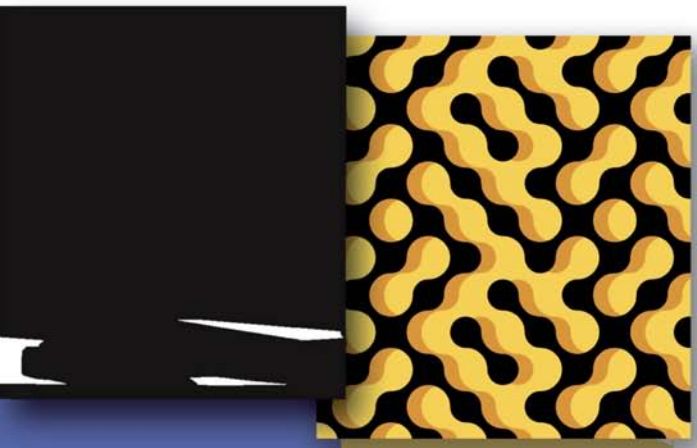
Владимир Дронов

bhv

PHP, MySQL, HTML 5 и CSS 3

Разработка современных динамических Web-сайтов



- АУДИО И ВИДЕО HTML 5
- ПРЕОБРАЗОВАНИЯ И АНИМАЦИЯ CSS 3
- JavaScript, DOM и AJAX
- БИБЛИОТЕКА Yii
- ИНТЕРАКТИВНЫЕ ЭЛЕМЕНТЫ: СПОЙЛЕР, ЛАЙТБОКС И БЛОКНОТ
- УНИВЕРСАЛЬНОЕ ФАЙЛОВОЕ ХРАНИЛИЩЕ
- ПОДДЕРЖКА BBCode
- ПУБЛИКАЦИЯ САЙТА

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Материалы на www.bhv.ru

Владимир Дронов

PHP, MySQL, HTML5 и CSS 3

Разработка современных динамических Web-сайтов

Санкт-Петербург
«БХВ-Петербург»
2016

УДК 004.43+004.738.5
ББК 32.973.26-018.1
Д75

Дронов В. А.

Д75 PHP, MySQL, HTML5 и CSS 3. Разработка современных динамических Web-сайтов. — СПб.: БХВ-Петербург, 2016. — 688 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-3529-8

Книга посвящена разработке динамических Web-сайтов с применением HTML5, CSS 3, PHP и MySQL. Описаны возможности HTML5 по работе с текстом, графикой, аудио и видео, таблицами, средствами навигации и Web-формами, а также способы представления, преобразования и анимации Web-страниц с помощью CSS 3. Рассказано о языке JavaScript, объектной модели документа DOM, разработке Web-сценариев и технологии AJAX. Рассмотрены серверное программирование, язык PHP и сервер данных MySQL. Дано описание библиотеки Yii, предоставляющей Web-программисту готовый набор инструментов для написания серверных приложений. На практических примерах показана разработка дизайна страниц, интерактивных элементов — спойлера, лайтбокса и блокнота, создание универсального файлового хранилища и реализации поддержки тегов BBCode для форматирования текста. Рассмотрен процесс создания полнофункционального сайта и его публикации в Интернете. Все исходные коды доступны для загрузки с сайта издательства.

Для Web-программистов

УДК 004.43+004.738.5
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 31.08.15.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 55,47.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

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

ISBN 978-5-9775-3529-8

Оглавление

Введение

Давайте посмотрим, какие боевые действия происходят на фронте интернет-технологий, какие на нем случились победы и поражения.

С одной стороны, уверенно шагают вперед и одерживают вполне убедительные победы новые версии языков HTML и CSS — HTML5 и CSS 3. Современные Web-обозреватели обзаводятся поддержкой все новых и новых возможностей, предлагаемых этими языками, а Web-разработчики стараются использовать их на практике.

С другой стороны, "ветераны" рынка Web-разработки — сервер данных MySQL и платформа PHP — отнюдь не сдают свои позиции в битве с новичками. И можно с уверенностью говорить о том, что эти технологии будут существовать и оставаться актуальными еще очень и очень долго.

Существует много книг, рассказывающих об HTML5, CSS 3 и JavaScript. Книг, повествующих о PHP и MySQL, издано еще больше. Но автору не известна ни одна книга, которая бы рассказывала обо всех этих технологиях и их совместном применении для создания сайтов. "Надо бы восполнить этот пробел", — решил автор...

В результате появилась эта книга, которую следовало бы озаглавить "Все современные интернет-технологии под одной обложкой".

Некоторые замечания

Нужно заметить сразу, что стандарты HTML5 и CSS 3 еще окончательно не утверждены, более того, работа над ними продолжается до сих пор. И не все Web-обозреватели поддерживают полный набор средств, предоставляемых этими языками.

Эта книга описывает лишь подмножество HTML5 и CSS 3, поддерживаемое Web-обозревателями, в которых заявлена поддержка этих технологий. Ниже перечислены самые ранние версии программ Web-обозревателей, в которых появилась поддержка ключевых возможностей HTML5 и CSS 3:

- Microsoft Internet Explorer — 9;
- Mozilla Firefox — 16;

- Google Chrome — 26;
- Opera — 12.1;
- Apple Safari — 6.1.

Возможности, поддерживаемые не всеми Web-обозревателями, равно как и те, что стали поддерживаться более поздними версиями, здесь описаны не будут.

Отметим, что это касается лишь ключевых инструментов, необходимых для того, чтобы страницы успешно открывались и выводились на экран.

Также книга описывает только основные инструменты библиотеки Yii, необходимые для разработки учебного сайта. Что касается PHP и MySQL, то по ним дается лишь базовый учебный курс.

В конце концов, цель автора — не описать всю функциональность, предлагаемую HTML5, CSS 3, PHP, MySQL и Yii, а научить с их помощью разрабатывать сайты. Если же кому-то из читателей понадобится полное описание вышеупомянутых технологий, он может обратиться к справочникам, доступным на их "домашних" сайтах (список которых приведен в *заключении*).

В качестве практики мы создадим полнофункциональный сайт — систему электронных публикаций "СЭП", предоставляющую площадку для размещения в Интернете статей на различные темы. Этот сайт можно свободно использовать как основу для разработки других, более сложных решений.

> ***Материалы полнофункционального сайта***
>
> Электронный архив с материалами сайта "СЭП: Система электронных публикаций" можно скачать с FTP-сервера издательства "БХВ-Петербург" по ссылке **ftp://ftp.bhv.ru/9785977535298.zip** или со страницы книги на сайте **www.bhv.ru** *(см. приложение 4)*.

Автор применял в работе следующие версии ПО:

- Microsoft Windows 8 и 8.1, русская 64-разрядная редакция со всеми установленными обновлениями;
- Open Server 5.1.1;
- Yii 1.1.15.

Созданные Web-страницы проверялись в следующих Web-обозревателях:

- Microsoft Internet Explorer 10 и 11;
- Mozilla Firefox 32 и 33;
- Google Chrome 37.

Для написания программного кода применялся текстовый редактор Notepad+, который можно найти по интернет-адресу **http://notepad-plus-plus.org/**.

Типографские соглашения

В книге будут постоянно приводиться форматы написания различных конструкций, применяемых в языках HTML, CSS, JavaScript и PHP. В них использованы особые типографские соглашения, которые мы сейчас изучим.

ВНИМАНИЕ!

Все эти типографские соглашения применяются автором только в форматах написания языковых конструкций. В реальном программном коде они не имеют смысла.

- В угловые скобки (<>) заключаются наименования различных значений, которые дополнительно выделяются курсивом. В реальный код, разумеется, должны быть подставлены реальные значения. Например:

```
<страница или элемент>.getElementsByClassName(<имя стилевого класса>)
```

Здесь вместо подстроки *<страница или элемент>* должны быть подставлены реальная страница или реальный элемент страницы, а вместо подстрок *<имя стилевого класса>* — реальное имя стилевого класса.

- В квадратные скобки ([]) заключаются необязательные фрагменты кода. Например:

```
background-position: <горизонтальная позиция> [<вертикальная позиция>]
```

Здесь *вертикальная позиция* может присутствовать, а может и отсутствовать.

