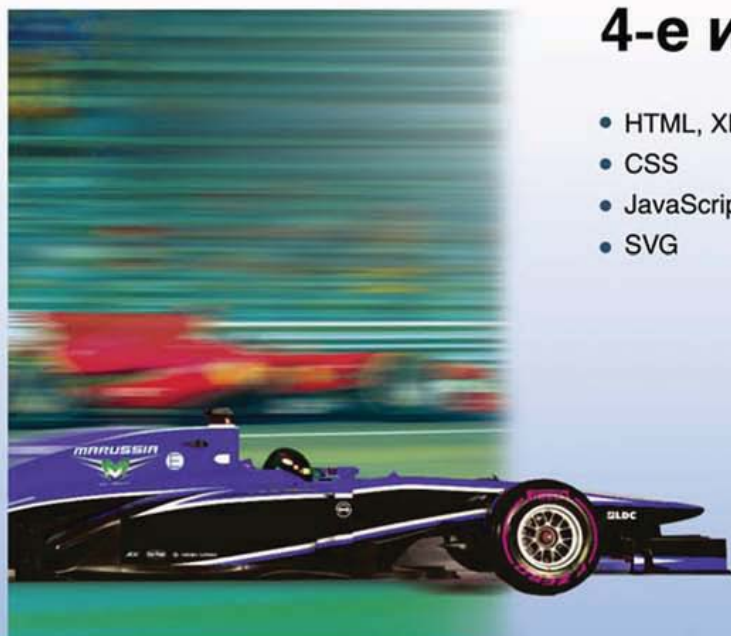


HTML, СКРИПТЫ И СТИЛИ

4-е издание

- HTML, XHTML
- CSS
- JavaScript, PHP
- SVG



Наиболее
полное
руководство

В ПОДЛИННИКЕ®

УДК 004.43+004.738.5
ББК 32.973.26-018.1
Д83

Дунаев В. В.

Д83 HTML, скрипты и стили. — 4-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2015. — 816 с.: ил. — (В подлиннике)

ISBN 978-5-9775-3317-1

Рассмотрены средства создания Web-сайтов — языки разметки гипертекста (XHTML, HTML 4 и HTML 5), каскадные таблицы стилей (CSS 2 и CSS 3), а также языки сценариев JavaScript и PHP. Изложены краткие теоретические сведения и приведены многочисленные примеры решения типичных задач разработки сайтов. Четвертое издание книги является результатом существенной переработки третьего издания с учетом современного состояния HTML, CSS и ведущих браузеров, удалены некоторые разделы, ставшие неактуальными, а также исправлены замеченные ошибки.

Для Web-дизайнеров

УДК 004.43+004.738.5
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Капалыгина</i>
Редактор	<i>Леонид Кочин</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Марины Дамбиевой</i>

Подписано в печать 31.07.14.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 65,79.

Тираж 1300 экз. Заказ №

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3317-1

© Дунаев В. В., 2015
© Оформление, издательство "БХВ-Петербург", 2015

Оглавление

Предисловие к четвертому изданию	15
Введение	17
ЧАСТЬ I. HTML И СТИЛИ	19
Глава 1. Что такое HTML и таблицы стилей CSS.....	21
1.1. Языки разметки документа	21
1.2. Что такое таблицы стилей.....	31
Глава 2. Структура (X)HTML-документа	35
2.1. Определение типа документа: дескриптор <code><!DOCTYPE...></code>	35
2.2. Структура собственно (X)HTML-кода.....	38
2.3. Раздел заголовка документа <code><head></code>	41
2.3.1. Тег <code><meta></code>	41
Группа <i>HTTP-EQUIV</i> (HTTP-эквиваленты)	42
Группа <i>NAME</i> (имя)	43
2.3.2. Тег <code><base></code>	45
2.3.3. Другие теги внутри <code><head></code>	45
2.4. Раздел тела документа <code><body></code>	47
2.5. Основные атрибуты тегов	48
2.6. Отображение элементов в нормальном потоке.....	50
Глава 3. Основы CSS	54
3.1. Присоединение таблиц стилей к (X)HTML-документу.....	54
3.2. Правила форматирования	55
3.2.1. Селекторы	56
3.2.2. Контекстные селекторы	58
3.2.3. Псевдоселекторы и псевдоэлементы	59
3.3. Приоритеты определений параметров стилей.....	61
3.4. Размерность и цвета	63
3.5. Блоки: поля, отступы, границы и размеры	66
3.6. Наследование параметров.....	72

Глава 4. Позиционирование с помощью CSS	74
4.1. <i>position:static</i>	75
4.2. <i>position:relative</i>	75
4.3. <i>position:absolute</i>	76
4.4. <i>position:fixed</i>	80
4.5. Отсчет координат.....	82
4.6. Слои	82
4.7. Обтекание	84
4.8. Видимость	87
4.8.1. <i>overflow</i>	87
4.8.2. <i>clip</i>	87
4.8.3. <i>visibility</i>	87
4.8.4. <i>display</i>	88
4.9. Размеры	90
4.10. Практические примеры	91
4.10.1. Центрирование элемента	92
4.10.2. Управление положением элемента с помощью мыши.....	94
4.10.3. Раскрывающаяся панель	95
Глава 5. Фон элементов и границ	97
5.1. <i>background</i>	97
5.2. <i>opacity</i>	101
5.3. <i>border-image</i>	104
5.4. <i>border-radius</i>	107
5.5. <i>box-shadow</i>	108
5.6. Градиенты.....	108
5.6.1. Линейный градиент	108
5.6.2. Радиальный градиент	109
Глава 6. Ссылки	112
6.1. Текстовые ссылки.....	113
6.1.1. Простое меню ссылок	113
6.1.2. Двухуровневое меню ссылок.....	117
6.2. Графические и комбинированные ссылки.....	120
6.3. Графические карты ссылок.....	120
6.3.1. Клиентский вариант графической карты ссылок.....	121
6.3.2. Серверный вариант графической карты ссылок	123
6.4. Внутренние ссылки.....	124
6.5. URL-адреса.....	126
6.5.1. Структура URL	126
6.5.2. Абсолютные и относительные пути.....	129
6.5.3. Кодирование URL.....	129
6.5.4. Псевдо-URL JavaScript.....	132
Глава 7. Тексты	133
7.1. Шрифты.....	133
7.2. Основные теги разметки текстов.....	136
7.3. Специальные символы	137

7.4. Форматирование текста	138
7.4.1. Красная строка	138
7.4.2. Выравнивание	138
7.4.3. Межстрочное расстояние	139
7.4.4. Межсловное расстояние	140
7.4.5. Межбуквенное расстояние	140
7.4.6. Декорация	141
7.4.7. Индексы	142
7.4.8. Выделение первой буквы строки и первой строки в блоке текста	143
7.4.9. Объемный текст	144
7.4.10. Преобразование регистра	146
7.4.11. Мультиколоночная верстка	146
7.5. Предварительно отформатированный текст	146
7.6. Генерируемое содержимое	147
Глава 8. Списки	152
8.1. Маркированный список	152
8.2. Нумерованный список	154
8.3. Автоматическая нумерация элементов списка	155
8.4. Иерархический раскрывающийся список	158
8.5. Меню на основе списка	164
8.6. Выравнивание элементов списка	167
8.7. Список определений	168
Глава 9. Таблицы	170
9.1. Табличные теги	170
9.2. Рамки таблицы	172
9.3. Размеры таблицы	176
9.4. Выравнивание содержимого ячеек таблицы	180
9.5. Задание параметров столбцов	183
9.6. Сложные таблицы	185
9.6.1. Расширение ячеек	185
9.6.2. Прокручиваемая таблица	188
9.7. Декорирование таблицы	191
Глава 10. Элементы пользовательского интерфейса и формы	193
10.1. Поля ввода, кнопки и переключатели: тег <code><input></code>	194
10.2. Кнопка: тег <code><button></code>	197
10.3. Раскрывающийся список: тег <code><select></code>	200
10.4. Текстовая область: тег <code><textarea></code>	202
10.5. Декорации	204
10.6. Форма: тег <code><form></code>	206
Глава 11. Вставка внешнего содержимого в (X)HTML-документ	209
11.1. Графические изображения	209
11.1.1. Основные форматы графики	209
Растровая графика	209
Векторная графика	212

