его конструктора. В ней свойствам объекта определяются требуемые значения:

```
function Name(first, last) {
  this.first = first;
  this.last = last;
}
```

Новый объект создается и с помощью оператора new:

```
Fred new Name ("Fred", "Smith");
```

Настройка объектов

Новые свойства для уже созданного объекта определяются очень просто:

```
Fred.middle = "Clarence";
```

Определяемые таким образом свойства добавляются только к указанным экземплярам объектов, а не ко всем объектов этого типа. Чтобы добавить свойство к прото-

http://ITBookz.ru/

типу объекта (в его определении) используется оператор prototype. В этом случае любой будущий объект этого типа будет обладать указанным свойством:

```
Name.prototype.title = "Citizen";
```

Этот метод используется для добавления свойств в определениях встроенных объектов. Например, следующий оператор добавляет свойство num ко всем будущим строковым объектам и задает ему значение 10.

```
string.prototype.num = 10;
```

Операторы JavaScript

Ниже приведен перечень операторов JavaScript.

Комментарии

Комментарии используются для вставки в код программы замечаний. Они игнорируются интерпретатором. Комментарии задаются двумя способами:

```
//Это комментарий
/*это тоже комментарий*/
```

Только второй вариант приемлем при введении комментариев в несколько строк. Первый тип комментариев вводится в одну строку.

Прерывание

Оператор break используется для прерывания выполнения цикла for или while.

Продолжение

Оператор continue вызывает продолжение выполнения цикла for и while.

Цикл for

Оператор for определяет цикл, начинающийся с определенного числа, которое впоследствии изменяется, задавая последующие итерации. В приведенном ниже цикле переменная i принимает значения 1—9:

```
for (i=1; i<10; i++;) {операторы}
```

Цикл for...in

Этот цикл предназначен для просмотра свойств объекта или элементов массива. В приведенном ниже примере перебираются свойства объекта Scores. Свойства объекта задаются переменной x:

```
for (x in Scores) {операторы}
```


http://ITBookz.ru/

Функция

Определенная в JavaScript функция используется в любом месте документа HTML. Функции иногда возвращают значения (с помощью оператора return). В этом примере в функции вычисляется сумма двух значений:

```
function (n1, n2) {
  result = nl + n2;
  return result;
}
```

Условие if...else

Это условное выражение. Если условие выполняется (вводится после ключевого слова if), то в силу вступают операторы, введенные далее; если нет, то выполняются операторы, введенные после ключевого слова else. В приведенном ниже примере переменная a сравнивается со значением 10:

```
if (a>10) {
  document.write("Больше десяти");
}
else {
  document.write("Десять или меньше");
}
```

Этот оператор можно записать и в краткой форме. После оператора ? вводятся операторы, выполняемые при выполнении условия, а после оператора : заданы операторы, выполняющиеся, если условие неправдиво:

```
document.write((a>10)? "Больше десяти" : "Десять или меньше");
```

Условные операторы детально рассмотрены в главе 7.

Оператор return

Этим оператором заканчивается определение функции, которая возвращает заданное значение. Оператор return необходимо вводить только в том случае, если функция возвращает какое-то значение.

Оператор var

Этот оператор используется при объявлении переменных. Если переменные объявляются в функции с помощью оператора var, то они гарантированно локальные. Если вне функции, то они глобальные:

```
var students = 30;
```

Поскольку JavaScript без претензий относится к формату объявления переменных, ключевое слово var можно упускать:

```
. students = 30;
```

Но оператором var все же стоит пользоваться. Он устраняет конфликты между глобальными и локальными переменными. Примите к сведению, что массивы не рассматриваются в JavaScript как наборы переменных, а только как объект. Детально о массивах рассказано в уроке 6.

http://ITBookz.ru/

Цикл while

Оператор while задает цикл, который выполняется, пока справедливо указанное условие. В приведенном ниже примере цикл справедлив до тех пор, пока строковая переменная имеет значение до:

```
while (document.form1.text1.value !="go") {операторы}
```

Встроенные функции JavaScript

Приведенные в этом примере функции встроены в JavaScript. Они не относятся к методам ни одного объекта.

Функция eval

Эта функция воспринимает значение строковой переменной как оператор, который необходимо выполнить. В следующем примере в качестве аргумента функции задан оператор JavaScript:

```
a = eval("add(x,y);");
```

Функция eval обычно используется для выполнения операторов, заданных пользователем в форме введения данных.