- Символом вертикальной черты (|) разделяются параметры или значения, из которых в коде должен присутствовать лишь один. Например:

```
font-style: normal|italic|oblique
```

Здесь допускается указание лишь одного из перечисленных значений — либо `normal`, либо `italic`, либо `oblique`.

- Слишком длинные, не помещающиеся на одной строке языковые конструкции автор разрывал на несколько строк и в местах разрывов ставил знаки ↳. Например:

```
"<div class='media'><audio controls preload='metadata'>↳
<source src='\\1'><source src='\\2'></audio></div>"
```

Приведенный код разбит на две строки, но должен быть набран в одну. Символ ↳ при этом нужно удалить.

- Многоточием (. . .) помечены пропущенные по тем или иным причинам фрагменты кода. Обычно такое можно встретить в исправленных впоследствии фрагментах кода — приведены лишь собственно исправленные выражения, а оставшиеся неизмененными пропущены.

Также многоточие используется, чтобы показать, в какое место должен быть вставлен вновь написанный код, — в начало исходного фрагмента, в его конец или в середину, между уже присутствующими в нем выражениями.

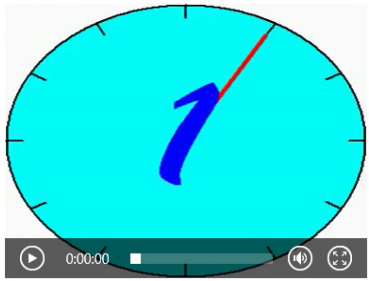
ЕЩЕ РАЗ ВНИМАНИЕ!

Все приведенные здесь типографские соглашения имеют смысл только в форматах написания конструкций языков HTML, CSS, JavaScript и PHP. В коде примеров используется только знак ↳и многоточие.

Благодарности

Автор приносит благодарности своим родителям, знакомым и коллегам по работе.

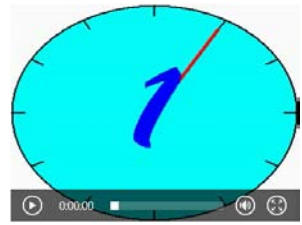
- Белову Алексею Васильевичу, начальнику (теперь уже бывшему) отдела ОИТ Волжского гуманитарного института (г. Волжский Волгоградской обл.), где работает автор, — за понимание и поддержку.
- Всем работникам отдела ОИТ — за понимание и поддержку.
- Родителям — за терпение, понимание и поддержку.
- Архангельскому Дмитрию Борисовичу — за дружеское участие.
- Шапошникову Игорю Владимировичу — за побуждение начать писательскую деятельность.
- Рыбакову Евгению Евгеньевичу, заместителю главного редактора издательства "БХВ-Петербург", — за неоднократные побуждения к работе, без которых автор давно бы обленился.
- Издательству "БХВ-Петербург" — за издание моих книг.
- Разработчикам всех использованных автором программных продуктов — за их труд.
- Всем своим читателям и почитателям — за прекрасные отзывы о моих книгах.
- Всем, кого я забыл здесь перечислить, — за все хорошее.



ЧАСТЬ I

Содержимое Web-страниц. Язык HTML5

ГЛАВА 1



Современный Web-дизайн. Web 2.0

Всемирная паутина, WWW (World Wide Web), Web-дизайн, Web-сайт, Web-страница — все знают, что это такое. Но что такое современная Всемирная паутина, современный Web-дизайн и современная Web-страница? Именно с ответов на все эти вопросы начнется данная книга.

Современный Web-дизайн. Концепция Web 2.0

Раньше доступ в Интернет можно было получить только с компьютеров. Потом в Интернет стали выходить с мобильных телефонов. Сейчас к Сети подключились мультимедийные плееры, устройства чтения электронных книг, появившиеся недавно планшетные компьютеры и "умные" телевизоры. А завтра — кто знает; может быть, мы будем выходить на Web-сайты с холодильника или пылесоса...

"Я буду везде, — заявляет Интернет. — Я стану вездесущим. Все готовьтесь к моему приходу!"

Что требуется от современного Web-сайта

Будем готовиться... Но что нам, как будущим Web-дизайнерам, для этого следует сделать? Соблюсти три несложных правила.

1. Строго соблюдать все интернет-стандарты.
2. Тщательно продумать наполнение Web-страниц.
3. Позаботиться о доступности Web-страниц.

Рассмотрим их подробнее.

Интернет грозится прийти на самые разные устройства, которые могут быть основаны на различных аппаратных и программных платформах, зачастую сильно отличающихся друг от друга. Так, персональные компьютеры построены на аппаратной платформе Intel и программной платформе Microsoft Windows (по крайней

мере, большинство). Мобильный телефон и планшет автора работают под управлением операционной системы Google Android. А на подходе — уже так называемые "интернет-вещи", "умные вещи", обладающие возможностью подключаться к Сети, и на каких платформах они будут работать, пока неизвестно...

Одно объединяет все это аппаратно-программное многообразие — строгое соблюдение интернет-стандартов. Устройства, не соблюдающие эти стандарты, в лучшем случае будут отображать Web-страницы неправильно, в худшем — вообще не будут работать.

Из этого следует *первое правило* из перечисленных ранее — Web-дизайнеры при создании Web-страниц обязаны строго придерживаться современных интернет-стандартов, чтобы их творения одинаково (ну, или почти одинаково) отображались на всех устройствах.

Первое правило также требует отказа от устаревших и закрытых, фирменных интернет-технологий. С устаревшими технологиями все понятно: старье — не помощник новому. Закрытые же технологии неудобны тем, что зачастую контролируются единственной фирмой, которая единолично "заказывает музыку" и далеко не всегда прислушивается к мнению интернет-сообщества. К таким технологиям относится, в частности, до сих пор популярная, но постепенно сдающая позиции Adobe Flash.

Открытыми интернет-стандартами, в том числе и Web-стандартами, занимается организация World Wide Web Consortium (Консорциум Всемирной паутины), или сокращенно W3C. Она разрабатывает стандарты, согласует их с требованиями участников рынка и публикует на своем Web-сайте **http://www.w3.org**. Все опубликованные там стандарты обязательны к применению всеми разработчиками устройств, интернет-программ и Web-страниц.

Интернет когда-то начинался как сеть ученых, которым было нужно обмениваться результатами исследований. А что представляли собой эти результаты? В основном, текст, возможно, с иллюстрациями. Ученые — публика в этом смысле невзыскательная, им вполне хватало скромных возможностей тогдашнего WWW.

Теперь же абсолютное большинство пользователей Интернета — обычные обыватели. Им мало простого текста с парой картинок, им подавай хорошо оформленный текст, музыку и видео. Они требовательнее первых обитателей Сети.

Отсюда вытекает *второе правило* — Web-дизайнеры должны заботиться о полноте и удобстве наполнения страниц.

- Структура Web-страниц должна быть хорошо продумана, чтобы посетитель сразу смог найти на них все, что ему нужно.
- Web-страницы должны легко читаться и не "резать" глаза.
- К важным материалам желательно привлечь внимание посетителя, а маловажные скрыть. В этом могут помочь динамические элементы: раскрывающиеся при щелчке мышью абзацы (спойлеры), гиперссылки, выделяющиеся при наведении курсора мыши, раскрывающиеся меню и пр.

- Если сайт посвящен музыке или видео, все это должно быть доступно для воспроизведения прямо на его страницах, без загрузки как собственно аудио- или видеофайла, так и какой-либо дополнительной программы.
- Одним словом — все для удобства посетителя! (Пожалуй, это правило следовало бы поставить в начале списка...)

Интернет грозится прийти на самые разные устройства с различными характеристиками: быстродействием процессора, объемом памяти, разрешением экрана, скоростью доступа к Сети. Но все они должны обеспечивать единообразный вывод Web-страниц. Как этого достигнуть?

Вот и *третье правило* — Web-дизайнеры должны заботиться о доступности страниц.

- Web-страницы следует делать как можно более компактными. Чем компактнее файл, тем быстрее он загружается по сети — это аксиома.
- Web-страницы не должны быть чересчур сложными. Чем сложнее страница, тем больше времени и системных ресурсов требует ее обработка и вывод.
- Web-страницы не должны требовать для отображения никакого дополнительного программного обеспечения. В идеале для их вывода должно быть достаточно только Web-обозревателя.
- Наконец, Web-страницы должны автоматически адаптироваться под устройства с различными параметрами экрана.

