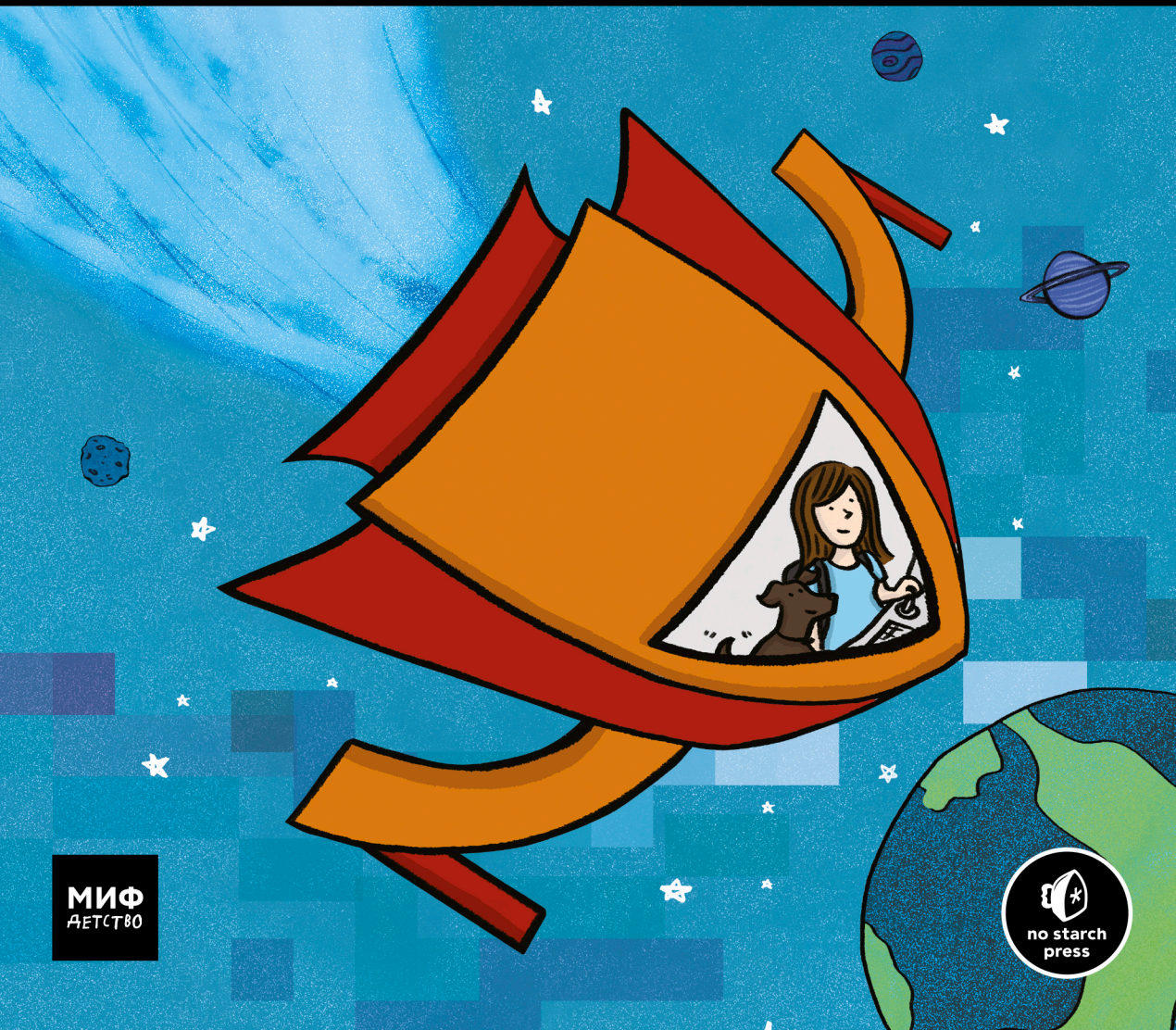


ЛЕГКОЕ
ПРОГРАММИРОВАНИЕ

КАК СОЗДАТЬ САЙТ

КОМИКС-ПУТЕВОДИТЕЛЬ
ПО HTML, CSS И WORDPRESS

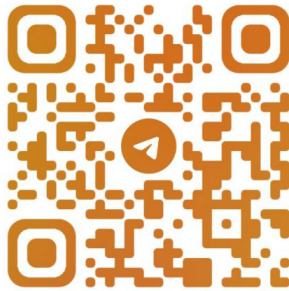
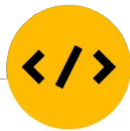
НЕЙТ КУПЕР



МИФ
ДЕТСТВО



Как создать сайт



@CODELIBRARY_IT

Нейт Купер

Как создать сайт

Комикс-путеводитель по HTML,
CSS и WordPress

Перевод с английского Станислава Ломакина

Иллюстрации Ким Джи

«Манн, Иванов и Фербер»

2019

УДК 087.5+004.439
ББК 32.973.4
К92

Издано с разрешения No Starch Press, Inc., a California Corporation

На русском языке публикуется впервые

Перевод с английского Станислава Ломакина

Научные редакторы Аркадий Аввакумов, Валерий Артюхин

Возрастная маркировка в соответствии с Федеральным законом №436-ФЗ: 0+

Купер, Нейт

К92 Как создать сайт. Комикс-путеводитель по HTML, CSS и WordPress. / Купер, Нейт; ил. Ким Джи. Пер. с англ. С. Ломакина. — М.: Манн, Иванов и Фербер, 2019. — 266 с.: ил.

ISBN 978-5-00100-914-6

Пытаясь сделать сайт для своего портфолио, художница Ким попадает на удивительную планету. Чтобы вернуться домой, ей придется выучить азы HTML, победить злобного дракона 404, подружиться с веб-гуру и доброй колдуньей CSS и выяснить, что таится за стенами WordPress-сити.

В формате увлекательного комикса книга познакомит детей с языками HTML и CSS, а также с конструктором сайтов WordPress.

УДК 087.5+004.439
ББК 32.973.4

Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Copyright © 2014 by Nate Cooper.

Title of English-language original: Build Your Own Website:

A Comic Guide to HTML, CSS, and WordPress,

ISBN 978-1-59327-522-8, published by No Starch Press.

Russian language edition copyright © 2019 by Mann, Ivanov and Ferber.

All rights reserved.

© Перевод на русский язык, издание на русском языке

ООО «Манн, Иванов и Фербер», 2019



ISBN 978-5-00100-914-6

Содержание

Об авторе.....	9
Об иллюстраторе	9
Предисловие автора.....	10
Благодарности	11

1

Первый день занятий.....	13
Основы веб-разработки	15
Что тебе потребуется.....	23
Веб-браузер	23
Текстовый редактор	23
Что нужно знать	24
Как читать веб-адреса?	24
Что такое клиент и сервер?	24
Что такое хостинг?.....	25
FTP-клиент	26

2

Приключения с HTML.....	27
Ким учит основные HTML-теги.....	30
Пути и правила именования.....	38
Соглашения об именах HTML-документов	43
Добавление изображений	45
Структура файлов и папок	51
Эксперименты с HTML	59
Приступаем	59
Использование основных HTML-тегов	60
Добавление картинок на веб-страницу	64
Добавляем секцию head	66
Несколько полезных HTML-тегов.....	69

3

Ким наводит красоту с помощью CSS	71
Знакомство с CSS	75
Ким изучает основы CSS	84
Углубляясь в CSS.....	91
Ким узнает о классах и метках CSS	96
Каскадные таблицы стилей	105
Создание таблицы стилей и ее подключение к HTML	105
Твоя первая таблица стилей.....	106
CSS: язык стилей	107
Синтаксис CSS	107
Классы, метки и наследование	110
Цвета	114
Тег <div> и выравнивание в CSS	115
Поля и внутренние отступы.....	119
Разметка структуры и <div>	122

4

Ким исследует WordPress-сити	127
Добро пожаловать в WordPress-сити	129
Ким осваивается в WordPress-сити.....	134
Ким создает первую страницу в WordPress	140
Ким обустроивает свой сайт	145
Ким добавляет фотографии на сайт.....	150
Начало работы с WordPress	157
Вход в WordPress и выход из него	157
Забыл, как войти на сайт?	159
Проверка результатов	160
Контент: посты и страницы.....	161
Структура твоего сайта.....	162
Создание первой страницы.....	163
Добавление изображений	167
Публикация страницы.....	172
Управление страницами	174
Создание поста в блоге	176

Задаем структуру: рубрики и метки постов	178
Верхнее изображение.....	180
Посты с видео, фотографиями и цитатами	181
Правка и удаление содержимого	183

5

Индивидуальная настройка WordPress	185
Панель «Внешний вид»	188
Ким устанавливает на сайт плагины	194
Ким заглядывает за занавеску.....	198
Меняем оформление: главное о темах	205
Индивидуальная настройка темы.....	208
Настройка меню навигации	211
Вкладка «Настройки экрана»	214
Настройки для всего сайта.....	215
Расширенные настройки	217
Плагины.....	218
Виджеты	222
Изучение виджетов	224
Обновления.....	224
Перенос сайта на новый хостинг	226
Дополнительная информация.....	227

6

Запуск сайта	229
Нет ничего лучше хостинга.....	231
Дом для портфолио Ким.....	240
Дружеская сеть	246
Итак, ты готов запустить сайт.....	249
Покупка хостинга: твой дом в сети.....	251
Запуск простого сайта на HTML/CSS	252
Установка WordPress	253
О покупке хостинга для WordPress	254
Заключение.....	255
Указатель.....	256

Об авторе



Фото: Аманда Гануни

Нейт Купер — писатель и консультант из Нью-Йорка. Он занимался розничным маркетингом в компании Apple и завоевал авторитет в среде предпринимателей своими статьями о бизнес-стратегиях. Его компания Simple Labs учит клиентов работать с WordPress и организует публичные мероприятия, посвященные технологиям и средствам коммуникации.

Об иллюстраторе

Ким Джи — иллюстратор и графический дизайнер. Живет в Нью-Йорке со своим бойфрендом и песиком Тофу. В 2010 году нарисовала веб-комикс о себе самой, а позже собственными силами опубликовала графический роман и сборник мини-комиксов.

Предисловие автора

Открывая свою консультационную фирму, я предвидел, что в среде дизайнеров возникнет спрос на базовые навыки создания веб-сайтов. Если раньше для работы хватало знания Photoshop, Illustrator и InDesign, то сегодня клиенты ждут умения обращаться с HTML и CSS. По мере того как компании переходят от проприетарных систем управления контентом к универсальным CMS вроде WordPress, растет спрос на дизайнеров с базовыми техническими навыками. Сегодня надо уметь создавать контент как для печати, так и для веба, а это подразумевает знание HTML и CSS.

Тут нет ничего удивительного: СМИ все активнее уходят от печатных носителей в онлайн. Но для меня стало сюрпризом, что потребность в этих навыках распространилась далеко за пределы дизайнерской и медиатусовки. Наверняка ты догадываешься, что раньше компьютеры стояли далеко не на каждом столе. Когда я был старшеклассником, я часто заходил к отцу на работу в муниципальный колледж. Папу считали прогрессивным: он не только установил компьютер на рабочем столе, но и проверял свою электронную почту — вещь среди его коллег неслыханная.

Того мира больше нет. Прошло не так уж много времени, но интернет, персональный компьютер, текстовые редакторы, умение печатать быстро и пользоваться электронной почтой стали стандартным набором инструментов и требований практически для любой специальности. Теперь не только дизайнерам, но и профессионалам из самых разных областей нужны базовые знания о HTML и CSS, чтобы создавать онлайн-контент. Интернет завоевывает мир, но освоиться в нем не так уж и сложно.

Я надеюсь, что эта книга заинтересует и тех, кто чувствует, что отстал от жизни, и тех, кто хочет быть впереди планеты всей. Мы живем в постоянно меняющемся мире, где знания, актуальные сегодня, завтра могут оказаться бесполезными. Чтобы добиться профессионального успеха в будущем, недостаточно знать только то, что необходимо для решения текущих задач, — нужно осваивать и развивать новые навыки. Учеба не должна стать рутинной или каторжным трудом! Когда ты поймешь, как сделать обучение интересным, оно будет приносить тебе радость.

Дизайнер ли ты, писатель, студент или кто угодно еще — я надеюсь, этот комикс покажется тебе забавным и интересным. А еще я надеюсь, что для тебя станут мантрой слова «Учиться — круто»!

С нетерпением жду будущего! 😊

Нейт Купер
Июль 2014 года

Благодарности

Эта книга зародилась как проект на Kickstarter. И пусть со временем он сильно отступил от первоначального плана, я хочу выразить искреннюю благодарность тем, кто с самого начала поддерживал мое стремление создать обучающую книгу-комикс.

Эти первопроходцы убедили меня, что такая книга нужна людям и что люди готовы поддержать проект не только словом, но и деньгами. Спасибо вам — тем, кто поверил в меня и в мою затею.

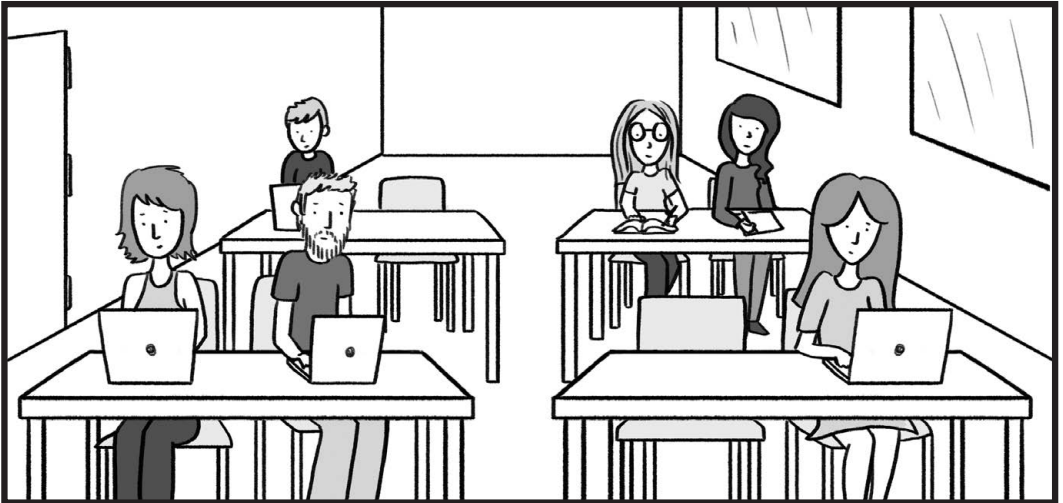
Моими главными спонсорами были Мэтью Бергман, Дуайт Бишоп, Дин Куни, Джеймс Кропчо, Сью Мейсонёв, Стивен Моррисон, Эдвард О'Нил и Юхан Уле. Выражаю им особую благодарность.

Гейл Амурао, Энджи Холл Андерсон, Ари Арсайди, Тони Стивен Беннет, Клэр Бёрнс, Николь Каласич, Люк Чемберлен, Сара Чиппс, Эрни Купер, Джессика Купер, Катрина Э. Дамкоэлер, Колин Диб, Марта Дентон, Эми Донелли, Дэнни Дозрти, Тэринн Фармер, Эдвард Дж., Джордж Хейнс, Стивен Ходас, Джим Хопкинсон, Билл Джонсон, Рейган Келли, Митч Косен, Марисса Леви Лерер, Джонатан Левин, Анна Лубрехт, Мишель Маззара, Колетт Маццучелли, Бренна МакЛафлин, Лура Мильнер, Джон Мёрч, Стефан Никам, Джейсон Ноу, Пол Орландо, Эрик Пэн, Крейг Планкетт, Джули Рош, Сет, Марни Смит, Шакти Андреа Смит, Кимберли Энн Саутвик, Бобби Стоскопф, Эрика Суоллоу, Харрисон Свифт, Кара Жалковски, Шон Толтс, София Тэпер, Дженнифер Цахи, Джереми Уодхэмс, Джо Уоткинс, Стефан Вермейер и Кэти Зак — спасибо вам!

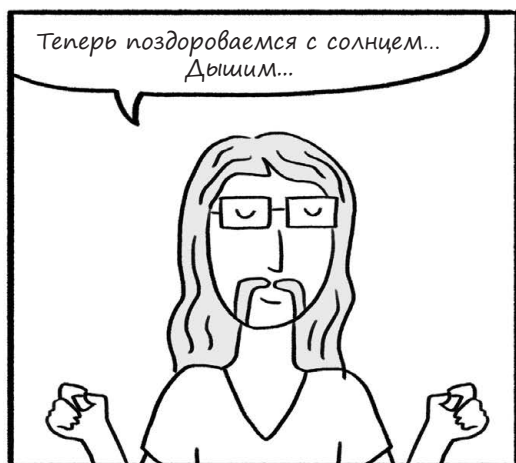
Хочу также поблагодарить Шея Хоуи за его ценные отзывы по ходу работы над книгой и всех сотрудников компании No Starch Press, которые помогли воплотить мой замысел в реальность.

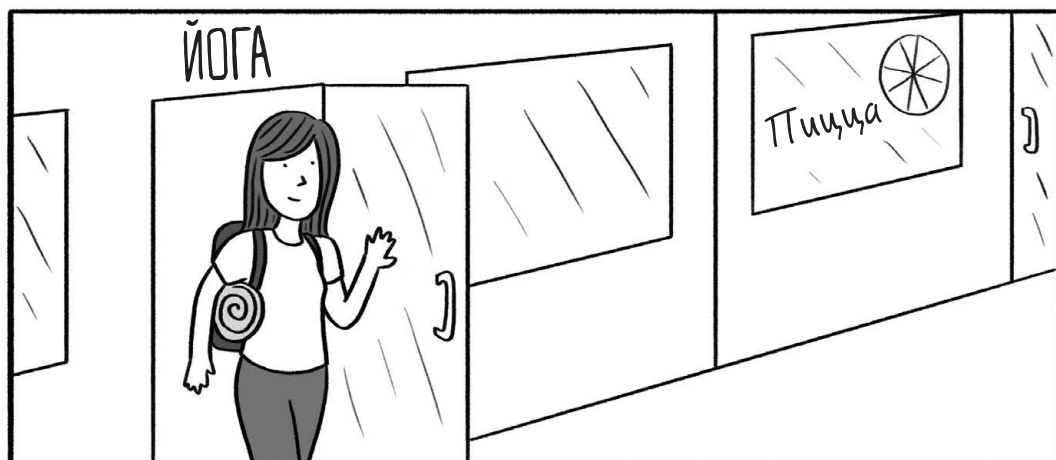
1

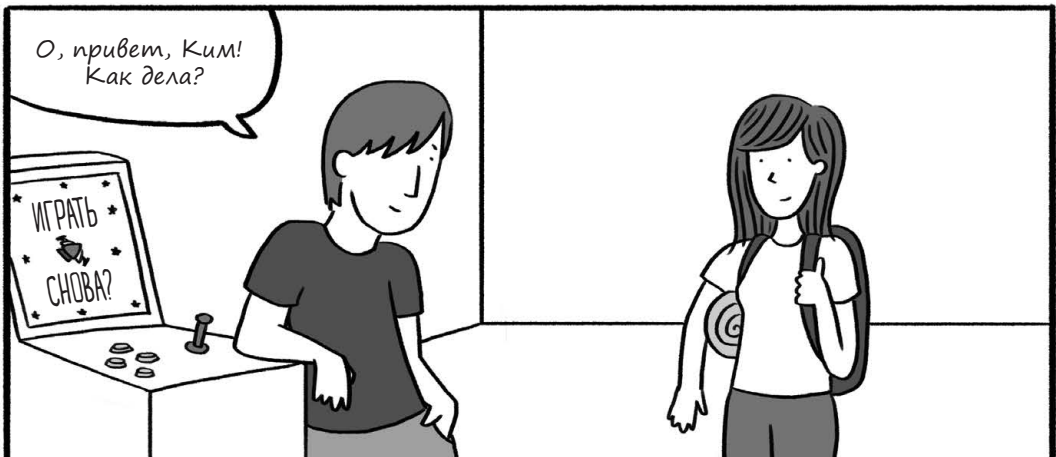
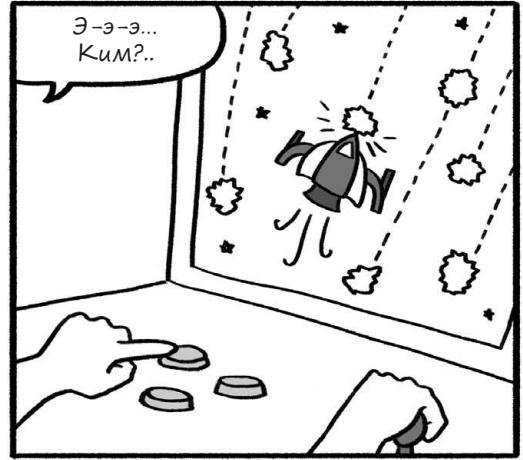
Первый день занятий



Основы веб-разработки

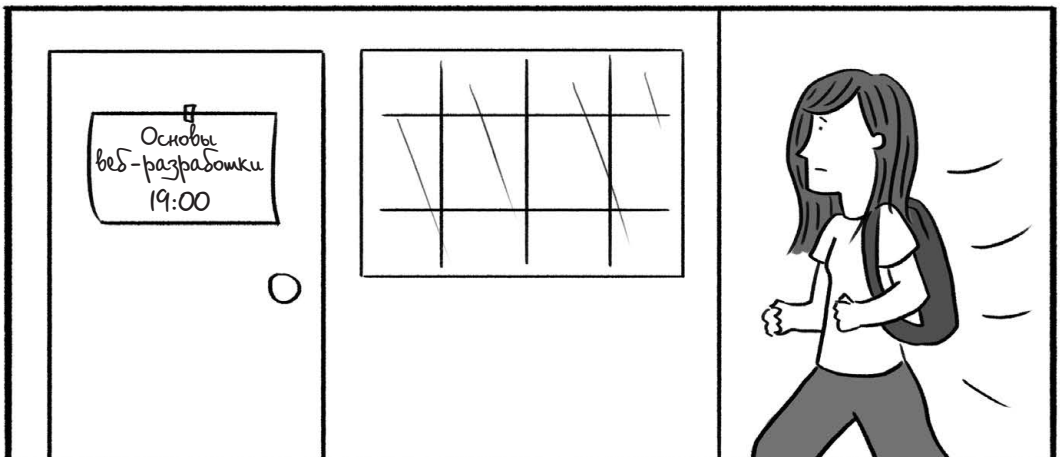
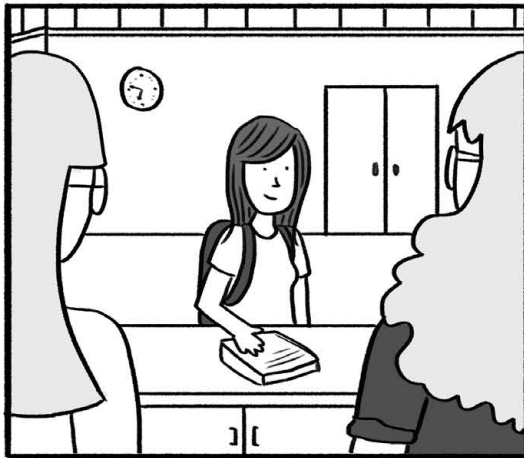
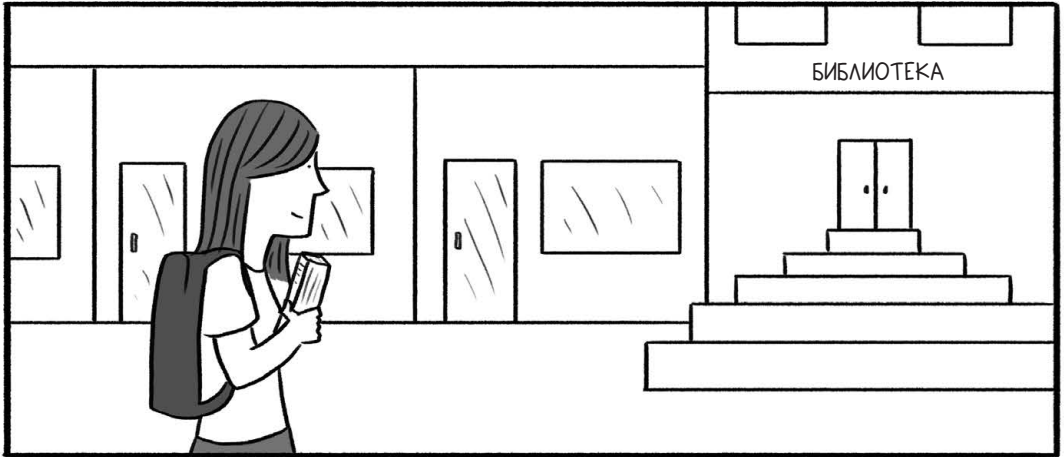


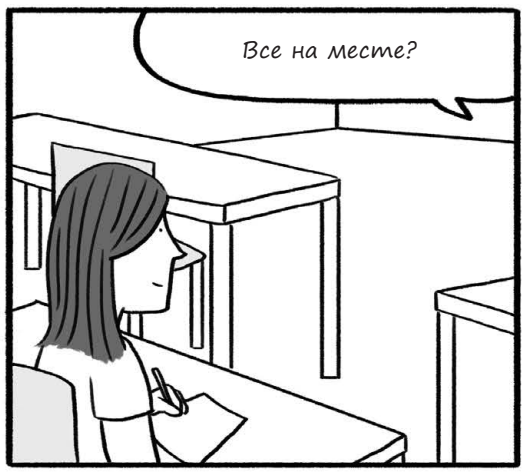
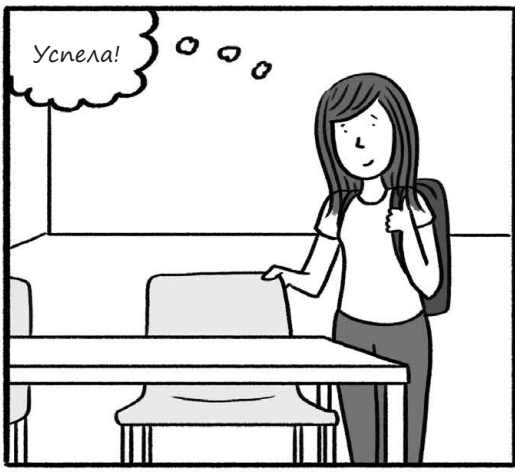
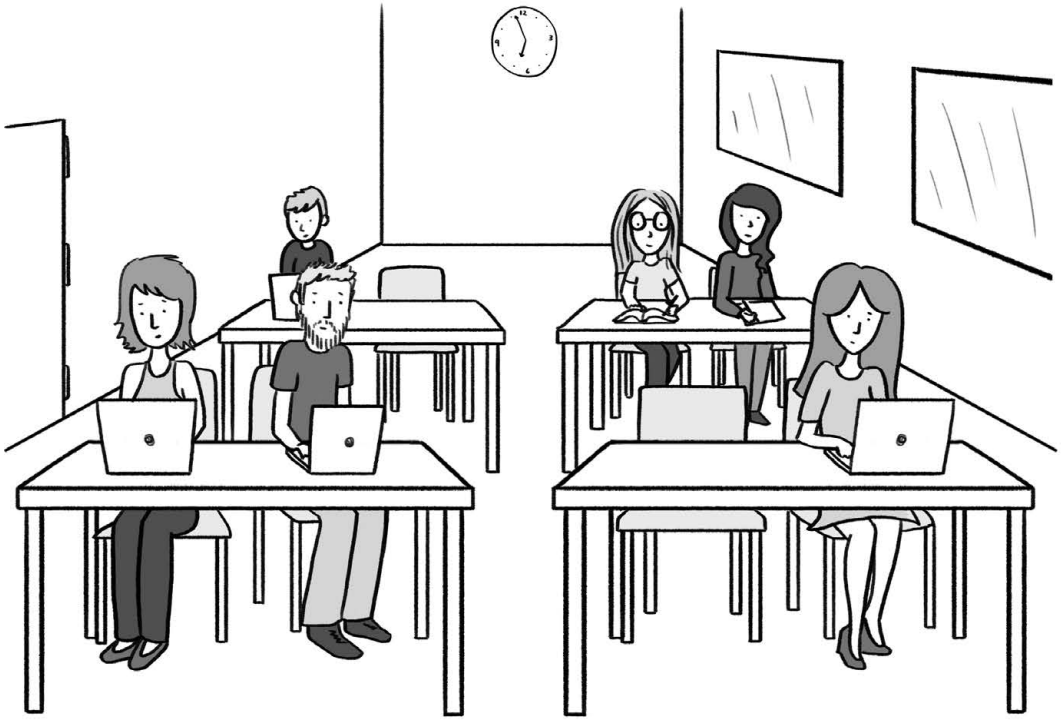














Что тебе потребуется

Эта книга научит тебя базовым приемам создания сайтов, а также основам HTML, CSS и WordPress. К концу курса ты получишь достаточно знаний, чтобы создать собственный веб-сайт. Впрочем, в одной небольшой книжке я не смогу рассказать о веб-разработке все. Обучение — это непрерывный процесс, и я надеюсь, что книга поможет тебе сделать первые шаги к настоящему мастерству.

Тебе предстоит *самостоятельно* выполнять задания, чтобы развивать необходимые навыки. В этой книге используется подход «обучение на практике», так что тебе потребуется компьютер и несколько программ.

Веб-браузер

Первым делом тебе понадобится *веб-браузер* — программа для просмотра интернет-страниц. Если твой компьютер работает под Windows, лучше всего скачать Chrome или Firefox, а Internet Explorer не использовать. Его старые версии не поддерживают последние веб-стандарты. Для большинства примеров из этой книги Internet Explorer вполне подойдет, но если ты займешься веб-разработкой более серьезно, то быстро почувствуешь разницу.

Если у тебя Mac OS, там уже установлен Safari, можешь использовать его. Впрочем, полезно установить также Firefox и Chrome, чтобы проверять результаты твоей работы. Кроме того, некоторые инструменты Firefox или Chrome могут показаться тебе более удобными.

Установить несколько браузеров — отличная идея. Так ты не только узнаешь, чем они отличаются, но и сможешь увидеть свой сайт глазами посетителей, использующих другие браузеры.

Текстовый редактор

Далее тебе понадобится редактор, в котором ты будешь править код. Зачем связываться с новой программой для работы со старым добрым текстом? Затем, что в хорошем редакторе кода писать HTML и CSS проще. На первый взгляд может показаться, что такая программа мало чем отличается от Microsoft Word и других текстовых редакторов, но скоро ты поймешь, что она специально создана для работы с кодом (см. рис. 1.1).

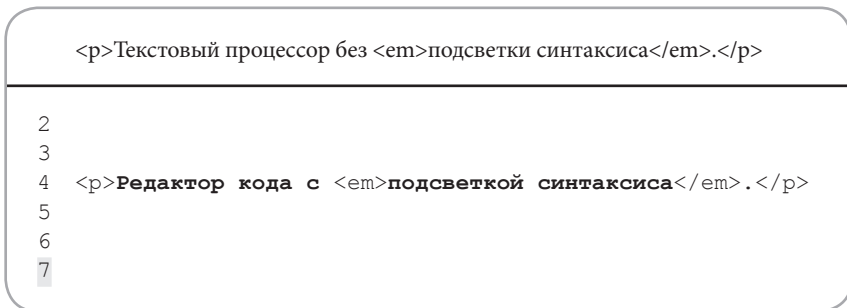


Рис. 1.1. Не используй Word и другие текстовые процессоры для правки HTML (вверху)! Хороший редактор кода облегчит твою работу (снизу). В нем используется моноширинный шрифт, он подсвечивает парные теги и сохраняет файлы в нужном формате

Если у тебя Windows, отличный бесплатный вариант — NotePad++ (доступен по адресу <http://www.notepad-plus-plus.org>). Если же у тебя Mac OS, можно бесплатно скачать TextWrangler (<http://www.barebones.com/products/textwrangler>). Также обрати

внимание на Sublime Text — это превосходный бесплатный редактор, который я настоятельно рекомендую. Он подходит как для Mac OS, так и для Windows (<http://www.sublimetext.com>). Выбери редактор на свой вкус и осваивай его.

Веб-браузер и текстовый редактор — все, что будет нужно в главах про HTML и CSS. Умение пользоваться редактором кода пригодится, если ты возьмешься за более продвинутый язык скриптов, например PHP, JavaScript или Ruby.

Что нужно знать

Я предполагаю, что ты умеешь пользоваться файловым менеджером (Finder в Mac OS, «Проводник» в Windows), открывать и сохранять файлы, устанавливать программы и в целом представляешь, как обращаться с компьютером. Есть еще несколько базовых вещей, о которых тебе следует знать.

Как читать веб-адреса?

Думаю, тебе уже доводилось встречаться с веб-адресами — такими как <http://nytimes.com/>, <http://en.wikipedia.org/> или <http://nostarch.com/websitecomic>. Мы, гики, обожаем хитрые термины, поэтому веб-адрес у нас называется *URL (uniform resource locator — единообразный указатель ресурса)*. Впрочем, я буду называть его просто ссылкой или адресом.

Когда ты бродишь по просторам интернета, адрес в строке браузера меняется, словно ты гуляешь по городским кварталам, заглядывая в новые магазины.

Давай разберемся, для чего нужна каждая из частей адреса.

<http://nytimes.com/articletitle>



- 1 В начале адреса стоит **http://**. Это значит, что мы используем *HTTP (HyperText Transfer Protocol — протокол передачи гипертекста)*. Для браузера это указание на то, что перед ним HTML-документ. Писать свой HTML-код мы начнем в главе 2.

HTTP — самый распространенный протокол в интернете. Также стоит знать про *HTTPS (HyperText Transfer Protocol Secure — защищенный протокол передачи гипертекста)*. Он предназначен для страниц, запрашивающих конфиденциальные данные, например пароли или номера кредитных карт. Ты можешь встретить и другие протоколы, например *ftp:// (File Transfer Protocol — протокол передачи файлов)* — см. «FTP-клиент» на стр. 26).

- 2 Затем следует доменное имя — в нашем случае это **nytimes**.
- 3 Далее мы видим **.com**. Это значит, что перед нами коммерческий сайт. Хотя **.com** все еще остается самым распространенным доменом верхнего уровня, сегодня в сети хватает и других **доменов**.
- 4 Остаток URL указывает на статью, пост в блоге, веб-страницу или другой конкретный ресурс.

Что такое клиент и сервер?

Тебе интересно, что происходит в сети, когда ты открываешь сайт, и как вообще устроен интернет? Так вот, это просто куча компьютеров, которые переговариваются между собой.

Когда ты открываешь «Википедию», твой компьютер обращается к серверу «Википедии» примерно так:



Когда ты запрашиваешь в браузере статью из «Википедии», сервер «Википедии» (показан на рис. 1.2) выдает ее тебе, примерно как официант в ресторане по твоей просьбе приносит меню или круассан.