11.1.2. Вставка графики в (X)HTML-документ.....	213
Применение тега <code></code>	213
Применение тегов <code><iframe></code> и <code><object></code> для вставки изображений	216
Пример простой фотогалереи	218
11.2. Звук и видео	219
11.2.1. Основные форматы звуковых и видеофайлов	219
11.2.2. Вставка звука и видео в (X)HTML-документ	221
Теги <code><audio></code> и <code><video></code>	221
Теги <code><object></code> и <code><embed></code>	223
11.2.3. Вставка FLV-видео в (X)HTML-документ	225
11.2.4. Вставка Flash-фильмов	227
11.3. Вставка (X)HTML-документов.....	231
11.4. Вставка элементов управления ActiveX	233
11.4.1. Что такое ActiveX	233
11.4.2. Примеры элементов ActiveX	234
Flash-проигрыватель	234
Adobe SVGViewer	235
Календарь.....	235
Обслуживание табличных данных в текстовых файлах.....	235
11.4.3. О безопасности ActiveX	240
11.5. Вставка апплетов Java	242
11.5.1. Что такое апплет	242
11.5.2. Вставка апплета посредством тега <code><applet></code>	243
11.5.3. Вставка апплета посредством тега <code><object></code>	244
Глава 12. Трансформация и анимация с помощью CSS	246
12.1. Трансформация	246
12.2. Анимация.....	250
Глава 13. Применение SVG	257
13.1. Что такое SVG.....	257
13.2. Создание простых фигур.....	262
13.2.1. Прямоугольник	263
13.2.2. Круг.....	264
13.2.3. Эллипс	264
13.2.4. Многоугольник	264
13.2.5. Линии.....	265
13.3. Создание сложных фигур (тег <code><path></code>)	268
13.4. Вставка растровых изображений.....	273
13.5. Применение CSS.....	273
13.6. Группировка элементов.....	275
13.7. Третье измерение, определения и клонирование элементов	276
13.8. Градиентная заливка.....	278
13.8.1. Линейный градиент	278
13.8.2. Радиальный градиент	282
13.9. Маски.....	285
13.9.1. Тег <code><mask></code>	285
13.9.2. Тег <code><clipPath></code>	288

13.10. Тексты.....	289
13.11. Трансформации.....	297
13.11.1. Перенос.....	298
13.11.2. Поворот	299
13.11.3. Наклон	301
13.11.4. Масштабирование.....	304
13.11.5. Отражение	305
13.11.6. Трансформация посредством матрицы.....	307
13.12. Анимация.....	309
13.12.1. Тег <code><animate></code>	311
13.12.2. Тег <code><animateTransform></code>	316
13.12.3. Тег <code><animateMotion></code>	318
13.12.4. Тег <code><animateColor></code>	322
13.12.5. Тег <code><set></code>	322
13.13. Интерактивность.....	323
13.13.1. Гиперссылки	323
13.13.2. Обработка событий.....	324
Применение тегов и атрибутов SVG	325
Применение JavaScript.....	329
13.14. Вставка в SVG-документ XHTML-кода	336
Глава 14. Компоновка страницы	341
14.1. Базовые схемы компоновки страницы.....	341
14.2. Жесткая схема.....	347
14.3. "Резиновая" схема.....	348
14.4. Центрирование страницы.....	349
14.5. Декорация схемы	349
14.6. Вставка плавающего фрейма (<code><iframe></code>).....	352
ЧАСТЬ II. СКРИПТЫ НА JAVASCRIPT	355
Глава 15. Что такое JavaScript	357
15.1. Немного истории	357
15.2. Общая характеристика языка.....	359
15.3. Вставка сценариев в (X)HTML-документ	360
Глава 16. Основы JavaScript.....	366
16.1. Ввод и вывод данных	366
16.1.1. Метод <code>alert()</code>	367
16.1.2. Метод <code>confirm()</code>	367
16.1.3. Метод <code>prompt()</code>	368
16.1.4. Метод <code>document.write()</code>	369
16.2. Типы данных.....	370
16.2.1. Примитивные типы данных.....	371
16.2.2. Составные типы данных	372
16.2.3. Автоматическое преобразование типов данных	374
Преобразование строк (<code>String</code>).....	376
Преобразование чисел (<code>Number</code>).....	376
Преобразование логических значений (<code>Boolean</code>)	376

Преобразование пустого значения (<i>null</i>).....	376
Преобразование неопределенного значения (<i>undefined</i>)	377
16.2.4. Принудительное преобразование типов данных.....	377
16.3. Переменные и оператор присваивания.....	380
16.3.1. Имена переменных	380
16.3.2. Создание переменных	381
16.3.3. Операторы присваивания.....	388
16.3.4. Проверка типа переменной.....	390
16.4. Операторы	391
16.4.1. Комментарии.....	391
16.4.2. Арифметические операторы	392
16.4.3. Дополнительные операторы присваивания.....	394
16.4.4. Операторы сравнения.....	395
16.4.5. Логические операторы	397
16.4.6. Операторы условия.....	399
Оператор <i>if</i>	399
Оператор условия <i>?:</i>	401
Оператор <i>switch</i>	401
16.4.7. Операторы цикла	403
Оператор <i>for</i>	403
Оператор <i>while</i>	406
Оператор <i>do-while</i>	407
16.4.8. Об условиях в операторах условия и цикла	408
16.4.9. Побитовые операторы	408
16.4.10. Другие операторы.....	409
16.4.11. Приоритет операторов	409
16.5. Функции.....	411
16.5.1. Встроенные функции.....	412
16.5.2. Пользовательские функции.....	415
16.5.3. Объект <i>Function</i>	417
16.6. Строки.....	422
16.6.1. Кавычки и специальные символы	422
16.6.2. Объект <i>String</i>	424
16.6.3. Функции вставки и замены подстрок.....	430
16.6.4. Функции удаления ведущих и заключительных пробелов.....	431
16.7. Массивы	433
16.7.1. Создание массива	433
16.7.2. Многомерные массивы	435
16.7.3. Копирование массива.....	437
16.7.4. Объект <i>Array</i>	438
16.7.5. Функции обработки числовых массивов	443
16.8. Числа.....	444
16.8.1. Числа целые и с плавающей точкой.....	444
16.8.2. Объект <i>Number</i>	447
16.8.3. Объект <i>Math</i>	449
16.9. Дата и время.....	451
16.9.1. Создание объекта <i>Date</i>	451
16.9.2. Методы объекта <i>Date</i>	452

16.10. Объекты	458
16.10.1. Создание объекта	458
16.10.2. Свойства и методы объекта <i>Object</i>	463
16.10.3. Объектные операторы	464
16.10.4. JSON	467
16.11. Операторы обработки исключительных ситуаций	469
Глава 17. Объектная модель браузера и документа	473
17.1. Общие сведения	473
17.2. Доступ к объектам	476
17.3. Доступ к свойствам элементов документа	481
17.3.1. Доступ к атрибутам	481
17.3.2. Доступ к свойствам CSS	482
Параметры CSS, определенные в атрибуте <i>style</i>	483
Параметры CSS, определенные в теге <code><style></code> или во внешнем файле	483
17.3.3. Доступ к содержимому элемента	487
17.4. Обработка событий	490
17.4.1. Привязка обработчиков событий	491
Атрибуты-события	491
Регистрация обработчика события	492
Средства DOM	495
17.4.2. Область видимости обработчиков событий	496
17.4.3. Изменение поведения элементов по умолчанию	497
17.4.4. Программный вызов обработчика события	498
17.4.5. Прохождение событий	501
17.4.6. Информация о событии: объект <i>Event</i>	504
Доступ к <i>Event</i>	504
Кто является целевым объектом?	506
Какое событие произошло?	507
Основные свойства объекта <i>Event</i>	508
17.4.7. Основные события	510
17.5. Основные объекты браузера и документа	513
17.5.1. Объект <i>window</i>	513
Свойства объекта <i>window</i>	513
Методы объекта <i>window</i>	514
17.5.2. Объект <i>screen</i>	516
17.5.3. Объект <i>location</i>	516
Свойства объекта <i>location</i>	516
Методы объекта <i>location</i>	517
17.5.4. Объект <i>history</i>	517
Свойства объекта <i>history</i>	517
Методы объекта <i>history</i>	517
17.5.5. Объект <i>navigator</i>	518
Свойства объекта <i>navigator</i>	518
Коллекции объекта <i>navigator</i>	519
Методы объекта <i>navigator</i>	519
17.5.6. Объект <i>document</i>	520
Свойства объекта <i>document</i>	520