Функция parseInt

Эта функция находит целое число в начале строки и возвращает его. Если число не найдено, то возвращается значение NaN.

Функция parseFloat

Эта функция находит любое число в начале строки и возвращает его. Если число не найдено, то возвращается значение NaN.

http://ITBookz.ru/

Предметный указатель

A

ActiveX, 11; 12
Adobe Acrobat, 28, 212
ASCII, 12

B

Beatnik, 212

C

CGI, 12; 26
CSS, 180
Свойства, 87

D

DHTML, 266
DOM, 96; 185; 192; 264

E

ECMA, 10

F

Forth, 76

H

HomeSite, 12
HTML, 7

I

Internet Explorer, 10; 98; 195; 226

J

Java, 11
JavaScript Console, 18
JScript, 10

L

LiveAudio, 212
LiveConnect, 28; 213
LiveScript, 8

M

Mac, 11; 12
Microsoft FrontPage 2000, 12
MIME, 101; 212

N

Netscape Navigator, 10; 98; 195
Notepad, 12; 16

P

Perl, 12

Q

QuickTime, 25; 28; 212

R

RealAudio, 28
RealPlayer, 212

S

SGML, 265
ShockWave, 25; 28; 212

T

TextPad, 12

U

Unix, 11; 12

V

VBScript, 12
Visual Basic, 12

W

W3C, 98; 201; 266
Web-технологии, 264
WWW, 6

http://ITBookz.ru/

X

XML, 265
XSL, 265

А

Автоматизация введения данных, 26
Авторские права, 30
Анимация, 25; 161; 196
Анкер, 101
Аплет, 11
Аргументы, 41
Атрибуты, 87; 142

Б

Бегущая строка, 29; 61
Бесконечный цикл, 78
Броузеры, 10
  независимых производителей, 170
Будущее JavaScript, 264
Булевы операторы, 68

В

Версии JavaScript, 10
Внедряемый модуль, 28
Внешняя таблица стилей, 185
Возвращение
  значений, 90
  значения, 43
  символа, 58
Возможности JavaScript, 8; 22
Воспроизведение музыки, 216
Временные
  задержки, 130
  зоны, 91
Вставка
  звука, 216
  сценария на Web-страницу, 15
Встроенные объекты, 35; 110
Выбор имени переменной, 45
Вывод на экран, 14
Вызов функции, 42
Выравнивание текста, 181
Выражение
  инкремента, 76
  условия, 67
Выявление ошибок, 18

Г

Генератор случайных чисел, 89
Глобальные переменные, 45; 224
Границы и поля, 183

Д

Дата изменения Web-страницы, 99
Декремент, 47
Диалоговое окно, 133
Динамические
  рисунки, 25
  стили, 186
  рисунок, 158
Длина строки, 57
Добавление
  метода, 109
  описания ссылки, 121
  сценария на Web-страницу, 8
Документ HTML, 8
Дочерний объект, 193

З

Задачи JavaScript, 8
Закрытие окон, 130
Запрос, 133

И

Изменение фона, 182
Изменяющиеся рисунки, 25; 159
Инкремент, 47
Инструменты создания сценария, 12
Интерпретация, 7
Исправление ошибок, 230
История
  JavaScript, 7
  броузера, 101

К

Каскадные таблицы стилей, 180
Клиентный разделенный рисунок, 156
Кнопка, 145 .
  перемещения по страницам, 103
Комментарий, 19; 37
Компиляция, 7
Комплексное условие, 70
Консоль JavaScript, 18; 225
Конструктор объектов, 108
Контейнер, 192
Копирование сценария, 29

Л

Логические операторы, 68
Локальные переменные, 45; 224

http://ITBookz.ru/

М

Массив, 59; 136
Метод, 35; 87; 202
управления текстом, 145

Н

Навигационный фрейм, 136
Надстройка, 28
Названия страниц, 237
Настраиваемый документ HTML, 50
Настройка встроенных объектов, 110
Начало сценария, 13
Начальное выражение, 76

О

Область действия переменной, 45
Обновление Web-страницы, 100; 132
Обработчик событий, 35; 115; 116; 241
Обший шлюзовой интерфейс, 12
Объект, 34; 87; 108
Объектная модель документа, 97; 185
Округление, 88
Описания ссылок, *239*
Определение
  слоя, 193
  стилей, 180
  формы, 142
  функции, 41
Открытие
  Web-страницы, 16
  окон, 130
Отладка, 225; 229
Отладчик JavaScript, 227
Отображение
  данных на форме, 148
  объектов, 202
Отправка данных формы, 143; 149
Очистка
  строки состояния, 121
  формы, 143
Ошибки
  задания объектов, 224
  синтаксиса, 223