Фото: Victorgrigas

Рис. 1.2. Сервер может быть как обычным настольным компьютером, так и группой специальных компьютеров — например как сервера «Википедии» на этой фотографии. Чем больше трафик у твоего сайта, тем более качественный и вместительный сервер тебе потребуется

Обычно ты запрашиваешь страницы из сети, как посетитель ресторана. Теперь тебе предстоит примерить на себя роль официанта. Но где взять собственный сервер?

Что такое хостинг?

Чтобы созданные тобой страницы были видны всему миру, необходим сервер. Содержать свой сервер непросто, поэтому люди чаще всего арендуют место на чужом. Компании, продающие серверные ресурсы, называются *хостинг-провайдерами*. За оговоренную плату ты можешь размещать на *хостинге* свои страницы, и они будут доступны всем желающим в режиме 24/7. Размещение сайта на хостинге обычно происходит в два этапа: сперва ты регистрируешь домен (например, www.natecooper.co), а затем выбираешь хостинг, оплачивая его на месяц или на год вперед.

FTP-клиент

Со временем тебе понадобится загружать на хостинг файлы — так ты будешь добавлять страницы и редактировать их. Для этого нужен *FTP-клиент*, то есть программа передачи файлов. На Mac OS и Windows можно воспользоваться отличным бесплатным FTP-клиентом FileZilla (www.filezilla-project.org/). Купив хостинг, ты получишь данные для подключения по FTP от хостинг-провайдера. Пока что тебе не нужен FTP-клиент, но он пригодится, когда ты создашь свой первый сайт.

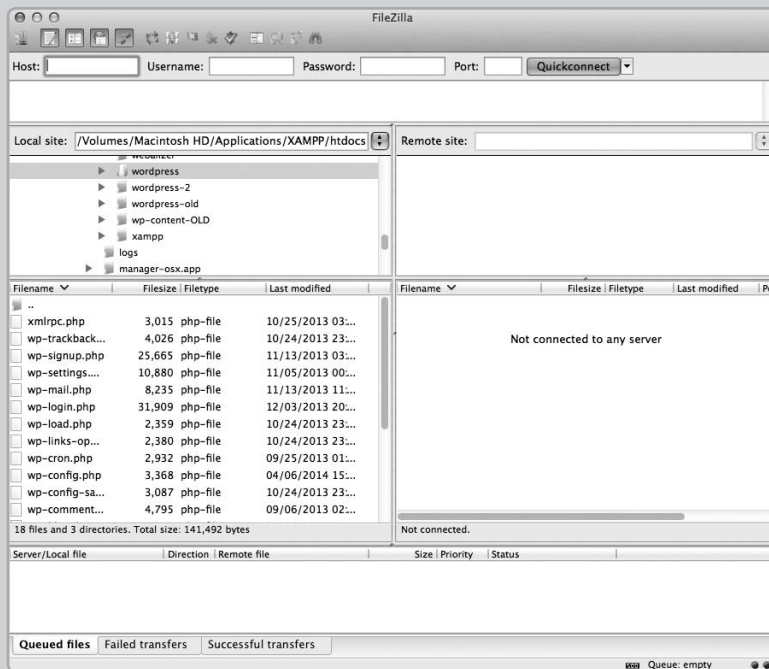


Рис. 1.3. FileZilla — это бесплатный FTP-клиент, позволяющий загружать файлы с компьютера на сервер

Впрочем, для работы с этой книгой не нужно ничего покупать. Создавая страницы на HTML и CSS, ты будешь тестировать их на своем компьютере, в браузере. Ты сможешь проверять, как они выглядят, не выкладывая их в интернет.

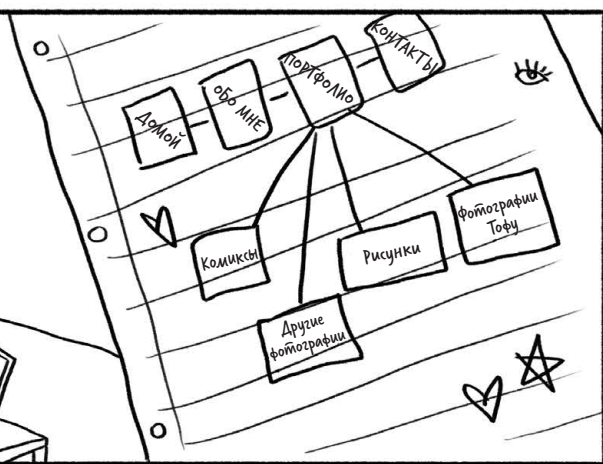
В главе 4 мы начнем экспериментировать с WordPress, и тогда тебе понадобится хостинг, поскольку WordPress требует установки на сервер. Ты можешь либо создать бесплатный аккаунт на сайте WordPress.com, либо приобрести хостинг на стороннем сайте (например, HostGator.com) и бесплатно установить WordPress там (см. главу 6).

Мы еще коснемся некоторых особенностей покупки хостинга в последней главе, а пока отложим этот вопрос. Веб-браузер и редактор кода уже установлены? Тогда сейчас тебе понадобится только тяга к приключениям. Итак, за учебу!

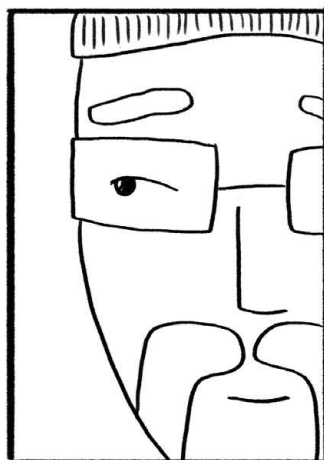
2

Приключения с HTML

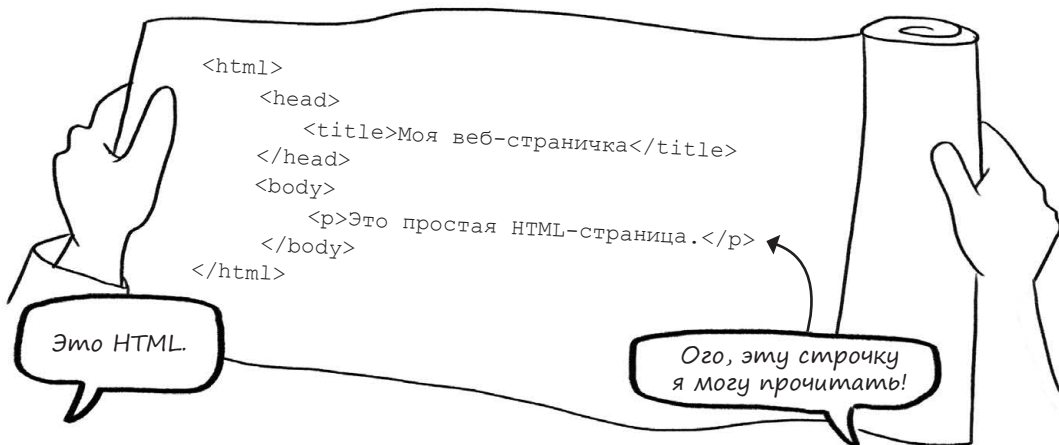




Ким учит основные HTML-теги

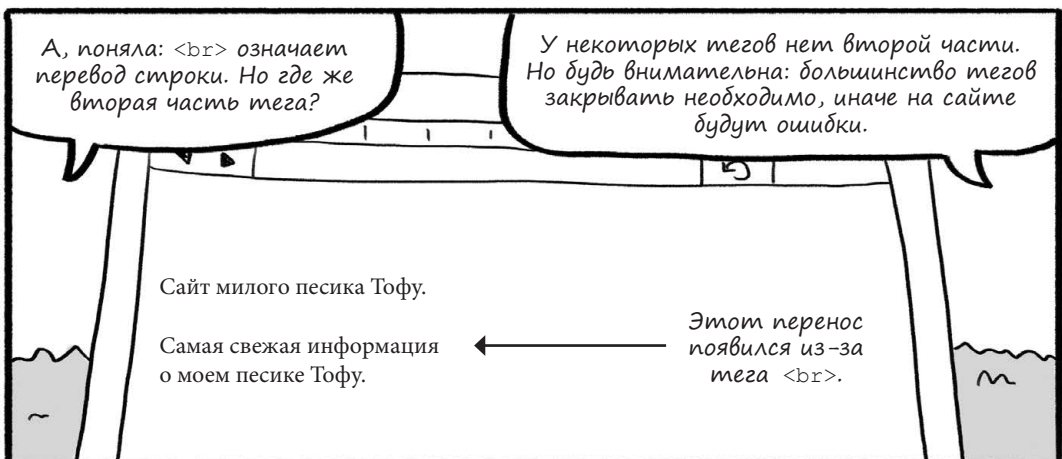




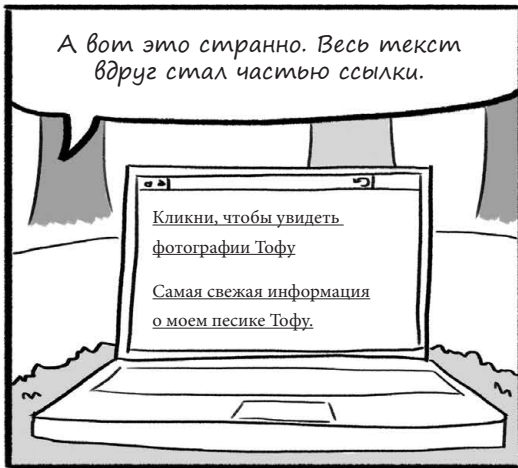
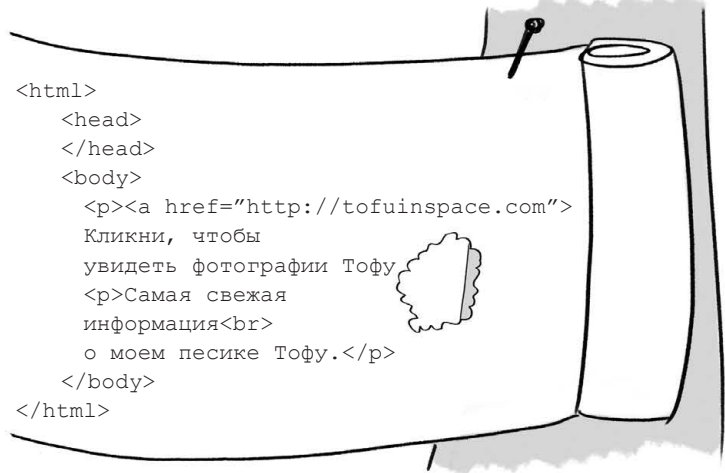








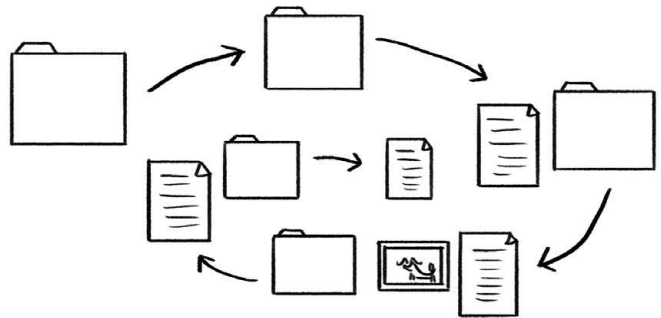




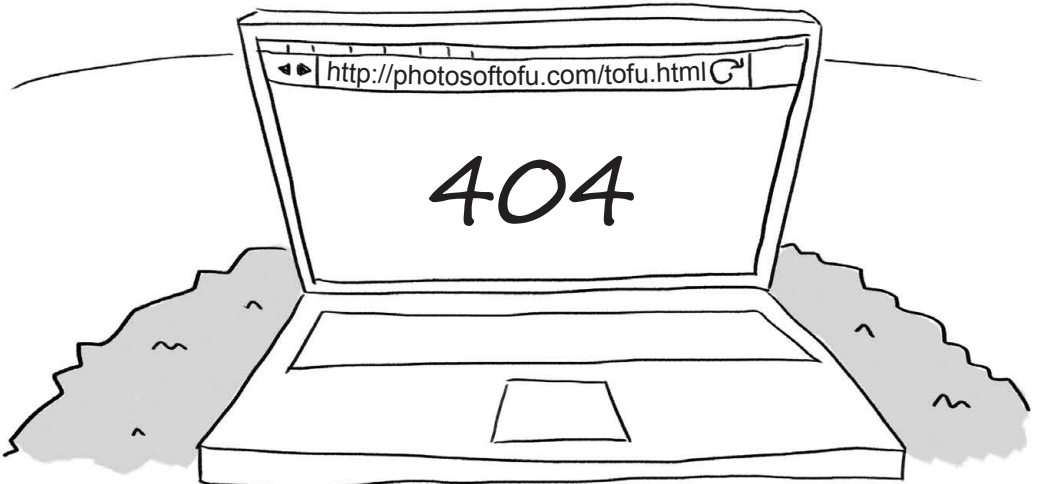
Пути и правила именовани



Видишь ли, давным-давно ученые, создавшие операционную систему UNIX, придумали иерархическую файловую систему.







Появился
злой дракон
404.





Спокойно, я с ним разделился. Так что ты ввела в адресной строке браузера?



А, понятно. Ты ввела tofi.html вместо ToFi.html. Веб-сервер выдает ошибку 404, если не может найти файл. Запомни: сервера отличают прописные буквы от строчных, для них это разные вещи.

Неужели одна крохотная ошибка вызвала целого дракона?



А ты чего ожидала? Просто увидеть сообщение об ошибке.*

* Ошибка 404 просто выводится на экран, однако веб-разработчикам следует остерегаться ее, как огнедышащего дракона. Ведь если на сайте есть ссылки с ошибками, посетители не смогут нормально им пользоваться.



В сети может быть опасно, Тофу.



Путешествовать без оружия — вот что опасно! Возьми этот Меч Стандартов и Соглашений.

Ну и названия вы придумываете, старички.



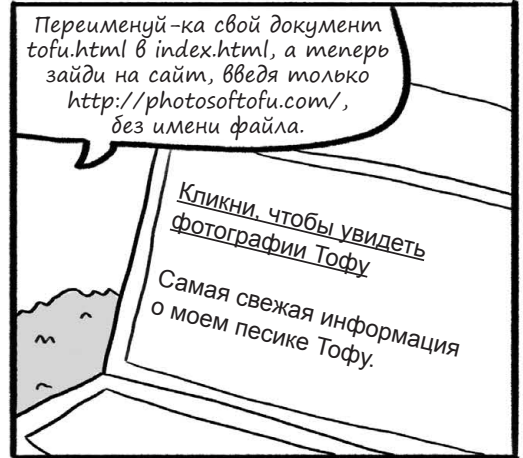
Есть такое. Но ты никогда не ошибешься, если будешь следовать заветам мудрецов.

Соглашения об именах HTML-документов

Если, давая своим документам имена, вы будете следовать этим правилам, драконы (ошибки) больше вас не побеспокоят!

- Осторожнее с заглавными буквами. Лучше вообще не использовать их в названиях документов, ведь *Tofu.html*, *ToFu.html* и *TOFU.HTML* — это три разных имени. Лучше называть страницы только строчными буквами, например *tofu.html*.
- Не используйте в именах пробелы. Если на компьютере есть файл *tofu in space.html*, в браузере он, скорее всего, откроется как *tofu%20in%20space.html*. Однако нет гарантии, что так будет на всех компьютерах. Лучше просто не ставьте в именах пробелы либо заменяйте их дефисами: *tofu-in-space.html*.
- Файлы с именем *index.html* — особенные. Если не указано имя файла, браузер автоматически попытается открыть именно индекс.

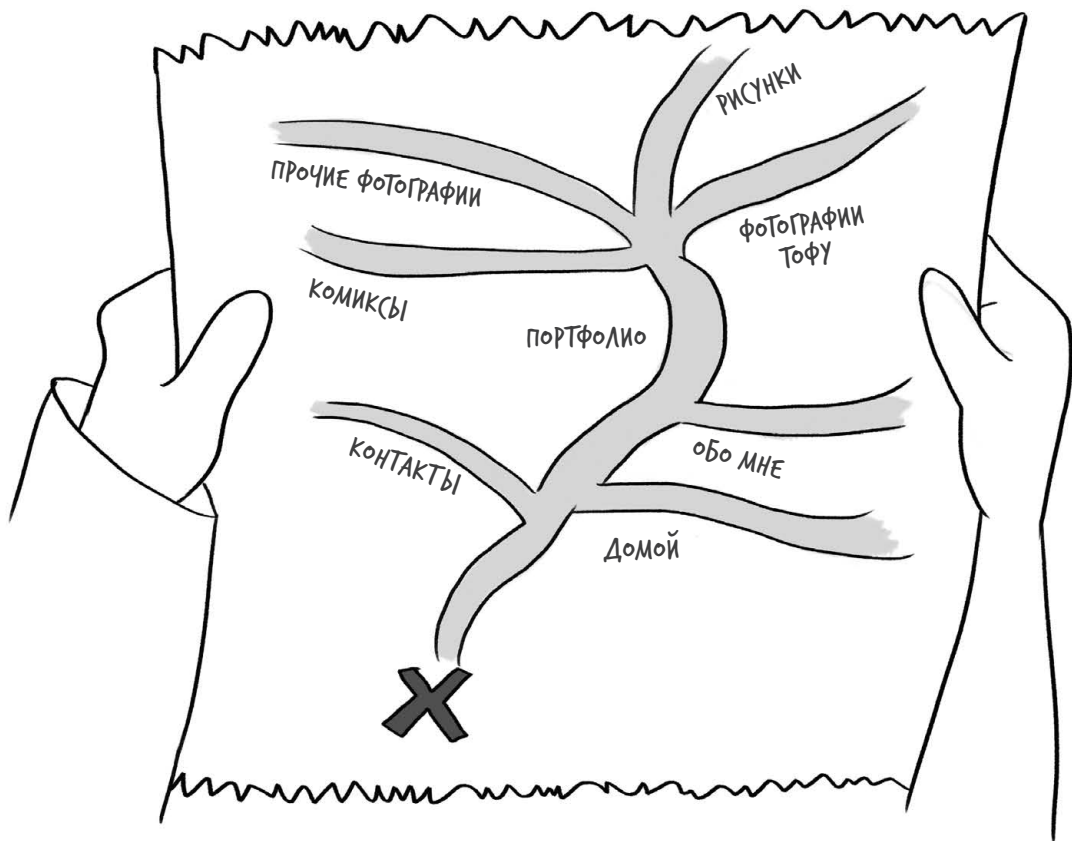




Добавление изображений

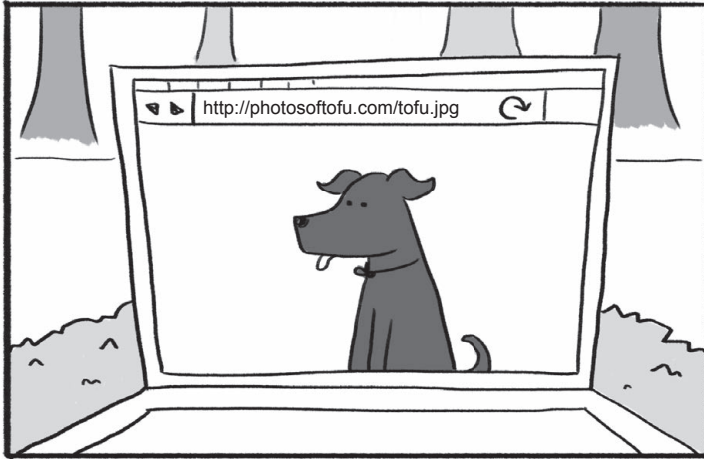


Хм-м. Сверимся с картой.









Да! Это так называемый абсолютный URL, или полный путь до картинки. Код, выделенный жирным, добавит картинку в твой HTML.

```
<html>
<head>
</head>
<body>
  <p><a href="http://tofuinspace.com">
Клики, чтобы увидеть фотографии Тофу</a></p>
  <p>Самая свежая информация<br>
о моем песике Тофу.</p>
  
</body>
</html>
```

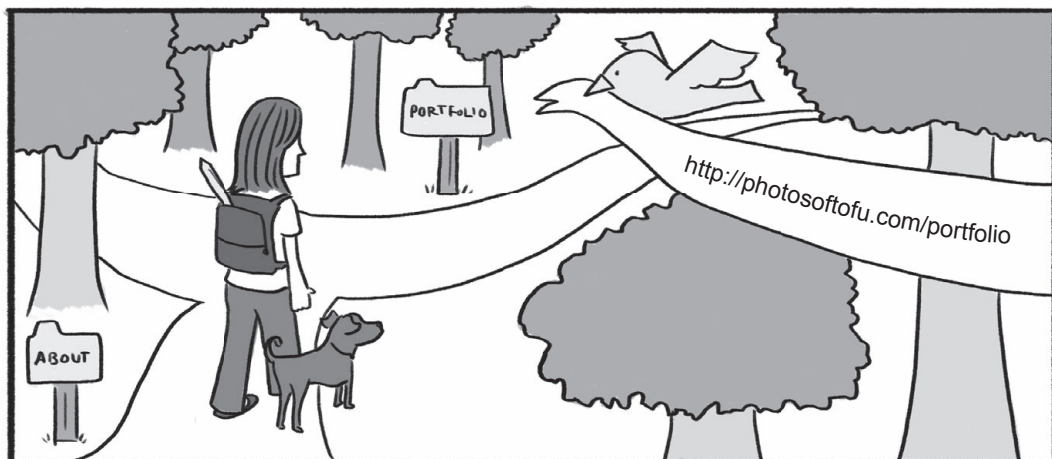





Структура файлов и папок







Ой, Тофу, какой бардак! Нужно
прибраться. Давай создадим
в папке portfolio отдельную папку
для картинок, images.



Ким права: хранить
картинки в отдельных
папках — хорошая идея.
Однако на ее странице уже
есть теги ``. Что с ними
будет, если переместить
картинки в другую папку?











Правила и стандарты помогли Ким победить дракона неверной ссылки.

Эксперименты с HTML

Сделать веб-сайт не так уж сложно, но сперва нужно все продумать. Хотя Ким и новичок в HTML, план сайта помог ей избежать ошибок. Создание сайта начинается с его первой страницы, а все страницы пишутся на языке HTML.

Но что такое HTML? Это *гипертекстовый язык разметки* — основной язык интернета. HTML-документ состоит из обычного текста, и чаще всего мы можем понять, о чем в нем идет речь, даже не разбираясь в программировании. С помощью угловых скобок (<>) в тексте задается разметка. Встретив такие скобки, браузер распознаёт их содержимое и меняет вид последующего текста в соответствии с правилами языка.

Как только ты выберешь редактор кода, для создания HTML-страниц потребуются лишь здравый смысл и немного знаний. Давай начнем делать веб-страничку, используя несколько простых тегов. Запускай текстовый редактор — и вперед!

Приступаем

Все веб-страницы должны начинаться с особой команды DOCTYPE. Она сообщает браузеру, какую версию HTML ты используешь, чтобы он знал, какому набору правил следовать. Мы будем использовать HTML5. Для этого надо, чтобы самой первой строкой в HTML-документе шла команда `<!DOCTYPE html>`. Название HTML5 означает, что это уже пятая редакция языка. Веб-разработчики все время совершенствуют свои инструменты, поэтому HTML эволюционирует, а теги входят в моду и устаревают, совсем как сленг или мемы.

Теперь начинается самое интересное. Поставь в начале документа тег `<html>`, а в конце — тег `</html>`. Весь остальной HTML-код должен находиться между этих тегов. Добавь туда какой-нибудь текст (это и называют *разметкой*), вот так:

```
<!DOCTYPE html>
<html>
  Моя первая страничка! Круто!
</html>
```

Помни, что пробельные символы и переводы строк в HTML не учитываются (точнее, браузер отобразит их как один-единственный пробел). Оставив между `<html>` и `</html>` пустые строки, мы лишь подготовили место для нового содержимого. Кроме того, код с отступами лучше читается. Сохрани документ под названием *test.html*.

Не используй в названиях HTML-страниц пробелы и специальные символы (например, #, \$, & и *). Если хочешь, чтобы имя файла состояло из нескольких слов, пиши их через дефис (например, *my-first-test.html*). Имена страниц чувствительны к регистру. Чтобы избежать ошибок, мы будем использовать только строчные буквы.

Эта страница уже будет работать в браузере, однако в ней отсутствуют некоторые стандартные части HTML-документа. Добавим в код еще несколько тегов, чтобы у нас получился каркас простейшей страницы. Введи между `<html>` и `</html>` теги `<head></head>` и `<body></body>` с переводом строки между ними, причем открывающий тег `<body>` должен быть перед фразой «Моя первая страничка! Круто!», а закрывающий `</body>` — после нее. В итоге у странички появится «тело» (`<body>`), где находится весь отображаемый контент, и «голова» (`<head>`), которая содержит информацию о документе (см. «Добавляем секцию head» на стр. 66). Для начала мы сосредоточимся на «теле» страницы. Твой документ должен выглядеть так:

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>

    Моя первая страничка! Круто!

  </body>
</html>
```

Еще раз сохрани файл и открой его в браузере, например в Chrome. Должен появиться текст, который ты ввел. Лично я вижу слова «Моя первая страничка! Круто!».

Кажется, что это мелочи. Но если кликнуть по странице правой кнопкой мышки и выбрать в меню пункт View page source, ты увидишь плоды своих трудов — HTML-код, из которого получилась эта страничка. Когда имя документа заканчивается на `.html`, браузер знает, что HTML-теги показывать не надо.

Переименуй файл в `test.txt`. Как думаешь, что выйдет, если открыть его в браузере? Попробуй, и ты увидишь: все теги теперь видны. Поскольку имя файла кончается на `.txt`, браузер считает его обычным текстом и отображает все содержимое, включая теги. Итак, расширение `.html` необходимо, чтобы браузер понимал, что делать со страничкой.

Снова поменяй имя на `test.html` и открой файл. Кстати, не нужно заново открывать документ каждый раз, когда ты что-то в него добавишь, — просто нажми в браузере кнопку «Обновить», и страница перезагрузится.

Использование основных HTML-тегов

Теперь можно добавить еще немного содержимого. Сперва попробуем тег `<p>`, создающий абзацы. Сотри тестовое сообщение и введи `<p>Привет, мир!</p>` между тегами `<body>` и `</body>`. Теги `<body>` обозначают начало и конец основного содержимого страницы. Любой текст, который ты добавишь внутрь этой пары тегов, отобразится в окне браузера. Сохранив и обновив страницу, ты увидишь новое сообщение, причем без тегов `<p>`. Отличная работа!

Теперь добавим второй абзац. Введи в конце первой строки `<p>Это второй абзац.</p>`. Должен получиться такой документ:

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>

    <p>Привет, мир!</p><p>Это второй абзац.</p>

  </body>
</html>
```

Обновив страницу в браузере, ты увидишь, что абзацы разделены пустой строкой, как на рис. 2.1.

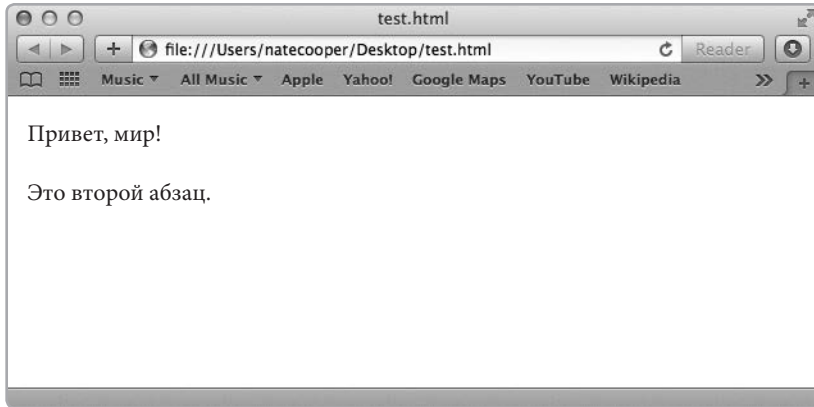


Рис. 2.1. Так наша страница должна выглядеть в браузере

Откуда между абзацами взялся отступ, если в HTML-коде они записаны на одной строке? Вспомни, что перевод строки в HTML роли не играет, так что начало и конец абзаца зависят только от тегов `<p>`.

Хотя ставить переводы строк в HTML необязательно, они помогают делать код читабельным. Запишем наш HTML-код в таком виде:

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>
    <p>Привет, мир!</p>
    <p>Это второй абзац.</p>
  </body>
</html>
```

Возможно, тебе бросилось в глаза, что я делаю отступы в начале строк. Они тоже не играют роли в HTML, но зато упрощают визуальное восприятие кода. Ты еще не раз увидишь написанный так код — это один из примеров общепринятых в HTML *соглашений*. Такой формат не только улучшает читабельность кода, но и помогает отследить, правильно ли закрыты теги.

Незакрытые теги — источник многих проблем. Познакомившись со следующими двумя тегами, мы увидим типичную ошибку такого рода. А сейчас добавим во второй абзац еще немного текста, оформив его с помощью тегов `` и ``.

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>
    <p>Привет, мир!</p>
    <p>Это второй абзац. <strong>Это жирный текст.</strong>
      <em>А это курсив.</em></p>
  </body>
</html>
```

Сохрани файл и обнови страницу. Фраза «**Это жирный текст**» должна отобразиться жирным шрифтом. Для такого выделения обычно используют тег ``. Фраза «*А это курсив*» будет набрана курсивом — это эффект тега ``.

Так что же случится, если вовремя не закрыть тег? Давай уберем закрывающий тег ``, сохраним файл и обновим страницу. Теперь весь текст, идущий после ``, стал жирным. Вот почему теги так важно закрывать. Если тег не закрыт, браузер не знает, когда его отключать, и тег продолжает действовать до конца страницы. В случае со `` мы видим неприятный эффект: несколько абзацев получили жирное начертание. Однако бывает еще хуже: иногда из-за незакрытого тега страница вообще перестает отображаться. Привыкай всегда закрывать теги и проверять код перед сохранением. И не забудь вернуть тег ``, который мы удалили!

Некоторые теги являются *одиночными*. Это значит, что у них нет закрывающей части — например, у тега `
`. Попробуй добавить `
` между фразами `Это жирный текст.` и `А это курсив.`.

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>
    <p>Привет, мир!</p>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
      <em>А это курсив.</em></p>
  </body>
</html>
```

Сохранив файл и обновив страничку, ты увидишь, что текст после `
` начинается с новой строки — как при использовании тега `<p>`, но с меньшим отступом. В текстовых процессорах это иногда называют *мягким переносом* строки.

Разделять контент можно не только абзацами и переводами строк — есть еще теги заголовка. HTML поддерживает заголовки шести уровней — от самого крупного, `<h1>`, до самого мелкого, `<h6>`. Обычно заголовки служат для создания разделов, как в книгах или статьях. Давай превратим абзац `<p>Привет, мир!</p>` в заголовок `<h1>`.

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
    <em>А это курсив.</em></p>
  </body>
</html>
```

Сохранив файл и обновив страницу, ты увидишь, что из-за тега `<h1>` фраза «Привет, мир!» стала крупнее. Будь наш документ длиннее, мы могли бы создать подразделы с заголовками меньшего размера, например `<h2></h2>` или `<h3></h3>`. Помни, что заголовки бывают шести уровней — от `<h1>` до `<h6>`. С их помощью можно визуально делить страницу на секции.

А теперь учимся создавать ссылки. Сначала это может показаться сложным, поскольку у тега ссылки есть особый атрибут с адресом. Выглядит это примерно так:

```
<a href="http://website.com">Клихни по мне!</a>
```

Открывающий тег URL Текст ссылки Закрывающий тег

Обрати внимание, какой здесь длинный открывающий тег! Атрибут `href` задает адрес сайта (URL), который будет загружен, если кликнуть по ссылке мышкой. Кликабельный текст ссылки, который видят посетители, вводится между открывающим и закрывающим тегами.

Попробуй сделать это самостоятельно, подставив любой URL.

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
    <em>А это курсив.</em></p>
    <p><a href="http://natecooper.co">Клихни сюда!</a></p>
  </body>
</html>
```

Открыв эту страничку в браузере, ты увидишь, что фраза «Клигни сюда!» подчеркнута и отображается синим цветом — так обозначаются гиперссылки. Перейдя по этой ссылке, ты попадешь прямоком на мой сайт. Каждая страничка в интернете имеет свой адрес (URL), который можно использовать для ссылок. Также можно связать ссылками разные страницы собственного сайта.

Очень важно знать, как размечать абзацы, заголовки, курсив, жирный текст и ссылки, поэтому постарайся запомнить эти теги и их назначение. Они встречаются практически на каждой веб-странице. Даже при работе с более сложными системами вроде WordPress знание основных HTML-тегов сослужит тебе хорошую службу.

Добавление картинок на веб-страницу

А теперь попробуем добавить картинку. Создавая свое портфолио, Ким выяснила, что это можно сделать разными способами и лучше быть настороже, чтобы не пригласить драконов. Для начала тебе понадобится изображение в формате JPG, GIF или PNG. Чтобы добавить его на страницу, используй тег ``. Как и `
`, этот тег одиночный, причем с подковыркой: для правильной работы ему нужно точно знать адрес картинки. Типичный тег `` выглядит так:

```

```

Как раз в строке `img src="path-to-image/image.jpg"` и таится сложность. Скажем, у меня на рабочем столе есть картинка. Открыв ее в браузере, я увижу в адресной строке путь к ней (см. рис. 2.2).

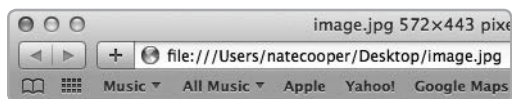


Рис. 2.2. Как узнать адрес картинки

Полный адрес картинки называется *абсолютным путем*, или *абсолютным URL*. Для нас это не лучший вариант, поскольку мы создаем страницу «локально» (то есть у себя на компьютере). Поэтому давай используем *относительный путь*. Просто возьмем и переложим картинку в ту же папку, где хранится HTML-документ. Тогда URL будет совсем простым.

Скопируй картинку в папку с веб-страницей, затем переименуй ее в `image.jpg` (или `image.gif`, или `image.png` — в зависимости от формата файла). В имени избегай пробелов и заглавных букв (как и в работе с HTML-страницами). Поскольку картинка и страница `test.html` теперь в одной папке, добавить картинку можно с помощью такого кода:

```

```

В итоге документ должен стать таким:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
    <em>А это курсив.</em></p>
    <p><a href="http://natecooper.co">Клигни сюда!</a></p>
    
  </body>
</html>
```

Сохрани файл и обнови страницу. Ну как, видишь картинку? Если нет, проверь, правильно ли введено имя файла и находится ли он в одной папке со страницей *test.html*.