Коллекции объекта <i>document</i>	521
Методы объекта <i>document</i>	521
Глава 18. Работа с основными объектами.....	522
18.1. Управление окнами и фреймами.....	522
18.1.1. Создание окон.....	522
18.1.2. Взаимодействие окон.....	524
18.1.3. Работа с фреймами.....	526
Взаимодействие фреймов.....	526
Предотвращение загрузки в чужой фрейм.....	531
Проверка загрузки всех фреймов.....	531
18.1.4. Виджет.....	532
18.2. Работа с таблицами.....	537
18.3. Работа с табличными данными в текстовых файлах.....	540
18.4. Работа с формами.....	543
18.4.1. Проверка данных перед отправкой.....	544
18.4.2. Баннер как форма.....	546
18.4.3. Переходы между полями по клавише <Enter>.....	547
18.5. Работа с локальным хранилищем данных.....	548
18.5.1. Cookie.....	549
18.5.2. Объект <i>localStorage</i>	554
18.6. Работа с графическими изображениями.....	555
18.6.1. Объект элемента	555
18.6.2. Объект <i>Image</i>	556
18.6.3. Управление свойствами изображения.....	557
18.6.4. Предварительная загрузка изображений.....	559
18.6.5. Апокрифические применения объекта <i>Image</i>	561
Передача данных на сервер.....	562
Парольная защита страницы на стороне клиента.....	564
18.7. Взаимодействие с сервером: объект <i>XMLHttpRequest</i> и AJAX.....	566
18.7.1. Объект <i>XMLHttpRequest</i>	567
Свойства объекта <i>XMLHttpRequest</i>	567
Передача данных.....	568
18.7.2. AJAX.....	576
18.8. Управление во времени.....	577
Глава 19. Примеры клиентских сценариев.....	581
19.1. Подсветка кнопки.....	581
19.2. Меню.....	584
19.2.1. Моментально раскрывающееся вертикальное меню.....	584
19.2.2. Плавно раскрывающееся меню.....	585
19.3. Раскрывающийся комбинированный список.....	588
19.4. Иерархический раскрывающийся список.....	589
19.5. Эффект пишущей машинки.....	591
19.6. Отображение кода на странице.....	592
19.7. Перемещение элементов мышью.....	595
19.8. Движение по траектории.....	599
19.8.1. Движение по произвольной кривой.....	599
19.8.2. Движение по эллипсу.....	601

19.9. Рисование линий посредством <code><div></code>	603
19.9.1. Прямая линия	603
19.9.2. Произвольная линия	606
19.9.3. Графики зависимостей	610
19.9.4. Перерисовка линий	612
19.10. Рисование посредством <code><canvas></code>	613
19.10.1. Как вставить <code><canvas></code> в (X)HTML-документ	613
19.10.2. Фигуры и линии	616
Прямоугольник	616
Путь	617
Линии	619
Панель с закругленными углами	621
19.10.3. Градиенты	622
19.10.4. Трансформации	623
19.10.5. Импорт растровых графических изображений	625
19.10.6. Анимация	628
19.10.7. Композиция графики	631
19.10.8. Текст	633
19.11. Дата и время	635
19.11.1. Отображение даты и времени в виде текста	635
19.11.2. Часы	636

ЧАСТЬ III. СКРИПТЫ НА PHP 639

Глава 20. Что такое серверные сценарии и PHP 641

20.1. Общая характеристика языка PHP	642
20.2. Как установить модуль PHP	642
20.3. Настройка Web-сервера	645
20.4. Проверка работоспособности Web-сервера с PHP	645
20.5. Проба пера	646
20.6. Включаемые файлы	647
20.7. Сообщения об ошибках	647
20.8. Принудительный выход из сценария	648
20.9. Справочная информация по PHP	648

Глава 21. Основы PHP 650

21.1. Вывод данных	650
21.2. Типы данных	651
21.3. Переменные и оператор присваивания	654
21.3.1. Имена переменных	654
21.3.2. Создание переменных	654
21.3.3. Отображение значений переменных	656
21.3.4. Переменные переменные	658
21.3.5. Область действия переменных	659
21.3.6. Проверка существования переменных и их типов	660
21.4. Константы	661
21.5. Операторы	663
21.5.1. Комментарии	663
21.5.2. Арифметические операторы	663
21.5.3. Строковый оператор	665

21.5.4. Дополнительные операторы присваивания.....	665
21.5.5. Операторы сравнения.....	665
21.5.6. Логические операторы.....	667
21.5.7. Побитовые операторы.....	668
21.5.8. Операторы условного перехода.....	669
Оператор <i>if</i>	669
Оператор <i>switch</i>	669
Оператор условия <i>?:</i>	670
21.5.9. Операторы цикла.....	670
21.6. Строки.....	670
21.6.1. Двойные и одинарные кавычки.....	670
21.6.2. Склеивка строк.....	673
21.6.3. Преобразование строк.....	673
21.6.4. Форматирование строк.....	678
21.7. Числа.....	681
21.7.1. Математические функции.....	681
21.7.2. Математические константы.....	682
21.7.3. Представление чисел в различных системах счисления.....	683
21.7.4. Форматирование чисел.....	685
21.8. Дата и время.....	686
21.9. Массивы.....	689
21.9.1. Создание массива.....	689
21.9.2. Многомерные массивы.....	692
21.9.3. Отображение массивов.....	692
21.9.4. Операции над массивами.....	694
Копирование массивов.....	694
Сортировка массивов.....	694
Перемещение по массиву.....	696
Запись значений элементов массива в переменные.....	698
Преобразование массива в текстовую строку.....	699
Преобразование текстовой строки в массив.....	699
Другие операции над массивами.....	700
21.10. Глобальные предопределенные переменные.....	702
21.11. Функции.....	704
21.11.1. Пользовательские функции.....	704
21.11.2. Переменные функции.....	708
21.11.3. Встроенные функции.....	709
21.11.4. Как узнать, есть ли такая функция?.....	709
21.12. Классы и объекты.....	709
21.12.1. Определение класса.....	710
Свойства и методы.....	710
Конструктор.....	711
21.12.2. Применение объектов.....	713
21.12.3. Ограничение доступа к свойствам и методам.....	714
21.12.4. Клонирование и удаление объектов.....	716
21.12.5. Использование методов несозданных объектов.....	716
21.12.6. Обработка исключений.....	717
21.12.7. Пример класса формы.....	718
21.13. Выполнение PHP-кода в текстовых строках.....	719

Глава 22. Примеры серверных сценариев	721
22.1. Получение данных из (X)HTML-форм клиента.....	721
22.1.1. Получение данных из HTML-форм.....	721
22.1.2. Передача файлов на сервер.....	729
22.2. Переходы и передача данных между Web-страницами	731
22.2.1. Вывод ссылок.....	732
22.2.2. Применение форм.....	732
22.2.3. Применение функции <i>header()</i> для переадресации.....	732
22.2.4. Добавление информации к URL-адресу	734
22.2.5. Применение cookie	735
22.2.6. Применение сеансов.....	736
Создание сеанса.....	736
Особенности сеансов	737
Пример организации сеанса	738
Защита страниц паролем	740
22.3. Работа с графикой.....	743
22.3.1. Создание и отправка изображения браузеру	743
22.3.2. Масштабирование изображения.....	744
22.3.3. Поворот изображения.....	744
22.3.4. Композиция нескольких изображений.....	745
22.3.5. Вставка текста в изображение	746
22.3.6. Рисование линий.....	747
22.4. Работа с файлами.....	748
22.4.1. Открытие файла	749
22.4.2. Закрытие и удаление файлов	750
22.4.3. Чтение файла.....	751
Чтение файла в переменную.....	751
Чтение файла в массив.....	751
Чтение файла с удалением тегов HTML.....	752
22.4.4. Запись в файл	753
22.4.5. Работа с папками	753
22.4.6. Простой счетчик посещений страницы	755
22.4.7. Работа с CSV-файлами.....	756
Чтение CSV-файла	756
Функции работы с табличными данными	758
Запись двумерного массива в CSV-файл	758
Чтение двумерного массива из CSV-файла	759
Поиск строки в массиве	760
Выборка строк в массиве.....	761
Сложный счетчик посещений страницы	762
Распространяемый счетчик посещений	764
Баннер	766
Гостевая книга	769
22.5. Работа с базами данных	775
22.5.1. Общие сведения о базах данных	775
22.5.2. Установка СУБД.....	776
22.5.3. Основные средства PHP для взаимодействия с базой данных	777
Подключение к базе данных.....	777

Передача запросов к базе данных	778
Обработка данных в сценарии	780
22.5.4. Создание гостевой книги	780
Создание базы данных	781
Сценарии для взаимодействия с посетителем	782
Сценарии для владельца гостевой книги.....	785
22.6. Другие возможности PHP	786
ПРИЛОЖЕНИЯ	787
Приложение 1. Перечень тегов HTML 5.....	789
Приложение 2. Перечень параметров CSS.....	795
Позиционирование	795
Размеры	795
Цвет и фон.....	796
Текст	796
Шрифты.....	796
Блоки (поля, отступы и границы).....	796
Таблицы.....	797
Печать	797
Интерфейс	797
Звук	797
Прочее.....	797
Литература	799
Предметный указатель	801

ГЛАВА 1



Что такое HTML и таблицы стилей CSS

В данной главе мы рассмотрим язык разметки и таблицы стилей. Здесь важно, не вникая в тонкости синтаксиса, понять их назначение, взаимосвязь и основные направления применения. Далее мы все проясним и поставим на свои места.