П

Параметры
  функции, 41
  циклов, 76
Переключатель, 146
Переменная, 14; 44; 45
Перемещения по страницам, 103
Перечень утилит, 214
Поддержка JavaScript в броузерах, 10; 172, 195
Подстроковые переменные, 57
Подтверждение, 133
Поиск
  символов, 58
  текста, 59
Пользовательские объекты, 35
Поток, 100
Правила, 181
Предварительная загрузка рисунка, 159
Предупреждение, 24
Преобразование данных, 49
Прерывание, 228
  цикла, 79
Присваивание значений, 47; 55; 90
Проверка
  данных, 26
  переменных, 151, 229
  правильности данных, 71
  утилит, 213
Продолжение выполнения цикла, 79
Прототип, 110

Р

Разделение строковой переменной, 60
Разделенный рисунок, 156
Разделитель, 14
Размещение сценария на Web-странице, 9
Разработка стандартов, 266
Разрыв, 228
Раскрывающийся список, 147; 237
Расположение слоя, 195
расширение документа HTML, 16
Регистр символов, 57; 116
Родительский объект, 193

С

Сведения о броузере, 27; 99; 102; 169
Свойства, 34; 97
  границ и полей, 183
  переключателя, 147
  раскрывающегося списка, 147
  рисунка, 158
  слоя, 194
  события, 117
  текстового поля, 144
  флажка, 146
  фона, 182

http://ITBookz.ru/

Свойства
формы, 143
шрифтов, 182
элемента, 201
Связи элемента, 201
Серверный разделенный рисунок, 156
Скрытие объектов, 202
Слой, 25
Собственные функции, 34
Событие, 35; 116
формы, 143
связанные с клавишами, 120
связанные с мышью, 118
Совместимость сценариев, 266
Создание
диалогового окна, 134
массива, 59
нового окна, 129
обработчика событий, 116
объекта, 87
слоя, 193
строковой переменной, 55
таблицы стилей, 183
экземпляра объекта, 109
Сообщение, 133
об ошибках, 226
Сортировка элементов массива, 61
Сохранение
данных, 14
значений, 49; 111
Сравнение значений, 68
Среднее арифметическое, 93
Ссылка, 101
Строка состояния, 23; 226
Строковая
переменная, 55; 87
массив, 60
Строковый объект, 55
Структура
DOM, 192
документа, 180
объектов DOM, 97

Т

Таблица стилей, 180
Текстовое поле, 144
Текстовые панели, 144
Текстовый процессор, 12
Тестирование сценария, 16
Техника программирования, 223
Тип данных, 48
MIME, 212
кнопок, 145
Точность, 93

У

Улучшение Web-страницы, 23
Управление
Web-документами, 98
датами, 89
динамическими рисунками, 158
мультимедиа, 28
объектами броузера, 96
объектами утилит, 213
окнами броузера, 24; 98; 129
подстроковой переменной, 58
событиями, 116
стилями, 180
строкой состояния, 23
фреймами, 135
циклами, 78
шрифтами, 182
элементами массива, 60; 80
Упрощение сценариев, 108
Уровень структуры, 193
Условие цикла, 76
Условный оператор, 67
Устранение ошибок, 18

Ф

Флажок, 146
Форма, 25; 142
Формат представления даты, 91
Функции, 34; 41

Ц

Цикл, 75

Э

Элемент, 193; 201
управления, 11
формы, 143

Я

Язык подготовки сценариев, 7

http://ITBookz.ru/

Научно-популярное издание

Майкл Монкур

Освой самостоятельно JavaScript за 24 часа, 2-е издание

Литературный редактор *Л.Н. Важенина*
Верстка *Е.Г. Рублев*
Художественный редактор *В.Г. Павлютин*
Технический редактор *Г.Н. Горобец*
Корректоры *Л.А. Гордиенко, О.В. Мишутина, Л.В. Чернокозинская*

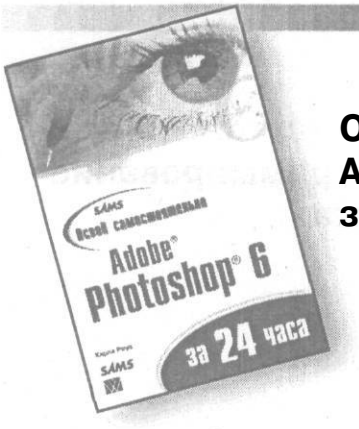
Издательский дом "Вильяме".
101509, Москва, ул. Лесная, д. 43, стр. 1.
Изд. лиц. ЛР № 090230 от 23.06.99
Госкомитета РФ по печати.