Если ты делаешь портфолио, как Ким, и хочешь, чтобы все изображения находились в отдельной папке, вернись туда, где лежит страничка *test.html*, и создай подпапку с именем *images*. Затем перемести в нее файл с картинкой *image.jpg*. Для проверки обнови страницу в браузере, не трогая код. Должно появиться сообщение о неверной ссылке — то есть картинка больше не видна. Чтобы это исправить, нужно поменять путь в HTML-коде. Найди тег ``.

```

```

Вместо старого пути введи следующий:

```

```

Символ `/` означает переход в подпапку текущей папки. Например, если ты переместишь свою картинку в папку *black-and-white*, которая находится в папке *portfolio*, которая в свою очередь находится в папке *images*, нужно будет указать в коде такой путь: `images/portfolio/black-and-white/image.jpg`.

Преимущество относительных путей в том, что, в отличие от путей абсолютных, они не привязывают код к одному конкретному месту. Если тебе придется перенести весь сайт (в нашем случае — страницу *test.html* и папку *images*) в другое место или на новый хостинг, путь по-прежнему будет работать. С абсолютным путем так не выйдет: как ни перемещай файлы, он будет указывать на прежнее место, и картинка не загрузится.

Есть еще один способ задать относительный путь (но он подходит только для файлов на сервере) — поставить в начале пути `/`. Тогда браузер будет обращаться к корневой папке хостинга. Это удобно, если у тебя есть картинка, которая нужна на каждой странице (например, логотип). Написав нечто вроде ``, ты дашь браузеру понять, что файл *logo.gif* находится в корневой папке хостинга. Опять же, это упростит перенос сайта в другое место.

Добавляем секцию `head`

Если тебе удалось выполнить все предыдущие задания, сейчас твой HTML-код должен выглядеть так:

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
    <em>А это курсив.</em></p>
    <p><a href="http://natecooper.co">Клигни сюда!</a></p>
    
  </body>
</html>
```

Видишь секцию в начале, которая сейчас пустует?

```
<head>

</head>
```

До сих пор мы ограничивались тем, что отображается в окне браузера, и ничего не писали в секции `<head>`.

Секция `<body>` содержит все то, что показывается в окне браузера. До сих пор мы добавляли содержимое только туда. Теперь настало время познакомиться с `<title>` — тегом, который принадлежит секции `<head>`.

Все, что мы добавим в секцию `<head>`, либо появится за пределами основного окна браузера, либо будет невидимым. Давай поместим внутрь `<head>` код `<title> Моя веб-страничка</title>`.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя веб-страничка</title>
  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
    <em>А это курсив.</em></p>
    <p><a href="http://natecooper.co">Клигни сюда!</a></p>
    
  </body>
</html>
```

Сохрани и обнови страницу. Видишь разницу? В одних браузерах ее заметить легко, в других сложнее. Название, которое мы только что ввели, должно появиться в заголовке окна (см. рис. 2.3) или на вкладке браузера.



Рис. 2.3. Текст, стоящий между тегов <title>, появляется в заголовке окна браузера

Что еще может находиться внутри секции <head>? Возможно, тебе встречался термин *метаданные*. В отличие от содержимого документа, метаданные — это информация о нем. Тег <title> — тоже своего рода метаданные (в частности, название страницы). Еще в секции <head> могут быть теги <meta>, позволяющие подробно описать документ. В браузере ты этих описаний не увидишь, зато их считывают поисковые системы, индексирующие сайт. К примеру, тег <meta name="description" content="Описание страницы"> позволяет задать краткое описание страницы, которое появится в результатах поиска. Добавь этот тег, чтобы код стал таким:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя веб-страничка</title>
    <meta name="description" content="Тестовая страница, созданная
    для изучения HTML">
  </head>
  <body>
    <h1>Привет, мир!</h1>
```

```

<p>Это второй абзац. <strong>Это жирный текст.</strong><br>
<em>А это курсив.</em></p>
<p><a href="http://natecooper.co">Кликни сюда!</a></p>

</body>
</html>

```

Даже после того как ты сохранишь и обновить страницу, это описание нигде не появится, поскольку метаданные предназначены в основном для сторонних программ. Например, Google может вывести описание в поисковой выдаче, после ссылки (см. рис. 2.4).

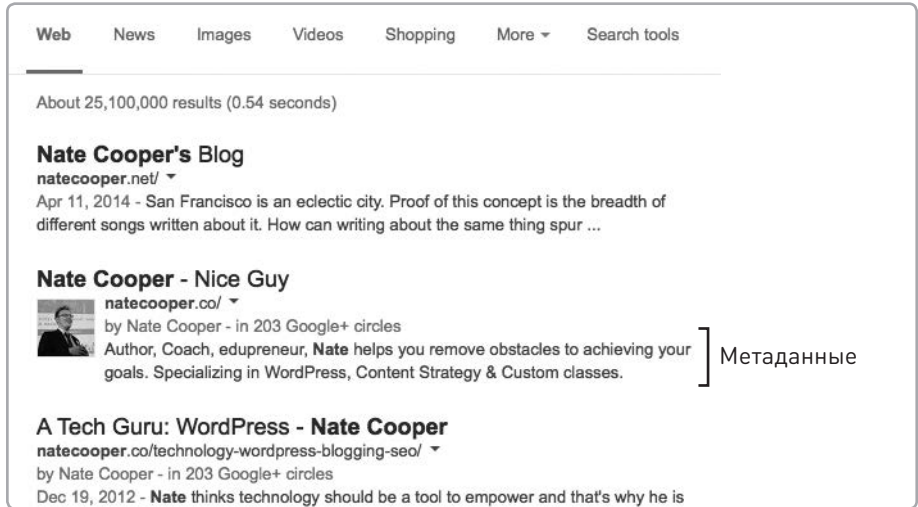


Рис. 2.4. Метаописание выводится после ссылки в результатах поиска

Хоть теги `<meta>` и считываются поисковиками, не надейся, что они волшебным образом поднимут твой поисковый рейтинг. Раньше они сильно влияли на него, но сегодня это отнюдь не решающий фактор. Тем не менее добавить тег `<meta name="description">` в код страницы лишним не будет.

Теперь у нас есть готовая страничка. Можно приступать к знакомству с CSS. Каждый HTML-документ должен обладать изученной нами базовой структурой, в которую входят секция `<head>` с метаданными и не отображаемой на странице информацией и секция `<body>`, содержимое которой видно в браузере. Советую тебе повторить основные теги, изученные в этой главе. Они наверняка понадобятся еще не раз.

Несколько полезных HTML-тегов

Вот тебе для справки перечень часто используемых тегов.

`<!DOCTYPE html>`

Сообщает браузеру, какую версию HTML ты собираешься использовать (в нашем случае HTML5). Отмечу, что сама эта строчка не HTML-элемент, а объявление, которое должно быть в самом верху документа, перед открывающим тегом `<html>`.

`<html></html>`

Все, что находится внутри этих тегов, браузер будет обрабатывать как HTML-код. Теги `<html>` и `</html>` должны быть в каждом HTML-документе.

`<head></head>`

Внутри этих тегов можно задавать метаданные, которые обрабатываются браузером, но не отображаются в его окне.

`<body></body>`

Все, что находится внутри этих тегов, будет показано в окне браузера. По сути это и есть содержимое страницы.

`<p></p>`

Текст внутри этих тегов браузер воспринимает как отдельный абзац.

``

Этими тегами помечают особо важный текст, который обычно выделяется жирным шрифтом.

``

Этими тегами помечают текст, который нужно акцентировать. Как правило, он выделяется курсивом.

`Нажмите сюда`

Это тег гиперссылки. На странице будет показано только ее название — в данном случае это «Нажмите сюда». Название обычно подчеркивается и выводится другим цветом — сразу видно, что по нему можно кликнуть. Адрес ссылки задается атрибутом `href` и берется в кавычки. Для ссылок на внешние источники используй абсолютный URL.

`<h1></h1>`, `<h2></h2>`, `<h3></h3>`, `<h4></h4>`, `<h5></h5>`, `<h6></h6>`

Это теги заголовков. Подобно абзацу, заголовок отделен от соседних элементов. Отсюда следует, что внутри абзаца заголовков быть не должно. Номер обозначает важность (уровень) заголовка, причем наименьший номер (`h1`) соответствует наибольшей важности, а наибольший номер (`h6`) — наименьшей. От важности заголовка обычно зависит размер его шрифта. Чаще всего заголовки выводятся крупными жирными буквами, однако CSS позволяет менять их стиль по твоему усмотрению.

`
`

Этот тег задает перевод строки. Он является одиночным, то есть не содержит текста и его не нужно закрывать. `
` можно использовать внутри другого элемента, например абзаца, чтобы начать текст с новой строки. Вертикальный отступ перед началом новой строки, как правило, меньше, чем расстояние между абзацами.

```

```

Этот одиночный тег добавляет на страницу изображение. В атрибуте src задается абсолютный или относительный путь к файлу с картинкой. С помощью alt можно сделать описание картинка для поисковых систем.

```
<title>Моя страничка</title>
```

Тег <title> задает название HTML-документа. Обычно оно показано в заголовке окна браузера. Поисковые системы часто используют <title>, чтобы определить тематику страницы. Этот тег должен находиться в секции <head>.

```
<meta name="description" content="Описание твоего сайта">
```

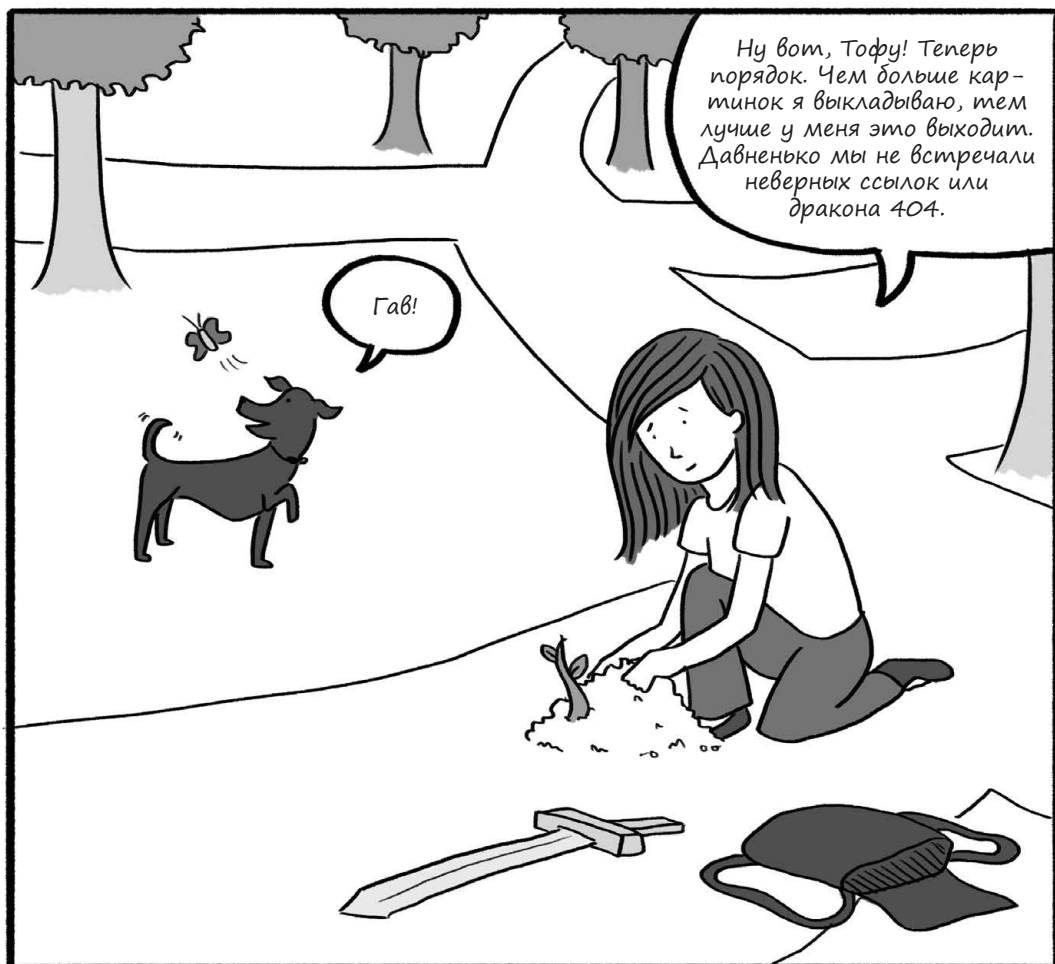
Тег, который задает краткое описание страницы для поисковых систем вроде Google. Посетителям сайта оно, как и прочие метаданные, не показывается. Этот тег должен находиться в секции <head>.

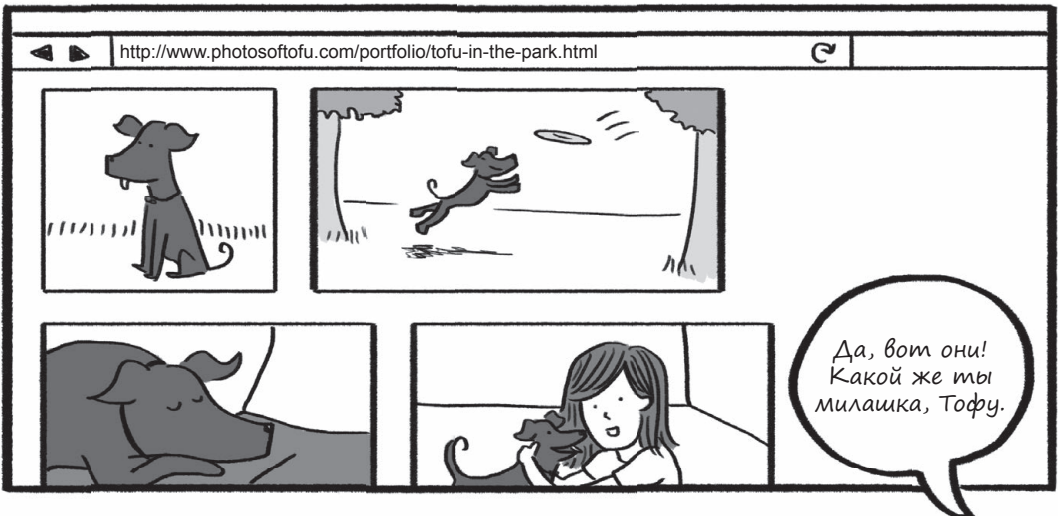
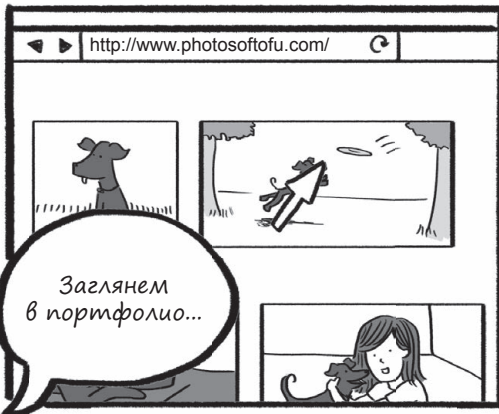
Разумеется, HTML-тегов гораздо больше, чем здесь перечислено. Но этими ты будешь пользоваться особенно часто. Познакомившись с основами, ты заложишь базу для дальнейшего обучения. А теперь давай посмотрим, как дела у Ким!

3

Ким наводит красоту
с помощью CSS







Знакомство с CSS





font-size: 12px;

color: #fff;

font-family: sans-serif;

border: thin solid 1px #000000;



Я Глинда, добрая
CSS-колдунья.

CSS — это каскадные
таблицы стилей. HTML
задает структуру
и содержание, а CSS —
оформление.

Например, при
помощи CSS ты
можешь выбрать
шрифт для всех
абзацев на своей
страничке.

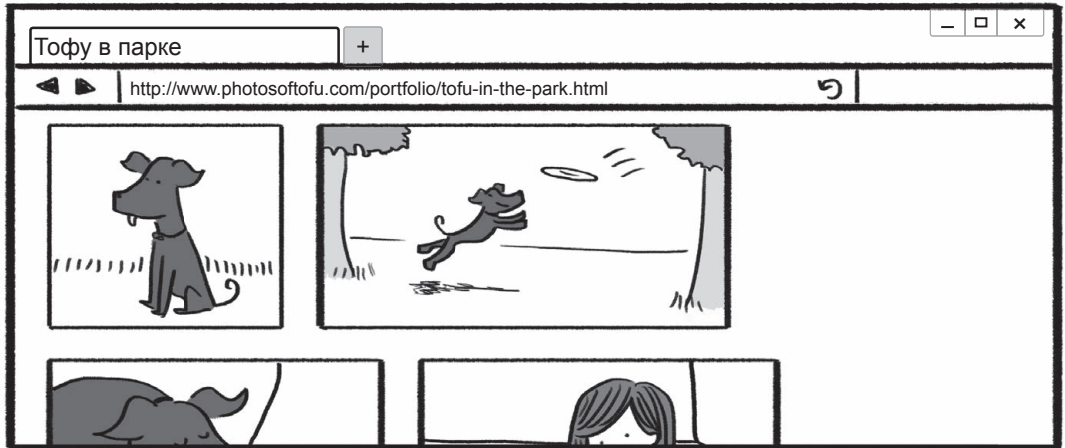


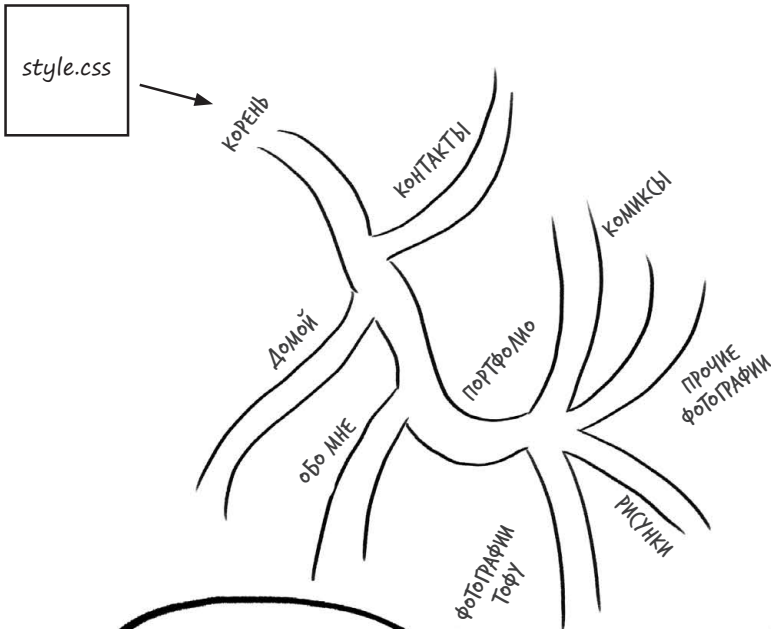
Чтобы CSS-таблица работала, нужно правильно подключить ее в секции `<head>` твоего HTML-документа. Вспомни его основные части:

```
<html>
  <head>
  </head>
  <body>
    <p>В прошлое воскресенье мы с Тофу гуляли в Центральном парке.</p>
    <p></p>
    <p></p>
    ...
  </body>
</html>
```

То, что мы добавляем в секцию `<body>`, отображается в окне браузера. Но содержимого `<head>` ты на странице не увидишь. Хороший пример — название документа.

```
<head>
  <title>Тофу в парке</title>
</head>
```





Обычно CSS находится в отдельном файле, лежащем в корне сайта. На сайтах посложнее бывает несколько CSS-файлов, но нам хватит и одного. Назвать его можно как угодно, лишь бы в конце было .css. Например, style.css.



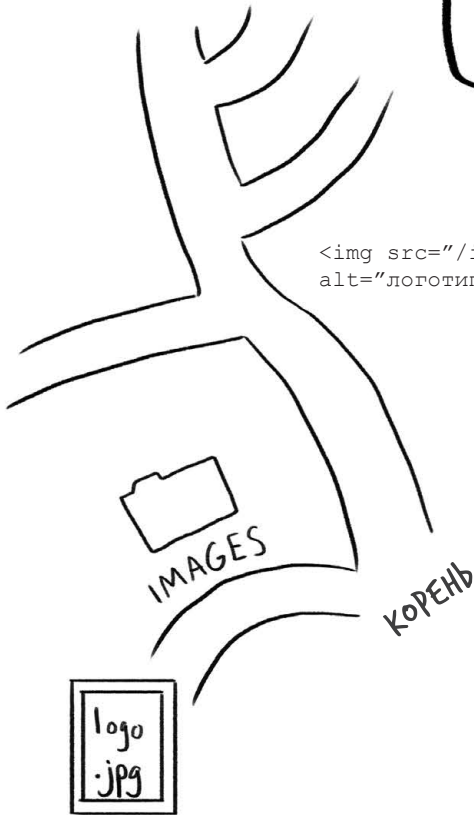
Так, мы помещаем этот файл в корень сайта, а дальше что?

```
<head>
<title>Тофу в парке</title>
<link rel="stylesheet" href="style.css">
</head>
```

Дальше нужно подключить CSS к страницам сайта, добавив особый тег в <head> каждого из документов. Выглядит это так:



Ага. Символ / можно использовать где угодно. Предположим, у тебя в папке `images` лежит логотип, который нужно показывать на всех страницах сайта.



```

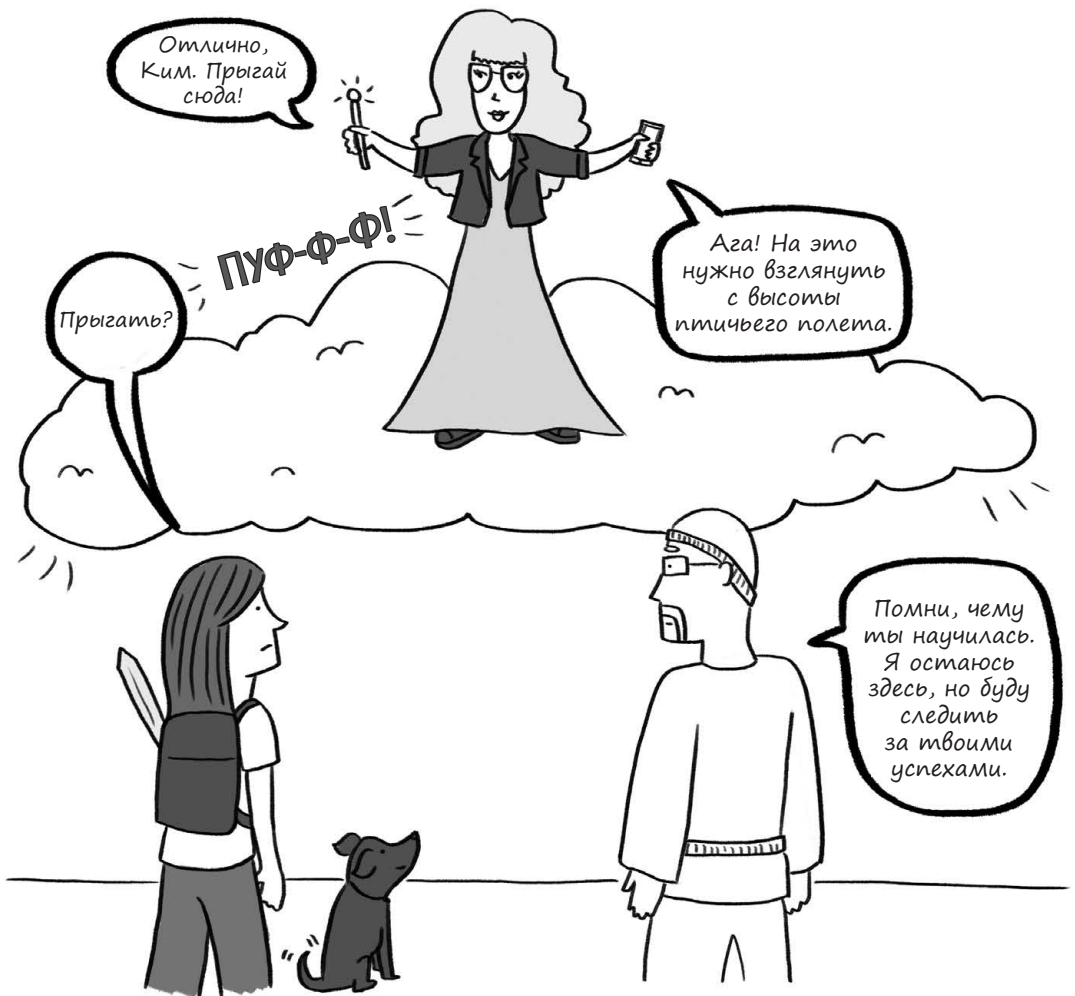
```

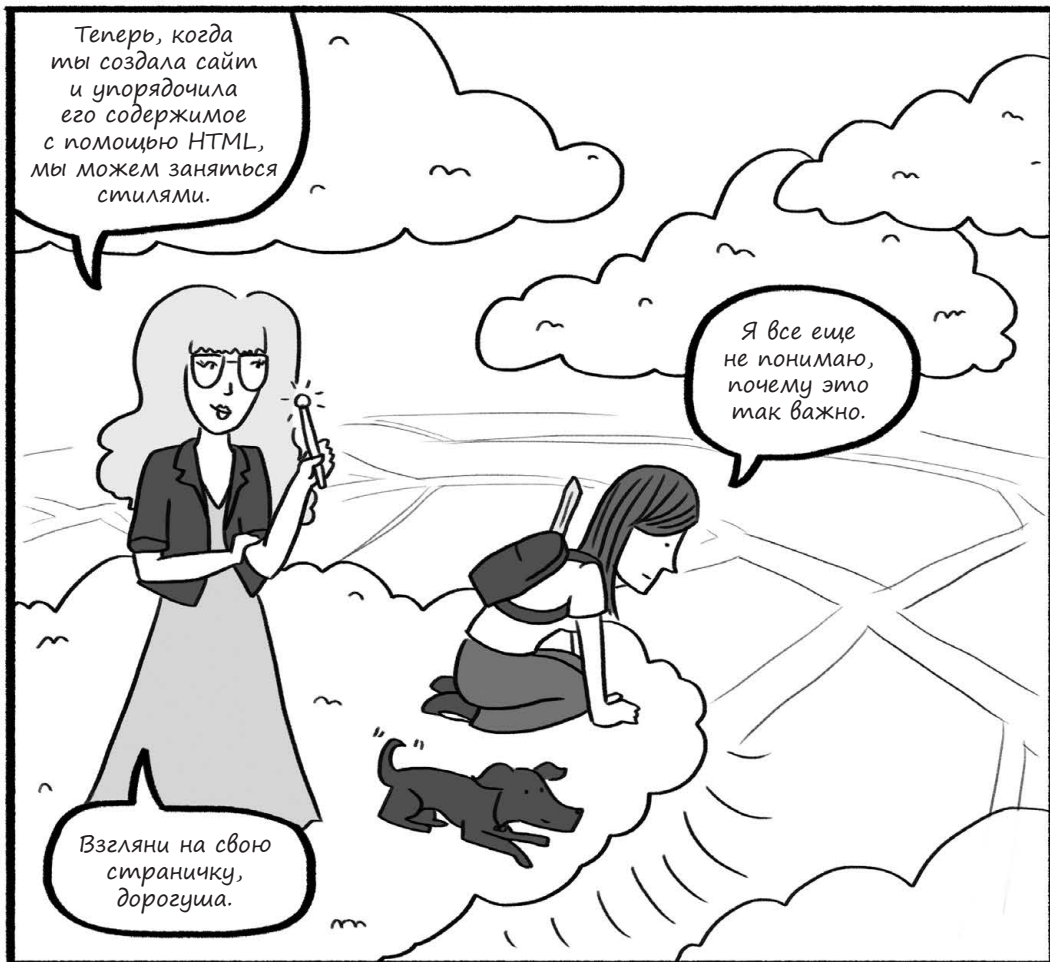
Используя этот код, ты запросто добавишь картинку на любую страницу сайта. Поскольку путь начинается с `/`, браузер поймет, что файл всегда нужно искать, начиная с корня.

Кстати, это грамотный подход. Если логотип нужно разместить в нескольких местах, лучше использовать одну и ту же картинку. Тогда сайт будет грузиться быстрее — посетители такое любят.

Я уже спец по грамотным подходам.

Ким изучает основы CSS





Ну и как тебе этот шрифт?



Раньше я об этом не думала, но мне кажется, буквы слишком угловатые. Из-за этих выступов он напоминает печатную машинку. Давай попробуем шрифт без засечек, например Helvetica или Arial.



ПУ-УФ!

Тофу любит бегать по травке, особенно если погода хорошая.

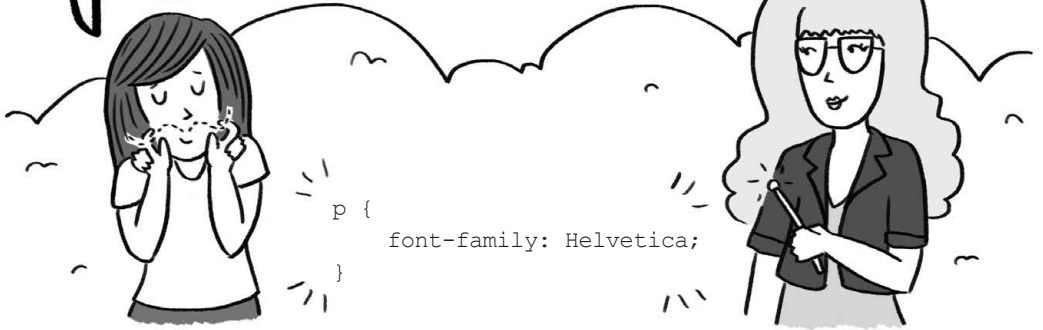
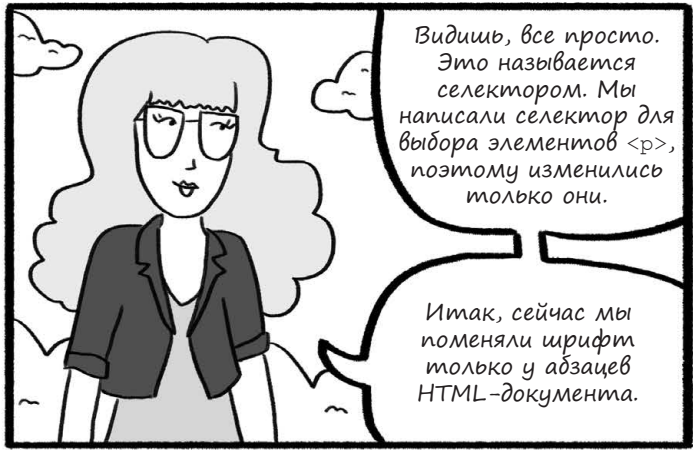


Как вы это сделали?

С помощью CSS! Видишь ли, CSS позволяет изменять вид отдельных HTML-элементов через их теги. Весь твой текст заключен в теги `<p>`, поэтому чтобы поменять разом все абзацы, хватит лишь одной строчки CSS, например такой:

```
p {font-family: Helvetica;}
```







Шрифты — дело особое. Они должны быть установлены на компьютере посетителя, если только не подгружать их из сети.

Так вот, на большинстве компьютеров Windows нет шрифта Helvetica. Но после него я указала Arial. Это значит, что, когда нет Helvetica, подойдет и он. А если нет ни того ни другого, сгодится любой из шрифтов без засечек — sans-serif.



Вы ведь и размер меняли? Я почему-то не вижу разницы.



Верно подмечено! Если не указывать размер шрифта, компьютер использует настройку по умолчанию. У тебя это как раз 16px, вот ничего и не поменялось.



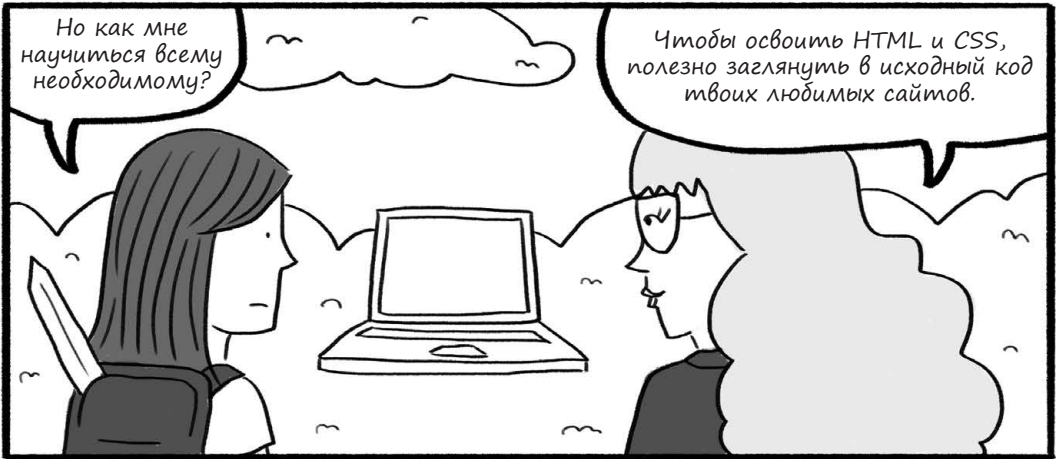
Но у смартфона, например, может стоять другой шрифт по умолчанию. Подробно описав CSS-стили, мы получим сайт, который одинаково выглядит в разных браузерах и операционных системах. Есть и другие способы задать размер шрифта:

- px Фиксированный размер в пикселях (точках компьютерного экрана)
- pt Фиксированный размер в пунктах (1/72 дюйма)
- em Относительный размер (1em — это размер шрифта по умолчанию, 2em — вдвое больше него и так далее)
- % Относительный размер в процентах от размера по умолчанию





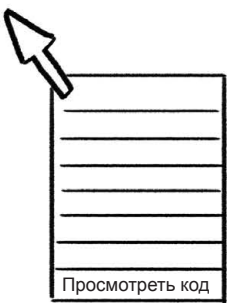
Углубляясь в CSS

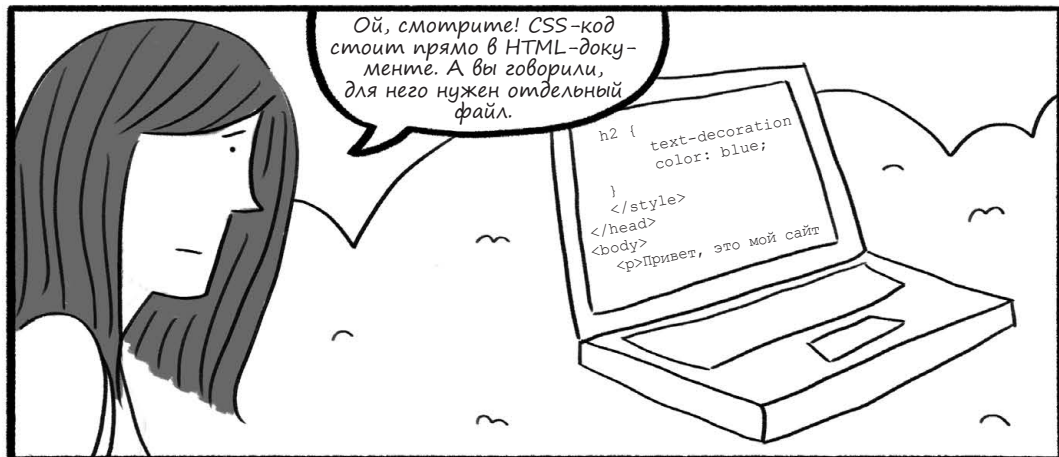


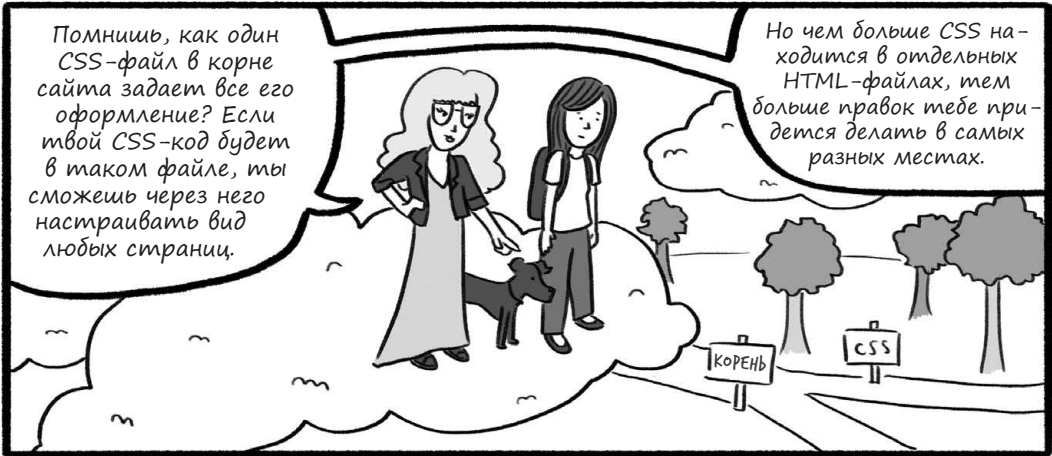
МАГИЯ ПРАВОЙ КНОПКИ!

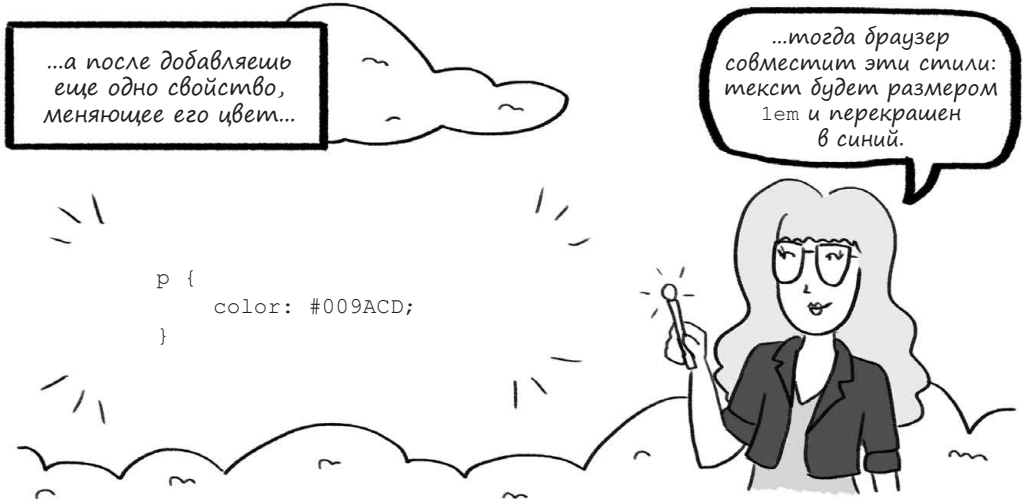
Клики по странице правой кнопкой мыши. Так ты увидишь и HTML, и CSS.

Многие браузеры позволяют изучать отдельные элементы страницы, в том числе смотреть их CSS-код.

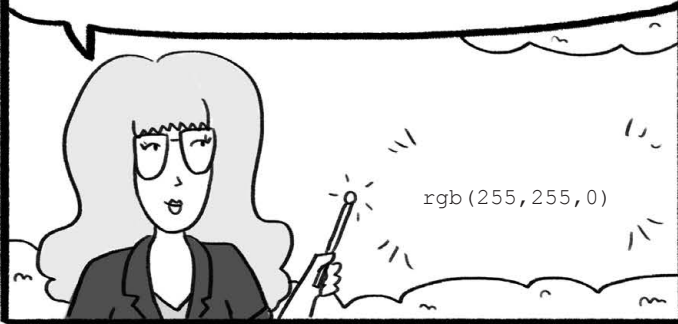








Здесь я задала один из оттенков синего. Еще можно использовать названия вроде blue и red. Можно даже указывать насыщенность красного, зеленого и синего по отдельности:



Ой, боюсь, с этим будут проблемы.



В смысле?

Что, если я сделаю несколько строк на разных страницах синими, а потом передумаю? Если я задам этот стиль в HTML, мне придется вспомнить и вручную изменить каждую страницу.



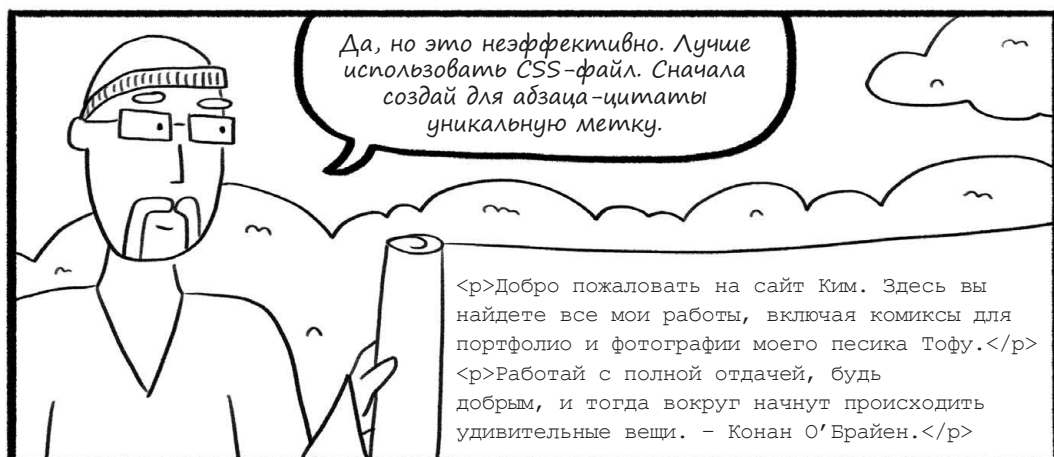
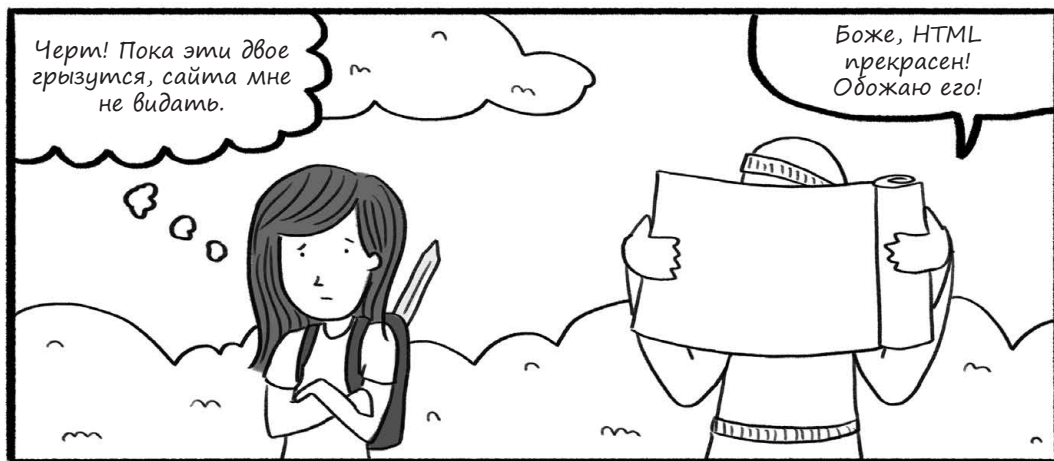
Именно. Поэтому существуют продвинутые способы выборочной привязки стилей к разным частям разных страниц.

Ох... Не знаю, готова ли я к чему-то «продвинутому».



Ким узнает о классах и метках CSS





Кажется, я поняла.
Для обоих абзацев
используется тег `<p>`,
поэтому сейчас их нельзя
различить в CSS-коде.

Однако с помощью атрибутов
`id` и `class` элементам можно
присваивать уникальные
метки. Тебе уже встречались
некоторые атрибуты: `href`
для ссылок, `src` и `alt` для
изображений. Изменим наш
HTML-код так:

```
<p>Добро пожаловать на сайт Ким. Здесь вы найдете все мои работы,  
включая комиксы для портфолио и фотографии моего песика Тофу.</p>  
<p id="quote">Работай с полной отдачей, будь добрым, и тогда  
вокруг начнут происходить удивительные вещи. – Конан О'Брайен</p>
```

Что значит
`id="quote"`?
Это подходит
только для цитат?
И что получится
в итоге?

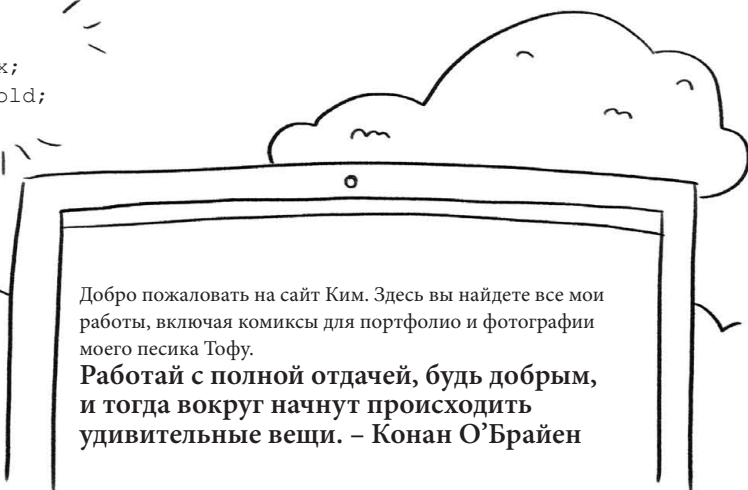
Атрибут `id`
задает метку.
Имя подойдет
любое, главное —
чтобы оно было
уникальным.

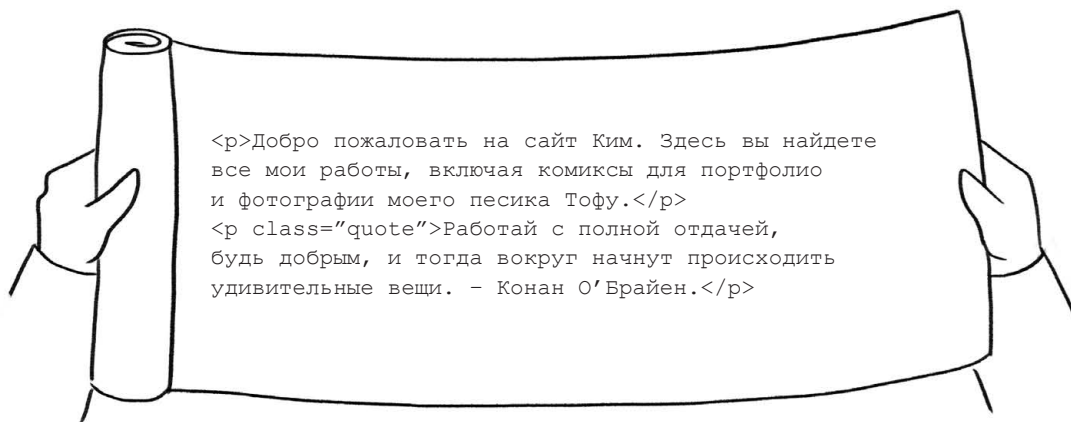
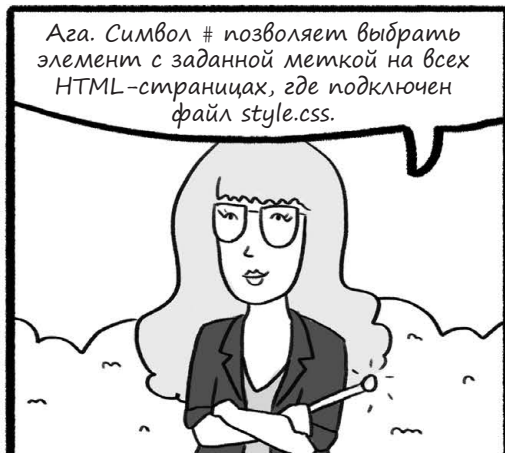
И тогда
у цитаты
изменится
шрифт?

Не совсем...



```
#quote {  
  font-size: 18px;  
  font-weight: bold;  
}
```





Но тогда изменится и CSS-код. Вот как назначить стиль для класса:

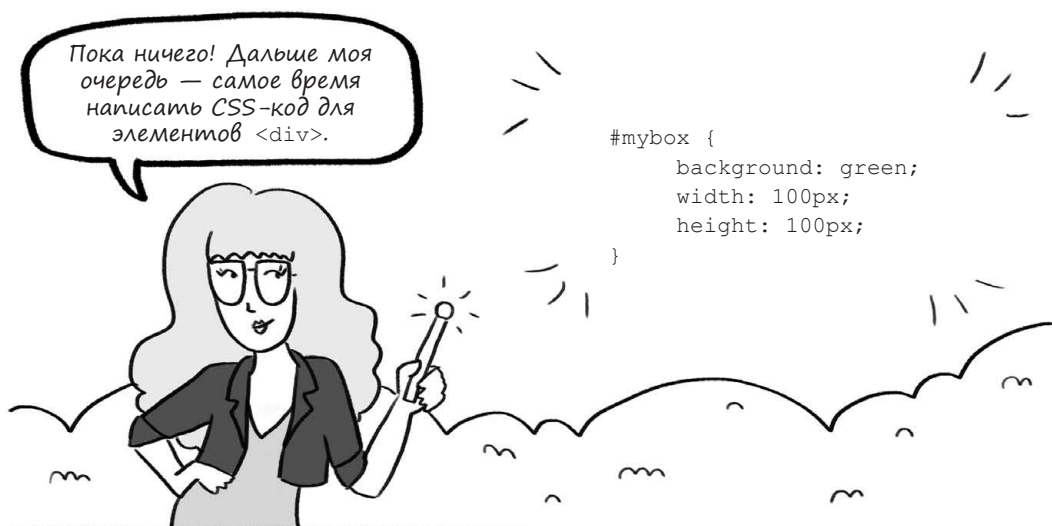
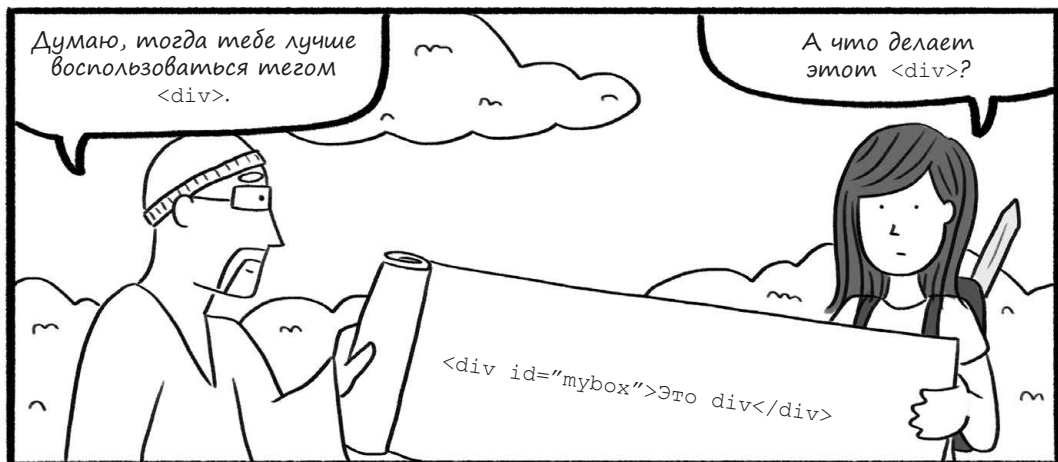
```
.quote {  
  font-size: 18px;  
  font-weight: bold;  
}
```

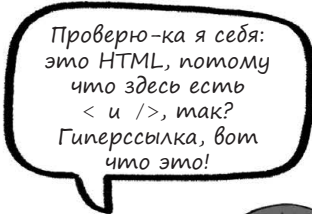
Нужно ставить # для id, и . для class, верно?

Да. И помни, что это подходит не только для абзацев — так ты можешь обратиться к любому тегу в HTML-документе.

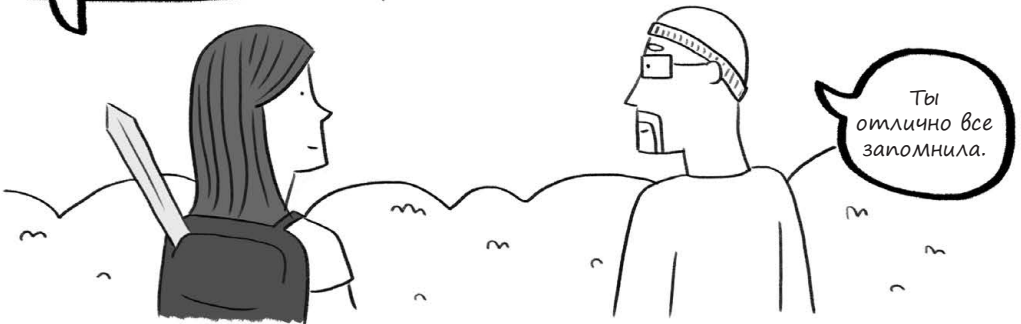
А если я хочу добавить на страницу фрагмент текста, не помещая его внутрь абзаца? Например, сделать колонку, как в газете?

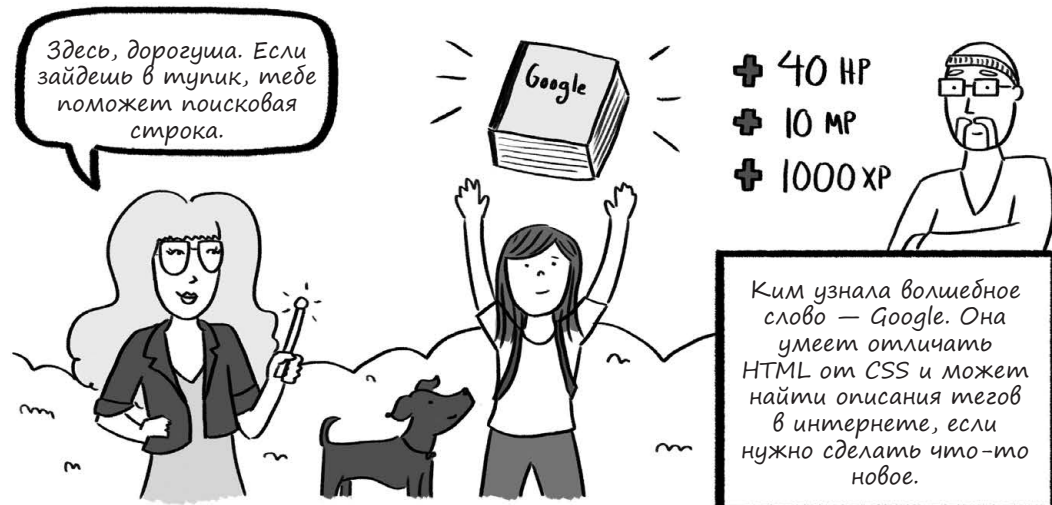
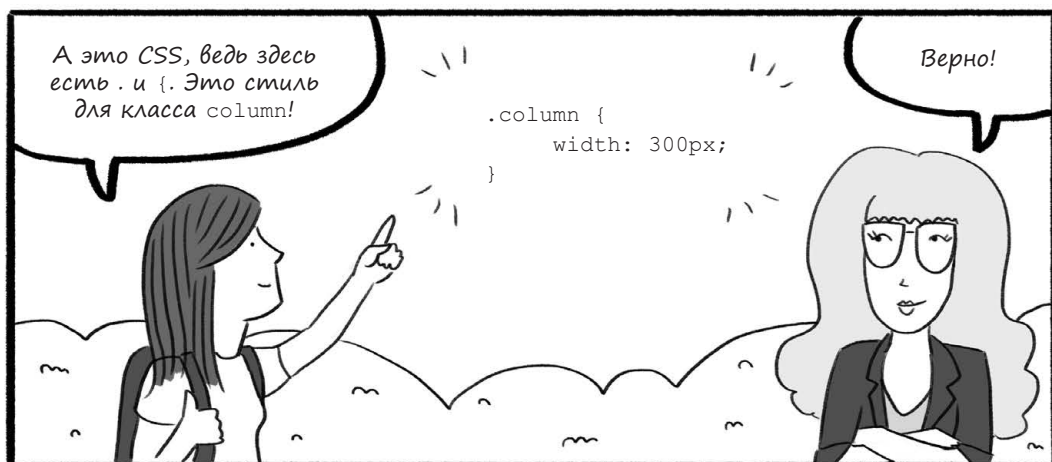
Новости все еще печатают на бумаге?





`Клихни сюда`





Каскадные таблицы стилей

Прежде чем углубиться в каскадные таблицы стилей (CSS), нужно еще немного рассказать о подключении CSS к HTML. Эти два языка работают в связке и зависят один от другого. HTML похож на архитектора, который возводит каркас дома, то есть задает базовую структуру страницы. А CSS напоминает дизайнера интерьеров, решающего, как что должно выглядеть, в какой цвет красить стены и куда ставить мебель. В этой главе мы выясним, как добавить скелету HTML-кода стиля и изящества при помощи CSS.

Есть несколько способов подключить CSS к сайту. Можно использовать *встроенный* CSS, то есть добавлять его прямо к тегам, а можно прописать CSS-код в шапке документа, внутри секции `<head>`. Но чаще всего для CSS создают отдельный файл, подключая его к каждой HTML-странице. Это самый надежный и распространенный способ, и в данной главе мы будем делать именно так.

Создание таблицы стилей и ее подключение к HTML

Термин «каскадные таблицы стилей» пришел из мира журналистики. *Таблица стилей* — это способ оформления журнальных страниц в программах издательской верстки, когда шрифты, отступы и все остальное задано в отдельном файле. Это позволяет управлять версткой журнала, так чтобы изменения стилей применялись сразу ко всем страницам. Используя отдельный CSS-файл, ты берешь пример с проверенных систем журнальной верстки. Для этого нужно создать CSS-файл и упомянуть его в секции `<head>` каждого из HTML-документов. В итоге получится структура как на рис. 3.1.

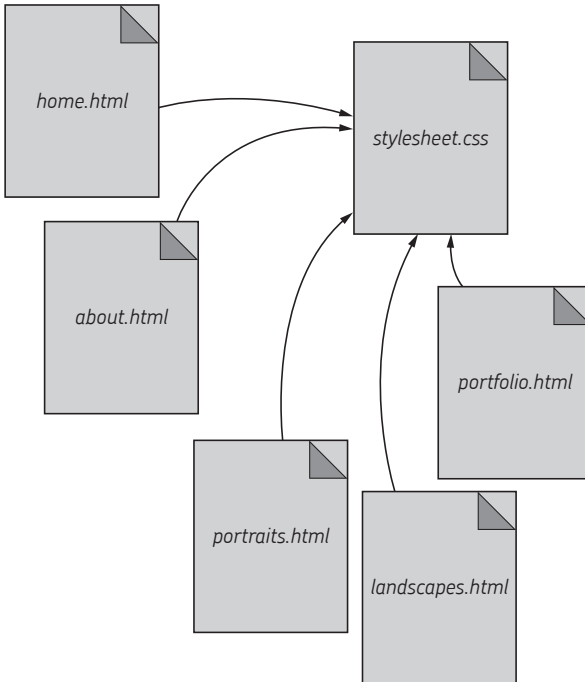


Рис. 3.1. Все страницы сайта могут ссылаться на один общий CSS-файл

Твоя первая таблица стилей

Запусти редактор кода и создай новый документ под названием *style.css*. В принципе, можно дать ему и другое имя, которое заканчивается на *.css*. Как и в случае с HTML-кодом, это будет обычный текстовый файл, однако по расширению *.css* браузер и сервер определяют, что внутри находится CSS-код. Сохрани файл в той же папке, где лежит твоя страничка *test.html*. Это важно, ведь подключить CSS-файл к HTML-документу можно, лишь зная его путь, — так же было в случае с картинками.

Теперь отредактируй файл *test.html*, добавив в секцию `<head>` следующий код: `<link href="style.css" rel="stylesheet">`. Неважно, введешь ты эту строчку перед тегами `<title>` и `<meta>` или после них (главное, чтобы она находилась между `<head>` и `</head>`). Я добавил ее прямо перед тегом `</head>`. Теперь твой HTML-документ должен выглядеть так:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя веб-страничка</title>
    <meta name="description" content="Тестовая страница, созданная
    для изучения HTML">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
    <em>А это курсив.</em></p>
    <p><a href="http://natecooper.co">Клигни сюда!</a></p>
    
  </body>
</html>
```

Выделенная строчка — это все, что нужно для подключения нашей таблицы стилей к HTML-документу. Чтобы все страницы сайта использовали один CSS-файл, мы добавим точно такой же код в секцию `<head>` каждого документа.

Таблица стилей — важная часть современного сайта, поэтому обычно ее хранят в главной, или «корневой», папке. Как мы уже знаем из недавних приключений Ким, чтобы сослаться на корень сайта, достаточно поставить в начале пути символ `/`. Например, для страницы в папке *portfolio* ссылка на таблицу стилей будет выглядеть так:

```
<link href="/style.css" rel="stylesheet">
```

Такая строчка в секции `<head>` позволит тебе задать общую таблицу стилей для множества HTML-страниц, даже если они находятся в разных папках.

Чтобы научиться работать с путями, тебе понадобятся время и опыт. Но не пугайся. Главное — помнить, что после каждой папки нужно добавлять символ `/` и что относительные пути начинаются с текущей папки. Поэтому если файл находится в той же самой папке, добавлять `/` не надо, а если в другой — используй `/`, чтобы обозначить переход на уровень выше. Сейчас твои файлы находятся на компьютере, а не на сервере, поэтому храни *style.css* там же, где и *test.html*, ссылаясь на него только по имени.

CSS: язык стилей

Теперь у нас есть отдельная таблица стилей *style.css* и ссылка на нее из *test.html*. Пора этим воспользоваться. Открой пустой файл *style.css* и напиши там:

```
p {  
  font-family: Helvetica, Arial, sans-serif;  
}
```

Сохрани файл и обнови страницу *test.html* в браузере. Видишь разницу? У всех абзацев должен смениться шрифт. Как это получилось? Наш тег `<link>`, добавленный в секцию `<head>`, означает, что HTML-страница использует свойства, заданные в файле *style.css*. Чтобы добавить CSS-свойство, первым делом нужно указать селектор. *Селектор* — это идентификатор, указывающий на фрагменты HTML, которые ты хочешь изменить. В данном случае мы использовали селектор `p`, управляющий HTML-тегами `<p>`. Мы задали для него свойство `font-family` со значением `Helvetica, Arial, sans-serif`. Оно позволяет сменить шрифт, а его значение указывает, какой именно шрифт нужно выбрать. В нашем случае браузер получает указание выбрать шрифт `Helvetica`, а если он недоступен — шрифт `Arial`. Если же и его нет, следует взять любой доступный шрифт семейства `sans-serif`. У таких шрифтов нет засечек — коротких перпендикулярных штрихов, которыми оканчиваются все линии, например как в `Times New Roman` (см. рис. 3.2).



Рис. 3.2. Шрифты без засечек, например *Arial* (слева), выглядят более современно, а у шрифтов с засечками, например *Times New Roman* (справа), есть фигурные выступы

Как и HTML, CSS нечувствителен к пробелам и переводам строк. Даже если записать наш код одной строкой, он будет работать так же:

```
p {font-family: helvetica, arial, sans-serif; }
```

Повторюсь: отступы принято делать исключительно для того, чтобы код легче читался.

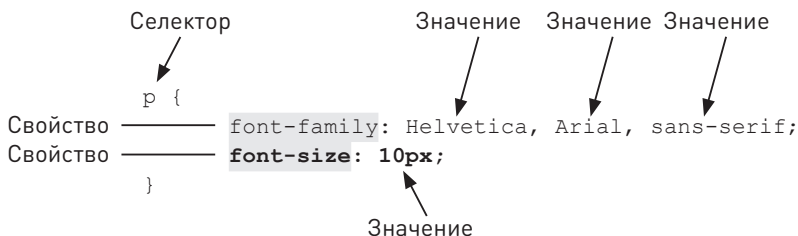
Итак, мы указали, что все абзацы должны следовать заданному свойству, ведь `p` в нашем CSS соответствует всем элементам `<p>` во всех HTML-документах, использующих эту таблицу стилей. Думаю, ты уже начинаешь понимать, почему CSS — очень мощный инструмент. Представь, что наш сайт состоит из 10, 20, 200 или 2000 страниц. Если каждая страница использует файл *style.css*, изменив только его, мы сможем поменять шрифты сразу на всем сайте.

Синтаксис CSS

CSS вовсе не похож на HTML, правда? Это другой язык, а значит, при написании CSS-кода нужно соблюдать другие правила. В CSS есть две ключевые конструкции: селектор и фигурные скобки `{}` после него. Селектор обозначает, у каких HTML-элементов (например, `<p>` или ``) нужно поменять свойства. Сами свойства задаются внутри фигурных скобок.

Значения свойств нужно вводить после двоеточия (:). Если их несколько, то через запятую. Для разделения разных свойств служит точка с запятой (;). Нарушишь хоть одно из этих правил — и CSS работать не будет!

Добавим к селектору абзацев еще одно свойство и посмотрим, что выйдет. Нажми Enter (или Return на Mac OS) в конце строки с `font-family` и задай свойству `font-size` (размер шрифта) значение `10px`. Код должен стать таким:



Сохрани файл `style.css` и обнови страничку `test.html` в браузере. Помни: поскольку мы подключили `style.css` к `test.html`, любые изменения в `style.css` автоматически отразятся на `test.html`. Размер шрифта абзацев должен уменьшиться.

Размеры в CSS можно задавать с помощью разных единиц измерения. Для шрифтов чаще всего выбирают `px` и `em`. В последнем примере мы использовали `px`, то есть число пикселей. Поскольку количество и размер экранных пикселей зависит от разрешения монитора, на маленьком экране (где пикселей меньше) шрифт 10-го размера может выглядеть крупнее, а на огромном (где куча пикселей) — мельче.

С другой стороны, если задать размер в `em`, он будет зависеть от размера шрифта по умолчанию. В результате шрифт будет подстраиваться под браузер, в котором открыта страничка. На мобильных устройствах размер шрифта по умолчанию обычно меньше, чем в браузерах для настольных компьютеров, однако этому размеру всегда соответствует значение `1em`.

Задав размер меньше `1em` (например, `0.8em`), ты получишь шрифт мельче заданного по умолчанию, а задав размер больше `1em` (например, `1.2em`) — крупнее. Если нужно, чтобы сайт гибко адаптировался к разным браузерам, есть смысл использовать `em`. Но пока ты только осваиваешься с размерами, лучше остановиться на `px`.

Не спеши хвататься за голову. Как ты уже знаешь на примере других технологий, часто существует несколько способов сделать одно и то же, но получить при этом разные результаты. Представь, что ты рисуешь на холсте. Можно использовать валик, или губку, или кисть любого размера — вопрос лишь в том, что лучше подходит для твоих целей. За одну ночь Рембрандтом не станешь, но со временем ты начнешь понимать достоинства и недостатки инструментов вроде `px` и `em`.

Давай посмотрим, как выглядит наша страница в браузере (рис. 3.3).

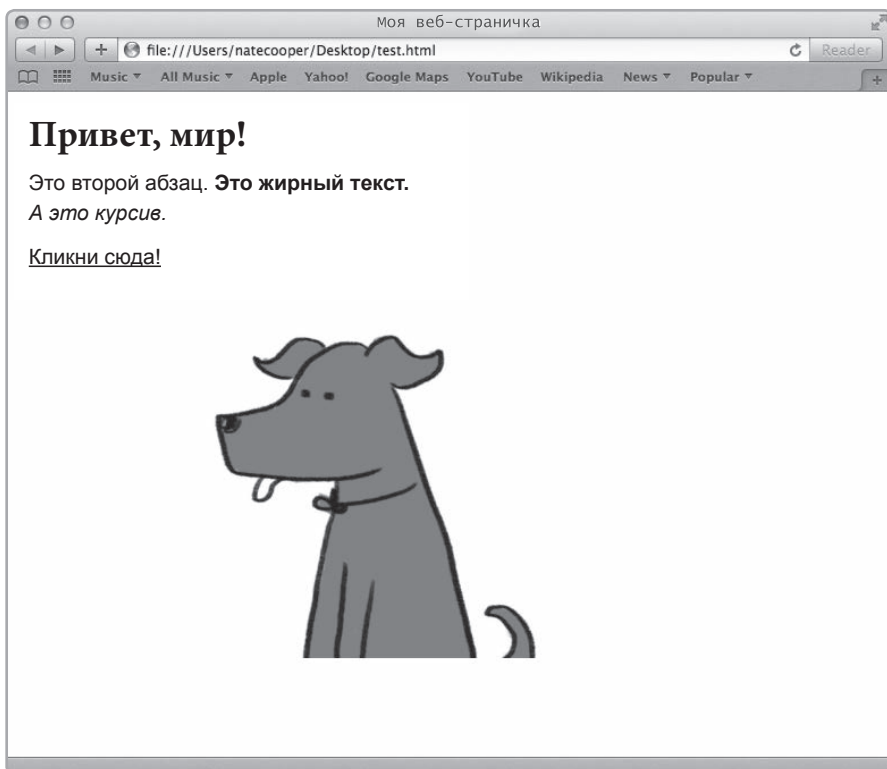


Рис. 3.3. Свойства шрифта из нашего CSS применяются к абзацам, но не к заголовкам

До сих пор мы меняли лишь один элемент нашего HTML — тег `<p>`. При этом фраза «Привет, мир!» осталась прежней. Она заключена в теги `<h1>`, а значит, ни одно из свойств, что мы задали для селектора `p`, на нее не влияет. Предположим, шрифт тегов `<h1>` нам тоже хочется поменять. Открой CSS-файл и назначь селектору `h1` такие же свойства, как у абзацев, только размер шрифта пусть будет `1.3em`. А размер шрифта абзацев поменяй на `.8em`. Код должен стать таким:

```
p {
    font-family: Helvetica, Arial, sans-serif;
    font-size: .8em;
}
h1 {
    font-family: Helvetica, Arial, sans-serif;
    font-size: 1.3em;
}
```

Сохрани файл и обнови страницу *test.html* в браузере. Шрифт изменился. Возможно, ты спросишь: «Разве заголовок `<h1>` не был и без того большего размера? Зачем задавать ему свойство `font-size`, если он по умолчанию больше `<p>?`» Отличный вопрос. Для всего, чем мы не управляем явно через CSS, используются настройки по умолчанию. Почти во всех браузерах `<h1>` больше `<p>`, однако *насколько* больше — зависит от конкретного браузера. CSS позволяет взять управление в свои руки и сказать браузеру, каким именно должен быть тот или иной элемент. С помощью CSS мы шлифуем базовую структуру, заданную в HTML-файле, чтобы сайт получился точно таким, как нам нужно.