1.1. Языки разметки документа

Допустим, вам требуется создать документ (письмо другу, резюме для работодателя, объявление о продаже какого-то имущества, прайс-лист, литературное произведение и т. п.), который вы намерены опубликовать — распечатать на принтере, а затем отправить в издательство или разместить на Web-сайте. Ваш документ будет содержать обычный текст и, возможно, картинки. Вот и все, что вы можете позволить себе при публикации документа на бумажных носителях. Разумеется, все то же самое можно опубликовать и в Интернете, стремясь максимально расширить аудиторию читателей. Однако даже в этом простейшем случае нередко возникают порой нетривиальные задачи форматирования (другими словами, оформления) документа. Текст должен быть структурирован, т. е. разбит на части (разделы, подразделы, абзацы), иногда снабженные заголовками. Даже обычное письмо объемом всего в одну страницу, как правило, разбивают на несколько абзацев, не говоря уже о статьях и более крупных произведениях, таких как повести и романы. Заголовки и абзацы выделяются, по крайней мере, своим местоположением относительно друг друга. Кроме того, заголовки разделов и даже некоторые словосочетания обычно выделяют из общего текстового потока шрифтом — гарнитурой, размером, цветом и другими характеристиками. Если в тексте предусмотрены графические иллюстрации, то задача форматирования документа еще более усложняется: необходимо определить, каким образом текст будет "обтекать" вставленные картинки.

При публикации документа в Интернете вы можете сделать значительно больше, чем при печати: вставить в документ аудио- и видеоклипы, формы, заполняемые читателем и отправляемые на сервер, элементы интерфейса (ссылки на другие документы, кнопки, поля ввода, переключатели и т. п.), с помощью которых можно управлять отображением содержимого и решать другие задачи.

Даже в обычных бумажных документах нередко встречаются ссылки на разделы этого же или другого документа (например, "см. разд. 2.5", "в книге [7] можно найти..."),

"в приложении 1 приведен список ..." и т. п.). Встретив такую ссылку в книге, читатель при желании должен либо перелистывать ее страницы, либо искать другую книгу, чтобы затем в поисках нужной информации "порыться" в ней. В Web-документах, просматриваемых в браузерах, щелчок левой кнопкой мыши на ссылке приводит к автоматическому поиску в Сети и отображению в окне браузера соответствующего (релевантного) документа. Такие ссылки в связи с их мощной и удобной функциональностью называют гиперссылками, а тексты, их содержащие, — гипертекстами (HyperText). Гиперссылки могут указывать на документы различного характера — обычные и гипертексты, графические изображения, звуковые и видеоклипы и др. Активизация гиперссылки означает запрос к Web-серверу указанного в ней документа и, если он найден в Сети, пересылку и отображение его в Web-браузере. Благодаря гиперссылкам множество различных документов можно связать в некую единую систему, образовав сайт или, другими словами, Web-узел. Так вы можете создать, по крайней мере, один документ, содержащий гиперссылки на другие, чтобы не только представить самого себя в Интернете, но и послужить проводником к другим ресурсам Всемирной сети. Простая в своей основе идея и адекватный ей инструмент гиперссылки позволил создать и далее развивать технологии, лежащие в фундаменте Всемирной паутины (World Wide Web), преобразившей в последнее время все информационное пространство и коммуникации в нем.

Итак, при подготовке любого документа к публикации его требуется отформатировать или, иначе говоря, разметить — с помощью специальных символов явно указать, что и каким образом следует представить пользователю (читателю) этого документа. Набор специальных символов (меток, дескрипторов, команд или тегов), а также правила их употребления составляют то, что называют языком разметки (Markup Language). В настоящее время существуют несколько таких языков, среди которых наиболее популярны HTML (HyperText Markup Language — язык разметки гипертекста), XHTML (eXtensible HyperText Markup Language — расширяемый язык разметки гипертекста) и XML (eXtensible Markup Language — расширяемый язык разметки). HTML и XHTML применяются для разметки Web-документов, а XML — как средство структуризации информации для обмена между компьютерными программами. Между этими языками очень много общего, хотя имеются и различия, о которых будет рассказано позже.

Разметка документа, как уже упоминалось, осуществляется посредством специальных дескрипторов, называемых тегами (tag — метка, признак, ярлык). В языках разметки в большинстве случаев они имеют вид: `<имя_тега>` и `</имя_тега>`. Например, `<p>` и `</p>`, где `p` — имя тега. Эти две формы одного и того же тега играют роль подобно открывающей и закрывающей скобкам при написании математических формул. Между ними можно разместить некоторое содержимое — обычный текст и/или тег, содержащий, возможно, другие теги. Тег, имеющий открывающую и закрывающую части, называют контейрным. Вот его синтаксис:

```
<имя_тега > содержимое контейнерного тега </имя_тега >
```

Существуют и неконтейрные теги (без закрывающей части), например `` — для вставки графического изображения и `
` — для указания перехода на новую строку.

Теги не отображаются в окне браузера, а лишь указывают ему, какие элементы входят в документ и как они между собой соотносятся. Так, текстовый документ состоит из абзацев, которые могут объединяться в разделы. Разделы, в свою очередь, могут входить

в более крупные структурные единицы, например главы или части и т. д. Язык HTML обеспечивает поддержку иерархической декомпозиции документа довольно простыми средствами. Например, чтобы указать, что данный фрагмент текста относится к одному и тому же абзацу, достаточно перед этим фрагментом написать тег `<p>`, а после него — `</p>`. Другими словами, текст абзаца следует заключить в контейнер `<p>`, например `<p>Здравствуйте, дорогой Иван Иванович.</p>`.

Чтобы объединить в один раздел несколько элементов документа (например, абзацев), их следует заключить в какой-нибудь подходящий для этой цели контейнер, например `<div>` (листинг 1.1).

Листинг 1.1. Пример использования контейнера `<div>`

```
<div>
  <p>
    Здравствуйте, дорогой Иван Иванович.
  </p>
  <p>
    В первых строках своего письма передаю поклон Марье Ивановне.
  </p>
</div>
```

Здесь контейнер `<div>` содержит два контейнера `<p>`, каждый из которых заключает в себя некий текст.

Самый крупный контейнер для содержательной части документа — тег `<body>` (тело документа), он содержит другие контейнеры. Например, формируя письмо, мы могли бы написать фрагмент кода на языке разметки, приведенный в листинге 1.2.

Листинг 1.2. Фрагмент кода на языке разметки

```
<body>
  <div>
    <p>
      Здравствуйте, дорогой Иван Иванович.
    </p>
    <p>
      В первых строках своего письма передаю поклон Марье Ивановне.
    </p>
  </div>
</body>
```

Итак, мы уже упомянули несколько тегов: `<p>`, `
`, `<body>`. В HTML имеется много других тегов для группировки элементов документа в логические блоки, образующие в совокупности структуру документа.

На рис. 1.1 в окне обычного текстового редактора показан пример текста, размеченного тегами абзаца `<p>` и перевода строки `
`, а также его вид в окне браузера. Хотя данный документ в целом оформлен не по всем правилам и даже совсем не по правилам HTML, основные Web-браузеры все-таки отображают его, причем одинаково. Дело в

том, что файл `text.htm`, содержащий теги HTML, был открыт в Web-браузере. Все современные браузеры при открытии в них файлов с расширениями `htm` или `html` пытаются интерпретировать последние как HTML-коды, т. е. как поток тегов языка разметки. Однако это еще не означает, что формировать Web-документ можно "как попало", вспоминая и используя те или иные теги. Правила создания Web-документов существуют, и их настоятельно рекомендуется соблюдать, чтобы избежать несовместимости с различными браузерами и других проблем.