Но как эти правила реализуются на практике? Давайте откроем какой-нибудь современный Web-сайт, например, принадлежащий организации W3C (рис. 1.1). Как мы помним, его можно найти по интернет-адресу **http://www.w3.org**.

Что же мы здесь видим?

- Web-сайт создан с учетом всех современных интернет-стандартов. Он отображается во всех Web-обозревателях практически одинаково.
- Web-сайт не использует ни устаревших, ни закрытых интернет-технологий.
- Структура страниц исключительно ясна — мы можем без проблем найти все, что нужно. Слева находится набор гиперссылок, ведущих на другие страницы сайта (панель навигации), посередине — список новостей и гиперссылки на избранные статьи, справа — гиперссылки на дополнительные материалы.
- Web-страница прекрасно читается. Тонкий шрифт без засечек, спокойная серо-голубая цветовая гамма, тонкие рамочки, минимум графики — ничто не бросается в глаза.
- Web-страница быстро загружается и мгновенно выводится на экран.
- Web-страница ничего не требует для своего вывода, кроме Web-обозревателя.

Налицо и соблюдение стандартов, и наполнение, и доступность. Три из трех!

Именно такие страницы мы и будем учиться создавать в данной книге.

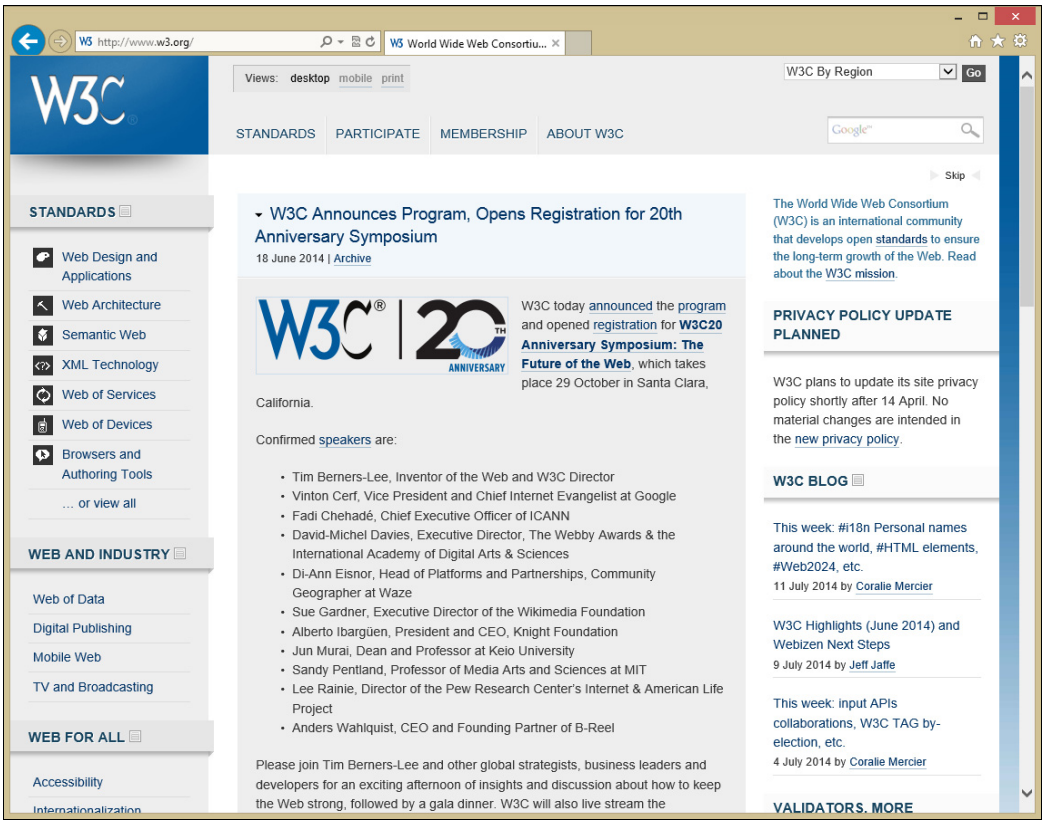


Рис. 1.1. Главная страница Web-сайта организации W3C

Концепция Web 2.0

Давайте еще раз обратимся к рассмотренным ранее правилам и немного расширим их.

- При создании Web-страниц следует придерживаться современных интернет-стандартов. При этом нужно полностью отказаться от устаревших и закрытых интернет-технологий, как не укладывающихся в современную парадигму Web-дизайна и зачастую не поддерживаемых всеми Web-обозревателями.
- Особое внимание нужно обратить на структуру и наполнение Web-страниц. Структура страниц должна быть максимально простой, а наполнение — достаточно богатым, чтобы посетитель быстро нашел нужную ему информацию. Кроме того, необходимо создавать страницы так, чтобы дизайн не мешал восприятию информации.
- Web-страницы обязательно следует делать максимально доступными на любых устройствах. Страницы должны быстро загружаться и выводиться на экран. Страницы должны нормально отображаться на экранах с различными характеристиками. Наконец, страницы не должны требовать для отображения никакого дополнительного программного обеспечения.

Фактически здесь мы привели постулаты так называемой концепции *Web 2.0*. Это список правил, которым должен удовлетворять любой Web-сайт, претендующий на звание современного. Образно выражаясь, это флаг, который совместно несут труженики Web-индустрии, шагая в ногу со временем.

Также концепция Web 2.0 предусматривает четыре принципа, являющиеся "передним краем" Web-дизайна. Давайте их рассмотрим.

Принцип первый — разделение содержимого, представления и поведения Web-страницы. Здесь *содержимое* — это информация, которая выводится на Web-странице, *представление* описывает формат вывода этой информации, а *поведение* — реакцию Web-страницы или отдельных ее элементов на действия посетителя. Благодаря их разделению мы сможем править, скажем, содержимое, не затрагивая представление и поведение, или поручать создание содержимого, представления и поведения разным людям.

Принцип второй — *подгружаемое содержимое*. Вместо того чтобы обновлять всю Web-страницу в ответ на щелчок на гиперссылке, мы можем подгружать только ее часть, содержащую необходимую информацию. Это заметно: время загрузки страницы и объем передаваемой по сети информации (сетевой трафик) позволят выполнять какие-либо действия с данными после их подгрузки.

Принцип третий — *адаптируемое содержимое*. Страницы должны сами подстраиваться под экраны с различными параметрами. Так, на обычном компьютерном мониторе может выводиться полная версия страницы, а на экране телефона — ее упрощенная версия, на которой некоторые маловажные элементы скрыты.

Принцип четвертый — *семантическая разметка* данных. С ее помощью мы можем дать определенным фрагментам страницы какое-либо особое значение. Например, мы можем пометить часть страницы — рисунок и подпись к нему — как иллюстрацию, и тогда Web-обозреватель сможет выделить ее определенным образом, основываясь на заданном нами представлении.

В качестве примеров можно привести два сайта. Во-первых, сайт популярной социальной сети "ВКонтакте" (**http://vk.com/**). Вы, вероятно, замечали, что, стоит прокрутить страницу вниз, как на ней появляются более ранние записи — это работает механизм подгрузки данных, полученных от Web-приложения (о которых мы поговорим в *главе 24*). Во-вторых, это сайт интернет-энциклопедии Википедия (**http://ru.wikipedia.org/**) — его страницы успешно подстраиваются под самые разные экраны.

Очень многие сайты в настоящее время следуют концепции Web 2.0, выполняя если не все ее принципы, то бóльшую их часть. И неудивительно, ведь это позволяет радикально сократить время разработки, повысить скорость загрузки, дать страницам дополнительную функциональность и без проблем адаптировать их для отображения на мобильных устройствах.

Да-да, Web 2.0 — это не самоцель, а всего лишь средство достижения задач, поставленных перед Web-дизайнерами и перечисленных в самом начале этой главы. Хотите разрабатывать современные, "легкие" и доступные для всех устройств сайты? Следуйте данной концепции. И читайте эту книгу!

Интернет: как все это работает

Давайте еще раз посмотрим на Web-сайт организации W3C. И зададимся вопросом, вынесенным в заголовок данного раздела.