Классы, метки и наследование

Мы изучили основы работы со стилями в CSS. Теперь тебе должно быть понятно, что CSS позволяет менять стили любых HTML-элементов по их тегам. Но что, если тебе нужно настроить один конкретный элемент? Допустим, это будет цитата, которая должна отличаться от прочего текста. Открой свой HTML-документ и добавь код, выделенный жирным:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя веб-страничка</title>
    <meta name="description" content="Тестовая страница, созданная для
    изучения HTML">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
    <em>А это курсив.</em></p>
    <p><a href="http://natecooper.co">Клигни сюда!</a></p>
    
    <p><b>Я никогда не позволял школе вмешиваться в мое образование. —
    Марк Твен</b></p>
  </body>
</html>
```

Результат ожидаемый (см. рис. 3.4). Мы задали в свойствах абзаца определенный размер и тип шрифта, и теперь все абзацы выглядят соответствующе.

Но мы хотим, чтобы у цитаты были другие свойства. В коде этот абзац ничем не отличается от прочих. Как же обратиться к нему в таблице стилей? Нужно присвоить этому абзацу особое имя. Открой файл *test.html* в редакторе и добавь к тегу `<p>` следующий код:

```
<p id="quote">Я никогда не позволял школе вмешиваться в мое образование. —
Марк Твен</p>
```

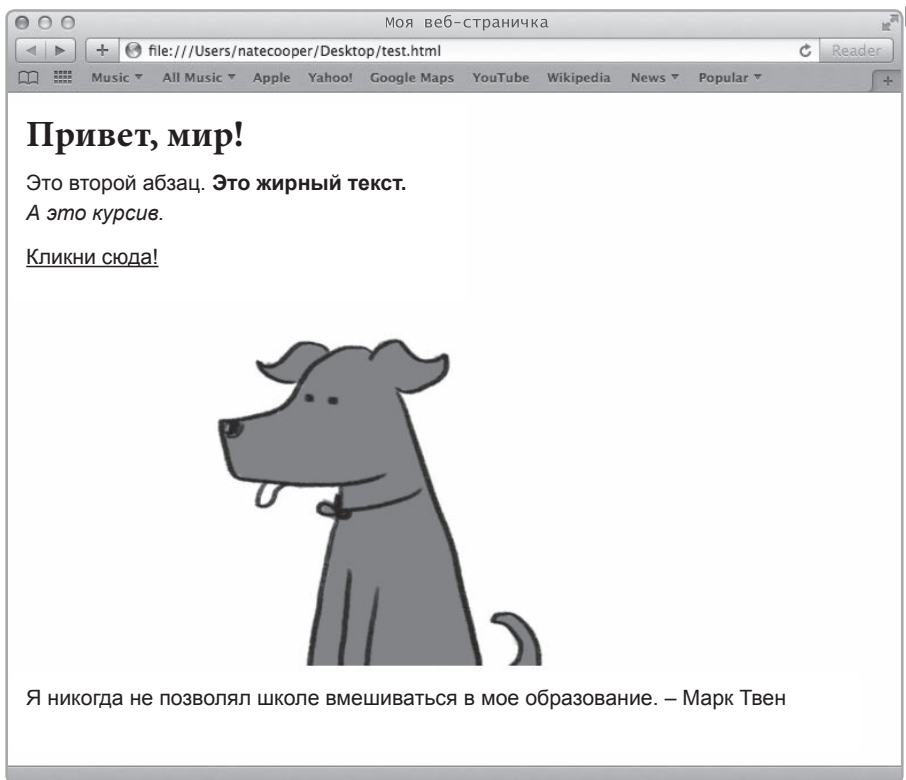


Рис. 3.4. Новый абзац внизу страницы использует свойства, заданные в CSS, ведь это тоже абзац

Атрибут `id` — это уникальный идентификатор, то есть метка, которую можно задавать почти для всех HTML-тегов, чтобы обращаться к ним из CSS. В данном случае мы выбрали в качестве метки слово `quote` («цитата»), но ты можешь называть свои метки как хочешь.

Если сохранить и обновить страничку, ты не заметишь ничего нового. Хотя мы и снабдили абзац уникальной меткой, это все еще абзац, и он наследует свойства, заданные для элементов `<p>`. Но мы хотим изменить его. Как же обратиться к нему из CSS? Чтобы задать свойства элемента по его метке, используй символ `#`. То есть вместо селектора `p` мы будем использовать селектор `#quote`.

Чтобы изменить оформление нашей цитаты, добавим в CSS следующий код:

```
#quote {
    text-align: center;
    color: gray;
}
```

Сохранив и обновив страницу, ты увидишь, что цитата теперь стоит в центре и перекрашена в серый (см. рис. 3.5), но ее шрифт остался таким же, как прежде. В CSS это называется *каскадированием*, или *наследованием*: свойства, не указанные явно, переходят (наследуются) от предыдущих определений. В нашем случае абзац `#quote` сперва наследует все свойства, заданные для абзацев, а затем принимает дополнительные свойства, которые мы указали отдельно для него.

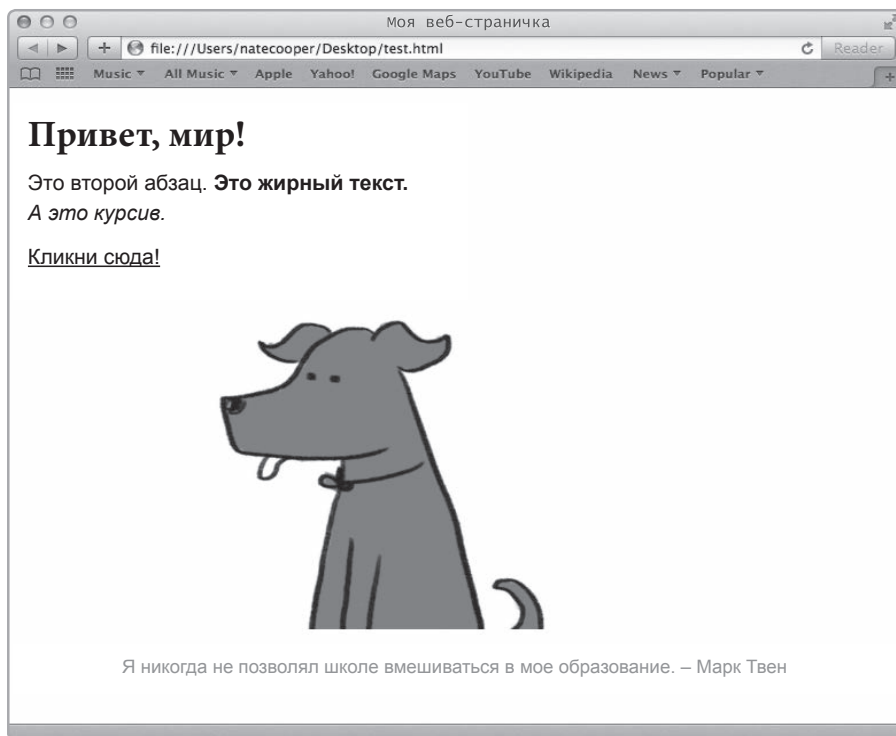


Рис. 3.5. Поскольку у абзаца `#quote` есть уникальная метка, можно задать ему дополнительные свойства, сделав его серым и выровняв текст по центру

Добавь код `font-family: Times, Courier, serif;` следующей строчкой после `color: gray.` CSS-файл должен принять такой вид:

```
p {
    font-family: Helvetica, Arial, sans-serif;
    font-size: .8em;
}
h1 {
    font-family: Helvetica, Arial, sans-serif;
    font-size: 1.3em;
}
#quote {
    text-align: center;
    color: gray;
    font-family: Times, Courier, serif;
}
```

Сохрани и обнови страницу. Теперь ты видишь, что шрифт цитаты отличается от других абзацев, хотя его размер не изменился. Метка отлично подходит, чтобы обратиться к единственному элементу на странице, но как быть, если их несколько? Нам на выручку приходит класс — инструмент, позволяющий одинаково оформить несколько частей страницы. В нашем случае это будет набор цитат. Добавим в *test.html* еще одну цитату и посмотрим, как работают классы.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя веб-страничка</title>
    <meta name="description" content="Тестовая страница, созданная для
изучения HTML">
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>Привет, мир!</h1>
    <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
<em>А это курсив.</em></p>
    <p><a href="http://natecooper.co">Клики сюда!</a></p>
    
    <p class="quote" id="twain">Я никогда не позволял школе вмешиваться
в мое образование. - Марк Твен</p>
    <p class="quote" id="einstein">Образование - это то, что остается,
когда забываешь все, чему тебя учили в школе. - Альберт Эйнштейн</p>
  </body>
</html>
```

Обрати внимание: мы добавили новую цитату, а также поменяли тег предыдущей. В обоих случаях мы указали атрибут `class="quote"`, обозначив, что оба абзаца принадлежат к одной группе. В отличие от меток, одно имя класса можно использовать на странице несколько раз. Для одного элемента можно указать и класс, и метку. При этом один класс можно назначить нескольким элементам, но присваивать разным элементам одну и ту же метку нельзя. Метки нужны, чтобы задавать одиночным элементам страницы уникальный стиль, тогда как классы позволяют выбрать группу элементов и одинаково их оформить. Как несложно догадаться, обращаться к классам и меткам из CSS надо по-разному. Измени свой код следующим образом:

```
p {
  font-family: Helvetica, Arial, sans-serif;
  font-size: .8em;
}
h1 {
  font-family: Helvetica, Arial, sans-serif;
  font-size: 1.3em;
}
.quote {
  text-align: center;
  color: gray;
  font-family: Times, Courier, serif;
}
#twain {
  color: blue;
}
#einstein {
  color: red;
}
}
```

Заметь: для обозначения класса мы использовали в CSS точку (.). На HTML-тег `<p class="quote">` из CSS можно сослаться через селектор `.quote`. Новичкам не всегда понятно, когда следует использовать класс, а когда — метку. Попрактикуйся немного, и все встанет на свои места. Самое важное состоит в том, что для элементов, свойства которых должны отличаться от общих свойств тега, можно задавать как метки, так и классы. При этом один и тот же класс можно назначать как угодно часто, а метка должна быть уникальной для страницы. Класс определяет стили, которые тебе, скорее всего, еще не раз понадобятся. В данном случае нам нужен класс `quote`, ведь цитат на странице несколько. Поскольку у меток более узкое применение, стиль, заданный для метки, может перекрывать стиль класса, как на рис. 3.6. Здесь мы используем класс и метку для тегов `<p>`, но можно делать это и с другими тегами, которые требуется как-то выделить на странице.

Цвета

В предыдущем примере мы задавали цвет абзацев с помощью слов `blue` (синий), `red` (красный) и `gray` (серый). Это лишь один из способов указывать цвет в CSS. Для многих цветов можно использовать английские названия, даже довольно странные, вроде `azure` (лазурный) или `salmon` (лососевый). Но есть другой способ — *шестнадцатеричные значения* цветов. Например, `#FFFFFF` — это белый цвет, а `#000000` — черный. Еще можно задавать цвета по модели RGB, то есть по содержанию в нем красного, зеленого и синего. Тогда черный цвет обозначается как `rgb(0, 0, 0)`, а белый — как `rgb(255, 255, 255)`. Шестнадцатеричные значения, RGB-значения и названия цветов — это просто разные способы выразить одно и то же. При этом (в отличие от `px` and `em`), какой способ ты ни выберешь, результат будет одинаковым.

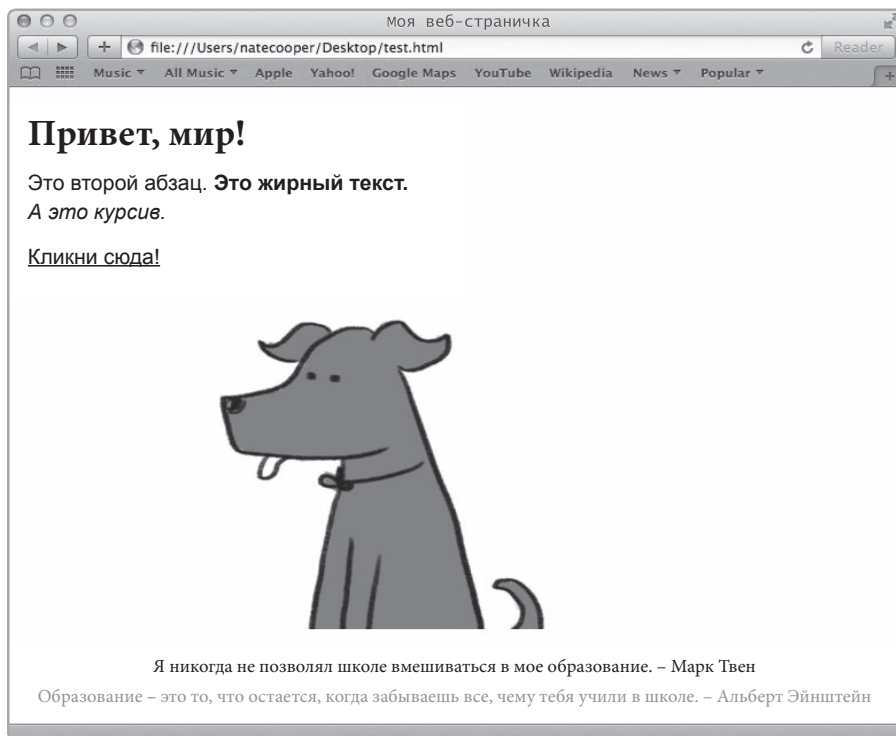


Рис. 3.6. Класс определяет характеристики обеих цитат, а метка задает цвет каждой по отдельности

Например, любая из этих трех записей задаст для #twain один и тот же цвет:

```
#twain {
    color: rgb(0,0,255);
}
#twain {
    color: #0000FF;
}
#twain {
    color: blue;
}
```

В таблице 3.1 перечислено несколько основных цветов, которые могут понадобиться тебе в CSS. Можешь выбрать ту систему записи цвета, которая больше нравится.

Таблица 3.1. Основные цвета

Цвет	Название	Шестнадцатеричный	RGB
черный	black	#000000	rgb(0,0,0)
серый	gray	#C0C0C0	rgb(192,192,192)
белый	white	#FFFFFF	rgb(255,255,255)
желтый	yellow	#FFFF00	rgb(255,255,0)
красный	red	#FF0000	rgb(255,0,0)
зеленый	green	#00FF00	rgb(0,255,0)
синий	blue	#0000FF	rgb(0,0,255)

Тег <div> и выравнивание в CSS

Для каждого из HTML-элементов, о которых мы до сих пор говорили, заданы стили по умолчанию. Например, тег <p> добавляет отступ между абзацами, а тег <h1> делает шрифт крупным и жирным. А теперь рассмотрим тег <div>. У него нет свойств по умолчанию, поэтому он особенно удобен для оригинального CSS-оформления. Открой свой HTML-документ и добавь следующий код:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя веб-страничка</title>
    <meta name="description" content="Тестовая страница, созданная для
изучения HTML">
    <link href="style.css" rel="stylesheet">
  </head>
```

```
<body>
  <h1>Привет, мир!</h1>
  <p>Это второй абзац. <strong>Это жирный текст.</strong><br>
  <em>А это курсив.</em></p>
  <p><a href="http://natecooper.co">Клихни сюда!</a></p>
  
  <p class="quote" id="twain">Я никогда не позволял школе вмешиваться
  в мое образование. - Марк Твен</p>
  <p class="quote" id="einstein">Образование - это то, что остается,
  когда забываешь все, чему тебя учили в школе. - Альберт Эйнштейн</p>
  <div>Это div Гуру</div><div>Это div Глинды</div>
</body>
</html>
```

Сохрани документ и обнови страничку — и обрати внимание на несколько особенностей. Во-первых, в обоих `<div>` не используется шрифт Helvetica, который задан для абзацев. Это логично, ведь мы меняли свойства только у тегов `<p>`. Еще одна любопытная деталь: отступ между элементами `<div>` не такой, как между абзацами. Он скорее напоминает `
`, чем `<p>`. Элемент `<div>` (от английского *division* — «разделитель») не обладает почти никакими свойствами, пока мы не укажем их в CSS. Так и сделаем. Но сперва зададим для каждого из двух `<div>` уникальную метку (`id`), чтобы оформить их по-разному. Измени HTML-код следующим образом:

```
<div id="guru">Это div Гуру</div>
<div id="glinda">Это div Глинды</div>
```

Теперь, когда у каждого `<div>` есть уникальная метка, можно настроить их стили в CSS:

```
#guru {
  height: 100px;
  width: 100px;
  background-color: blue;
  float: left;
}
#glinda {
  height: 100px;
  width: 100px;
  background-color: green;
  float: left;
}
```

Сохрани код и обнови страничку. Если все сделано правильно, результат будет как на рис. 3.7.

Здесь мы задали несколько CSS-свойств. Рассмотрим каждое из них. Во-первых, мы указали для каждого `<div>` ширину (`width`) и высоту (`height`). По умолчанию размеры для `<div>` не определены. Если бы мы не добавили внутрь текст и не указали ширину с высотой, наши `<div>` вообще не отобразились бы на страничке. Задав размеры 100px на 100px, мы получили небольшие квадратики. Кро-

ме того, мы выбрали для них разный цвет фона (свойство `background-color`). Свойство `color`, которым мы пользовались прежде, определяет цвет текста, а `background-color` относится именно к фону *контейнера*. Контейнер — это элемент, который содержит что-то между открывающим и закрывающим тегом. В данном случае контейнер — это `<div>`, но можно настроить фон и для `<p>`, для ``. И наконец, последнее свойство — это `float: left`. Свойство `float` служит для позиционирования: оно помогает выровнять элементы по правой (`right`) или левой (`left`) границе окна. После того как мы добавили `float: left`, элементы `<div>` прижались друг к другу так, будто они притягиваются к левой границе.

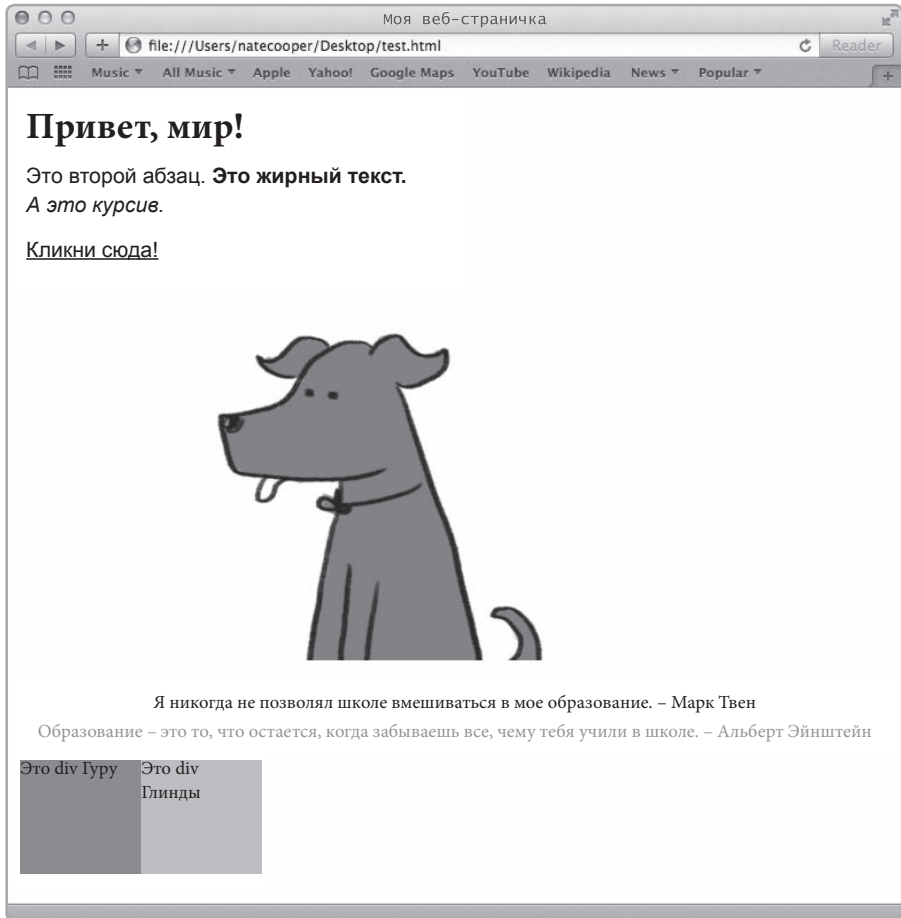


Рис. 3.7. Элементы `<div>` практически невидимы, если не задать им свойства. В данном случае мы изменили их цвет и форму

Давай снова посмотрим на CSS-код:

```
#guru {
    height: 100px;
    width: 100px;
    background-color: blue;
    float: left;
}
#glinda {
    height: 100px;
    width: 100px;
    background-color: green;
    float: left;
}
```

Некоторые свойства наших квадратиков дублируются. Должен быть способ обойтись без этого... Открой HTML-код и добавь к каждому <div> атрибут `class="box"`.

```
<div class="box" id="guru">Это div Гурю</div>
<div class="box" id="glinda">Это div Глинды</div>
```

Теперь можно отредактировать CSS и привязать общие свойства к классу, а уникальные оставить для меток:

```
.box {
    height: 100px;
    width: 100px;
    float: left;
}
#guru {
    background-color: blue;
}
#glinda {
    background-color: green;
}
```

Сейчас мы привязали к классу все, что относится к размерам и выравниванию, а меткам оставили только цвет (уникальные свойства). Теперь задавать общие свойства наших квадратиков будет проще. Давай настроим для `.box` выравнивание текста:

```
.box {
    height: 100px;
    width: 100px;
    float: left;
    text-align: center;
}
```

Сохранив `style.css` и обновив страницу, ты увидишь, что текст внутри квадратиков выровнялся по центру, хотя сами квадраты по-прежнему прижаты к левому краю. Получается, свойство `float` относится к контейнеру, а свойство `text-align` — к тексту внутри него. Ты еще увидишь, что зачастую CSS-свойства по-разному влияют на контейнеры и на их содержимое.

Поля и внутренние отступы

Поля и внутренние отступы — хороший пример того, как различаются свойства, относящиеся к внутренней и внешней областям контейнера. Открой свой CSS-файл и добавь к `.box` следующее:

```
.box {  
  height: 100px;  
  width: 100px;  
  float: left;  
  text-align: center;  
  margin: 5px;  
}
```

После сохранения и обновления страница должна выглядеть как на рис. 3.8.

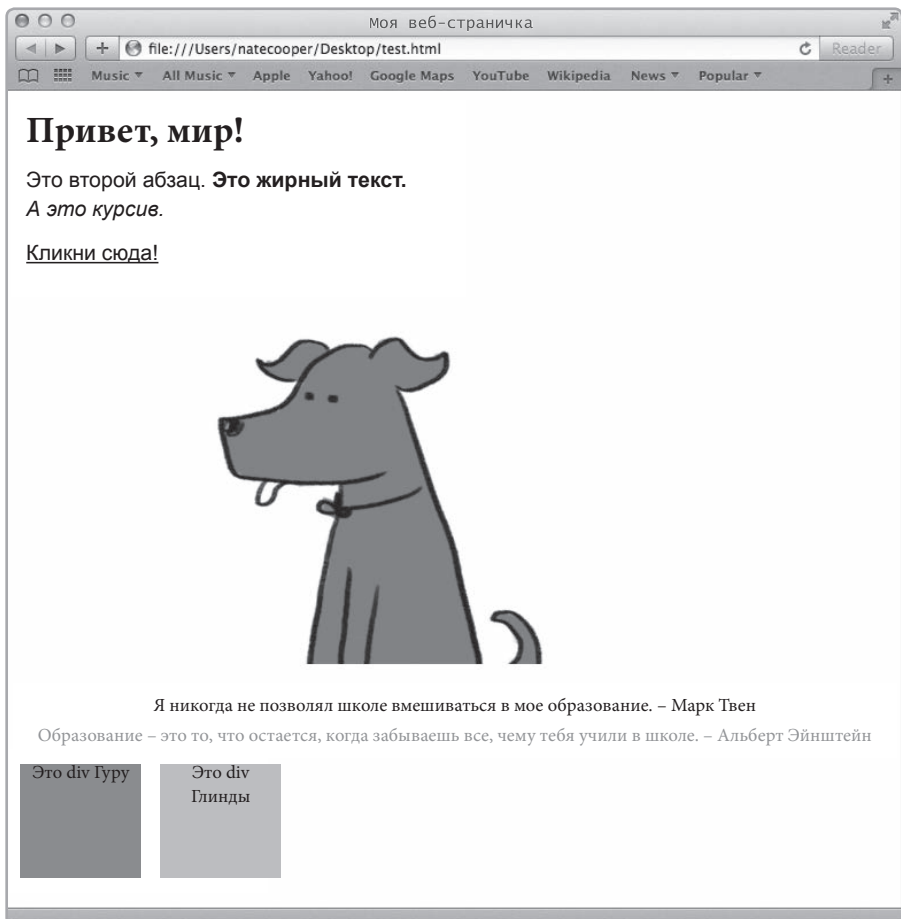


Рис. 3.8. Свойство `margin` добавило отступы вокруг элементов

Вокруг контейнеров появились отступы, отделяющие их друг от друга. А теперь замени в CSS свойство `margin` на `padding`, вот так:

```
.box {  
    height: 100px;  
    width: 100px;  
    float: left;  
    text-align: center;  
    padding: 10px;  
}
```

Результат можно увидеть на рис. 3.9.

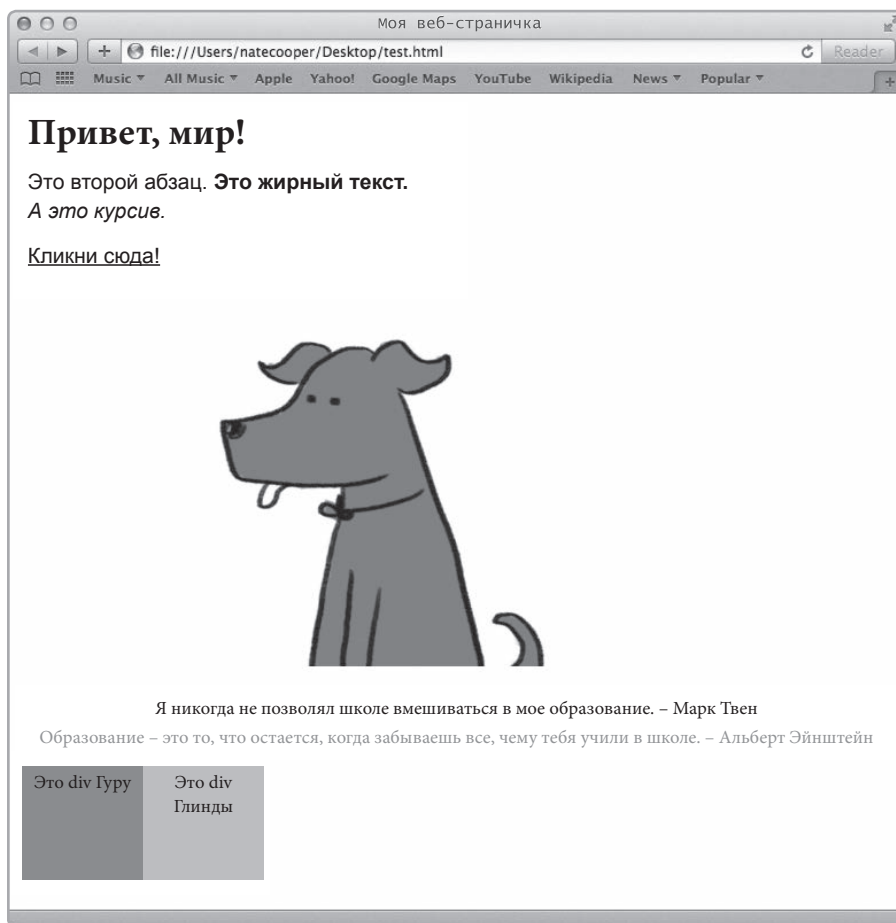


Рис. 3.9. Свойство `padding` добавило отступы внутри элементов `<div>`

Квадратики снова стоят вплотную друг к другу, но между их внутренними границами и текстом появились отступы. Поля (margin) — это пространство вокруг элемента, а внутренние отступы (padding) — пространство внутри элемента, как показано на рис. 3.10.

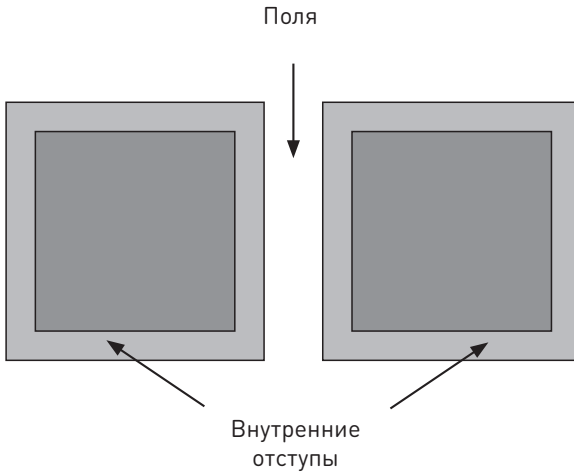


Рис. 3.10. Обрати внимание на разницу между полями, разделяющими два контейнера, и внутренними отступами, которые отделяют содержимое контейнера от его границ. Это тонкое, но важное отличие

Открыв CSS-код, верни свойство `margin`, а `padding` поменяй на `5px`. Должно получиться так:

```
.box {  
    height: 100px;  
    width: 100px;  
    float: left;  
    text-align: center;  
    padding: 5px;  
    margin: 5px;  
}
```

Теперь у нас есть отступы и между квадратиками, и между их границами и текстом. Сначала бывает трудно понять, когда нужно использовать `padding`, а когда `margin`. Но практика расставит все по местам. Не волнуйся, если первые попытки будут выглядеть не очень красиво. Все мы учимся на ошибках.

Разметка структуры и <div>

Если тебе интересно, как при помощи CSS разметить структуру страницы, вот один пример. Создай HTML-документ, сохрани его как *columns.html* и добавь в него такой код:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Пример с колонками</title>
    <link rel="stylesheet" href="columns.css">
  </head>
  <body>
    <div class="main">
      <div class="header">Шапка</div>
      <div class="column" id="leftcolumn">Левая колонка</div>
      <div class="column" id="rightcolumn">Правая колонка</div>
    </div>
  </body>
</html>
```

Теперь создай и сохрани CSS-файл *columns.css*, добавив в него следующий код:

```
.main {
  margin: 0 auto;
  width: 80%;
}
.main:after {
  display: block;
  clear: both;
  content: "";
}
.column {
  width: 50%;
  min-height: 500px;
  margin-bottom: 20px;
  float: left;
  display: block;
}
#leftcolumn {
  background-color: blue;
}
#rightcolumn {
  background-color: red;
}
.header {
  width: 100%;
  height: 100px;
  background-color: yellow;
  clear: both;
}
```

Результат можно увидеть на рис. 3.11.

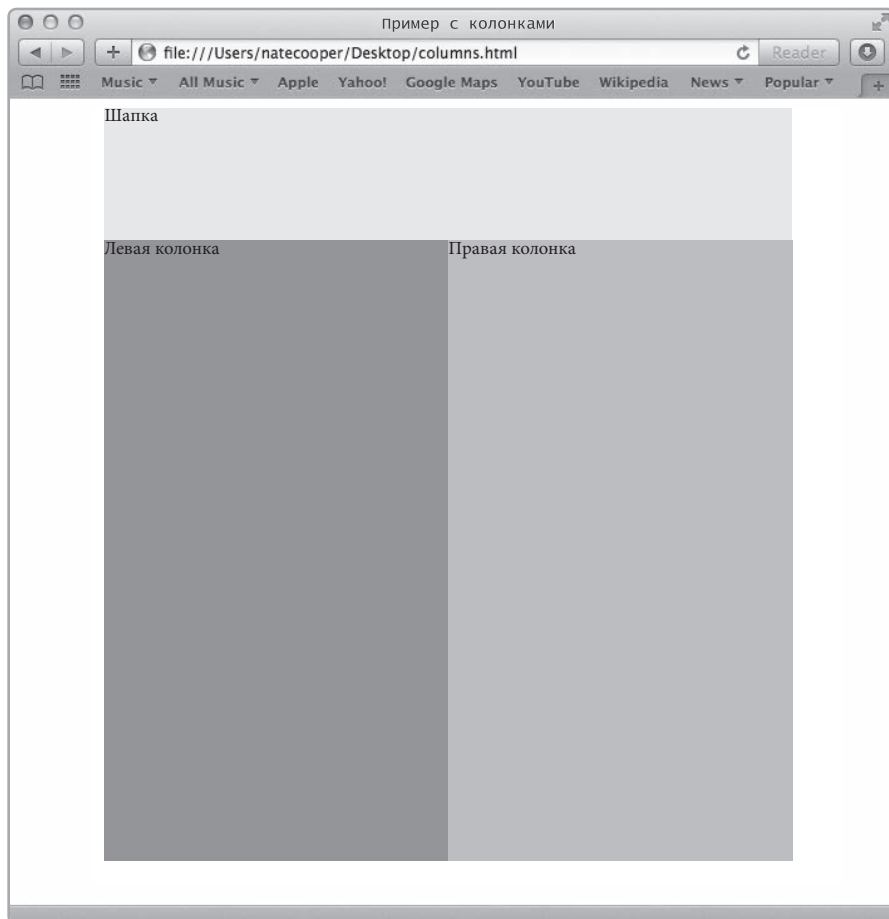


Рис. 3.11. В этом примере мы создали с помощью элементов `<div>` две колонки и поместили третий `<div>` сверху, в качестве газетной «шапки»

Как видишь, CSS хорош не только для шрифтов. С его помощью можно создавать сложную структуру, которая масштабируется и подстраивается под браузер. Здесь мы использовали для этого элементы `<div>`.