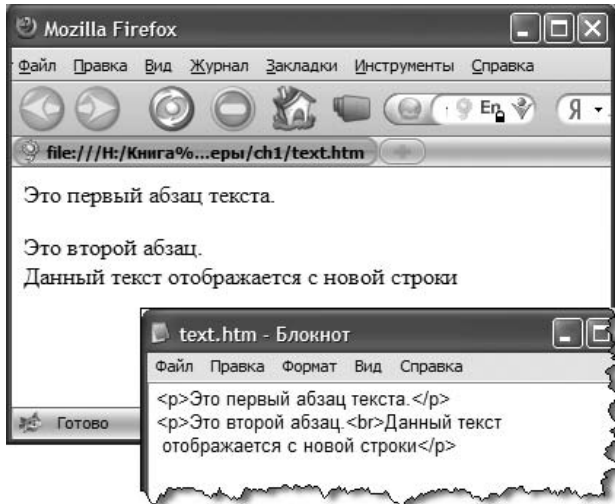


Рис. 1.1. Текст, размеченный тегами абзаца `<p>` и перевода строки `
`, в текстовом редакторе и в браузере

Обратите внимание, что в рассматриваемом примере расположение текста в окнах редактора и браузера различается. В текстовом редакторе символы могут располагаться в одной или нескольких строках, что еще не определяет однозначно их расстановку в окне браузера. При наборе символов в текстовом редакторе нажатие на клавиши `<Enter>`, `<Tab>` и т. п. приводит к форматированию текста, визуально контролируемому с помощью дисплея. Текстовый редактор отвечает на нажатие указанных клавиш расстановкой в вводимом тексте специальных управляющих символов, которые не видны в общем потоке печатаемого текста. Для управления расположением текста в окне браузера требуется явное задание управляющих символов в виде специальных тегов языка разметки. В рассматриваемом примере тег абзаца `<p>` указывает браузеру отобразить заключенный в него текст с новой строки и с отступом в одну строку. В обычном текстовом редакторе такого же эффекта можно добиться, дважды нажав на клавишу `<Enter>` в конце введенного текста. Далее, переход на новую строку внутри второго абзаца задан с помощью тега `
`. Без него этот переход либо отсутствовал бы, либо произошел лишь при достаточно малой ширине окна браузера. Результат зависит от используемого браузера. Дело в том, что браузеры пытаются, так или иначе, перенести поток символов на новую строку, если ширина окна не позволяет расположить текст вдоль одной горизонтальной линии.

Чтобы текст и другие элементы документа отображались инвариантно относительно различных браузеров, необходимо принудительно указывать переносы, отступы и дру-

гие управляющие символы с помощью тегов языка разметки и специальных средств (атрибутов тегов и параметров CSS, о чем будет рассказано далее).

О правилах оформления HTML-документов мы поговорим позже, а сейчас остановимся на одной важной особенности HTML. Не все, но многие теги определяют не только структуру документа, но и внешний вид его элементов в окне браузера, т. е. задают еще и способ отображения соответствующей информации в браузере. Например, тег абзаца `<p>` определяет, что все следующее за ним будет отображено с новой строки и с пропуском одной строки; текст, заключенный между тегами `<h1>` и `</h1>`, будет выведен с новой строки максимальным по размеру шрифтом, а последующая информация будет размещена с новой строки (иначе говоря, контейнер `<h1>` задает заголовок первого уровня); текст в контейнере `<i>` выводится курсивом.

Таким образом, разметка тегами HTML в большинстве случаев предопределяет не только структурную декомпозицию документа, но и его внешний вид в окне браузера. Разумеется, структурная декомпозиция, в конечном счете, должна быть как-то выражена на внешнем образе. Например, листовую книгу, мы видим ее структуру в виде иерархии глав, разделов, подразделов и абзацев. Вместе с тем, нетрудно понять различие между собственно структурной декомпозицией и внешним представлением ее элементов. В самом деле, главы, разделы и подразделы можно отобразить как линейно следующие друг за другом, так и расположенные на страницах в двух и более колонках; заголовкам можно придать тот или иной вид, манипулируя такими параметрами шрифта, как цвет, размер и начертание. При этом логическая структура документа остается постоянной, а изменяется только ее представление на бумаге или в окне браузера. Однако идея четко разделить структурный и "представительский" аспекты языка разметки в HTML не реализована в полной мере. В HTML средства структурной декомпозиции документа и его внешнего представления оказались изначально сильно связаны между собой. Это обстоятельство и сыграло, на мой взгляд, решающую роль в чрезвычайно широкой популярности данного языка разметки: разработчики Web-страниц хотели сразу видеть результат разметки документа в браузере и получили это (см. рис. 1.1).

Для определения элементов, вставляемых в документ, одних только тегов бывает недостаточно. Так, для вставки графического изображения служит тег ``. Однако требуется еще указать источник изображения, т. е. адрес или, точнее, URL (Uniform Resource Locator — унифицированный указатель ресурса) файла. В подобных случаях используются параметры тегов, называемые еще атрибутами. Один или несколько атрибутов записывают в открывающей части тега в произвольном порядке в следующем виде: *имя_атрибута*="значение". Атрибуты отделяют друг от друга пробелами. Вот пример тега для вставки графического изображения:

```

```

Тег `` не имеет заключительной части вида ``.

Чтобы указать размеры графического изображения на странице (в пикселах), можно записать следующее выражение:

```

```

Здесь `src`, `width` и `height` — имена атрибутов тега ``, за которыми следует знак равенства и, далее, значения, указанные в кавычках, двойных или одинарных.

Атрибуты позволяют указать, например, цвет фона, размеры, начертание и цвет шрифта, характеристики выравнивания элементов документа и т. п. Так, в контейнерном теге `<body>`, задающем основную часть документа, с помощью атрибутов `text` и `bgcolor` можно определить цвет текста и фона:

```
<body text="#ff0000" bgcolor="#00ffee">.
```

Таким образом, развитие HTML пошло в сторону добавления средств, определяющих внешнее представление документов. Стали появляться новые теги и их новые атрибуты. На первых порах производители браузеров (Microsoft, Netscape, Sun Microsystems и др.) создавали свои варианты HTML, отличающиеся не только наборами тегов, но и способами их интерпретации. В сложившихся условиях один и тот же документ мог выглядеть по-разному в зависимости от браузера, что явно противоречило цели всемирного распространения информации. Очевидно, стандарты стали насущной необходимостью.

Разработкой стандарта для HTML занялась международная организация World Wide Web Consortium (W3C, Консорциум Всемирной паутины), в состав которой вошли крупнейшие производители программного обеспечения для работы в Интернете (в том числе Microsoft, Sun Microsystems, Netscape и др.). Сведения о стандартах (спецификациях) можно найти на официальном сайте W3C по адресу www.w3.org. Схема подготовки спецификаций консорциумом примерно такая: сначала выпускается проект или черновик (draft) спецификации, в результате обсуждения которого появляется ее рабочий вариант. Последний предлагается к обсуждению в течение некоторого времени, по окончании которого рабочий вариант может стать рекомендацией — официальным вариантом спецификации. Браузеры должны интерпретировать язык разметки соответственно стандарту, хотя на практике это происходит не в полной мере. Одно дело идея, другое — ее реализация.

В июле 1997 г. консорциум W3C выпустил проект спецификации HTML 4.0, который уже в декабре стал официальной рекомендацией для производителей Web-браузеров, а с декабря 1999 г. существует версия 4.01. Тем не менее различия в интерпретации основными браузерами некоторых тегов и атрибутов официальной спецификации HTML все же остались. Для обеспечения дополнительных возможностей визуального представления, программирования поведения, динамичности и интерактивности документов появились каскадные таблицы стилей (CSS — Cascading Style Sheets) и скрипты (сценарии), написанные на специальных языках, наиболее популярный из которых — JavaScript.

Параллельно с HTML развивался другой язык разметки — XML (eXtensible Markup Language — расширяемый язык разметки). Он был создан как средство структуризации информации для обмена между программами. Web-браузер — одна из них, но далеко не единственная. При этом на способ внешнего представления структурированных данных никаких ограничений не накладывалось. Главная задача, которая ставилась при разработке этого языка, состояла в обеспечении совместимости между различными системами обработки структурированных данных. Передающей стороне не важно, как будет отображен XML-документ получателем, требуется лишь, чтобы он однозначно разобрался в структуре принятого документа.

В XML, так же как и в HTML, есть теги и атрибуты. Однако, в отличие от HTML, в языке XML они не предопределены изначально, а могут создаваться автором доку-

мента по своему усмотрению. Теги XML задают лишь структуру документа, но не его представление. На рис. 1.2 показан фрагмент XML-кода в текстовом редакторе и результат его обработки Web-браузером. Теги `<manual>` и `<step>` были придуманы автором данной книги для представления структуры инструкции по созданию сайта. Web-браузер не "знает" таких тегов, а потому просто показывает XML-код без какой-либо его интерпретации. Внешний вид элементов XML-документа определяется отнюдь не тегами, а специальными программами-анализаторами, часто называемыми парсерами (parser). Основные Web-браузеры имеют такие встроенные парсеры, но чтобы подключить их к работе и заставить отобразить собственно информацию (без тегов), необходимо указать, что содержимое документа является XML-кодом и, кроме того, сослаться на каскадную таблицу стилей (CSS).

