5

ОСНОВЫ HTML

Встроенная в браузер JavaScript-консоль, которой мы до сих пор пользовались, хороша, когда нужно протестировать небольшой фрагмент кода, но для создания более масштабных программ понадобится чуть более гибкое и универсальное средство — вроде веб-страницы со встроенным JavaScript-кодом. В этой главе мы как раз и научимся создавать несложные странички на языке HTML.

Гипертекстовый язык разметки HTML предназначен специально для создания веб-страниц. Слово *гипертекстовый* означает, что фрагменты текста связаны между собой *гиперссылками* — то есть ссылками в документе на другие объекты. А *язык разметки* — это способ встраивать в текст дополнительную информацию. Разметка указывает программам (таким как браузер), как отображать текст и что с ним делать.

В этой главе я покажу, как создавать HTML-документы в *текстовом редакторе* — программе, предназначенной для работы с простым текстом без форматирования, в отличие от текстовых процессоров вроде Microsoft Word. Документы текстовых процессоров содержат *форматированный* текст (с различными типами и размерами шрифтов, цветами и т. п.), и устроены эти программы так, чтобы форматирование было легко менять. Кроме того, многие текстовые процессоры позволяют вставлять в текст картинки и другие графические элементы.

Простой же текст является только текстом — без цветов, стилей, размеров и т. д. Вставить в такой текст картинку не выйдет, разве только составить ее из символов — скажем, как этого котика справа.

```
 /\_/\
=( °W° )=
  )   (  //
 (__ __)//
```

Текстовые редакторы

Мы будем создавать HTML-документы в кросс-платформенном (совместимом с Windows, Mac OS и Linux) редакторе Sublime Text. Скачать Sublime Text можно бесплатно, однако спустя некоторое время вас попросят приобрести лицензию. На случай, если вам такой вариант не по нраву, я отобрал несколько полностью бесплатных альтернатив. Хотя в этой главе я буду ориентироваться на Sublime Text, работа с другими редакторами будет не сильно отличаться — благодаря относительной простоте текстовых редакторов как таковых.

- Gedit — кросс-платформенный текстовый редактор, часть проекта GNOME (https://wiki.gnome.org/Apps/Gedit/).
- Для Microsoft Windows хорошей альтернативой будет Notepad++ (http://notepad-plus-plus.org/).
- В Mac OS вы можете воспользоваться TextWrangler (http://www.barebones.com/products/textwrangler/).

Чтобы установить Sublime Text, зайдите на сайт http://www.sublimetext.com/. Инструкции по установке редактора отличаются для каждой из операционных систем, но написаны просто и понятно. В случае каких-либо проблем загляните в раздел Support («Поддержка») на сайте приложения.

ПОДСВЕТКА СИНТАКСИСА

Sublime Text будет отображать ваши программы в цвете — это называется *подсветкой синтаксиса*. Смысл в том, что программы легче читать, когда разные конструкции языка выделены разными цветами. Например, строки могут отображаться зеленым цветом, а ключевые слова вроде `var` — оранжевым.

Sublime Text позволяет выбрать одну из множества схем подсветки. В этой книге используется схема IDLE — вы можете включить ее, войдя в меню Preferences → Color Scheme и выбрав там IDLE, чтобы у вас в редакторе программы выглядели так же, как примеры кода в этой главе и далее.

Наш первый HTML-документ

Установив Sublime Text, запустите его и создайте новый файл, выбрав File → New File. Затем выберите File → Save, чтобы сохранить новый, пустой файл; назовите его *page.html* и сохраните на рабочий стол.

Page — страница

Настало время писать HTML-код. Введите в файл *page.html* следующий текст:

```
<meta charset="UTF-8">
<h1>Привет, мир!</h1>
<p>Моя первая веб-страничка.</p>
```

Сохраните обновленный файл *page.html*, выбрав File ➝ Save. Теперь посмотрим, на что это будет похоже в веб-браузере. Откройте Chrome и, удерживая CTRL, нажмите O (в Mac OS вместо CTRL используйте клавишу COMMAND). В появившемся окне выберите файл page.html, находящийся на рабочем столе. То, что вы должны после этого увидеть, изображено на рис. 5.1.

Рис. 5.1. Ваша первая HTML-страница в Chrome

Вы только что создали свой первый HTML-документ! Вы просматриваете его через браузер, однако находится он не в интернете — Chrome открыл его с вашего компьютера и, считав разметку, определил, как нужно отображать текст.