Прежде всего у нас есть основной `<div>` с классом `main`, в который заключено остальное содержимое. Для него заданы свойства `margin: 0 auto; width: 80%;` — они центрируют элемент и определяют его ширину в процентах от ширины страницы.

Внутри главного элемента `<div>` находится `<div>`-шапка с классом `header` и еще два элемента `<div>` с классом `column`. Чтобы шапка занимала всю ширину основного `<div>`, мы указали для нее `width: 100%`. Она будет отображаться сверху как узкая полоса, поэтому мы задали высоту в пикселах: `height: 100px`. Под шапкой должны быть две колонки одинаковой ширины и высоты. Мы создали для них класс, назначив ему общие для обеих колонок свойства: `width: 50%; min-height: 500px`. Ширина равна 50%, поскольку колонка занимает половину основного элемента `<div>`. Также мы указали свойство `margin-bottom: 20px`, чтобы добавить отступ от нижнего края страницы.

ПРИМЕЧАНИЕ

В CSS вкладывание прямоугольных HTML-элементов один в другой (как в нашем примере шапка и колонки вложены внутрь основного `<div>`) называется блочной моделью.

Теперь для шапки и обеих колонок заданы подходящие размеры, но без `float` эти элементы стояли бы вертикально, один над другим. Свойство `float` часто используют, чтобы выстроить элементы в ряд. Например, если указать для картинке `float: left`, она «приклеится» к левой границе контейнера, а текст будет выведен справа. В нашем примере благодаря свойству `float: left` колонки прижимаются к левой стороне страницы.

Осторожнее с `float`: у него есть свои хитрости. Если назначить элементу это свойство, браузер будет считать, что его нужно вывести в один ряд с другими элементами. В нашем случае это помешало бы правильно отобразить шапку, поэтому мы задали ей свойство `clear: both` — оно позволяет игнорировать элементы с `float`, расположенные справа или слева. Кроме того, мы добавили `clear` для основного `<div>` с помощью так называемого псевдоэлемента `main:after`. Но это работает, только если задать свойства `display` и `content`, как в нашем примере. Псевдоэлементы позволяют автоматически изменять HTML, чаще всего — добавлять что-то перед элементом или после него, чтобы получился аккуратный, структурированный документ.

Осваивая более продвинутые свойства CSS, ты научишься размечать колонки, создавать боковые панели, скрывать элементы и делать их снова видимыми.

Хочешь еще глубже изучить CSS? Поэкспериментируй со свойствами из таблицы 3.2.

Таблица 3.2. Базовые CSS-свойства

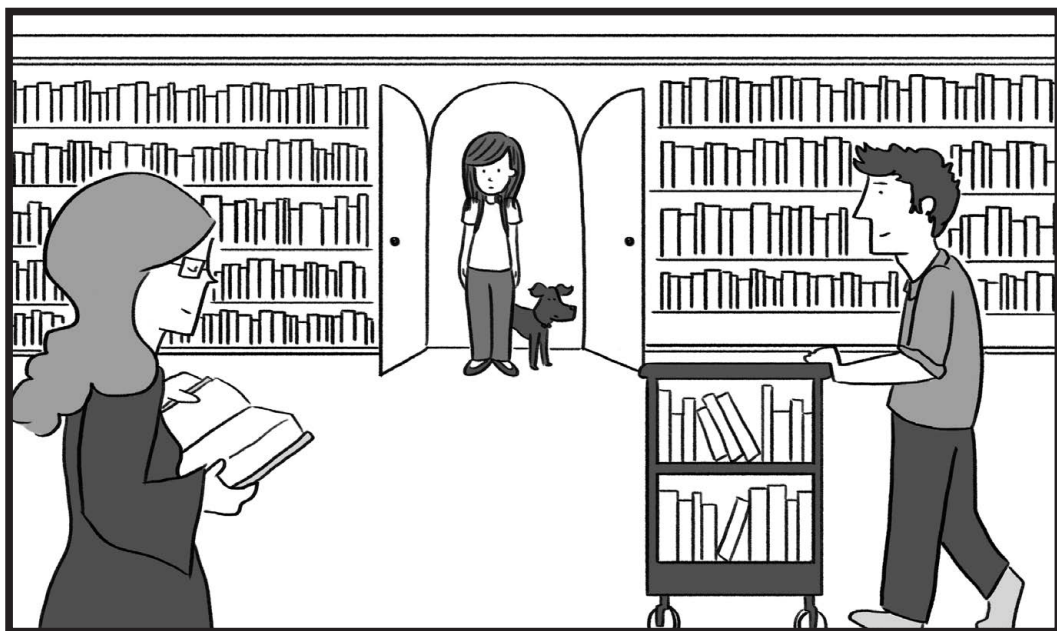
Свойство	Значение	Описание
<code>background-color</code>	<i>#-значение, rgb, название цвета</i>	Задает фоновый цвет контейнера (например, <code><p></code> или <code><div></code>)
<code>background-image</code>	<i>url ("http://адрес-картинки")</i>	Использует картинку с указанным адресом как фоновое изображение контейнера
<code>border-color</code>	<i>#-значение, rgb, название цвета</i>	Задает цвет рамки вокруг элемента
<code>border-style</code>	<i>dotted, solid, double, dashed</i>	Задает стиль рамки
<code>border-width</code>	<i>thin, thick, medium</i>	Включает рамку вокруг элемента и задает ее толщину
<code>border</code>	<i>(общее)</i>	Чтобы не настраивать разные свойства рамки по отдельности, можно воспользоваться общим свойством <code>border</code> (например, <code>border: thin solid black</code>)

Свойство	Значение	Описание
clear	left, right, none, both	Сообщает, должен ли элемент игнорировать float
color	<i>#-значение, rgb, название цвета</i>	Задаёт цвет текста внутри контейнера
float	left, right, none	«Приклеивает» элемент к правой или левой границе контейнера
font-family	<i>имена шрифтов через запятую</i>	Задаёт шрифты для элемента-контейнера
font-size	px, pt, in, cm, em	Задаёт размер шрифта для контейнера (в пикселах, пунктах, дюймах, сантиметрах или em)
font-weight	light, lighter, normal, bold, bolder	Меняет толщину шрифта
height	px, in, cm, %	Задаёт высоту элемента
margin	px, in, cm, %	Задаёт поля вокруг элемента
margin-bottom	px, in, cm, %	Задаёт поле снизу
margin-left	px, in, cm, %	Задаёт поле слева
margin-right	px, in, cm, %	Задаёт поле справа
margin-top	px, in, cm, %	Задаёт поле сверху
overflow	hidden, scroll, visible	Настраивает поведение контейнера, если его размер меньше, чем размер содержимого
padding	px, in, cm, %	Задаёт отступы между границами контейнера и его содержимым
padding-bottom	px, in, cm, %	Задаёт отступ между нижней границей контейнера и его содержимым
padding-left	px, in, cm, %	Задаёт отступ между левой границей контейнера и его содержимым

Свойство	Значение	Описание
padding-right	px, in, cm, %	Задаёт отступ между правой границей контейнера и его содержимым
padding-top	px, in, cm, %	Задаёт отступ между верхней границей контейнера и его содержимым
text-align	left, right, center, justify	Задаёт режим выравнивания текста в контейнере
text-decoration	underline, line-through, none	Задаёт стиль текста
width	px, in, cm, %	Задаёт ширину элемента

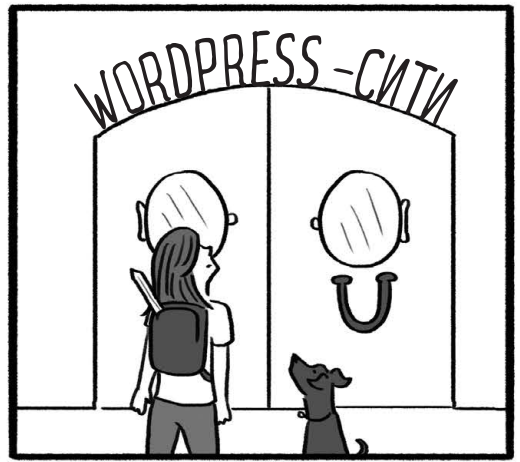
4

Ким иследует WordPress-сими

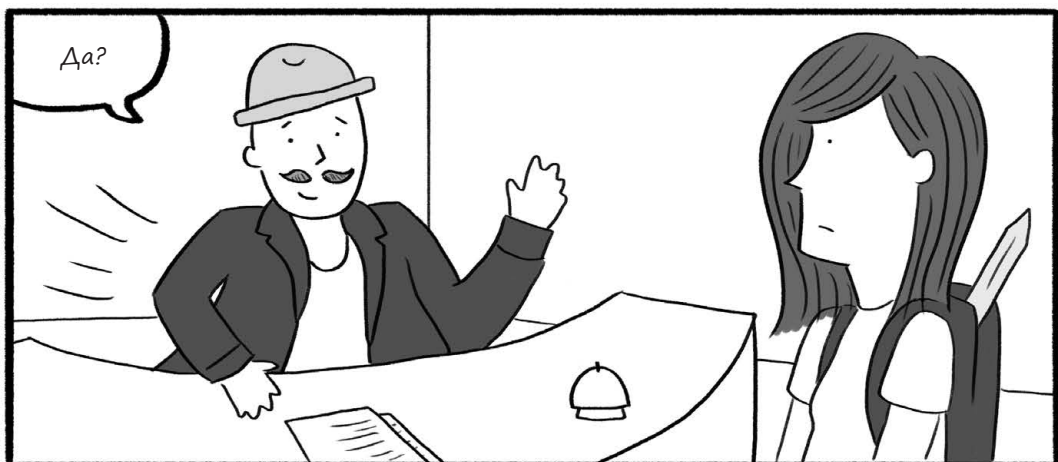
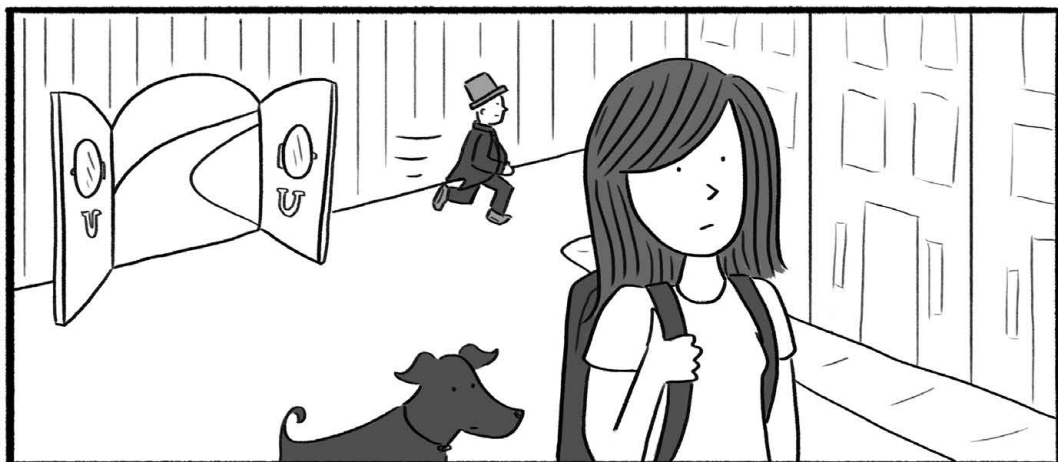


Добро пожаловать в WordPress-сити



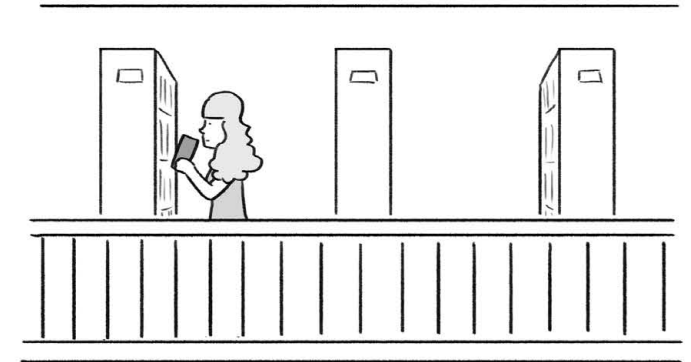






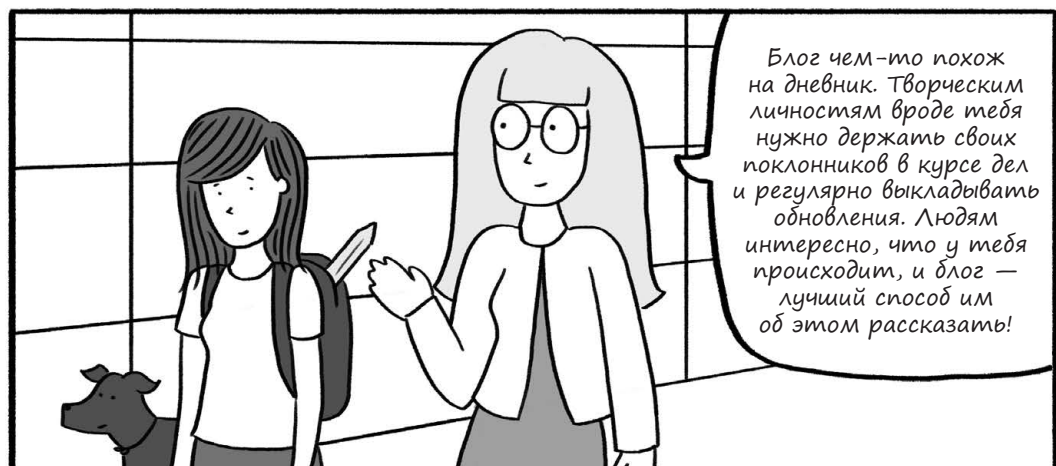


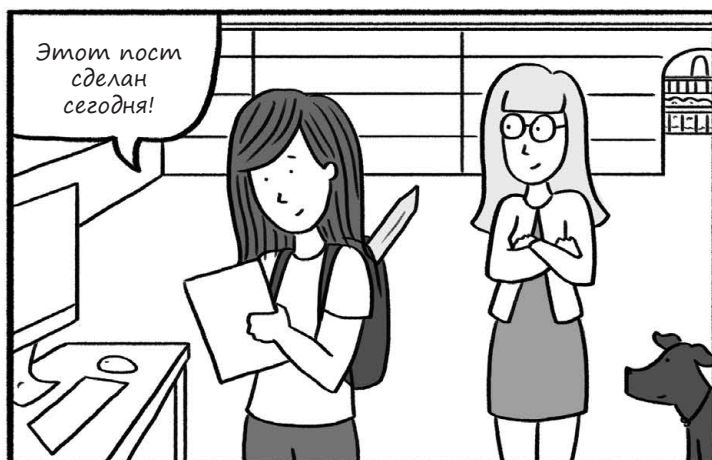
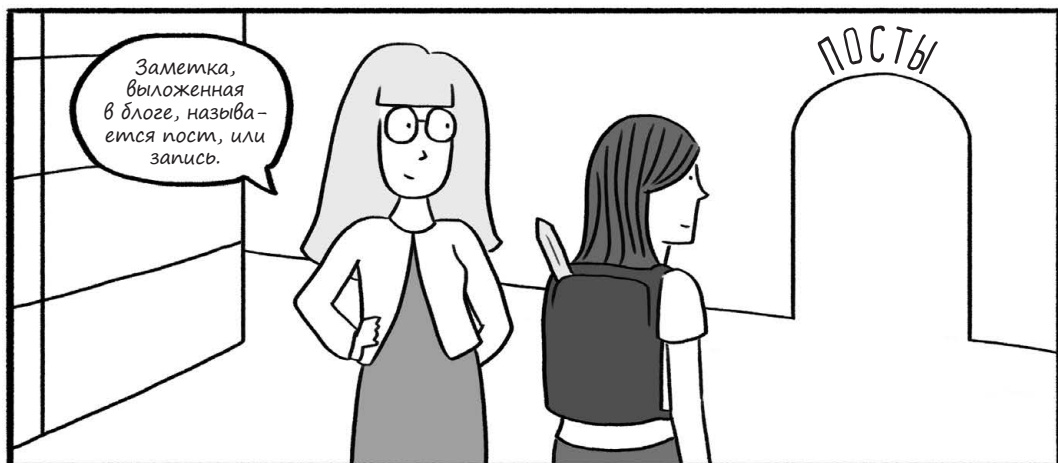
Ким осваивается в WordPress-сими

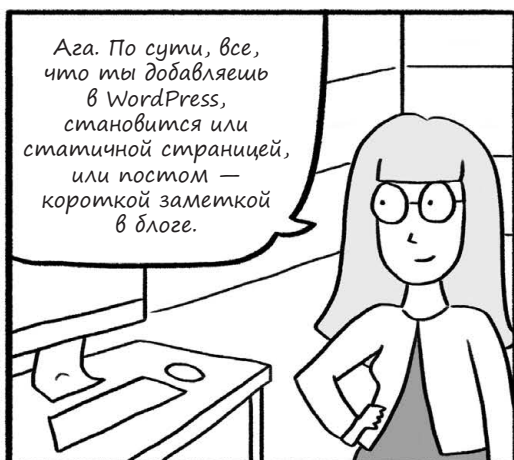




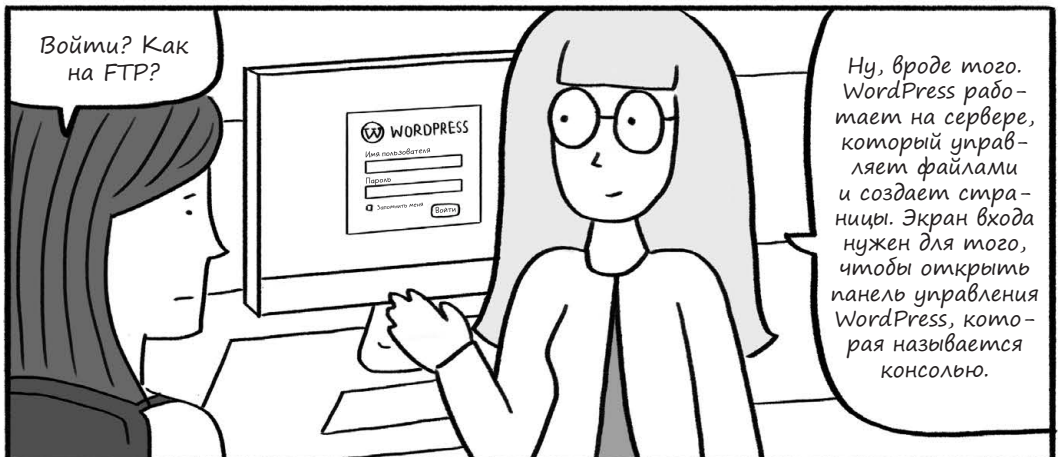
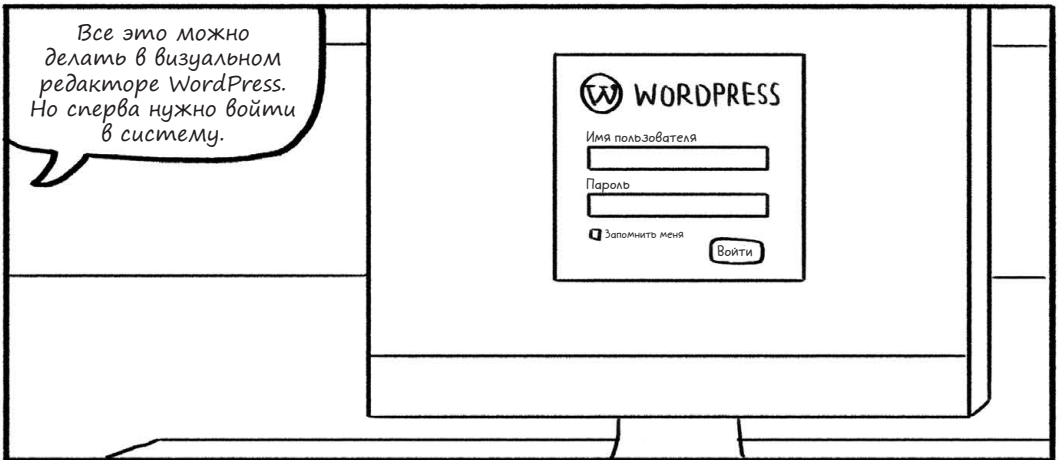




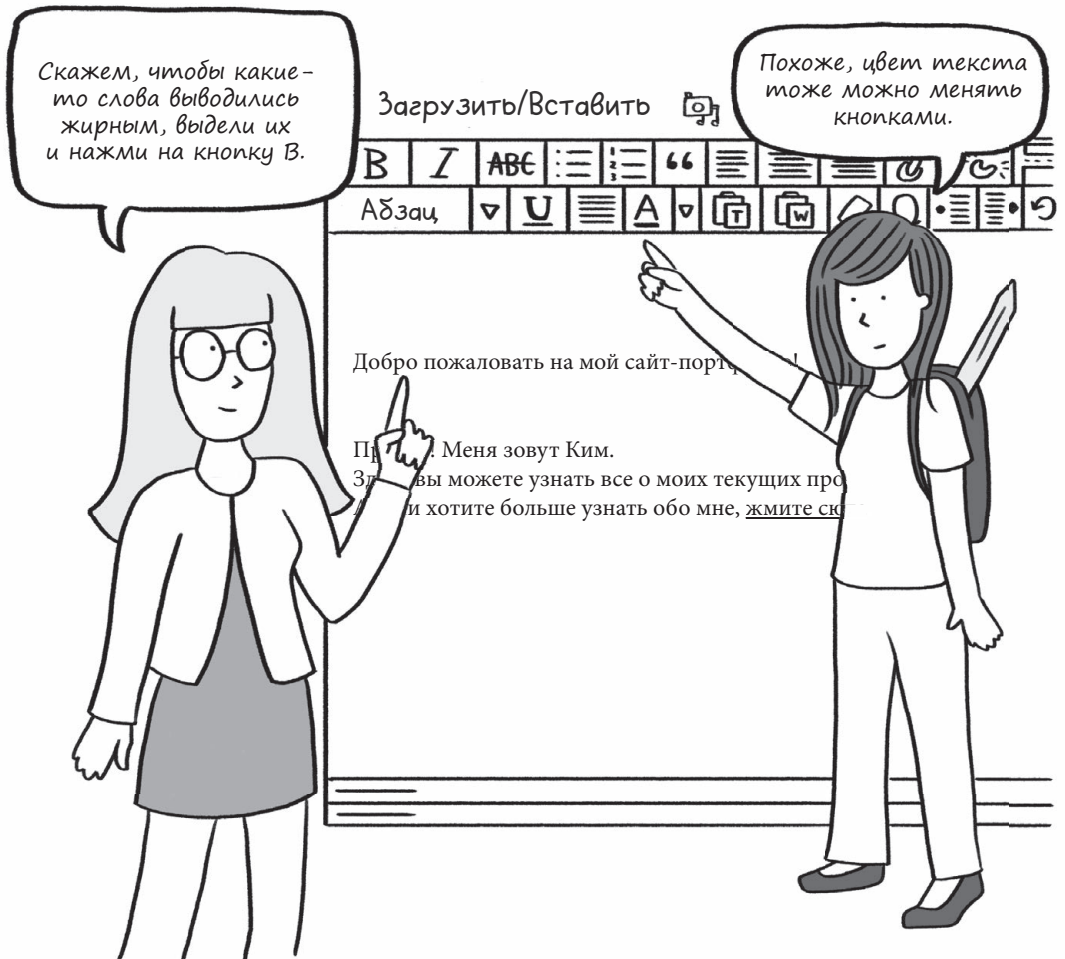


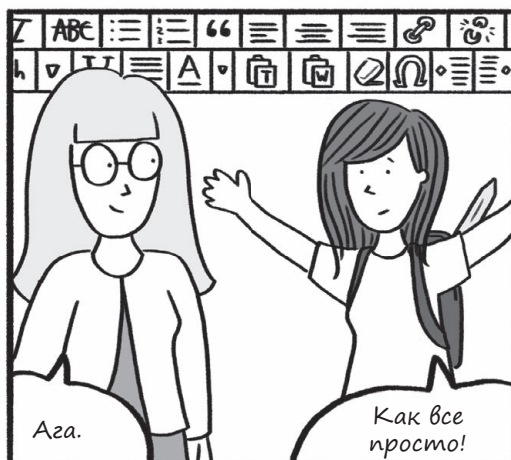
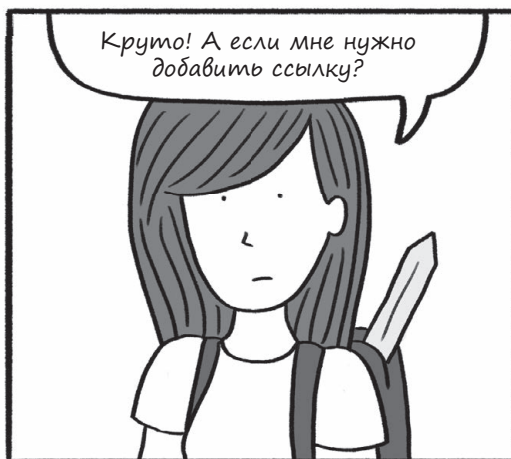


Ким создает первую страницу в WordPress



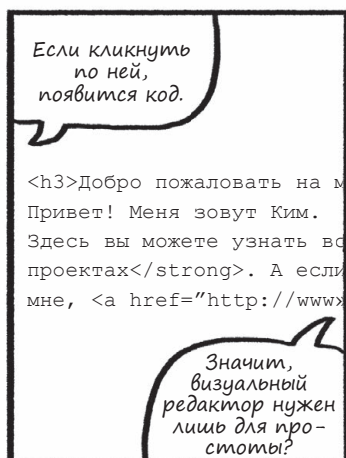
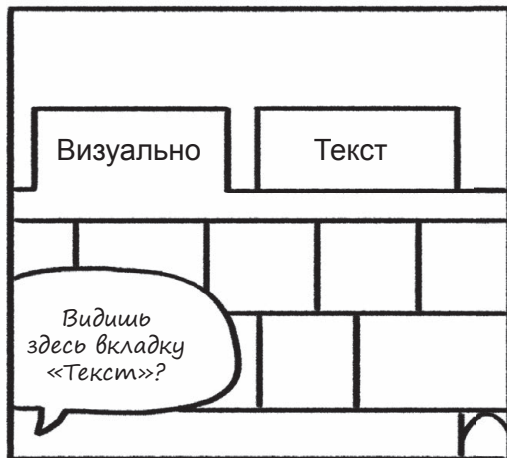




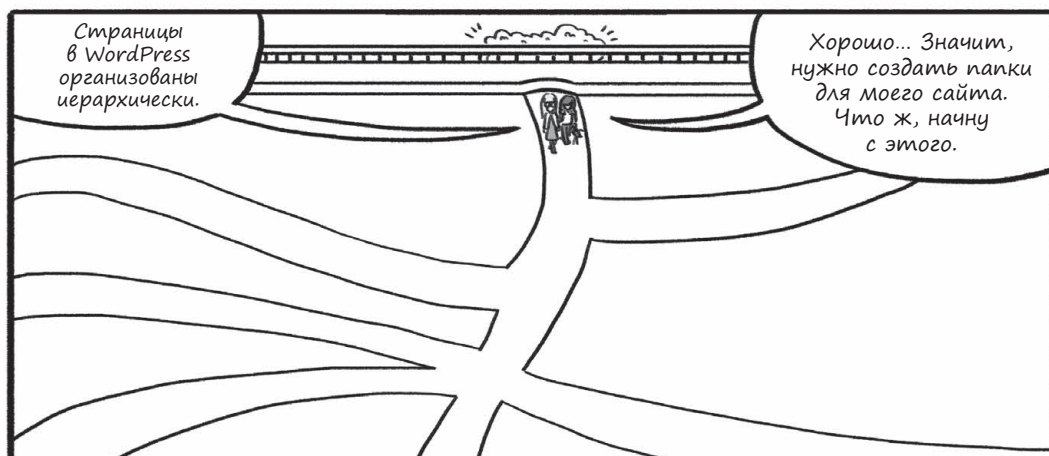
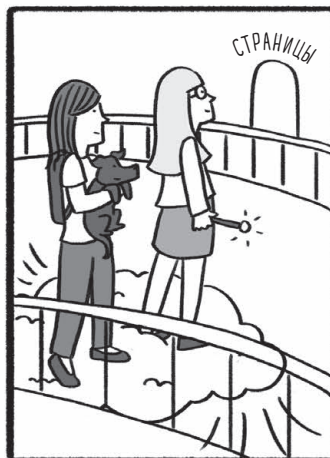
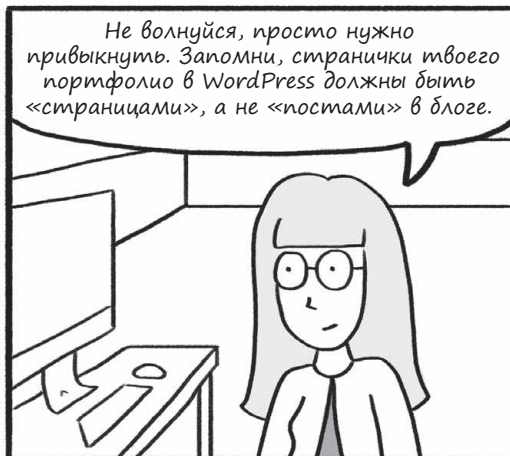
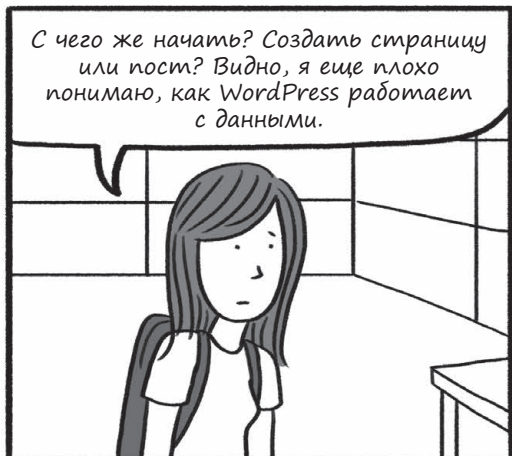


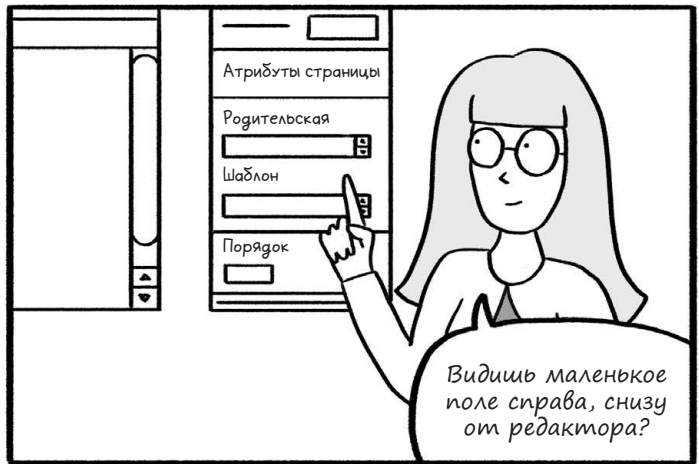
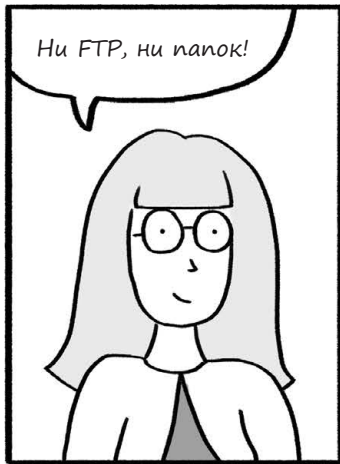
Добро пожаловать на мой сайт-портфолио!

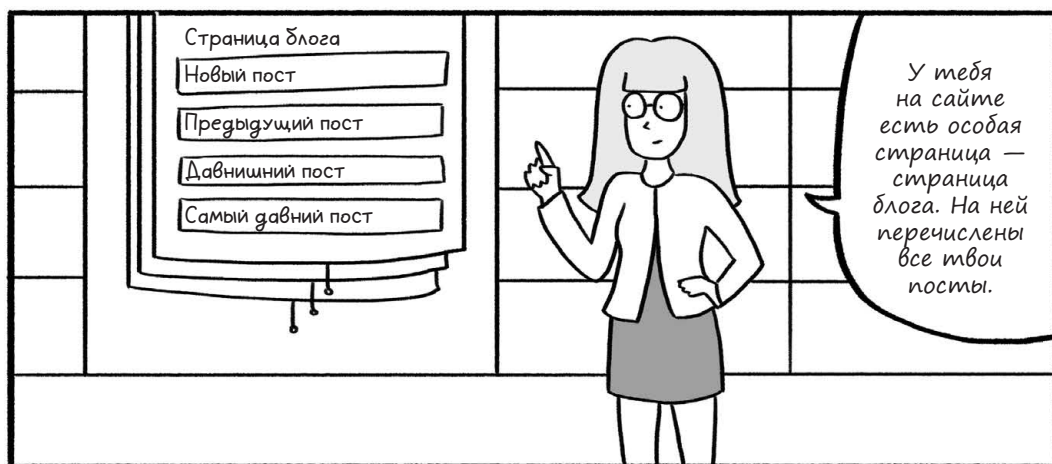
Привет! Меня зовут Ким.
Здесь вы можете узнать все о моих текущих проектах. А если хотите больше узнать обо мне, жмите сюда.