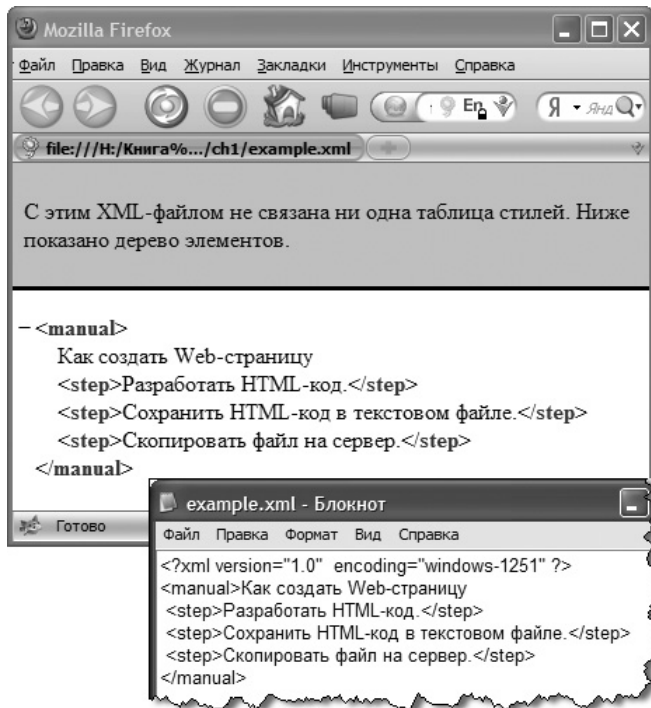


Рис. 1.2. Внешний вид XML-кода, загруженного в браузер

Таблицы стилей сохраняются в отдельных файлах и содержат параметры визуализации элементов (тегов) XML-документа. На рис. 1.3 показан пример, в котором XML-документ, представляющий инструкцию, в первой строке содержит дескриптор, указывающий тип документа, а во второй строке — ссылку на файл `example.css` с правилами внешнего представления тегов `<manual>` и `<step>`. В данном случае браузер не отображает теги, а выводит результат их интерпретации. Так, в таблице стилей указаны цвет и "жирность" шрифта, а также требование, согласно которому каждый элемент `<step>` (шаг инструкции) занимает по горизонтали 200 пикселей (`width:200px`) с отступом 20 пикселей слева (`padding-left:20px`) и начинается с новой строки (`display:block`).

При желании можно модифицировать параметры отображения элементов документа, оставляя неизменной его логическую структуру.

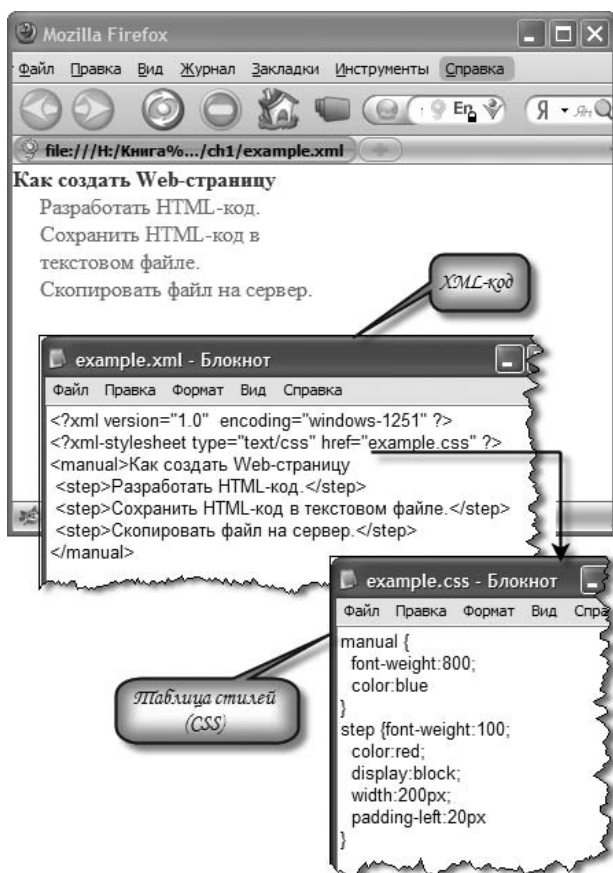


Рис. 1.3. Пример использования таблицы стилей для отображения XML-документа в окне браузера

Итак, в XML воплотилась хорошая идея разделения между структурным и внешним представлением содержания документа. В HTML это не так, но его можно немного модернизировать, чтобы существенно приблизить к XML. На пути к достижению этой цели появился XHTML (eXtensible HyperText Markup Language — расширяемый язык разметки гипертекста).

ПРИМЕЧАНИЕ

XML — подмножество более сложного языка SGML (Standard Generalized Markup Language — стандартный обобщенный язык разметки). На основе XML созданы более специализированные языки разметки, например XHTML, SVG, RSS, MathML и др. Такие языки еще называют словарями данных языка XML. С другой точки зрения XML представляет собой текстовый формат описания иерархических баз данных.

Еще до появления XHTML многие разработчики Web-сайтов практиковали следующий технологический прием. С помощью HTML-тегов создавалась лишь структура документа, при этом теги и атрибуты, предназначенные только для задания внешнего вида элементов, практически не использовались (насколько это было возможно), а отображение элементов документа в окне браузера или при печати определялось только в каскадных таблицах стилей. Данный прием можно назвать "применением HTML в стиле XML".

Разметку Web-страницы можно реализовать на основе языка как HTML, так и XHTML. Однако XHTML более строг, ближе к универсальному языку XML и должен поддерживаться одинаковым образом всеми современными браузерами. В данной книге все примеры написаны с использованием синтаксиса XHTML.

Читатели, знакомые только с HTML, увидят не слишком много отличий этого языка от XHTML. Перейти от HTML-кодов к XHTML-кодам довольно легко. Языки HTML и XHTML очень схожи, но различия все же есть. Вот основные (но далеко не все) из них:

- язык HTML регистронезависимый: не имеет значения, строчными или прописными буквами записаны имена тегов и атрибутов. В XHTML для записи имен тегов и атрибутов рекомендуется указывать только строчные буквы. Например, вместо

```
<IMG SRC="mypicture.jpg">
```

следует писать

```
;
```

- в HTML большинство тегов контейнерные (например, `<h1>...</h1>`), т. е. имеют открывающий и закрывающий дескрипторы. Вместе с тем есть и неконтейнерные теги: для дескриптора `<тег>` нет закрывающего дескриптора `</тег>`. Например, для вставки графического изображения в HTML применяется тег ``, а в XHTML — ``. Неконтейнерные теги еще называют пустыми. В XHTML для таких тегов необходимо указывать слэш (косую черту) перед закрывающей угловой скобкой: `<тег ... />`, например

```
 или <br/>;
```

- в некоторых тегах HTML имеются атрибуты без значений (так называемые булевы или логические атрибуты), например `<option selected>`. В XHTML для подобных атрибутов следует явно указывать строковое значение, совпадающее с именем соответствующего атрибута, например

```
<option selected="selected" />;
```

- в XHTML значения атрибутов тегов нужно заключать в кавычки. В HTML это делать не обязательно. Например, в XHTML следует писать ``, а не ``, как допустимо (хотя тоже не рекомендуется) в HTML. Некоторые атрибуты принимают числовые значения, которые в HTML можно (хотя и не рекомендуется) указывать без кавычек, но в XHTML их всегда следует заключать в кавычки, двойные или одинарные, например

```
;
```

- в содержательной текстовой части XHTML-документа нельзя непосредственно использовать символы, которые применяются в коде, такие как `<`, `>` и `&`, причем их недопустимо записывать даже в значении URL-адреса. Эти специальные символы следует заменять соответствующими буквенными или числовыми эквивалентами `<`, `>` и `&` (или `<`, `>` и `&`;) соответственно. Об эквивалентах специальных символов рассказано в *разд. 7.3*;

- кодировка файла по умолчанию для документа HTML 4 — ISO 8859-1, а для документов XHTML и HTML 5 — UTF-8.