Как все это работает? Откуда Web-обозреватель получает нужную Web-страницу? Кто отвечает за работу сложного механизма под названием Всемирная паутина?

Клиенты и серверы Интернета. Интернет-адреса

Возьмем для примера главную страницу сайта, который мы открыли. Она должна где-то храниться. Но где? На диске другого компьютера, подключенного к сети (в данном случае — к сети Интернет). Этот компьютер может принадлежать как автору Web-сайта, так и сторонней организации, предоставляющей доступ в Интернет *(интернет-провайдеру)* или площадку для публикации сайтов клиентов *(хостинг-провайдеру)*. И хранится она в виде файла или набора файлов, таких же, какие в изобилии "водятся" на нашем собственном компьютере.

Но как мы смогли получить и просмотреть содержимое этого файла? Во-первых, посредством самой сети — она связала компьютер, хранящий файл, с нашим. Во-вторых, с помощью особых программ, которые, собственно, и выполнили передачу файла. Эти программы делятся на две группы.

Программы первой группы взаимодействуют непосредственно с пользователем: принимают от него запросы на информацию, которая хранится где-то в сети, получают ее, выводят на экран и, возможно, позволяют ее править и отправлять обратно. Такие программы называют *клиентами*.

Для просмотра страниц мы пользуемся Web-обозревателем. Это программа-клиент; она принимает от нас интернет-адреса страниц, получает файлы, хранящие их содержимое, и выводит это содержимое на экран. Программа почтового клиента позволяет как извлекать из почтового ящика полученные письма, так и создавать новые. Существуют также клиенты чата, систем мгновенных сообщений и пр.

Но клиенты не имеют прямого доступа к хранящейся на других компьютерах информации. Они не могут просто "залезть" на жесткий диск удаленного компьютера и прочитать оттуда файл. Так сделано из соображений безопасности. Вместо этого они отправляют запросы программам второй группы — серверам.

Серверы работают на компьютерах, хранящих информацию, которая должна быть доступна в сети. Они принимают запросы от клиентов, извлекают требуемую информацию из файлов и отправляют им. Также они могут получать введенную пользователями информацию от клиентов и сохранять их в файлах, при этом, возможно, как-то обработав. Можно сказать, что серверы выступают посредниками между клиентами и запрашиваемой ими информацией.

Для управления Web-сайтами используются *Web-серверы*, которые принимают запросы от клиентов и отправляют им содержимое требуемых файлов. Для управления почтовыми службами применяются серверы электронной почты; они сохраняют пришедшие письма в файлах, выдают их почтовым клиентам по запросу, при-

нимают от клиентов новые сообщения и отправляют их по указанному адресу — в общем, работают как своего рода почтовое отделение. Службы чатов и мгновенных сообщений также имеют свои серверы.

Клиенты — лицо Интернета. Серверы — его сердце.

Но как указать, какая информация и с какого сервера нам требуется? С помощью определенным образом составленного интернет-адреса.

Каждая единица информации — файл, ящик электронной почты, канал чата, — доступная в сети, однозначно идентифицируется интернет-адресом, который представляет собой строку из букв, цифр и некоторых других символов. Этот интернет-адрес включает в себя две части:

- интернет-адрес программы-сервера, работающей на компьютере;
- указатель на нужную единицу информации, например, путь к файлу, имя ящика электронной почты, имя канала чата и др. (может отсутствовать).

Рассмотрим несколько примеров интернет-адресов.

В интернет-адресе **http://www.somesite.ru/folder1/file1.htm** присутствуют обе части. Здесь **http://www.somesite.ru** — интернет-адрес программы-сервера (или, как еще говорят, *имя хоста*), в данном случае — Web-сервера, а **/folder1/file1.htm** — путь к запрашиваемому файлу.

В интернет-адресе **http://www.othersite.ru** присутствует только интернет-адрес Web-сервера. Какая информация в этом случае будет отправлена клиенту (Web-обозревателю), мы узнаем потом.

А в интернет-адресе **user@mail.someserver.ru** мы видим интернет-адрес сервера электронной почты (**mail.someserver.ru**) и имя почтового ящика (**user**).

Разговор об интернет-адресах еще не закончен. Мы вернемся к нему в *главе 7*, когда будем рассматривать средства навигации по Web-сайту, в частности, гиперссылки. А пока что давайте подробнее поговорим о Web-серверах и их нелегкой "работе".

Web-сайты и Web-серверы

Как мы только что выяснили, все интернет-программы делятся на клиенты и серверы. Клиенты работают на стороне пользователя, получают от них интернет-адреса и выводят им полученную с этих адресов информацию. Серверы принимают запросы от клиентов, находят запрашиваемую ими информацию на дисках серверных компьютеров и отправляют ее клиентам.

Во Всемирной паутине WWW в качестве клиентов используются Web-обозреватели, а в качестве серверов — Web-серверы. Это мы тоже знаем.

Любая информация на дисках компьютера хранится в файлах. Ну, это знает любой более-менее подкованный пользователь...

Web-страницы также хранятся в файлах с расширением htm или html (или, с учетом описанных во введении типографских соглашений, htm[l]). Одна Web-страница занимает один или более файлов.

А теперь — внимание! Мы рассмотрим некоторые "интимные" подробности работы Web-серверов, которые знает не каждый интернетчик.

Прежде всего, для хранения всех файлов, составляющих Web-сайт, на диске серверного компьютера выделяется особая папка, называемая *корневой папкой Web-сайта*. Путь к этой папке указывается в настройках Web-сервера, чтобы он смог ее "найти".

Все, повторим — все файлы, составляющие Web-сайт, должны храниться в корневой папке или в папках, вложенных в нее. Файлы, расположенные вне корневой папки, с точки зрения Web-сервера не существуют. Это сделано для безопасности, чтобы злоумышленник не смог получить доступ к дискам серверного компьютера.

Когда в интернет-адресе указывается путь к запрашиваемому файлу, Web-сервер отсчитывает его относительно корневой папки. Это проще всего показать на примерах.

- **http://www.somesite.ru/page1.htm** — в ответ будет отправлен файл page1.htm, хранящийся в корневой папке сайта.
- **http://www.somesite.ru/chapter2/page6.htm** — в ответ будет отправлен файл page6.htm, хранящийся в папке chapter2, которая вложена в корневую папку сайта.
- **http://www.somesite.ru/downloads/others/archive.zip** — в ответ будет отправлен файл archive.zip, хранящийся в папке others, вложенной в папку downloads, которая, в свою очередь, вложена в корневую папку сайта.

Но ведь мы нечасто набираем интернет-адрес, включающий путь к запрашиваемому файлу. Гораздо чаще интернет-адреса включают только адрес программы-сервера, например, **http://www.somesite.ru**. Что в таком случае делает Web-сервер? Какой файл он отправляет в ответ?

Специально для этого предусмотрены так называемые *Web-страницы по умолчанию*. Такая Web-страница выдается клиенту, если он указал в интернет-адресе только путь к файлу, но не его имя. Обычно файл Web-страницы по умолчанию имеет имя default.htm[l] или index.htm[l], хотя его можно изменить в настройках Web-сервера.

Так, если мы наберем интернет-адрес **http://www.somesite.ru**, Web-сервер вернет нам файл страницы по умолчанию, хранящийся в корневой папке сайта. Практически всегда это будет *главная Web-страница* — та, с которой начинается "путешествие" по сайту.

Мы можем набрать и интернет-адрес вида **http://www.somesite.ru/chapter2/**. Тогда Web-сервер отправит нам файл страницы по умолчанию, хранящийся в папке chapter2, вложенной в корневую папку сайта.

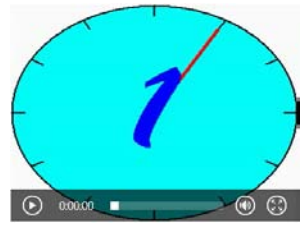
> ***Внимание!***
>
> Если Web-сервер не обнаружит указанный файл страницы, он вернет Web-обозревателю сообщение об ошибке с кодом 404 — "запрошенная страница отсутствует". Такое сообщение говорит о проблемах в работе сайта, и эта проблема должна быть устранена.

Что дальше?