Теги и элементы

HTML-документы состоят из *элементов*. Каждый элемент начинается с *открывающего тега* и оканчивается *закрывающим тегом*. Например, в нашем первом документе пока всего два элемента: `h1` и `p` (а также элемент `meta`, но его мы отдельно здесь рассматривать не будем. Он нужен, чтобы в браузере отображался русский текст). Элемент `h1` начинается с открывающего тега `<h1>` и заканчивается закрывающим тегом `</h1>`, а элемент `p` начинается с открывающего тега `<p>` и заканчивается закрывающим тегом `</p>`. Все, что находится между открывающим и закрывающим тегами, называют *содержимым* элемента.

Открывающие теги представляют собой название элемента в угловых скобках: < и >. Закрывающие теги выглядят так же, но перед именем элемента в них ставится наклонная черта `(/)`.

Элементы заголовков

У каждого элемента есть особое назначение и способ применения. Например, элемент `h1` означает «это заголовок верхнего уровня». Содержимое, которое вы введете между открывающим и закрывающим тегами `<h1>`, браузер отобразит на отдельной строке крупным жирным шрифтом.

Всего в HTML шесть уровней заголовков: `h1`, `h2`, `h3`, `h4`, `h5` и `h6`. Выглядят они так:

```
<meta charset="UTF-8">
<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>
```

На рис. 5.2 показано, как эти заголовки выглядят в браузере.

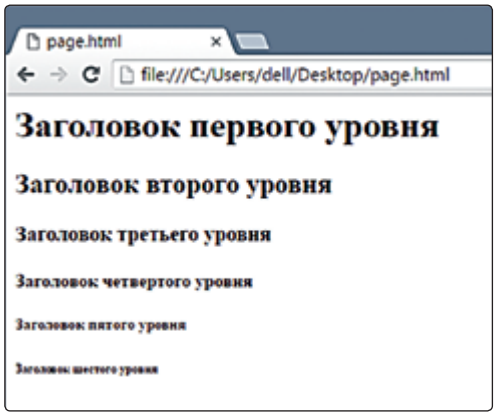

Рис. 5.2. Элементы заголовков разного уровня

Элемент p

Элемент `p` нужен для разделения текста на параграфы. Любой фрагмент текста, который вы поместите между тегами `<p>`, будет отображен как отдельный параграф, с отступами сверху и снизу. Давайте посмотрим, что происходит, если элементов `<p>` несколько. Для этого добавьте новую строку в документ *page.html* (прежние строки показаны серым цветом).

```
<meta charset="UTF-8">
<h1>Привет, мир!</h1>
<p>Моя первая веб-страничка.</p>
<p>Добавим-ка еще параграф.</p>
```

На рис. 5.3 показана страничка с нашим новым параграфом.

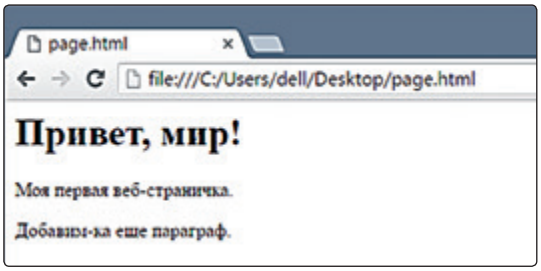

Рис. 5.3. Та же страничка с еще одним параграфом

Обратите внимание, что каждый параграф отображен с новой строки, а между параграфами сделан отступ. Все это благодаря тегу `<p>`.

Пробелы в HTML и блочные элементы

А как наша страничка будет выглядеть без тегов? Давайте посмотрим:

```
<meta charset="UTF-8">
Привет, мир!
Моя первая веб-страничка.
Добавим-ка еще параграф.
```

На рис. 5.4 показана страничка без тегов.

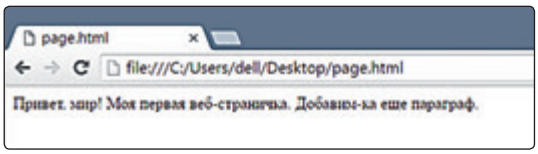

Рис. 5.4. Та же страничка без HTML-тегов

Мало того что пропало форматирование, теперь весь текст отображается в одну строку! Дело в том, что в HTML все *пробельные символы* преобразуются в единственный пробел. Пробельные символы — это любые символы, которые отображаются в браузере как пробелы или

отступы, — например, это пробел, символ табуляции и символ перевода строки (тот самый, который вы вводите, нажимая ENTER или RETURN). Поэтому все пустые строки, которые вы вставите между фрагментами текста в HTML-документе, сожмутся до одного пробела.