Ошибки в HTML-коде браузеры пытаются преодолеть тем или иным образом, так что даже весьма небрежно написанный код все же как-то отображается (во многих случаях вполне удовлетворительно). По рекомендации W3C в случае ошибки в XHTML-коде браузеры должны сообщать об этом и прекращать дальнейшую обработку. Таким образом, XHTML-код подвергается более тщательному анализу прежде, чем начнется его интерпретация — отображение в окне браузера. HTML-код, напротив, не анализируется предварительно на правильность, а отображается в порядке следования тегов. При этом сообщения об ошибках не выводятся, а отображение продолжается, даже если ошибки все же возникли. Очевидно, HTML более "дружественен" разработчику, чем XHTML. Но более других эту "дружественность" ценят малоопытные и небрежные разработчики своих домашних страниц, а не авторы серьезных Web-проектов. Для последних однозначность интерпретации и дисциплина важнее "угодливости" браузеров, вредность которой проявляется при поиске трудноуловимых смысловых ошибок и при попытках обеспечить межбраузерную инвариантность представления документов.

В XHTML сохранилось большинство тегов HTML 4. Однако некоторые старые HTML-теги, введенные в свое время для обеспечения дополнительных возможностей отображения, теперь применять не рекомендуется.

В настоящее время существуют официальные версии XHTML 1.0 и 1.1 и черновая спецификация XHTML 2.0. Однако работа над последней остановлена в пользу создания нового стандарта — HTML 5.

Начиная с 2004 г. совместными усилиями рабочих групп W3C HTML WG и WHATWG с привлечением таких компаний, как Apple, Google, Mozilla, Opera, Microsoft и др., ведется работа по созданию спецификации HTML 5. К середине 2010 г. существовал лишь черновой ее вариант, а к началу 2014 г. — кандидат в рекомендации (www.w3.org/TR/html5). Все современные ведущие браузеры поддерживают основные элементы HTML 5.

HTML 5 призван стать преемником как HTML 4, так и XHTML 1.0, обеспечивая совместимость с ними, по крайней мере, в самом важном. В HTML 5 возможен синтаксис как обычного HTML, так и XHTML. Вместе с тем, ряд тегов и атрибутов, не рекомендованных к использованию в прежних версиях, в новом стандарте отсутствуют. Например, теги `<applet>` для вставки Java-апплетов, `<center>` для центрирования содержимого по горизонтали и `<frameset>`, `<frame>` для разбиения окна браузера на статические фреймы в HTML 5 не поддерживаются. С целью обеспечения обратной совместимости в спецификации HTML 5 описывается, как должен поступить браузер при наличии непредусмотренных тегов. Уже существующие сайты не должны пострадать из-за появления нового языка разметки.

В HTML 5 вводится в оборот ряд новых тегов с отчетливо выраженной семантикой, предназначенных для повышения эффективности разработки Web-приложений, а также для помощи поисковым системам. Так, для создания логической структуры и упрощения типовой верстки документа предлагаются теги `<header>` (заголовок), `<nav>` (навигационная панель), `<article>` (статья), `<section>` (раздел, секция статьи), `<aside>` (боковая колонка), `<footer>` ("подвал"), `<menu>` (меню) и др. В прежних версиях указанные элементы обычно создаются тегами `<div>` и сразу понять смысл их применения довольно трудно. Иначе говоря, семантика для `<div>` неоднозначна.

Элементы `<input>` полей ввода данных в HTML 5 приобрели новые атрибуты, заметно облегчающие создание пользовательского интерфейса: `time`, `email`, `url`, `required` и др. Эти атрибуты во многих случаях позволяют обойтись без скриптов, проверяющих правильность введенных данных.

Особого внимания заслуживает элемент `<canvas>`, предоставляющий область на странице, в которой с помощью скриптов на языке JavaScript можно рисовать изображения, подобные создаваемым посредством SVG или Flash. Для вставки мультимедиа в HTML 5 предусмотрены теги `<audio>` и `<video>`, обеспечивающие внедрение на страницу звука и видео с возможностью управления.

HTML 5 предусматривает многие другие полезные и удобные возможности. В данной книге будут описаны лишь те из них, которые уже поддерживаются ведущими браузерами.

1.2. Что такое таблицы стилей

Как уже говорилось, параметры внешнего вида документа можно задать с помощью *каскадных таблиц (листов) стилей* (Cascading Style Sheets, CSS). Таблицу стилей, подобно шаблону форматирования текстов, можно разработать отдельно от конкретного (X)HTML-документа, а затем применить к нему. Модификация содержимого таблицы стилей меняет внешний вид (X)HTML-документов, не затрагивая их структуры и информационного содержания. Одна и та же таблица стилей может применяться к нескольким документам и, наоборот, к одному и тому же документу может быть применено несколько таблиц стилей. В последнем случае браузер учитывает приоритеты таблиц и по определенным правилам разрешает возникающие конфликты, в результате чего таблицы выстраиваются неким каскадом (отсюда и название — каскадные таблицы стилей).

Кроме технологичности стилизации (X)HTML-документов, CSS обеспечивает еще и произвольное позиционирование элементов. Для любого элемента можно задать размеры и координаты расположения, а также другие параметры визуализации. Так что, применяя CSS, можно обойтись без тегов таблиц и фреймов, которые широко используются как средство компоновки Web-страниц.

CSS содержит наборы стилевых параметров или, другими словами, правила форматирования. Например, если требуется определить для всех заголовков первого уровня шрифт Courier 24 пункта красного цвета, то соответствующее правило можно записать так:

```
h1 {font-family: Courier; font-size: 24pt; color: red}
```

Здесь в фигурных скобках указан список стилевых параметров — пар вида *имя:значение*, разделяемых точкой с запятой; левее этого списка указано имя `h1` тега, к которому данные параметры требуется применить. В результате заголовок первого уровня (содержимое тега `<h1>`) приобретает специальные параметры, установленные автором стиля, а не принятые по умолчанию, когда авторский стиль не задан.

Одно и то же правило можно применить к различным элементам и, наоборот, для одного и того же элемента можно задать несколько правил. Следующее правило определяет

одинаковые стилевые параметры одновременно для заголовков первого и второго уровней:

```
h1, h2 {font-family: Courier; font-size: 24pt; color: red}
```

А вот пример каскадного задания стилевых параметров:

```
h1, h2 {font-size: 24pt; color: red}
h1 {font-family: Arial}
h2 {font-family: Courier}
```

Здесь сначала задаются размер и цвет шрифта для обоих заголовков, а затем для каждого из них определяется своя гарнитура шрифта (имя шрифта).

Стилевые параметры, заключенные в фигурные скобки, можно назначить не только именам тегов, но и тегам по значениям их атрибута `id` (идентификатора элемента). Так можно изменить представление не всех, например, заголовков первого уровня (тегов `<h1>`), а только тех, которые имеют указанное значение атрибута `id`. Существует также возможность определить правила, изначально не связанные ни с какими элементами документа, а затем сослаться на них из любого элемента с помощью атрибута `class`. Указатели на элементы, к которым применяются стилевые параметры в фигурных скобках, называются селекторами. Селекторы, как мы только что видели, могут быть различных видов. Подробнее о них будет рассказано в *разд.* 3.2.

Правила CSS можно применить к любому видимому элементу (X)HTML-документа. Почти каждому элементу, заданному тем или иным тегом, можно придать произвольное представление. Поэтому оказывается вполне достаточно одного или нескольких тегов, например `<div>` и/или ``, а визуальное разнообразие их вхождений в документ можно обеспечить посредством только CSS. Однако при этом, возможно, будет нанесен ущерб свойству (X)HTML представлять логическую структуру документа. Действительно, глядя на (X)HTML-документ, содержащий одни только теги `<div>`, будет трудно понять, где заголовок текстового документа, а где абзац основного текста. Поэтому многие опытные разработчики рекомендуют не придерживаться чрезмерного "тегового минимализма", а использовать основные структурообразующие теги по прямому назначению. Например, теги `<h1>`, ..., `<h6>` следует применять только для создания заголовков, а не, скажем, оформления гиперссылок. Кстати, HTML 5 явно поддерживает идею семантической определенности тегов, вводя в оборот элементы `<header>`, `<nav>`, `<article>`, `<section>` и т. п.