В этой главе мы познакомились с современными веяниями в Web-дизайне, узнали о концепции Web 2.0 и поняли, как работает Интернет в целом и WWW в частности. В общем, узнали самые основы.

Теперь на какое-то время мы оставим в покое принципы и концепции и сосредоточимся на практике, а именно на языке HTML. В следующей главе мы рассмотрим основные его понятия: теги, их атрибуты и вложенность и структурирование страниц.

ГЛАВА 2



Введение в язык HTML5

В предыдущей главе мы ознакомились с современными тенденциями Web-дизайна, концепцией Web 2.0 и принципами работы Интернета. Все эти тенденции, концепции и принципы не отняли у нас много времени...

...Чего не скажешь о языке разметки страниц HTML. Ему будет посвящена вся первая часть книги.

Основные принципы HTML

Язык *HTML* (HyperText Markup Language, язык гипертекстовой разметки) предназначен для описания содержимого Web-страниц, т. е. той информации, которая, собственно, будет выводиться на экран: абзацев, заголовков, списков, таблиц, графических изображений, гиперссылок, аудио и видео.

Первая версия HTML была представлена еще в 1992 году. В настоящее время в разработке находится пятая версия этого языка — HTML5, — и, хоть окончательная ее спецификация до сих пор не выпущена, все современные Web-обозреватели уже поддерживают, по крайней мере, большинство предлагаемых ею возможностей. Эту версию мы и будем использовать для написания наших страничек.

> ***Внимание!***
>
> Поддержка HTML5 появилась в Microsoft Internet Explorer, начиная с версии 9, в Mozilla Firefox — начиная с версии 3.5, в Opera — с версии 10, в Apple Safari — с версии 4, в Google Chrome — с самых первых версий.

Язык HTML5 довольно прост и основан на нескольких совсем не сложных принципах. Сейчас мы в этом убедимся.

Первая Web-страница

Изучать HTML лучше всего на примере. Так что давайте сразу же создадим нашу первую Web-страничку. Сделать это можно в любом простейшем текстовом редакторе, например, входящем в комплект поставки Windows Блокноте.

Откроем Блокнот и наберем в нем текст (или, как говорят бывалые программисты, код), приведенный в листинге 2.1.

Листинг 2.1

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Пример Web-страницы</title>
  </head>
  <body>
    <h1>Изучаем язык HTML</h1>
    <p>Язык <strong>HTML</strong> предназначен для создания содержимого
    Web-страниц.</p>
  </body>
</html>
```

Проверим набранный код на ошибки и сохраним в файл с именем 2.1.html. Только сделаем при этом две важные вещи.

1. Сохраним HTML-код в кодировке UTF-8. Для этого в диалоговом окне сохранения файла Блокнота найдем раскрывающийся список **Кодировка** и выберем в нем пункт **UTF-8**.
2. Заключим имя файла в кавычки. Иначе Блокнот добавит к нему расширение txt, и наш файл получит имя 2.1.html.txt.

Все, наша первая Web-страница готова! Теперь осталось открыть ее в любом Web-обозревателе, поддерживающем язык HTML5 (автор применил Internet Explorer 10), и посмотреть на результат (рис. 2.1).

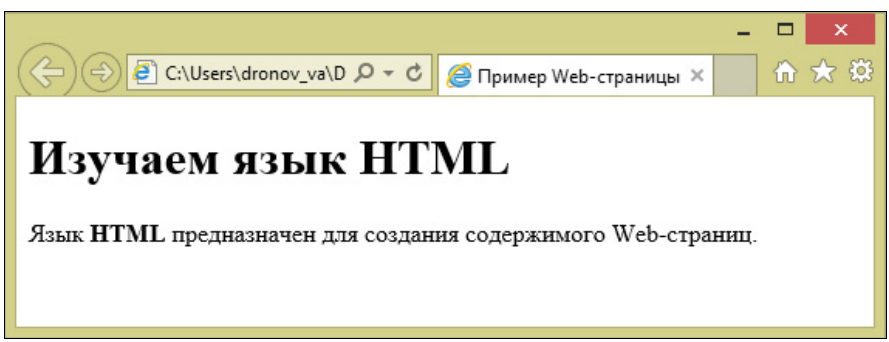


Рис. 2.1. Наша первая Web-страница

Видите? Мы создали Web-страницу, содержащую большой "кричащий" заголовок, абзац текста, который включает в свой состав фрагмент текста, выделенный полужирным шрифтом (аббревиатура "HTML"). И все это — в "голом" тексте, набранном в Блокноте!

Теги и атрибуты тегов

Теперь посмотрим, что же мы такое написали в файле 1.1.htm. Пока что ограничимся следующим небольшим фрагментом HTML-кода:

```
<h1>Изучаем язык HTML</h1>
<p>Язык <strong>HTML</strong> предназначен для создания содержимого
Web-страниц.</p>
```

Здесь мы видим текст заголовка и абзаца. И еще странные слова, взятые в угловые скобки — символы < и >. Что это такое?

Это *теги* HTML, особые команды, задающие назначение того или иного фрагмента содержимого — будет он абзацем, заголовком или важным текстом, который следует выделить полужирным шрифтом.

Начнем с тегов <h1> и </h1>, поскольку они идут первыми. Эти теги превращают фрагмент текста, находящийся между ними, в заголовок. Тег <h1> помечает начало фрагмента, на который распространяется действие тега, и называется *открывающим*. А тег </h1> устанавливает конец "охватываемого" фрагмента и называется *закрывающим*. Что касается самого фрагмента, заключенного между открывающим и закрывающим тегами, то он называется *содержимым тега*. Именно к содержимому применяется действие тега.

Все теги HTML представляют собой символы < и >, внутри которых находится *имя тега*, определяющее назначение тега. (Имена тегов можно набирать как строчными, так и прописными буквами, но в HTML5 обычно используются строчные буквы.) Закрывающий тег должен иметь то же имя, что и открывающий; единственное отличие закрывающего тега — символ /, который ставится между символом < и именем тега.

Рассмотренные нами теги <h1> и </h1> в HTML фактически считаются одним тегом <h1>. Такой тег называется *парным*.

Другой парный тег — <p> — создает на Web-странице абзац из текста, являющегося содержимым этого тега. Такой абзац будет отображаться с отступами сверху и снизу. Если он полностью помещается по ширине в окне Web-обозревателя, то отобразится в одну строку; в противном случае сам Web-обозреватель разобьет его на несколько более коротких строк. (Это же справедливо и для заголовка.)

Третий парный тег — — помечает свое содержимое как важный текст, на который следует обратить особое внимание и который выделяется полужирным шрифтом. Как мы видим, этот тег вложен внутрь содержимого тега <p>. Это значит, что содержимое тега будет отображаться как часть абзаца (тега <p>).

А теперь рассмотрим вот такой тег:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Это *одинарный тег*, не имеющий закрывающей пары. Такие теги действуют в той точке HTML-кода, где они находятся, и либо задают какие-либо сведения о самой странице (как только что рассмотренный нами тег <meta>), либо помещают в соответствующее место страницы какой-либо элемент, не относящийся к тексту.

А еще мы видим, что в данном теге между символами < и >, помимо имени тега, присутствуют еще какие-то данные. Это *атрибуты тега*, задающие его параметры. В частности, атрибуты `http-equiv` и `content` тега `<meta>` указывают тип документа и его кодировку.

Каждый атрибут тега имеет *имя*, за которым ставится знак равенства и *значение* данного атрибута, взятое в двойные кавычки. Так, атрибут с именем `http-equiv` имеет значение `"Content-Type"`, указывающее, что данный тег задает тип документа. А атрибут с именем `content` имеет значение `"text/html; charset=utf-8"`, обозначает тип документа "Web-страница" и указывает, что он набран в кодировке UTF-8.

Атрибуты тегов бывают обязательными и необязательными. *Обязательные* атрибуты должны присутствовать в теге в обязательном порядке. *Необязательные* же атрибуты могут быть опущены; в таком случае тег ведет себя так, будто соответствующему атрибуту присвоено значение по умолчанию. Атрибуты `http-equiv` и `content` тега `<meta>` являются обязательными.

Вложенность тегов

Если мы снова посмотрим на приведенный в листинге 2.1 HTML-код, то заметим, что одни теги вложены в другие. Так, тег `<strong>` вложен в тег `<p>`, являясь частью его содержимого. Тег `<p>`, в свою очередь, вложен в тег `<body>`, а тот — в тег `<html>`. (Теги `<body>` и `<html>` мы рассмотрим чуть позже.) Такая *вложенность тегов* в HTML — обычное явление.

Когда Web-обозреватель встречает тег, вложенный в другой тег, он как бы накладывает действие "внутреннего" тега на эффект "внешнего". Так, действие тега `<strong>` будет наложено на действие тега `<p>`, и фрагмент абзаца окажется выделенным полужирным шрифтом, при этом оставаясь частью этого абзаца.

Теперь — внимание! Порядок следования закрывающих тегов должен быть обратным тому, в котором следуют теги открывающие. Говоря иначе, теги со всем их содержимым должны полностью вкладываться в другие теги, не оставляя "хвостов" снаружи.

Осталось выучить несколько новых терминов. Тег, в который непосредственно вложен данный тег, называется *родительским*, или *родителем*. В свою очередь, тег, вложенный в данный тег, называется *дочерним*, или *потомком*. Так, для тега `<p>` в приведенном ранее листинге тег `<body>` — родительский, а тег `<strong>` — дочерний. Любой тег может иметь сколько угодно дочерних тегов, но только один родительский (что, впрочем, понятно — не может же он быть непосредственно вложен одновременно в два тега).

Аналогично, элемент Web-страницы, в который вложен элемент, создаваемый данным тегом, называется *родительским*, или *родителем*. А элемент Web-страницы, который вложен в данный элемент, — *дочерним*, или *потомком*.

Уровень вложенности того или иного тега показывает количество тегов, в которые он последовательно вложен. Так, если принять за точку отсчета тег `<html>`, то тег

<body> будет иметь первый уровень вложенности, т. к. он вложен непосредственно в тег <html>. Тег <p> же будет иметь второй уровень вложенности, т. к. он вложен в тег <body>, а тот, в свою очередь, — в тег <html>. В сложных же Web-страницах уровень вложенности иных тегов может составлять несколько десятков.

Форматирование Web-страницы

А теперь рассмотрим несколько важных моментов, касающихся структуры HTML-кода страницы и сведений о ней, что необходимы Web-обозревателю для успешного вывода ее содержимого.

Секции Web-страницы

Прежде всего, содержимое любой страницы разделяется на две *секции*. Для этого применяются особые теги, которые часто называют *невидимыми*, поскольку они никак не отображаются на экране, по крайней мере, напрямую.

Давайте мысленно удалим из листинга 2.1 все содержимое, за исключением этих тегов, в результате чего получим листинг 2.2.

Листинг 2.2