Элементы `p` и `h1` — *блочные;* это значит, что их содержимое отображается отдельными блоками текста с новой строки и любое содержимое, идущее после такого блока, тоже начнется с новой строки.

Строчные элементы

А теперь добавим к нашему документу еще два элемента, `em` и `strong`:

На рис. 5.5 показано, как выглядит страница с новыми тегами.

```
<meta charset="UTF-8">
<h1>Привет, мир!</h1>
<p>Моя <em>первая</em> <strong>веб-страничка</strong>.</p>
<p>Добавим-ка еще <strong><em>параграф</em></strong>.</p>
```

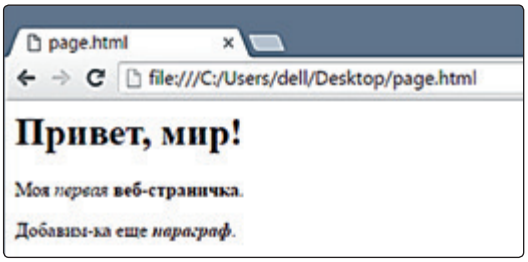

Рис. 5.5. Элементы `em` *и* `strong`

Элемент `em` отображает свое содержимое курсивом, а элемент `strong` — жирным шрифтом. И `em`, и `strong` относятся к строчным элементам, поскольку они, в отличие от блочных элементов, не выводят свое содержимое отдельной строкой.

Чтобы отобразить текст одновременно жирным шрифтом и курсивом, поместите его внутрь обоих тегов. Обратите внимание, что в последнем примере теги стояли в такой последовательности: `<strong><em>параграф</em></strong>`. Очень важно правильным образом *вкладывать* элементы друг в друга: если один элемент находится внутри другого элемента, то его открывающий тег и его

закрывающий тег также должны находиться внутри этого элемента. Например, такой вариант недопустим:

```
<strong><em>параграф</strong></em>
```

Закрывающий тег `</strong>` расположен здесь перед закрывающим тегом `</em>`. Как правило, браузеры никак не сообщают о подобных ошибках, однако неправильно вложенные теги приведут к неверному отображению страниц.

Полноценный HTML-документ

До сих пор мы имели дело лишь с фрагментами HTML, тогда как полноценный HTML-документ должен включать некоторые дополнительные элементы. Давайте посмотрим на законченный HTML-документ и разберемся, зачем нужна каждая его часть. Добавьте в файл *page.html* следующие элементы:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Моя первая настоящая HTML-страничка</title>
</head>