Правила CSS можно записать непосредственно в (X)HTML-документе или сохранить в отдельном файле, чтобы применить их одновременно к нескольким документам. В последнем случае браузер кэширует таблицу стилей и, следовательно, последующие страницы, задействующие ту же таблицу стилей, загружаются быстрее. Правила, записываемые непосредственно в самом (X)HTML-документе, заключают в контейнерный тег `<style>`, а для ссылки на таблицу стилей, расположенную во внешнем файле, в контейнер `<head>` вставляют тег

```
<link href="URL css-файла".../>
```

или в контейнер `<style>` добавляют директиву

```
@import url("URL css-файла");
```

На рис. 1.4 показан пример таблицы стилей, непосредственно вставленной в (X)HTML-документ. Если правила этой таблицы разместить в текстовом файле `mystyle.css`, то вместо тега `<style>` следует указать ссылку на данный файл, например, так:

```
<link rel="stylesheet" type="text/css"
href="http://www.anyserver.ru/mystyle.css" />
```

или (в рамках тега `<style>`) так:

```
@import url("http://www.anyserver.ru/mystyle.css");
```

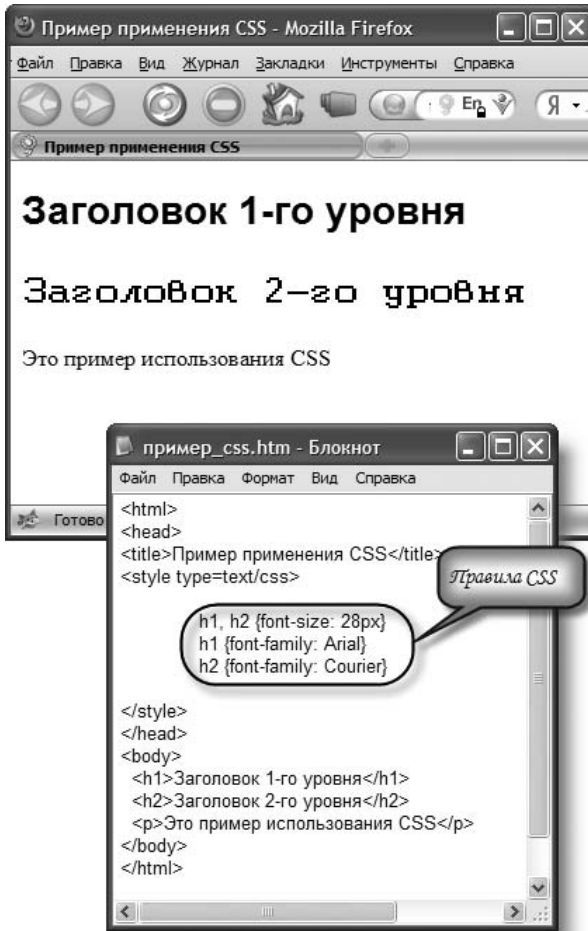


Рис. 1.4. Пример использования таблицы стилей в (X)HTML-документе

Спецификация каскадных таблиц стилей уровня 1 (CSS 1) была принята в качестве рекомендации W3C в декабре 1996 г., а в январе 1999 г. вышел ее откорректированный вариант. Рекомендация спецификации уровня 2 (CSS 2) была выпущена в мае 1998 г. Она сохраняла обратную совместимость с CSS 1. Пересмотренный ее вариант (CSS 2.1) увидел свет в апреле 2008 г. и вошел составной частью в спецификацию CSS 3, достигшую стадии рабочего проекта (working draft). CSS 3 предоставляет существенно новые возможности, начиная от простых визуальных эффектов (тени, скругленные углы)

и заканчивая трансформацией и анимацией. С текущим состоянием работ по CSS 3 можно познакомиться, например, по адресу www.w3.org/Style/CSS/current-work.

Хотя спецификация CSS 3 пока находится в разработке, различные браузеры уже поддерживают в той или иной степени многие из ее элементов. При этом имена параметров, возможно, придется модифицировать путем добавления префикса в зависимости от движка браузера. Так, для Mozilla Firefox (движок Gecko) добавляется префикс `-moz-`, для Google Chrome и Apple Safari (движок WebKit) — `-webkit-`, для Opera — `-o-`, для Microsoft Internet Explorer (движок Trident) — `-ms-`. В последующих версиях браузеров необходимость добавления к именам параметров специфических префиксов, возможно, отпадет.

Более подробно таблицы стилей мы рассмотрим в *главе 3*, а сейчас вернемся к языку разметки документа.

ГЛАВА 2



Структура (X)HTML-документа

Как известно, Web-узел (сайт) состоит из одной или нескольких страниц, каждая из которых представлена своим кодом, написанным на языке разметки HTML или XHTML. Такой код мы будем также называть (X)HTML-документом. Он имеет определенную структуру и сохраняется в обычном текстовом файле с типичными расширениями `htm` или `html`. Файл с (X)HTML-документом главной страницы сайта, которая загружается в браузер первой, обычно называется `index.htm` или `index.html` и размещается в корневой папке сайта на Web-сервере. При активизации ссылки на сайт без указания имени файла загружается именно `index.htm` или `index.html`, если он будет найден на сервере по указанному адресу в корневой папке сайта. Разумеется, вы можете назначить любое имя файлу с (X)HTML-кодом вашей главной страницы, но тогда URL-адрес, помимо адреса сайта, должен будет содержать и имя файла.

Прежде всего, (X)HTML-документ, соответствующий вашей Web-странице, должен содержать описание типа документа (как требует стандарт DTD — Document Type Definition), после чего начинается собственно код — набор тегов, т. е. дескрипторов языка разметки (X)HTML.

Теги записывают в одну или несколько строк, с отступами или без, что не влияет на отображение соответствующих элементов в окне браузера. Выбирая способ расположения элементов кода, следует исходить из удобства его чтения, но не забывать при этом, что символы пробелов, табуляции и перевода строки вносят свою лепту в общий объем файла. Важно, чтобы после открывающей угловой скобки (`<`) сразу же (без пробелов) следовало имя тега, в то время как закрывающая угловая скобка может находиться сколь угодно далеко. Это не единственное правило (X)HTML, но новички очень часто игнорируют его, из-за чего возникают досадные ошибки.

2.1. Определение типа документа: дескриптор `<!DOCTYPE...>`

В разд. 1.1 упоминалось, что для разметки документа могут использоваться различные языки, такие как HTML, XHTML, XML и др. Для правильной интерпретации языка, на котором написан документ, браузеру следует сообщить, какой именно язык выбран. Это делается посредством специального дескриптора (тега).

Собственно (X)HTML-код Web-страницы заключается в контейнерном теге `<html>` (т. е. между дескрипторами `<html>` и `</html>`). Однако при создании настоящего (X)HTML-документа рекомендуется дать его описание, расположенное перед тегом `<html>` и состоящее из одного или двух специальных дескрипторов. В одном из таких дескрипторов указывается тип документа (DTD — Document Type Definition).

Тип документа HTML 5 задается специальным дескриптором:

```
<!DOCTYPE html>
```

Если вы хотите создать Web-страницу (HTML-документ) с самого начала (как говорят, с нуля), то в первой строке своего HTML-кода напишите `<!DOCTYPE html>` и можете сразу перейти к следующему разделу.

Дескриптор `<!DOCTYPE html>` подойдет и в случае, если код вашего документа использует лишь возможности прежнего HTML 4 или же написан с применением синтаксиса XHTML. Стандарт HTML 5 предоставляет новые возможности и обеспечивает совместимость с устаревшими версиями путем указаний, как браузеры должны действовать, если встретились с теми или иными записями в документе, которые не соответствуют HTML 5.

Ранее, до появления HTML 5, для документов XHTML и HTML 4 дескриптор `<!DOCTYPE...>` имел более сложную структуру. Вы можете заменить DTD своих старых (X)HTML-документов новым дескриптором `<!DOCTYPE html>`. Однако в этом нет необходимости. Поскольку многие разработчики ничего не изменяют в своих DTD, я приведу в качестве справочной информации общую схему и примеры DTD, используемые для XHTML, HTML 4 и некоторых других языков разметки. Эта схема имеет следующий вид:

```
<!DOCTYPE элемент_верхнего_уровня публичность
"регистрация//организация//тип имя//язык" "url">.
```

Атрибуты дескриптора `<!DOCTYPE...>`:

- элемент верхнего уровня* — для XHTML и HTML элемент верхнего уровня — `html`;
- публичность* — возможны значения `PUBLIC` (для публичных документов) и `SYSTEM` (для системных ресурсов);
- регистрация* — возможны значения `+` (разработчик DTD зарегистрирован в международной организации по стандартизации ISO) и `-` (разработчик не зарегистрирован в ISO); для консорциума W3C указывается `-`;
- организация* — уникальное название организации, разработавшей данный DTD; для (X)HTML-документов такой организацией является W3C;
- тип* — тип описываемого документа; для (X)HTML-документов указывается DTD;
- имя* — уникальное имя документа с описанием DTD, например `XHTML 1.0 strict` (для XHTML версии 1.0 строгой схемы) или `HTML 4.01 transitional` (для HTML версии 4.01 переходной схемы);
- язык* — язык, на котором описан объект; значение состоит из двух букв в верхнем регистре; для (X)HTML-документов указывается английский язык (EN);
- url* — URL-адрес файла с описанием DTD, например `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd`.