```
<html>
  <head>
    . . .
  </head>
  <body>
    . . .
  </body>
</html>
```

Начнем с парного тега <body>, о котором уже упоминали ранее. Он формирует *секцию тела* страницы, которая описывает собственно содержимое страницы, выводящееся на экран. Все теги, что формируют содержимое, должны находиться в этой секции:

```
<body>
  <h1>Изучаем язык HTML</h1>
  <p>Язык <strong>HTML</strong> предназначен для создания содержимого
  Web-страниц.</p>
</body>
```

А в парном теге <head> находится *секция заголовка* Web-страницы. (Не путать с заголовком, который создается с помощью тега <h1>!) В эту секцию помещаются сведения о параметрах Web-страницы, не отображаемые на экране и предназначенные исключительно для Web-обозревателя (например, знакомый нам тег <meta>):

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Пример Web-страницы</title>
</head>
```

И заголовок, и тело Web-страницы находятся внутри парного тега <html>. Этот тег расположен на самом высшем (нулевом) уровне вложенности и не имеет родителя.

Любая Web-страница должна быть правильно отформатирована: иметь секции заголовка и тела и необходимый набор метатегов (о которых мы также поговорим). Только в этом случае она будет считаться корректной с точки зрения стандартов HTML.

Метаданные и метатеги

Мы только что узнали о секции заголовка Web-страницы и о том, что она предназначена для задания параметров страницы. Эти параметры называются *метаданными*, т. е. данными, описывающими другие данные. А для их задания применяются *метатеги*, относящиеся к числу невидимых тегов.

Прежде всего, в метаданные входит *название* Web-страницы. Оно отображается в заголовке окна Web-обозревателя, где выводится содержимое данной Web-страницы, и хранится в "истории" (списке посещенных к настоящему времени Web-страниц). Название помещается в парный тег <title> и располагается в секции заголовка Web-страницы:

```
<head>
  . . .
  <title>Пример Web-страницы</title>
</head>
```

Далее, это уже знакомый нам тег <meta>, задающий тип документа и кодировку текста, которым он набран.

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  . . .
</head>
```

Приведенный тег указывает, что данный документ представляет собой Web-страницу, и задает для него кодировку текста UTF-8.

> ***На заметку***
>
> Кодировка *UTF-8* — это разновидность кодировки Unicode, предназначенная для Web-дизайна. Кодировка Unicode (а значит, и UTF-8) может закодировать все символы всех языков, имеющихся на Земле.

> ***Внимание!***
>
> Для кодирования текста страниц, написанных на языке HTML5, следует применять только кодировку UTF-8. Использование других кодировок не допускается.

Теперь осталось рассмотреть последний тег, находящийся в самом начале HTML-кода нашей Web-страницы. Этот тег находится даже вне "всеобъемлющего" тега

<html>. Это метатег <!doctype>, который задает, во-первых, версию языка HTML, на которой написана Web-страница, а во-вторых, разновидность данной версии.

В нашем случае тег <!doctype> выглядит так:

```
<!doctype html>
```

Он указывает, что для создания страницы применяется версия 5 языка HTML.

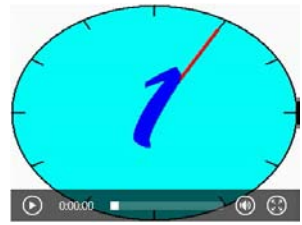
> ***Внимание!***
> Код любой Web-страницы, написанной на HTML5, должен включать метатег <!doctype html>, расположенный в самом его начале. В противном случае Web-обозреватель не сможет правильно обработать страницу.

Что дальше?

В этой главе мы начали изучать язык HTML. Мы познакомились с основными принципами и понятиями этого языка: тегами, атрибутами тегов, вложенностью, секциями страницы и метатегами. Без этих знаний мы просто не поймем, о чем же идет речь в последующих главах.

В очередной главе мы рассмотрим средства структурирования текста: абзацы, заголовки, списки, цитаты и теги семантической разметки. Так сказать, научимся делить текст на "куски".

ГЛАВА 3



Структурирование текста

В предыдущей главе мы изучили основные понятия и принципы языка HTML: теги, атрибуты тегов, вложенность, секции Web-страницы и ее метаданные. А уж узнав все это, мы можем продолжить знакомство с этим примечательным языком.

В этой главе мы рассмотрим средства HTML, предназначенные для структурирования текста — разбиения его на отдельные значащие фрагменты, имеющие разное назначение и несущие различный смысл. Такими фрагментами являются абзацы, заголовки, списки, цитаты и теги семантической разметки.

Простейшие средства структурирования текста

И начнем мы с самых простых и самых "старых", появившихся еще в первых версиях HTML-средств.

Абзацы и заголовки

Из чего состоит текст? Правильно, из отдельных абзацев, включающих логически законченные и относительно независимые фрагменты текста, и заголовков, дающих названия отдельным частям текста: разделам, главам и параграфам.

Как мы уже знаем из *главы 2*, для создания абзаца применяется парный тег `<p>`. Содержимое этого тега становится текстом абзаца:

```
<p>Я - совсем короткий абзац.</p>
<p>А я - уже более длинный абзац. Возможно, Web-обозреватель разобьет
меня на две строки.</p>
```

Теперь о заголовках. Скажем сразу, что в HTML есть такое понятие, как *уровень заголовка*. Он представляет собой целое число от 1 до 6, указывающее, насколько крупную часть текста открывает данный заголовок.

- Заголовок первого уровня (1) открывает самую крупную часть текста. Как правило, это заголовок всей Web-страницы. Web-обозреватель выводит заголовок первого уровня самым большим шрифтом.

- Заголовок второго уровня (2) открывает более мелкую часть текста. Обычно это большой раздел. Web-обозреватель выводит заголовок второго уровня несколько меньшим шрифтом, чем заголовок первого уровня.
- Заголовок третьего уровня (3) открывает еще более мелкую часть текста; обычно главу в разделе. Web-обозреватель выводит такой заголовок еще меньшим шрифтом.
- Заголовки четвертого, пятого и шестого уровней (4–6) открывают отдельные параграфы, крупные, более мелкие и самые мелкие соответственно. Web-обозреватель выводит заголовки четвертого уровня еще меньшим шрифтом, а шрифт заголовков пятого и шестого уровней — даже меньше, что шрифт, которым выводятся обычные абзацы.

Чтобы выделить заголовки, даже пятого и шестого уровней, Web-обозреватель выводит их полужирным шрифтом.

Если мы откроем нашу первую Web-страницу, сохраненную в файле 2.1.html, в Блокноте, заменим в ней содержимое секции тела (тега `<body>`) на код, приведенный в листинге 3.1, сохраним под именем 3.1.html и откроем в Web-обозревателе, мы увидим то, что показано на рис. 3.1. Так мы можем примерно оценить размер шрифта, которым выводятся заголовки различных уровней.

Листинг 3.1

```
<h1>Заголовок первого уровня</h1>
<h2>Заголовок второго уровня</h2>
<h3>Заголовок третьего уровня</h3>
<h4>Заголовок четвертого уровня</h4>
<h5>Заголовок пятого уровня</h5>
<h6>Заголовок шестого уровня</h6>
<p>Абзац</p>
```

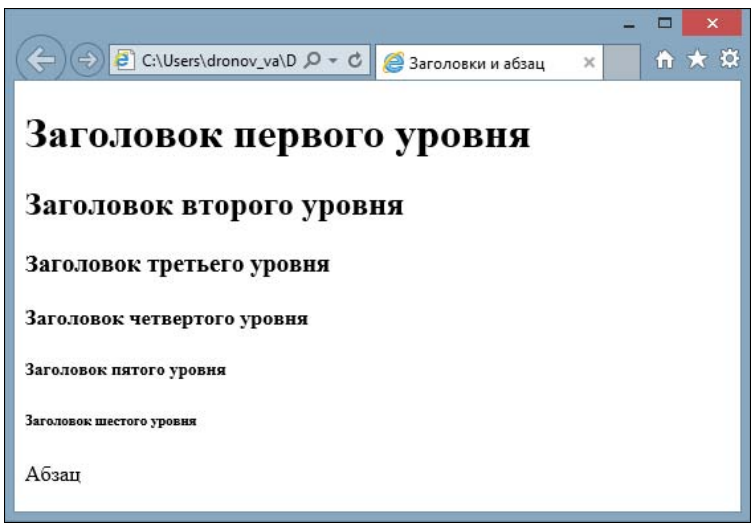


Рис. 3.1. Заголовки различных уровней и абзац

Блочные элементы HTML

А теперь рассмотрим один важный вопрос. Он касается типов элементов страниц, поддерживаемых HTML, и правил, согласно которым они выводятся на экран.

Уясним сразу, что абзацы и заголовки — это так называемые *блочные элементы*. Такие элементы:

- являются независимыми от остальных блочных элементов того же уровня вложенности (имеющих того же родителя);
- отделяются небольшими отступами от соседних блочных элементов;
- занимают все свободное пространство по ширине.

В случае необходимости Web-обозреватель сам выполняет перенос текста — содержимого блочных элементов.

И, если уж зашла речь о тексте, давайте перечислим правила, согласно которым выполняется его вывод:

- два и более следующих друг за другом пробела считаются за один пробел;
- перевод строки считается за пробел;
- пробелы и переводы строки между тегами, создающие блочные элементы, никак не отображаются на Web-странице. (Благодаря этому мы можем форматировать HTML-код для более удобного чтения, в том числе, ставить отступы для обозначения вложенности тегов.)

Все теги, что мы изучим в этой главе, создают блочные элементы. А в следующей главе мы познакомимся с другой разновидностью элементов Web-страниц — встроенными. А пока что не будем забегать вперед.

Списки

Списки используются для того, чтобы представить читателю перечень каких-либо позиций, пронумерованных или непронумерованных, — пунктов списка. Список с пронумерованными пунктами так и называется — *нумерованным*, а с непронумерованными — *маркированным*. В маркированных списках пункты помечаются особым значком — *маркером*, который ставится левее пункта списка.

Маркированные списки обычно служат для простого перечисления каких-либо позиций, порядок следования которых не важен. Если же необходимо обратить внимание читателя на то, что позиции должны следовать друг за другом именно в том порядке, в котором они перечислены, следует применить нумерованный список.

Любой список в HTML создается в два этапа. Сначала пишут строки, которые станут пунктами списка и каждую из этих строк помещают внутрь парного тега `<li>`. Затем все эти пункты помещают внутрь парного тега `<ul>` (если создается маркированный список) или `<ol>` (в случае создания нумерованного списка) — эти теги определяют сам список (листинг 3.2).

Листинг 3.2

```
<ul>
  <li>Я - первый пункт маркированного списка.</li>
  <li>Я - второй пункт маркированного списка.</li>
  <li>Я - третий пункт маркированного списка.</li>
</ul>
<ol>
  <li>Я - первый пункт нумерованного списка.</li>
  <li>Я - второй пункт нумерованного списка.</li>
  <li>Я - третий пункт нумерованного списка.</li>
</ol>
```

Web-обозреватель выводит список с отступом слева. Расстояние между пунктами списка он делает меньшими, чем в случае абзацев или заголовков. Также он сам расставляет необходимые маркеры или нумерацию. И списки, и их пункты относятся к блочным элементам страницы.

Списки можно помещать друг в друга, создавая *вложенные списки*. Делается это следующим образом. Сначала во "внешнем" списке (в который должен быть помещен вложенный) отыскивают пункт, после которого должен находиться вложенный список. Затем HTML-код, создающий вложенный список, помещают в разрыв между текстом этого пункта и его закрывающим тегом `</li>`. Если же вложенный список должен помещаться в начале "внешнего" списка, его следует вставить между открывающим тегом `<li>` первого пункта "внешнего" списка и его текстом. Что, впрочем, логично.

В листинге 3.3 представлен HTML-код, создающий два списка, один из которых вложен внутрь другого. Обратим внимание, где помещается HTML-код, создающий вложенный список.

Листинг 3.3

```
<ul>
  <li>Я - первый пункт внешнего списка.</li>
  <li>Я - второй пункт внешнего списка.
    <ul>
      <li>Я - первый пункт вложенного списка.</li>
      <li>Я - второй пункт вложенного списка.</li>
      <li>Я - третий пункт вложенного списка.</li>
    </ul>
  </li>
  <li>Я - третий пункт внешнего списка.</li>
</ul>
```

HTML позволяет вкладывать нумерованный список внутрь маркированного и наоборот. Количество последовательно вложенных друг в друга списков не ограничено.

Еще HTML позволяет создать так называемый *список определений*, представляющий собой перечень терминов и их разъяснений. Такой список создают с помощью парного тега <dl>. Внутри него помещают пары "термин — разъяснение", причем термины заключают в парный тег <dt>, а разъяснения — в парный тег <dd> (листинг 3.4).

Листинг 3.4

```
<dl>
  <dt>Содержимое</dt>
  <dd>Информация, отображаемая на Web-странице</dd>
  <dt>Представление</dt>
  <dd>Набор правил, определяющих формат отображения содержимого</dd>
  <dt>Поведение</dt>
  <dd>Набор правил, определяющих реакцию Web-страницы или ее элементов на
  действия посетителя</dd>
</dl>
```

Цитаты и адреса

HTML предлагает нам еще два полезных тега. Первый позволяет нам создать цитату, а второй — указать адрес, которым могут быть контактные данные разработчиков сайта, дату написания статьи или просто подпись ее автора. Оба этих тега являются блочными элементами.

Цитата создается парным тегом <blockquote>. В нем помещается HTML-код, собственно формирующий цитату (листинг 3.5).

Листинг 3.5