<body>
    <h1>Привет, мир!</h1>
    <p>Моя <em>первая</em> <strong>веб-страничка</strong>.</p>
    <p><strong><em>параграф</em></strong>.</p>
</body>
</html>
```

Head — здесь «шапка документа»

Title — название

Body — тело документа

Sublime Text автоматически ставит отступы при вводе некоторых строк кода, как показано в этом примере. По тегам (таким как `<html>`, `<h1>` *и т. д.) он определяет, внутри каких элементов находится каждая строка, и делает отступы в соответствии с этим. Перед тегами* `<head>` *и* `<body>` *Sublime Text, в отличие от некоторых других редакторов, отступов не ставит.*

На рис. 5.6 показан законченный HTML-документ.

Давайте по очереди рассмотрим элементы из файла *page.html*. Тег `<!DOCTYPE html>` — всего лишь объявление, он сообщает: «это HTML-документ». Далее следует открывающий тег `<html>` (закрывающий тег `</html>` находится в самом конце кода). Каждый

HTML-документ должен содержать элемент `html` верхнего уровня вложенности.

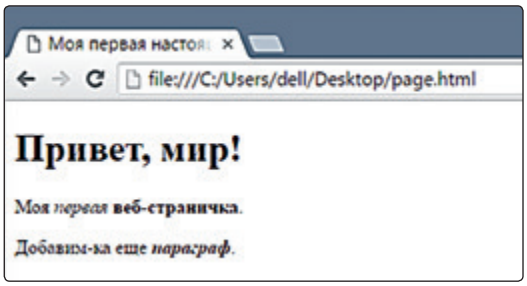

Рис. 5.6. Законченный HTML-документ

Внутри элемента `html` находятся элементы `head` и `body`. Элемент `head` содержит определенную информацию об HTML-документе, например элемент `title`, устанавливающий название документа, — обратите внимание, что текст на закладке браузера на рис. 5.6 («Моя первая настоящая HTML-страничка») соответствует содержимому `title`. Элемент `title` находится внутри элемента `head`, который, в свою очередь, находится внутри элемента `html`.

Внутри элемента `body` находится содержимое, которое отображается в браузере. В данном случае мы просто скопировали эти данные из предыдущего примера.

Иерархия HTML

HTML-элементы подчинены строгой иерархии, которую можно себе представить в виде перевернутого дерева. На рис. 5.7 в виде дерева показан наш документ.

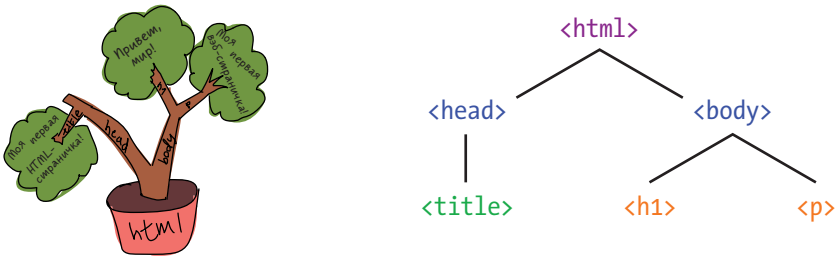

Рис. 5.7. Элементы страницы с рис. 5.6, показанные в виде дерева

Сверху находится элемент `html`. Он содержит элементы `head` и `body`. В свою очередь, `head` содержит элемент `title`, а `body` — элементы `h1` и `p`. Браузер интерпретирует наш HTML согласно этой

иерархии. О том, как менять структуру документа, мы узнаем позже, в девятой главе.

На рис. 5.8 показан другой способ изображения иерархии HTML — в виде вложенных прямоугольников.

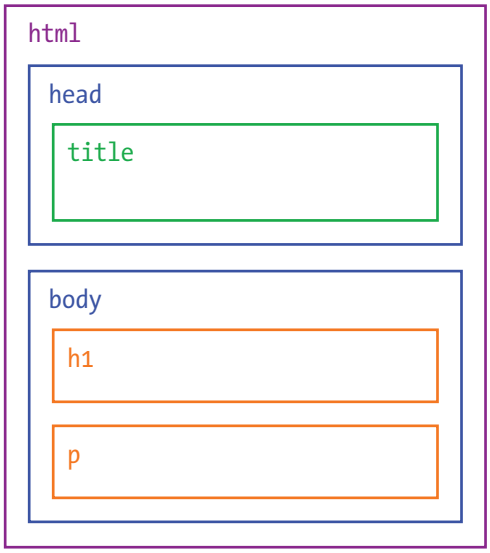

Рис. 5.8. Иерархия HTML в виде вложенных прямоугольников

Добавим в HTML ссылки

Ранее в этой главе мы узнали, что HTML — гипертекстовый язык. Это значит, что HTML-документы могут содержать *гиперссылки* (или просто *ссылки*), ведущие на другие веб-страницы. Такие ссылки можно создавать с помощью элемента `a` (от английского anchor — «якорь»).

Измените свой HTML-документ, чтобы он соответствовал следующему примеру: удалите второй элемент `p`, а также теги `em` и `strong` и добавьте выделенный цветом код, чтобы создать ссылку на интернет-адрес `http://comicsia.ru/collections/xkcd`:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Моя первая настоящая HTML-страничка</title>
</head>

<body>
    <h1>Привет, мир!</h1>
```

```
    <p>Моя первая веб-страничка.</p>
    <p><a href="http://comicsia.ru/collections/xkcd">Нажмите
сюда</a>, чтобы почитать отличные комиксы.</p>
</body>
</html>
```

Сохраните файл и откройте страничку в браузере — она должна выглядеть как на рис. 5.9.