Подписано в печать 29.03.2001. Формат 70x100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 18,7. Уч.-изд. л. 15,7.
Тираж 5000 экз. Заказ № 371.

Отпечатано с диапозитивов в ФГУП «Печатный двор»
Министерства РФ по делам печати,
телерадиовещания и средств массовых
коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.

http://ITBookz.ru/

Карла Роуз



Освой самостоятельно Adobe Photoshop 6 за 24 часа

В книге Карлы Роуз рассмотрены возможности последней версии популярного редактора изображений — Photoshop 6. Работа со слоями, каналами, инструментами; управление контурами, цветами и текстом — все нашло отражение в книге. Рассчитанная на начинающих пользователей, которые только делают первые шаги в освоении этого графического приложения, она станет полезной любому художнику и дизайнеру. Детальное описание фильтров и методов коррекции цветных и черно-белых рисунков делает книгу настольным пособием, используемым при редактировании сканированных изображений и фотографий.

Книга содержит красочные иллюстрации, наглядно демонстрирующие все применение средств и инструментов Photoshop 6. Несмотря на простоту, она содержит сведения о большинстве используемых средств. Советы, заметки и подробные инструкции обращают внимание читателя на наиболее важный материал. Упражнения и тесты, приведенные в конце каждой главы, помогают закрепить освоенный материал и расширить знания читателя.

Вся книга разбита на 24 занятия каждый примерно по часу так, чтобы максимально просто и эффективно ознакомить читателя со сложным и многосторонним графическим приложением — Photoshop 6. Каждое занятие относится к отдельной теме и тесно связано с предыдущими. Только последовательное изучение материала книги позволяет в полной мере узнать все возможности программы.

http://ITBookz.ru/

Рейф Колберн



Освой самостоятельно CGI-программирование за 24 часа

Книга знакомит читателя с технологией CGI. Вы узнаете, что такое CGI-программы, как с их помощью создаются Web-приложения и интерактивные Web-страницы, как обрабатываются переданные пользователем данные, как Web-приложения работают с базами данных, как обеспечить автоматическую отправку почтового сообщения с вашего Web-узла и многое другое. Изложение материала подкреплено многочисленными примерами — от простейшей обработки данных формы до создания сетевого магазина.

Особое внимание уделено вопросам безопасности CGI-программы и Web-узла в целом.

Книга предназначена для широкого круга читателей.

http://ITBookz.ru/

АЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

ДЛЯ ЗАМЕТОК

http://ITBookz.ru/

SAMS

Освой самостоятельно

JavaScript™

за 24 часа

Узнайте, как:

- создавать динамичные Web-страницы
- добавлять анимационные изображения и изменяющиеся рисунки
- писать сценарии, выполняющиеся в Internet Explorer и Netscape Navigator
- управлять формами на Web-страницах
- заказывать товары в Internet
- создавать игры
- отлаживать готовые сценарии
- создавать динамические Web-страницы с помощью слоев

Категория: создание Web-страниц
Уровень: для начинающих пользователей

ВИЛЬЯМС

SAMS

www.williamspublishing.com
www.samspublishing.com

Начните прямо сейчас!

Двадцать четыре практических занятия

Всего за 24 урока продолжительностью в один час вы сможете научиться программировать на JavaScript. С помощью доступных пошаговых инструкций и подробных примеров вы быстро изучите JavaScript



Советы указывают кратчайший путь к выполнению задачи



Замечания просто и доходчиво объясняют понятия и процедуры



Предостережения помогают избежать наиболее частых недоразумений

Майкл Монкур — владелец консалтинговой фирмы *Starling Technologies*, занимающейся предоставлением услуг в области сетевых технологий и Internet. Он не только блестящий Web-дизайнер, но и автор книги *Laura Lemay's Web Workshop:JavaScript*, а также многих других популярных книг по сетям Novell и подготовке сертифицированных специалистов *Microsoft*

ISBN 5-8459-0159-6



http://ITBookz.ru/