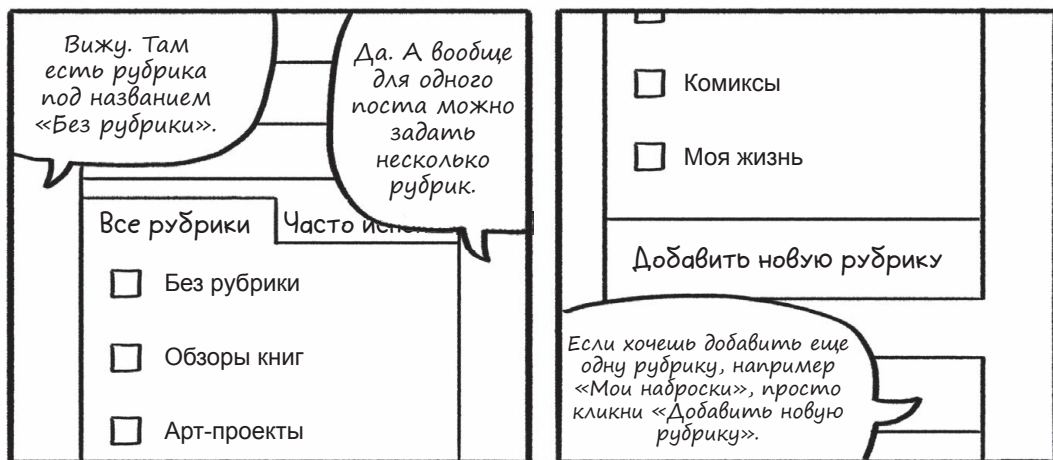


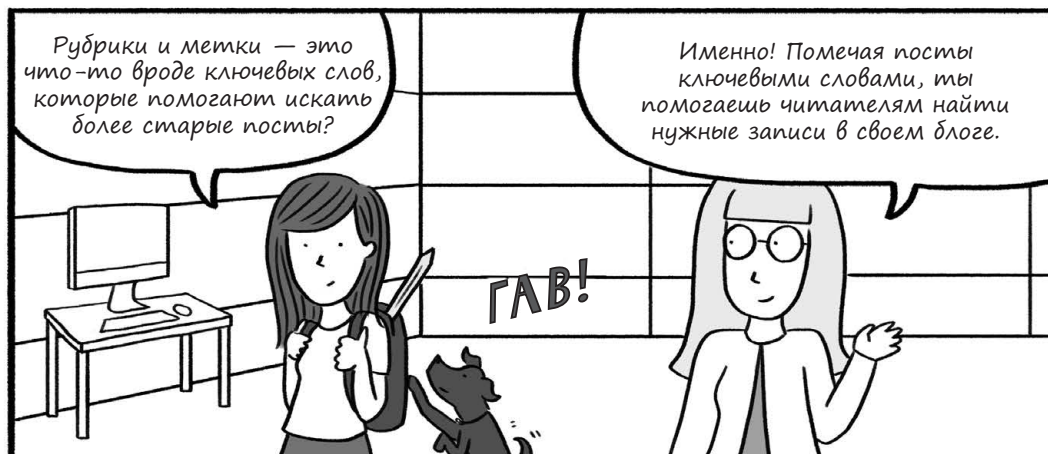
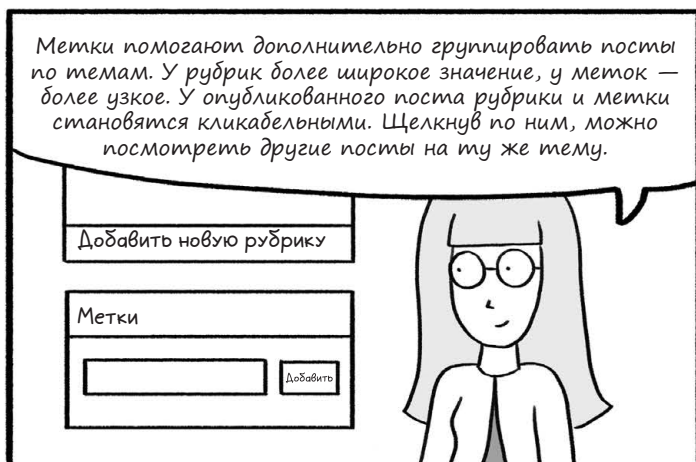
Ким обустроивает свой сайт



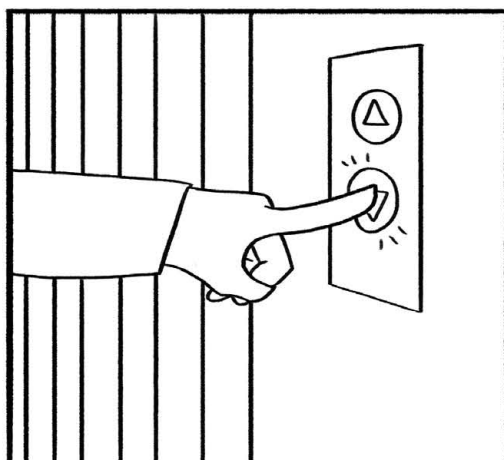


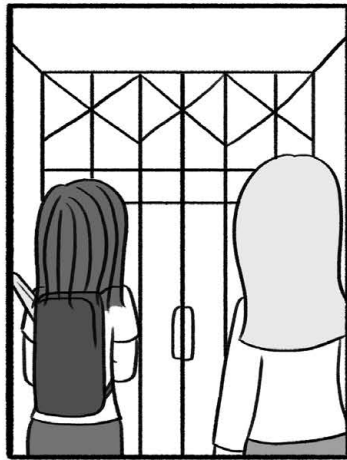






Ким добавляет фотографии на сайт





Ты можешь давать им названия с пробелами и длинные описания. Текст для атрибутов alt тоже можно вводить прямо здесь, а не в коде. Все это делается через библиотеку файлов WordPress.

Ого. А как насчет создания папок и управления фотографиями?

Библиотека файлов — это централизованное хранилище картинок, отсортированных по описанию. Раскладывать их по папкам не нужно.

Кликнув на раздел «Медиафайлы» и выбрав вкладку «Библиотека», ты увидишь их полный список.

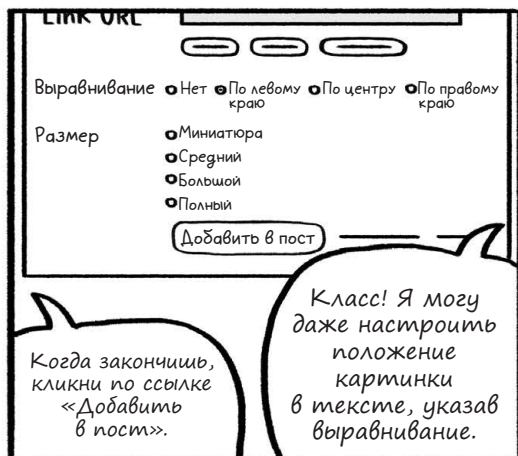
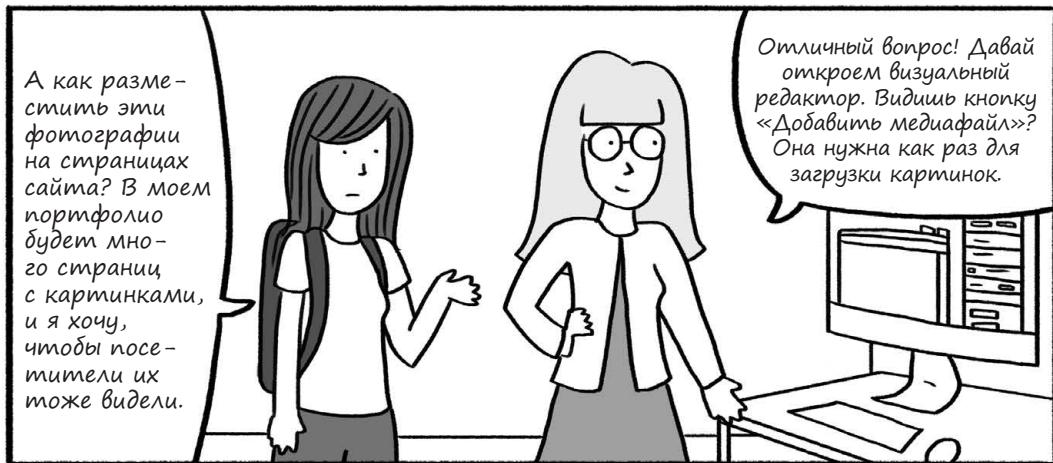
- Записи в блоге
- Медиафайлы**
- Библиотека
- Добавить новый
- Ссылки
- Страницы

Можно сортировать их по дате или искать по имени или описанию. Тебе не нужно думать о том, где и как хранится каждое фото, — об этом позаботится WordPress. Каждый раз, создавая новый пост или страницу, ты можешь искать нужные фотографии в этом закрытом от посторонних хранилище.

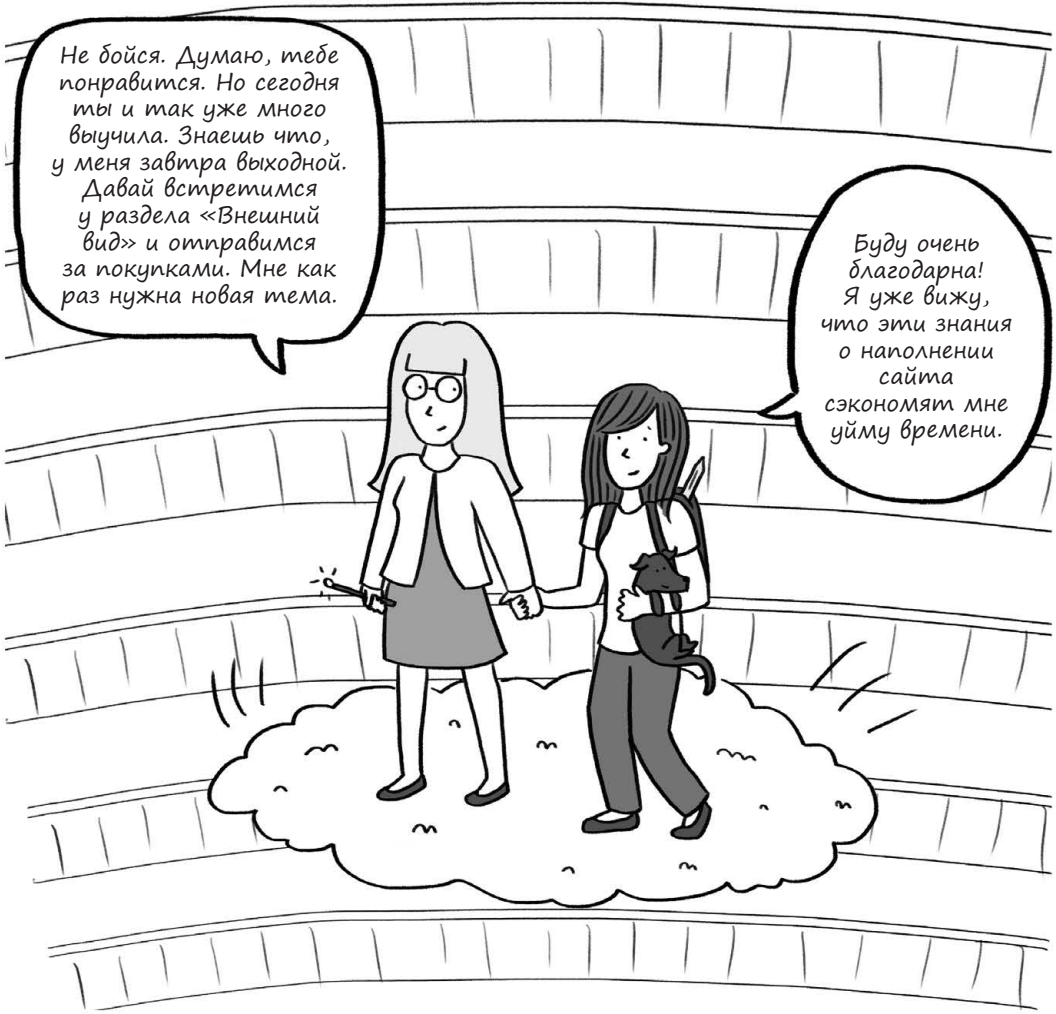
<input type="checkbox"/>	Файл	Автор	Загружен для	Дата
<input type="checkbox"/>		—	—	—
<input type="checkbox"/>		—	—	—

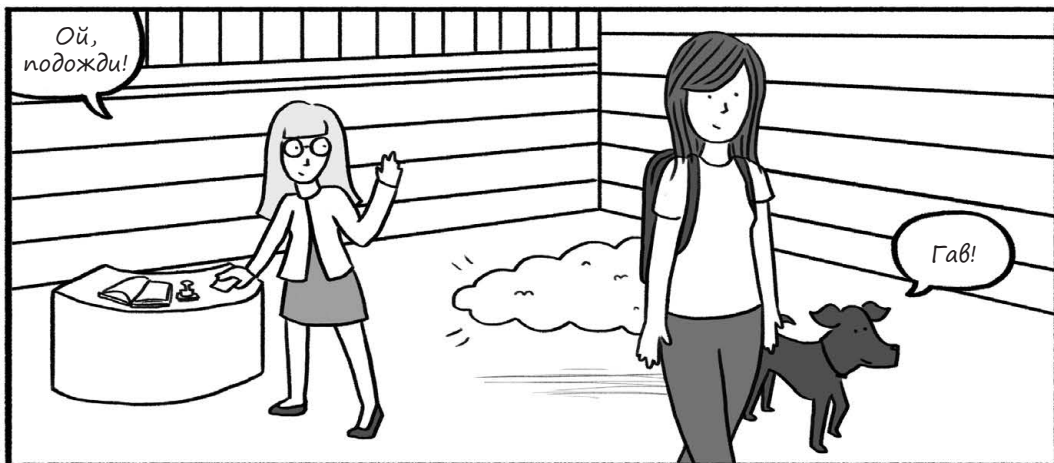
Значит, это моя личная библиотека файлов?

Да.









Начало работы с WordPress

WordPress — это мощный инструмент для ведения блогов, а также система управления контентом (*content management system*, или *CMS*). Такие системы позволяют автоматизировать многие задачи, связанные с созданием сайтов, например загрузку картинок и управление материалами. Кроме того, с WordPress тебе не придется вручную писать код для каждой страницы или поста в блоге!

В отличие от глав 2 и 3, где можно было проверить работу HTML и CSS у себя на компьютере, для того чтобы приступить к созданию сайта на WordPress, тебе понадобится хостинг (подробнее об этом в главе 5). Либо ты можешь зарегистрироваться на WordPress.com и бесплатно создать сайт там. Разумеется, у этой медали есть и обратная сторона: возможности бесплатных аккаунтов на WordPress.com ограничены, к тому же у тебя не будет адреса с доменом второго уровня (например *<твое название>.ru*). Придется выбрать адрес вида *<твое название>.wordpress.com*. За дополнительную плату возможности аккаунта можно расширить — например, подключить собственное доменное имя.

В этой книге мы будем ориентироваться на версию WordPress, которую необходимо устанавливать на хостинг (иногда ее называют WordPress.org). Это откроет для тебя дополнительные возможности вроде установки любого количества тем и плагинов. Чтобы установить WordPress на собственном хостинге, открой стр. 251 с советами по выбору хостинга для WordPress и следуй инструкциям. Не волнуйся, мыждемся твоего возвращения.

WordPress обладает неоспоримым достоинством: наполнение страниц в нем отделено как от структуры, так и от оформления сайта. Вспомни, как мы создавали HTML-странички: все их наполнение (текст), равно как и разметка (всевозможные теги), находились в одном документе. Чтобы изменить текст, нам приходилось открывать документ с кучей тегов. В WordPress редактировать текст с картинками гораздо проще, ведь в его редакторе показано только наполнение, без разметки. Если же тебе нужно поменять вид сайта, к твоим услугам отдельный инструмент — *темы*. Они выполняют ту же задачу, что и CSS в обычном HTML-документе.

Все эти удобства позволяют тебе сконцентрироваться на самом сложном этапе создания крутого сайта — написании интересных материалов — без лишних хлопот с разметкой. Стоит ли удивляться, что WordPress так популярен?

Итак, приступим.

Вход в WordPress и выход из него

Когда ты установишь WordPress (или зарегистрируешь аккаунт на WordPress.com), тебе потребуется создать учетную запись администратора, выбрав логин и пароль. Это ключи от всех дверей: администратор может редактировать любые материалы, создавать страницы и даже удалить весь сайт. У каждого сайта на WordPress должен быть администратор — один или несколько.

Теперь войди с этим логином и паролем в консоль WordPress (см. рис. 4.1), которая находится по адресу *http://<адрес-твоего-wordpress-сайта>/wp-admin/*. Добавь этот адрес в закладки браузера: ты будешь постоянно к нему обращаться.

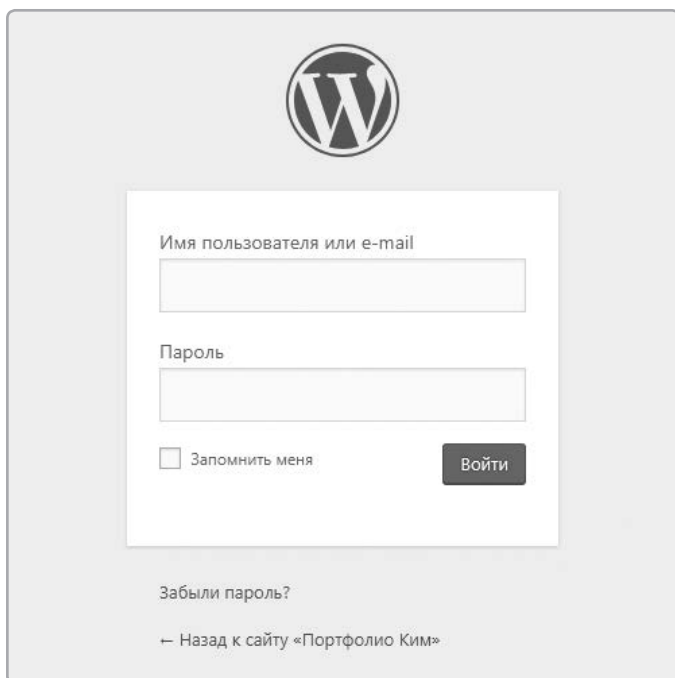


Рис. 4.1. Экран входа

Консоль WordPress выглядит приблизительно как на рис. 4.2. Она является бэкэндом твоего нового сайта. Это центр управления, откуда ты будешь добавлять контент или вносить правки. Попасть туда сможешь только ты (и те, для кого ты создашь учетную запись).

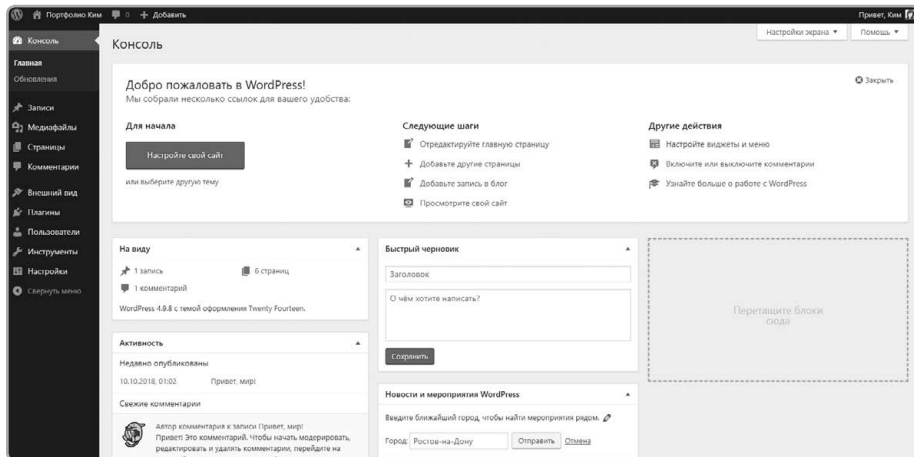


Рис. 4.2. Консоль WordPress. Используйте вкладки слева для управления разными частями сайта

Остальные посетители твоего сайта увидят только *фронтенд*: посты в блоге и прочее содержание. Поначалу он будет выглядеть примерно как на рис. 4.3.

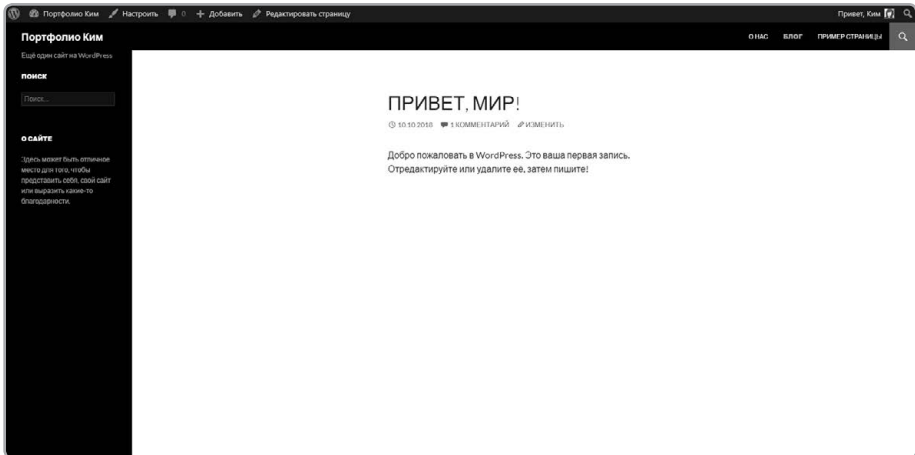


Рис. 4.3. Фронтенд твоего сайта. Именно его видят посетители. Изменения, которые ты внесешь через консоль (бэкенд WordPress), отразятся на конечном облике сайта, то есть здесь. В зависимости от того, какая тема активна, твой блог может выглядеть по-разному. Здесь используется тема *Twenty Fourteen*

ПРИМЕЧАНИЕ

Нужно помнить: как только ты настроишь сайт на WordPress, он станет доступен любому, у кого есть его адрес. Конечно, ты можешь создавать черновые посты, видимые только тебе. Но некоторые из действий, описанных в этой главе, связаны с публикацией данных в интернете. Впрочем, на этом этапе не стоит слишком беспокоиться о защите информации. Маловероятно, что кто-то найдет твой сайт, не зная его адреса. Кроме того, все тестовые странички потом можно будет удалить.

Забыл, как войти на сайт?

Если ты забыл URL страницы, с которой ты заходишь в панель управления сайтом, добавь к его адресу `/wp-admin`. Например, если сайт на WordPress располагается по адресу `http://<твой-сайт>/`, для входа в консоль укажи адрес `http://<твой-сайт>/wp-admin`.

Проверка результатов

Наверное, тебе хотелось бы видеть, как меняется вид сайта для посетителей по мере того, как ты добавляешь содержимое через бэкэнд. Войдя на сайт как администратор, ты увидишь сверху страницы панель администратора. Ее также называют верхней панелью (см. рис. 4.4).



Рис. 4.4. Панель администратора

Находясь в бэкэнде, кликни по названию сайта в верхней панели, чтобы перейти во фронтенд. В нашем случае название — это «Портфолио Ким», значит, нажимаем на него. А чтобы, находясь во фронтенде, перейти обратно в бэкэнд для добавления контента, открой выпадающее меню и выбери «Консоль» (см. рис. 4.5).

В правой части панели администратора есть меню, позволяющее выйти из панели управления (рис. 4.6). Если ты используешь чужой компьютер, после завершения работы всегда выходи из своей учетной записи! Ты же не хочешь, чтобы посторонние управляли твоим сайтом?

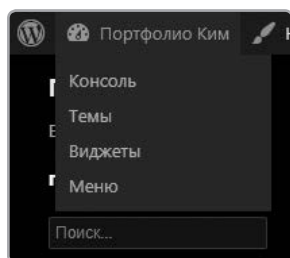


Рис. 4.5. Используй выпадающее меню слева для переключения между фронтендом и бэкэндом (консолью)

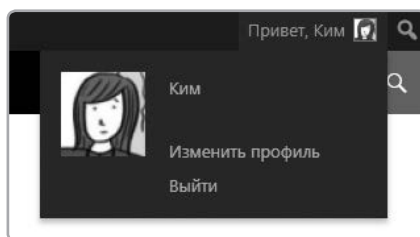


Рис. 4.6. Выпадающее меню справа позволит тебе отредактировать свой профиль или разлогиниться

Хватит прыгать от фронтенда к бэкэнду, пора уже что-нибудь сделать! Для начала включи тему *Twenty Fourteen*, чтобы твой WordPress выглядел так же, как на страницах этой книги. Открой консоль, в панели слева кликни на вкладку **«Внешний вид»** и затем **«Темы»**. Должен открыться экран выбора темы, который мы рассмотрим подробнее в разделе «Меняем оформление: главное о темах» на стр. 205. Пока просто проверь, активна ли тема *Twenty Fourteen*. Если нет, найди ее в списке и кликни «Активировать».

Контент: посты и страницы

В WordPress есть два основных типа контента: *посты* в блоге и *страницы*. И то и другое может содержать текст, ссылки, картинки, видео и так далее. Поскольку WordPress изначально был разработан для ведения блогов, зайдя на сайт, первым делом ты увидишь список постов, отсортированных по дате, подобно записям в дневнике.

С другой стороны, если ты добавишь новую *страницу*, например «Обо мне» или «Контакты», она будет показана отдельно, всегда оставаясь на своем месте. Нет смысла публиковать такую информацию в посте, который быстро потеряется под грудой новых записей.

На самом деле страницы важнее постов еще вот по какой причине: когда ты создаешь страницу, ссылка на нее обычно попадает в *меню навигации*. В теме *Twenty Fourteen* это горизонтальная панель со ссылками наверху каждой страницы сайта (см. рис. 4.7).

Посты в блоге — это просто лента коротких записей на странице блога (см. рис. 4.8).

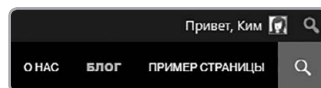


Рис. 4.7. Новые страницы отображаются в меню навигации, которое позволяет посетителям переключаться между разделами сайта. Потом можно будет удалить оттуда ненужные ссылки

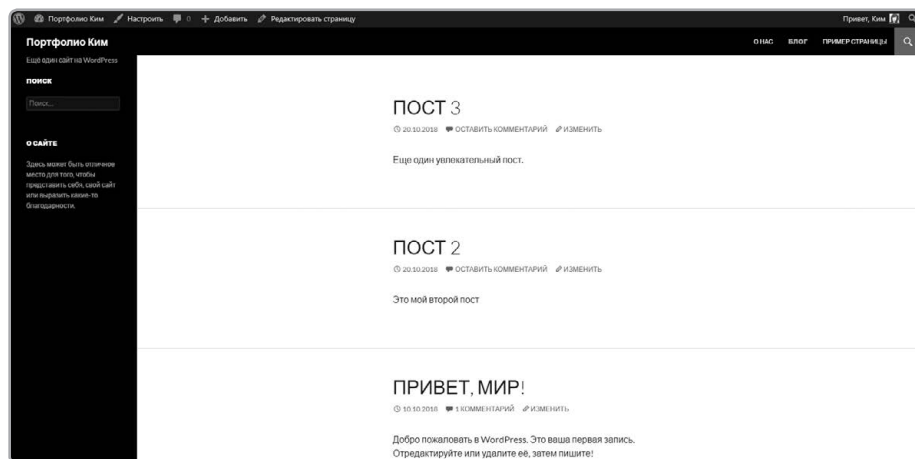


Рис. 4.8. В отличие от страниц, посты в блоге обычно не видны в меню — они перечислены в хронологическом порядке на странице блога, которая по умолчанию является главной страницей сайта

Разница между постами и страницами — самое простое и самое важное, что нужно понимать, работая над сайтом в WordPress. Помни: страницы нужны для отдельных блоков информации, а посты — для записей в блоге.

Структура твоего сайта

Пока ты осваиваешься и экспериментируешь с WordPress, планировать особо нечего. Но когда ты приступишь к наполнению сайта, желательно составить его план или карту. Сперва подумай о первом впечатлении — какую страницу ты хочешь показать новым посетителям? В WordPress это называется *главной страницей* сайта. Она может быть как постом блога, так и обычной страницей (см. раздел «Настройки для всего сайта» на стр. 215). Кстати, ты можешь вообще не заводить блог на своем сайте.

Что же делать дальше? Хороший план должен описывать взаимное расположение страниц и место, которое блог займет на твоём сайте. Если тебе нужен только блог, подойдет и простейшая структура, где одна страница отведена под посты, а вторая — под раздел «Обо мне», как на рис. 4.9.

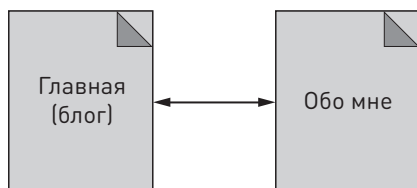


Рис. 4.9. Схема простого блога

Если же ты делаешь портфолио, подобно Ким, то, скорее всего, тебе понадобится несколько страниц. Тогда особенно важно составить четкий план на бумаге (см. рис. 4.10).

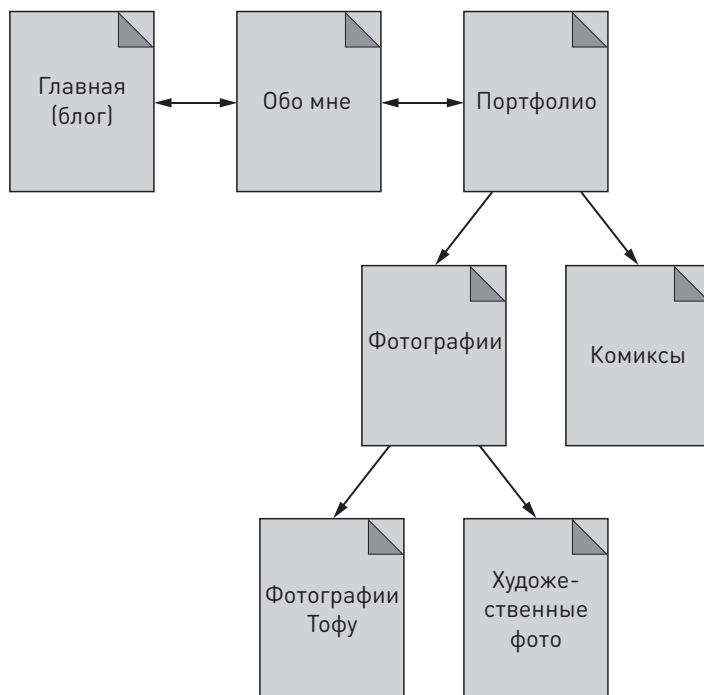


Рис. 4.10. Заранее составленный план сайта поможет тебе создавать контент. Так содержимое твоего сайта будет лучше структурировано

Вооружившись картой сайта, ты можешь серьезно и обстоятельно подойти к созданию страниц. Я еще раз напомним об этом в конце главы, когда у тебя сложится представление о взаимосвязи разных частей WordPress. А сейчас нужно освоиться с WordPress и начать создавать контент.

Создание первой страницы

Давай создадим страницу. Открыв консоль, клики по вкладке «Страницы» в панели слева. Затем выбери **«Добавить новую»**, как показано на рис. 4.11.

После этого откроется *визуальный редактор* (см. рис. 4.12), в котором можно добавлять и редактировать страницы или посты.

Теперь мы создадим в визуальном редакторе простую страницу. Для начала нужно указать ее название. Кликни по фразе «Введите заголовок» и набери на клавиатуре **«Моя страница»**. Затем нажми на кнопку **«Сохранить»** в панели **«Опубликовать»** справа. Сохранение черновика означает, что, пока мы работаем над содержимым, его никто больше не увидит. Теперь нажми на кнопку **«Просмотреть»**. Откроется новое окно или новая вкладка браузера, где ты увидишь свою страницу, оформленную в стиле выбранной тобой темы (см. рис. 4.13). Ты можешь в любой момент открыть предпросмотр черновика страницы (или поста), но посетители увидят ее лишь после того, как ты нажмешь на синюю кнопку «Опубликовать».

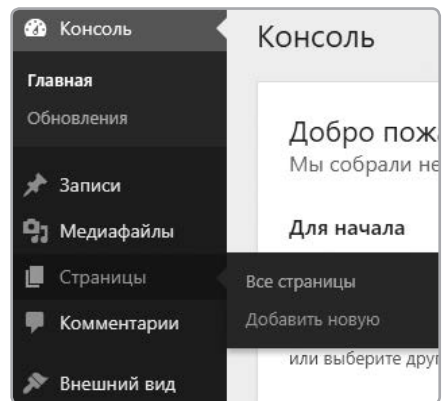


Рис. 4.11. В левой части консоли находится вкладка «Страницы». Чтобы создать новую страницу, нужно кликнуть по вкладке и выбрать «Добавить новую»

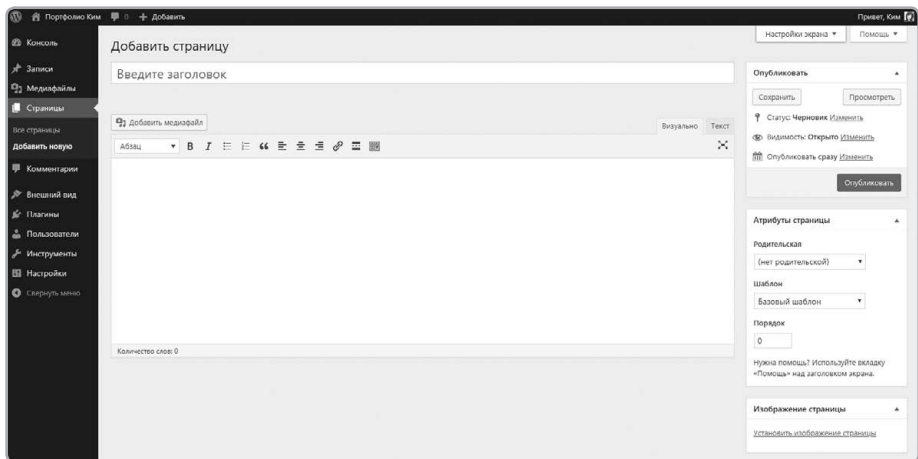


Рис. 4.12. Визуальный редактор страниц. Справа — настройки публикации и атрибутов страницы. В центре — окно редактора, в котором создаются посты, добавляются картинки и т. д. Вверху — поле для названия страницы



Рис. 4.13. Предпросмотр страницы позволяет оценить свою работу, прежде чем она будет выставлена на всеобщее обозрение

Как видишь, в WordPress создавать веб-страницы намного проще: не нужны ни тег `<title>`, ни CSS с настройками оформления. WordPress отделяет контент от структуры и оформления, поэтому, редактируя страницу в консоли, о них можно не думать. Мы просто добавляем нужное содержимое, а остальное WordPress делает за нас.

Закрой новое окно или вкладку, чтобы вернуться в консоль. Давай добавим еще содержимое на нашу тестовую страницу. Напечатай в визуальном редакторе следующее:

Это жирный текст.

Это курсив.

Теперь отформатируй этот текст с помощью *панели форматирования* (см. рис. 4.14). Выдели первую строку и нажми на кнопку **B**, чтобы сделать текст жирным. Затем выдели вторую строку и нажми на кнопку **I**, чтобы появился курсив.

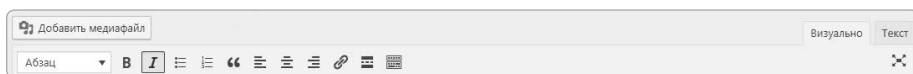


Рис. 4.14. Панель форматирования позволяет менять внешний вид текста, например делать его жирным или курсивным, без помощи HTML

Сохрани черновик и снова открой предварительный просмотр. Результат должен быть примерно таким, как на рис. 4.15.