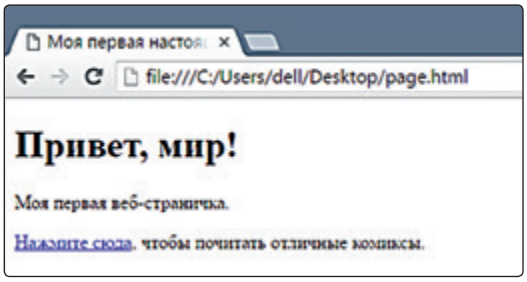

Рис. 5.9. Веб-страница со ссылкой на http://comicsia.ru/collections/xkcd

Если кликнуть по этой ссылке, браузер должен перейти по адресу `http://comicsia.ru/collections/xkcd`. Насладившись комиксами (выберите там тег «программисты» и почитайте смешные истории из жизни разработчиков), кликните на кнопку «назад», чтобы вернуться к нашей страничке.

Атрибуты ссылок

Давайте разберемся, как мы создали эту HTML-ссылку. Чтобы браузер знал, куда перейти по клику, мы добавили элементу `a` так называемый *атрибут*. Атрибуты HTML-документов напоминают пары «ключ-значение» в объектах JavaScript: у каждого атрибута есть имя и значение. Посмотрите еще раз на созданную нами ссылку:

```
<a href="http://comicsia.ru/collections/xkcd">Нажмите сюда</a>
```

Href — от hypertext reference — гипертекстовая ссылка

В данном случае у атрибута есть имя `href` и значение `"http://comicsia.ru/collections/xkcd"` — то есть веб-адрес.

На рис. 5.10 показаны все составные части этой ссылки.

Ссылка отправит вас по любому адресу, который указан в качестве значения атрибута `href`.

Рис. 5.10. Базовый синтаксис для создания гиперссылки

Атрибут title

Также к ссылкам можно добавлять атрибут `title` — он задает текст, который появляется при наведении курсора на ссылку. Например, давайте изменим открывающий тег `<a>`, чтобы он выглядел так:

```
<a href="http://comicsia.ru/collections/xkcd" ↵
title="xkcd: Комиксы для гиков!">Нажмите сюда</a>
```

Теперь перезагрузите страничку. При наведении мышки на ссылку должна появиться надпись: «`xkcd: Комиксы для гиков!`», как на рис. 5.11.

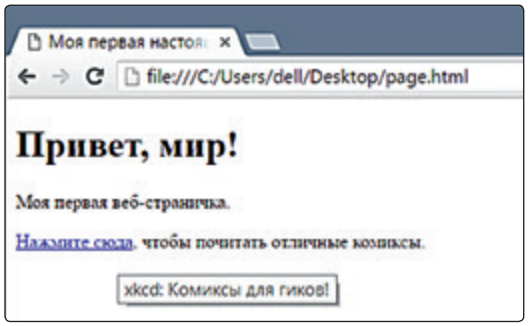

Рис. 5.11. Веб-страничка, содержащая ссылку на адрес http://comicsia.ru/collections/xkcd/ с атрибутом `title`

ПОПРОБУЙТЕ!

Создайте новый файл под названием *links.html*. Пусть его HTML-структура будет такой же, как у странички *page.html*, однако название и заголовок поменяйте на другие, а также добавьте три элемента `p` («параграф»). В каждом параграфе создайте ссылку на один из своих любимых сайтов. Убедитесь, что для всех элементов `a` заданы атрибуты `href` и `title`.

Links — ссылки

Что мы узнали

В этой главе мы познакомились с основами HTML — языка для создания веб-страниц. Также мы создали простой HTML-документ со ссылкой на другую страницу.

В следующей главе мы разберемся, как встраивать в нашу страничку JavaScript-код. Это облегчит создание более объемных программ по мере изучения новых возможностей JavaScript.

Эта книга посвящена JavaScript, а не HTML, поэтому я рассмотрел лишь самые азы создания HTML-документов. Вот некоторые ресурсы, где можно узнать о HTML больше:

На английском языке:

- Курс HTML и CSS от Codeacademy: http://www.codecademy.com/tracks/web/
- Mozilla Webmaker: https://webmaker.org/

На русском языке:

- Введение в HTML от Mozilla Developer Network: https://developer.mozilla.org/ru/docs/Web/Guide/HTML/Introduction
- https://htmlacademy.ru/

издательство
МАНН, ИВАНОВ И ФЕРБЕР
Максимально полезные книги