```
<blockquote>
  <p>Я - начало большой цитаты.</p>
  <p>Я - продолжение большой цитаты.</p>
</blockquote>
```

Как видим, в тег <blockquote> можно поместить несколько абзацев. Там также могут быть заголовки и списки (если уж возникнет такая потребность).

Цитата выводится на экран с отступом слева.

Что касается адреса, то он создается парным тегом <address>. Он ведет себя так же, как тег абзаца <p>, но его содержимое выводится курсивом:

```
<address>Я - адрес создателя данной Web-страницы: почтовый, электронный,
телефоны и факсы.</address>
```

Текст фиксированного формата

Ранее мы ознакомились с правилами, согласно которым Web-обозреватель выполняет вывод текста. Эти правила, в частности, гласят, что несколько стоящих подряд пробелов считаются за один пробел; также за пробел считаются переводы строк.

Однако часто бывает необходимо вывести какой-либо фрагмент текста как есть, со всеми множественными пробелами, стоящими подряд, и переводами строк. К таким фрагментам можно отнести, например, исходные тексты программ; такие тексты набираются согласно правилам, определяемым языком программирования, на которых они написаны, и, следовательно, выводиться они должны точно так же, как и набраны.

Специально для таких случаев HTML предусматривает парный тег `<pre>`. Внутри него помещается текст, который должен быть выведен так же, как он набран, или, как говорят Web-дизайнеры, текст *фиксированного формата*.

Опять откроем в Блокноте созданную нами ранее страницу 2.1.html, вставим в его секцию тела код из листинга 3.6, сохраним под именем 3.2.html и откроем в Web-обозревателе. В результате мы получим картину, показанную на рис. 3.2.

Листинг 3.6

```
<pre>Этот текст
  будет выведен
    на Web-страницу
      как есть,
        без всяких преобразований.</pre>
```

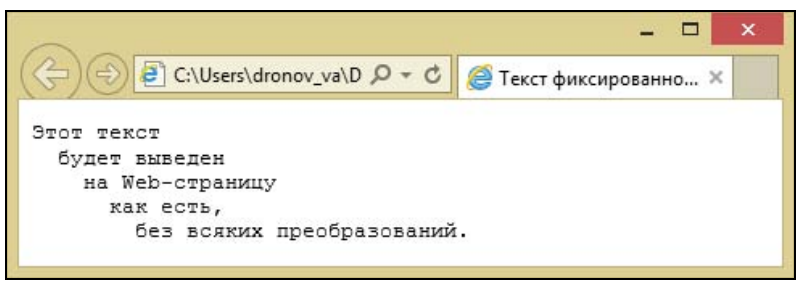


Рис. 3.2. Текст фиксированного формата

Правила отображения текста фиксированного формата:

- для вывода используется моноширинный, а не пропорциональный шрифт;
- все пробелы и переносы строк сохраняются при выводе (это мы уже знаем);
- если строка текста фиксированного формата не помещается в окне Web-обозревателя по ширине, она ни в коем случае не будет переноситься. Из-за этого может возникнуть потребность в горизонтальной прокрутке Web-страницы;

- допускается оформлять фрагмент текста фиксированного формата и создавать в нем гиперссылки, используя соответствующие теги (о которых будет рассказано в *главах 4* и *7*).

Текст фиксированного формата также является блочным элементом.

Блочные контейнеры

Иногда бывает необходимо объединить несколько блочных элементов — абзацев, заголовков, списков, тегов семантической разметки — в один элемент. Это может потребоваться при создании разметки страницы, интерактивного элемента или просто чтобы привязать к этому элементу какой-либо стиль.

Для таких случаев HTML предусматривает парный тег `<div>`. Внутри него помещается HTML-код, создающий содержимое, которое должно быть объединено в сущность, которая называется *блочным контейнером*, или *блоком*.

```
<div>
  <p>Я вхожу в состав блока.</p>
  <p>Я тоже вхожу в состав блока.</p>
</div>
```

Web-обозреватели при выводе никак не выделяют блочные контейнеры. Однако мы можем, как было сказано выше, привязать к ним стиль, который задаст для них нужное нам оформление (как это сделать, будет описано в *части II*).

Блоки очень пригодятся нам в дальнейшем, когда мы начнем работать со стилями, создавать разметку и интерактивные элементы страниц.

Семантическая разметка текста

Простейших инструментов для структурирования текста, предлагаемых HTML, хватает во многих случаях. Но если страница содержит большой текст, разбитый на множество частей, их может оказаться недостаточно. Поэтому HTML5 предоставляет в наше распоряжение набор новых тегов, выполняющих так называемую семантическую разметку текста.

Семантическая разметка заключается в разбиении текста на отдельные значащие блоки. Такими блоками могут быть сама статья, ее "шапка" или "поддон", отдельная часть статьи, примечание, иллюстрация в виде рисунка с подписью, набор гиперссылок для навигации и др.

Давайте перечислим все теги, используемые для семантической разметки, с указанием области их применения. Все они являются парными и создают блочные элементы.

- `<article>` — помечает независимый и самодостаточный фрагмент основного содержимого Web-страницы. Таким фрагментом может быть статья, отдельный фрагмент статьи, отдельная запись на форуме или в блоге или отдельный комментарий к ней.

- `<aside>` — помечает примечание к статье, обычно располагающееся сбоку от основного текста.

```
<article>
  ...Статья...
  . . .
  <aside>
    ...Примечание...
  <aside>
  . . .
</article>
```

  Может использоваться в разметке Web-страницы для формирования плавающих контейнеров, не относящихся к панели навигации. (О разметке страниц мы поговорим в *главе 13*.)

- `<figcaption>` — помечает подпись к иллюстрации. Может присутствовать только в теге `<figure>` *(см. далее)*.

- `<figure>` — помечает иллюстрацию к статье. Такой иллюстрацией может быть обычное графическое изображение, аудио-, видеоролик или даже фрагмент текста.

```
<article>
  ...Статья...
  . . .
  <figure>
    <img src="/images/screen1.png>
    <figcaption>
      <p>Скриншот 1</p>
    </figcaption>
  </figure>
  . . .
</article>
```

- `<footer>` — помечает "поддон" статьи, который может содержать приложения, предметный указатель, список сопутствующих материалов, сноски и т. п. Кроме того, при разметке часто используется для формирования "поддона" самой страницы.

- `<header>` — помечает "шапку" статьи, содержащую заголовок или группу заголовков, имя автора, дату публикации, содержание и пр. Помимо этого, часто используется для создания "шапки" самой страницы.

- `<mark>` — помечает важный фрагмент, на который следует обратить внимание посетителя. Таким фрагментом может быть, например, важное замечание в тексте статьи.

- `<nav>` — помечает набор гиперссылок, предназначенных для навигации по сайту, а также для создания панели навигации.

- `<section>` — помечает значимую часть материала, например, параграф большой статьи.

```
<article>
  ...Большая статья...
  <header>
    ...Заголовок...
    ...Имя автора...
    ...Оглавление...
  </header>
  <section>
    ...Первый параграф...
  </section>
  <section>
    ...Второй параграф...
  </section>
  . . .
  <footer>
    ...Предметный указатель...
    ...Список сопутствующих материалов...
  </footer>
</article>
```

На данный момент Web-обозреватели никак не выделяют визуально теги семантической разметки. Однако мы всегда сможем привязать к ним стили, чтобы задать нужное нам оформление.

Горизонтальные линии

Горизонтальная линия HTML иногда применяется для визуального отделения одного фрагмента текста от другого. Она создается с помощью одинарного тега `<hr>`:

```
<p>Я отделен от следующего абзаца горизонтальной линией.</p>
<hr>
<p>Я отделен от предыдущего абзаца горизонтальной линией.</p>
```

Горизонтальная линия HTML растягивается на все свободное пространство по ширине, имеет один-два пиксела в толщину и выпуклый или вдавленный вид (конкретные параметры зависят от Web-обозревателя). Она также является блочным элементом.

> ***НА ЗАМЕТКУ***
>
> В настоящее время применение горизонтальных линий HTML считается дурным тоном Web-дизайна. Вместо них рекомендуется применять возможности CSS по созданию рамок (подробнее — в *главе 14*).

Комментарии

Напоследок рассмотрим одну очень важную возможность HTML, которая, хоть и не касается напрямую Web-дизайна, но очень поможет забывчивым Web-дизайнерам.