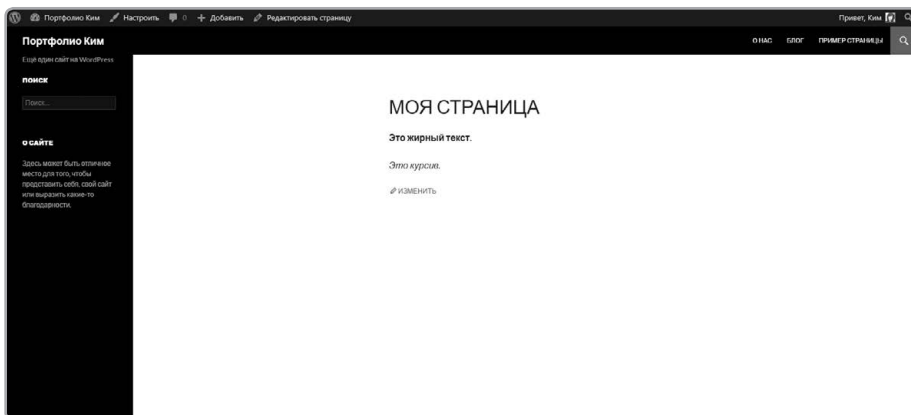


Рис. 4.15. Отформатируй текст в визуальном редакторе и нажми «Просмотреть» — ты увидишь страницу в законченном виде

WordPress отформатировал текст, избавив нас от необходимости вводить теги `` и ``. Кроме того, WordPress начинает каждый абзац с новой строки, хотя мы и не ставили тег `<p>`. Это здорово экономит время при добавлении контента. Визуальный редактор — это WYSIWYG-инструмент (от английского *What You See Is What You Get* — «что видишь, то и получишь»), гораздо более удобный для пользователей, чем HTML. Добавим в визуальном редакторе еще немного содержимого. Введи такую строку:

Кликните здесь

А теперь сделай из фразы «Кликните здесь» ссылку, выделив эту строку и нажав на кнопку **добавления ссылки** в панели редактора, как показано на рис. 4.16:

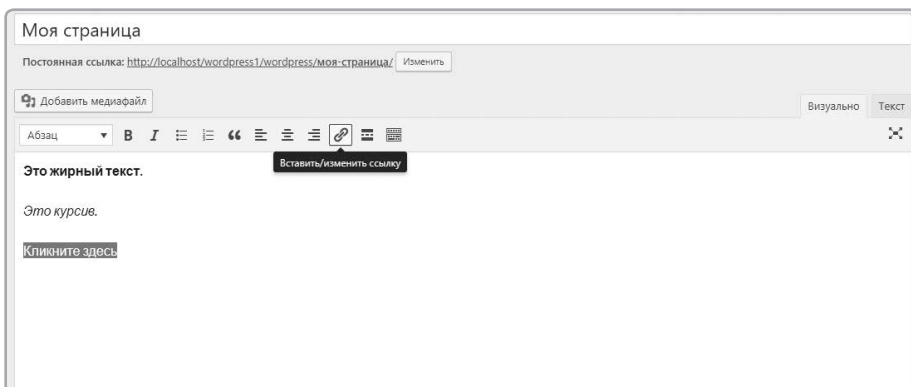


Рис. 4.16. Чтобы создать ссылку, просто выдели текст и нажми на кнопку с соответствующей пиктограммой

Появится диалоговое окно, где нужно указать URL-адрес (см. рис. 4.17). Введи <http://natecooper.co>, а затем нажми на большую синюю кнопку. Видишь, даже в редакторе текст теперь похож на ссылку. Удобно, правда?

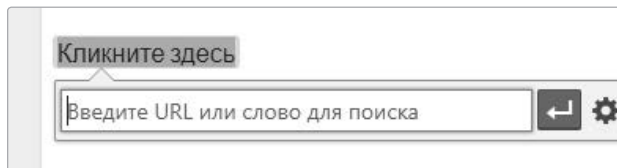


Рис. 4.17. Кнопка создания ссылки открывает диалоговое окно, где нужно указать адрес нужной страницы

Теперь добавим на страницу первую строку со словами «Привет, мир!». Давай сделаем ее заголовком. Однако на первый взгляд в панели редактора нет кнопки создания заголовка. Как же быть? В верхнем ряду слева есть выпадающее меню «Абзац» (см. рис. 4.18). Введи «Привет, мир!», выдели текст и выбери в меню «Абзац» опцию «Заголовок 1». Теперь на странице есть заголовок. Возможно, ты также заметил странную маленькую кнопку — крайнюю справа в верхнем ряду. Она открывает дополнительные опции форматирования.



Рис. 4.18. При нажатии на эту кнопку появляются дополнительные команды форматирования, которые раньше были скрыты

Сохрани черновик и открой предпросмотр страницы. Она должна выглядеть как на рис. 4.19.

Всего за пару минут мы создали в WordPress полноценную веб-страницу. Визуальный редактор позволил нам разметить документ без помощи HTML-тегов. Значит, теперь можно про них забыть, верно? Как бы не так! Знание HTML и CSS никогда не будет лишним. Визуальный редактор WordPress ставит теги за нас, и это сильно экономит время, но HTML все еще здесь. Закрой предпросмотр, чтобы вернуться к редактированию страницы, и кликни по вкладке «Текст» вверху панели редактора. Ты увидишь почти весь HTML-код нашей страницы (см. рис. 4.20).



Рис. 4.19. Когда мы сделали фразу «Привет, мир!» заголовком, она стала выделяться на фоне остального текста

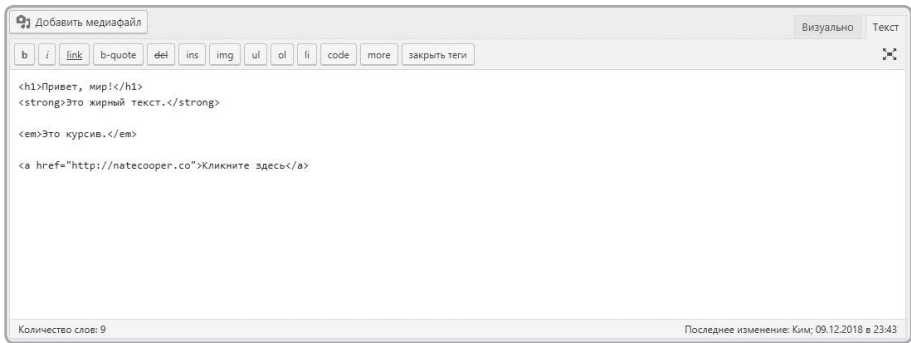


Рис. 4.20. Вкладка «Текст» переводит редактор в режим HTML. Если добавить здесь HTML-теги, это отразится на внешнем виде страницы

Страница все равно написана на HTML, а мы только использовали WordPress для автоматизации процесса. Тут нужно сделать пару замечаний. Во-первых, эта страница не содержит CSS. Как и при создании страниц вручную, CSS-файл хранится отдельно. Во-вторых, отметим специфическую особенность редактора WordPress: здесь нет тегов `<p></p>`. WordPress прячет их даже в HTML-режиме. Однако если, находясь в этом режиме, ты добавишь теги `<p></p>` или `
`, WordPress учтет их при форматировании текста.

Прелесть владения языком HTML в том, что ты не ограничен форматированием, которое WordPress выполняет автоматически. Если ты разбираешься в HTML-коде, ты запросто сможешь его отредактировать, если понадобится.

Миллионы людей каждый день используют WordPress, ничего не зная про HTML, но порой им сложно разметить абзацы, задать отступы и форматирование в точности так, как хочется. Тебе такая ситуация не грозит. Чем глубже твои познания в HTML и CSS, тем проще будет создавать страницы, точно отвечающие твоим потребностям, и тем меньше у тебя будет ограничений.

Добавление изображений

А теперь добавим на нашу страницу изображение. Можно взять ту же картинку, что мы использовали в главе 2. Сейчас все будет гораздо круче, поскольку WordPress хранит и упорядочивает изображения с помощью встроенной библиотеки файлов. Открой страницу в визуальном редакторе, выдели слова «Привет, мир!» и нажми клавишу Delete. Вместо этой фразы мы вставим картинку. Нажми на кнопку **«Добавить медиафайл»** (см. рис. 4.21), и на твоём экране появится окно вставки изображения, как на рис. 4.22.

Добавить медиафайл

Рис. 4.21. Кнопка добавления изображений

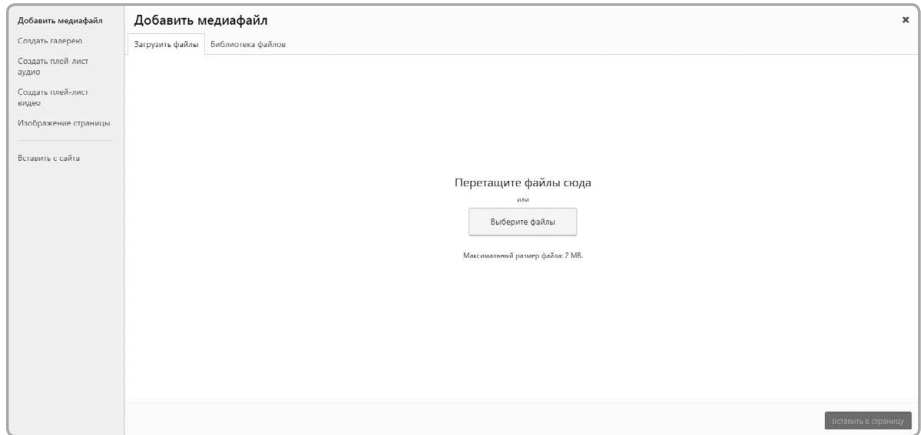


Рис. 4.22. Можно загрузить изображение с компьютера, нажав на кнопку «Выберите файлы», либо просто перетащить его на этот экран

Есть другой способ: можно перетащить картинку прямо в визуальный редактор. Тогда на экране появится окно, как на рис. 4.23.

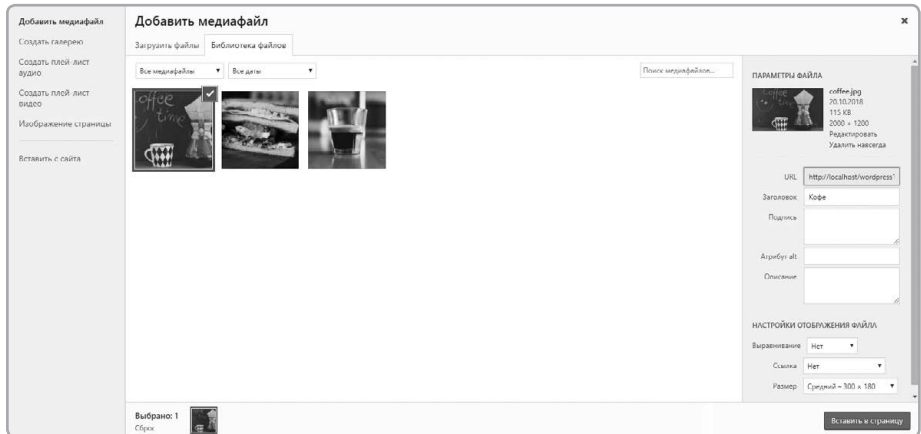


Рис. 4.23. В библиотеке файлов содержатся все изображения, когда-либо загруженные на сайт. Еще там можно хранить видеоролики, документы в форматах PDF и Word и многое другое

Сразу же появится индикатор загрузки. Когда он дойдет до конца, ты увидишь в библиотеке файлов свою картинку в синей рамочке и с флажком в верхнем правом углу (см. рис. 4.23). Это значит, что изображение загружено и готово к добавлению на страницу. Ловко, правда?

Итак, картинка хранится в библиотеке файлов. Теперь ее можно добавить на нашу страницу. Возможно, ты уже заметил в нижнем правом углу кнопку «Вставить в страницу», но не будем спешить — сперва нужно задать свойства картинки.

Справа от выбранного изображения указана информация о нем (см. рис. 4.24). Помнишь, как при использовании тега `` в HTML-коде мы указывали `alt="описание"`, чтобы поисковые системы знали, что изображено на картинке? В WordPress тоже можно задать `alt`, введя его в поле «Атрибут alt». Поля «Заголовок» и «Описание» заполнять необязательно. Обычно они не видны посетителям, но упрощают поиск изображения в консоли. Например, можно ввести в поле «Описание» несколько слов о картинке, чтобы потом быстро найти ее по ним. Поле «Подпись» также необязательное, но если его заполнить, то на странице отобразится подпись к картинке. Нужно взять в привычку хотя бы заполнять поле «Атрибут alt» — это очень важно для SEO. Также его используют программы для чтения с экрана, которыми пользуются слепые и люди с ослабленным зрением.

Выпадающее меню «Выравнивание» мы пока пропустим, зато выберем для поля **«Ссылка»** значение **«Нет»**. Не забывай: картинки могут быть ссылками, и по умолчанию в WordPress они ссылаются сами на себя. Это удобно, если на страницу вставлена миниатюра: кликнув по ней, посетитель увидит полноразмерную версию. Но сейчас нам нужна просто картинка, так что зададим для ссылки значение «Нет». Также WordPress может менять размер изображения на странице. Если картинка слишком велика, у тебя будет четыре варианта на выбор: «Миниатюра», «Средний», «Большой», «Полный». Давай выберем **«Средний»**, если есть такая возможность (рис. 4.25). Нажав на кнопку **«Вставить в страницу»**, ты увидишь что-то вроде рис. 4.26.

ПРИМЕЧАНИЕ

WordPress не увеличивает изображения. Если твоя картинка меньше размеров «Миниатюра», «Средний» или «Большой», ты таких опций не увидишь.

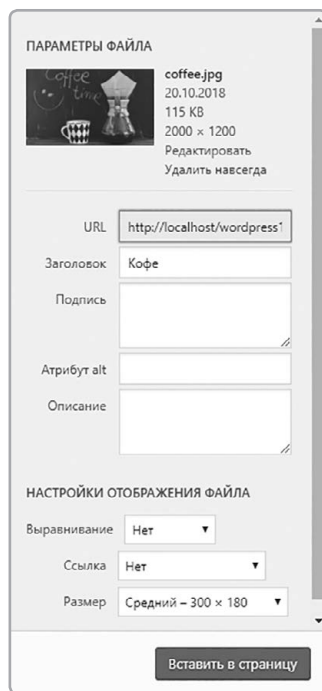


Рис. 4.24. В WordPress есть много опций для управления изображениями, включая подписи, атрибуты `alt` и даже возможность менять размер, чтобы картинка умещалась на странице

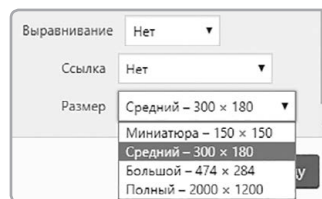


Рис. 4.25. Когда ты загружаешь картинку в WordPress, он автоматически создает ее миниатюру, а также версии среднего и большого размера. Иногда для маленьких картинок доступны только варианты «Миниатюра» и «Средний»

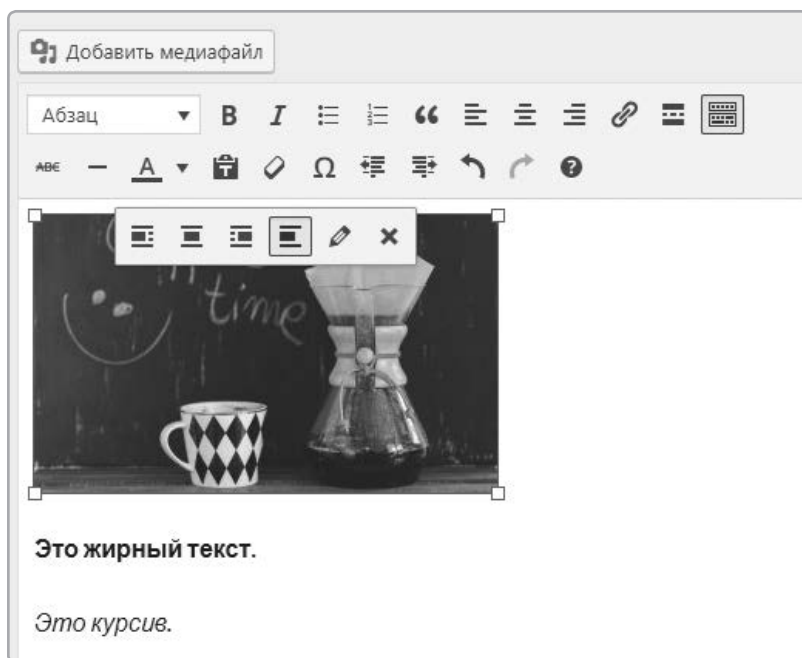



Рис. 4.26. Фотография, добавленная в визуальном редакторе, отображается как картинка, а не как HTML-код

Кликни по картинке, чтобы выделить ее. Вокруг нее появятся восемь квадратов, которые можно перетаскивать, чтобы изменить размер. А чтобы удалить картинку, нажми на иконку «X». Не волнуйся: она исчезнет со страницы, но останется в библиотеке файлов, откуда ее можно будет добавить снова. Иконка с карандашом вернет тебя к правке описания, подписи или атрибута `alt`. Пока изображение выделено, можно настроить его позицию кнопками выравнивания. Также можно сделать его ссылкой (тем же способом, что и обычный текст). На опубликованной странице картинка будет выглядеть в точности как в визуальном редакторе. Кликни **«Сохранить»** и **«Просмотреть»**, чтобы увидеть страницу целиком (см. рис. 4.27).

Теперь вернись в визуальный редактор. Кликни по картинке, а затем по кнопке **выравнивания влево**, как показано на рис. 4.28.

Кликнув **«Сохранить»** и **«Просмотреть»**, ты увидишь, что теперь текст выводится справа от картинки. Если тебя это не впечатлило, значит, тебе не доводилось тратить уйму времени на выравнивание картинок в HTML. Когда ты нажимаешь на кнопку выравнивания влево или вправо, WordPress назначает изображению особый класс, для которого в CSS текущей темы задано свойство `float:left` или `float:right`, в результате чего текст обтекает картинку с противоположной стороны.

МОЯ СТРАНИЦА



Это жирный текст.

Это курсив.

[Кликните здесь](#)




















 ИЗМЕНИТЬ

Рис. 4.27. Законченная страница выглядит почти так же, как в визуальном редакторе

Добавить медиафайл

Абзац **B** *I*   “ ”     

АВБ —         



Это жирный текст.

Это курсив.

Кликните здесь

Рис. 4.28. Когда картинка выделена, ее положение можно настроить кнопками выравнивания влево и вправо, и тогда текст будет ее обтекать — совсем как в текстовом редакторе

Публикация страницы

Мы завершили наполнение страницы и готовы выставить ее на всеобщее обозрение. На панели публикации в правой части страницы нажми большую синюю кнопку **«Опубликовать»** (см. рис. 4.29), чтобы сделать страницу доступной онлайн.

После этого перейди во фронтенд, чтобы оценить результат. Обрати внимание, что наша страница появилась в меню навигации (см. рис. 4.30).

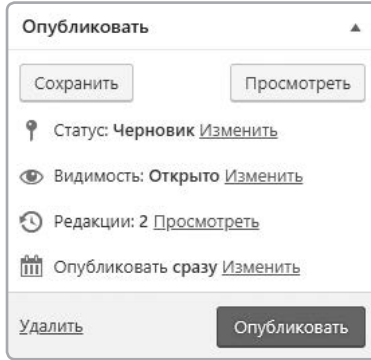


Рис. 4.29. С помощью панели публикации можно выложить страницу на сайт или сохранить ее как черновик, чтобы доделать потом

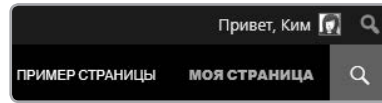


Рис. 4.30. Опубликованные страницы появляются в меню навигации фронтенда

Готово. После нажатия «Опубликовать» твою страницу увидят все посетители. А что, если ты поторопился и хочешь снова сделать ее черновиком? Никаких проблем. Вернись в консоль и кликни по вкладке «Страницы» на панели слева. Ты увидишь список всех страниц своего сайта (см. рис. 4.31).

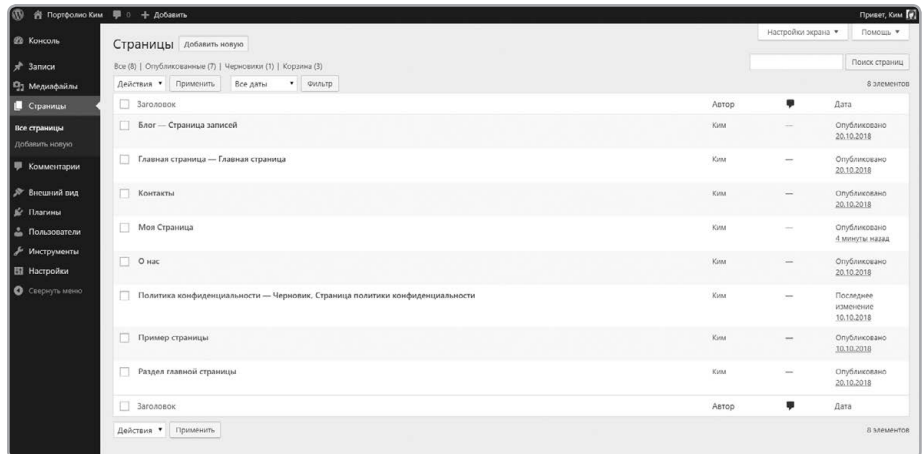


Рис. 4.31. В разделе «Все страницы» на вкладке «Страницы» перечислены все страницы сайта

Кликни по названию «Моя страница», и ты перейдешь к ее редактированию. В панели публикации нажми на «Редактировать» рядом со строчкой «Статус» и выбери опцию «Черновик». Нажми «ОК», а затем «Обновить» (см. рис. 4.32).

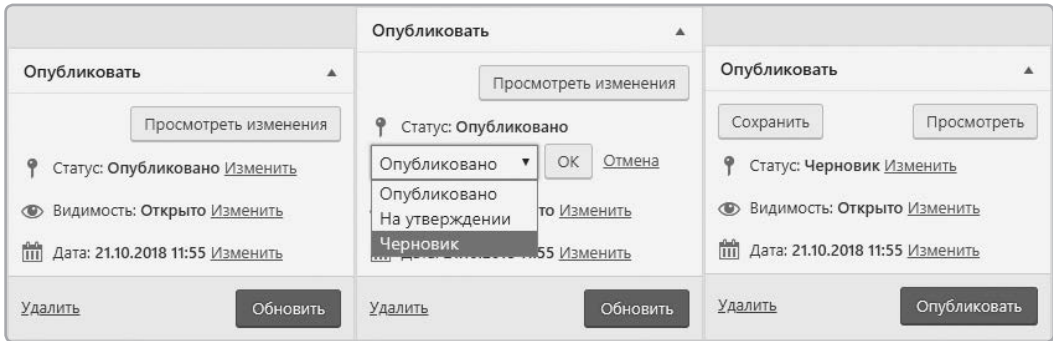


Рис. 4.32. Даже если страница уже опубликована, ее можно вернуть в черновики. Она останется доступной в консоли для просмотра и правки, но исчезнет из фронтенда, то есть посетители ее не увидят

Поздравляю! Ты только что отменил публикацию страницы со своего веб-сайта! Но данные сохранились, и ее всегда можно снова опубликовать. Страница просто вернулась в режим черновика, то есть она видна в консоли, но скрыта от посетителей. В WordPress также есть настройка «Видимость», которая позволяет создавать личные или защищенные паролем посты (см. рис. 4.33).

Одна из самых классных функций панели публикации — возможность выбрать любую дату публикации в прошлом или будущем. Если выбрать дату в прошлом, будет казаться, что страница действительно была опубликована тогда. Если же задать дату в будущем, то до нее страница будет видна только тебе, а затем WordPress автоматически опубликует ее. Здорово, да? Причем эта функция работает одинаково и для страниц, и для постов.

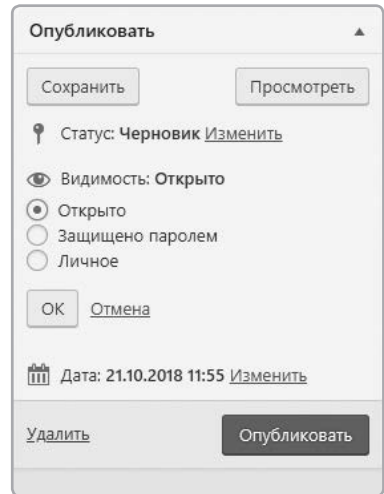


Рис. 4.33. Открытую страницу можно сделать личной (видимой только создателю) или защитить паролем

Управление страницами

Что, если тебе захочется сотворить со страницами что-нибудь поинтереснее? Ким собиралась создать портфолио, причем этот раздел у нее делится на два подраздела: «Рисунки» и «Фотографии». Сделать такую структуру сайта в WordPress совсем не сложно (см. рис. 4.34).

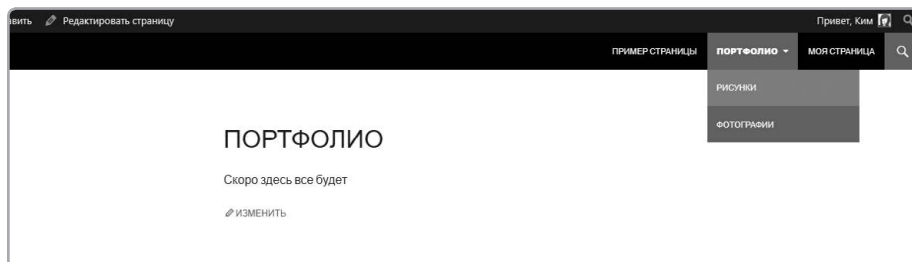


Рис. 4.34. Родительская страница — это более высокий уровень в иерархии. Выпадающее меню показывает, что в разделе «Портфолио» есть две страницы: «Рисунки» и «Фотографии». Сейчас мы находимся на странице «Рисунки», как ты можешь заметить по адресной строке

Страницы на верхнем уровне иерархии (как портфолио Ким) называют *страницами верхнего уровня*, или *родительскими*. А страницы второго и далее уровней называют *дочерними*. Панель навигации, помогающая посетителям ориентироваться на сайте, показывает взаимосвязь родительских и дочерних страниц.

Наполнять сайт мы начнем с раздела «Портфолио». Войди в консоль и затем кликни **«Страницы»** в панели слева. Ты увидишь список страниц.

Мы создадим три страницы: сначала «Портфолио», затем «Фотографии» и наконец «Рисунки». Потом мы вложим «Фотографии» и «Рисунки» внутрь «Портфолио». Для начала добавим на эти страницы текст «Скоро здесь все будет».

Кликни **«Добавить новую»**, дай странице название **«Портфолио»**, а затем нажми **«Опубликовать»**, как на рис. 4.35.

Страница «Портфолио» готова, теперь добавь еще две — «Фотографии» и «Рисунки». Нажми на кнопку **«Добавить новую»** вверху страницы, как показано на рис. 4.36.

Пусть это будет страница **«Фотографии»**. Дай ей соответствующее имя и введи в редакторе текст **«Скоро здесь все будет»**. Но прежде чем публиковать ее, взгляни в панель «Атрибуты страницы» в нижней правой части экрана. Там ты увидишь выпадающие меню «Родительская» и «Шаблон», а также поле «Порядок». Сейчас в **«Родительской»** указана опция «[нет родительской]». Поставь вместо нее значение **«Портфолио»**.

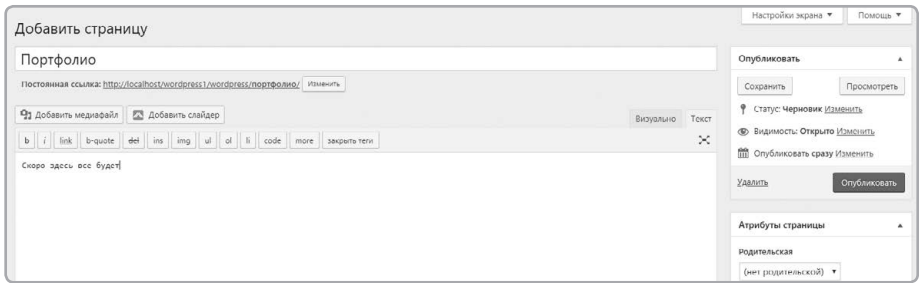


Рис. 4.35. Чтобы понять, как выстраивается иерархия сайта, создай пустую страницу «Портфолио». В качестве содержимого введи слова «Скоро здесь все будет»

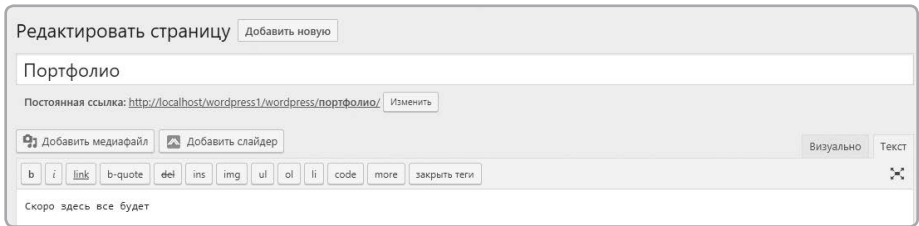


Рис. 4.36. Когда страница опубликована, вверху появляется кнопка «Добавить новую»

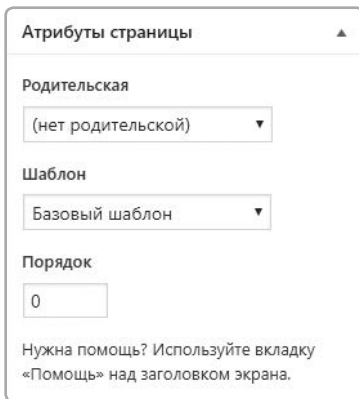


Рис. 4.37. В окне редактора есть панель «Атрибуты страницы», где можно выбрать родительскую страницу

Нажми **«Опубликовать»**. Затем кликни **«Добавить новую»** и повтори тот же процесс для страницы «Рисунки». Перейдем во фронтенд сайта. Обрати внимание, что теперь наверху показана страница «Портфолио» с двумя дочерними страницами в выпадающем меню. В консоли дочерние страницы помечены знаками тире — это отличает их от страниц верхнего уровня (см. рис. 4.38).

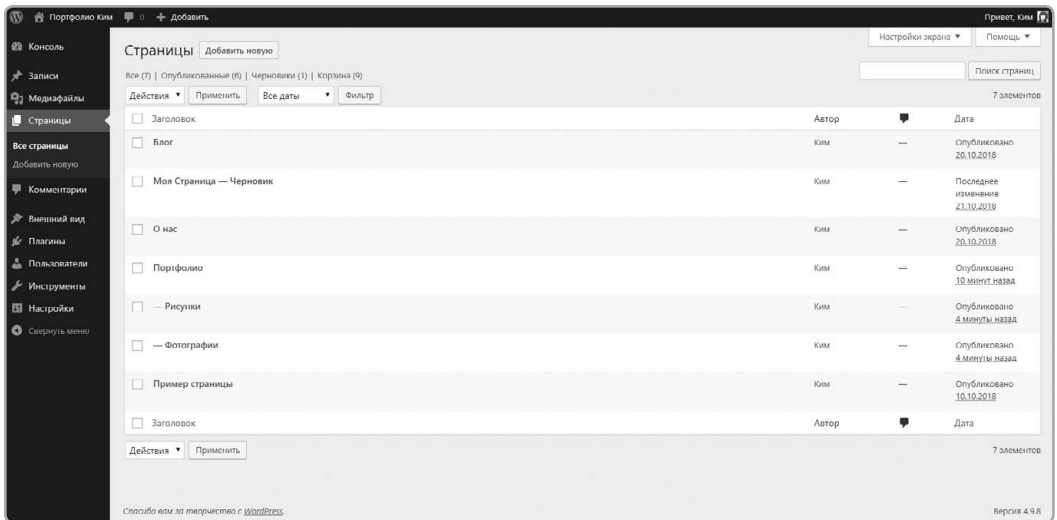


Рис. 4.38. В консоли дочерние страницы легко отличить от родительских: перед ними стоит тире

Создание поста в блоге

Теперь создадим пост в блоге. Перейди в консоль и выбери **«Записи»**, а затем **«Добавить новую»**. Ты снова окажешься в визуальном редакторе (см. рис. 4.39).

Назови страницу «Пост 2» и введи в редакторе текст **«Это мой второй пост»**. Затем нажми **«Опубликовать»**. Чтобы освоиться, можешь создать еще пару постов. Когда закончишь, кликни по названию сайта в левом верхнем углу, чтобы перейти во фронтенд.

Помни: посты отображаются на странице блога. Туда ты и попадешь, кликнув по названию сайта. Наверху страницы отображаются новые посты, а ниже — те, что были сделаны ранее (см. рис. 4.40). По умолчанию посты отображаются в обратном хронологическом порядке — от новых к старым.

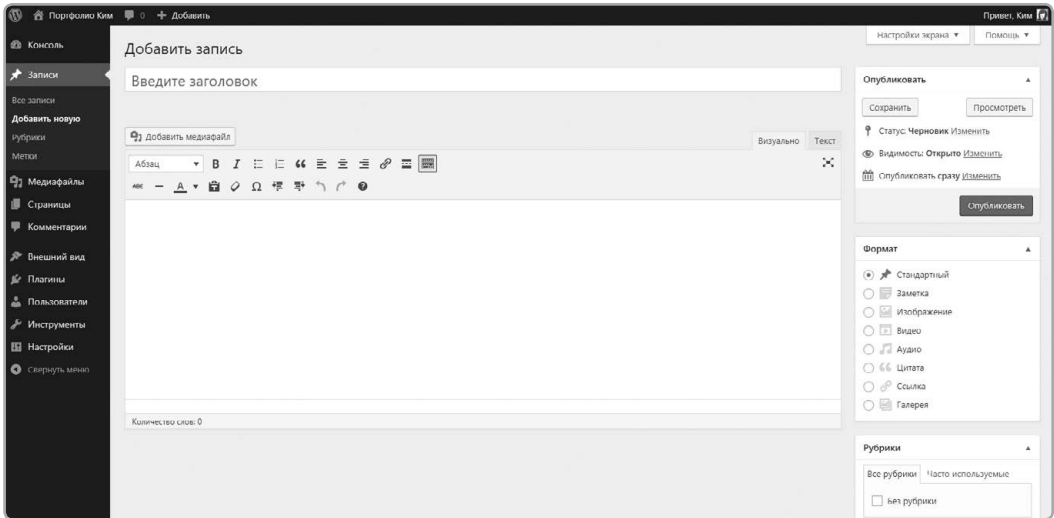


Рис. 4.39. Визуальный редактор постов почти не отличается от редактора страниц. Посты могут содержать то же, что и страницы: картинки, ссылки и отформатированный текст

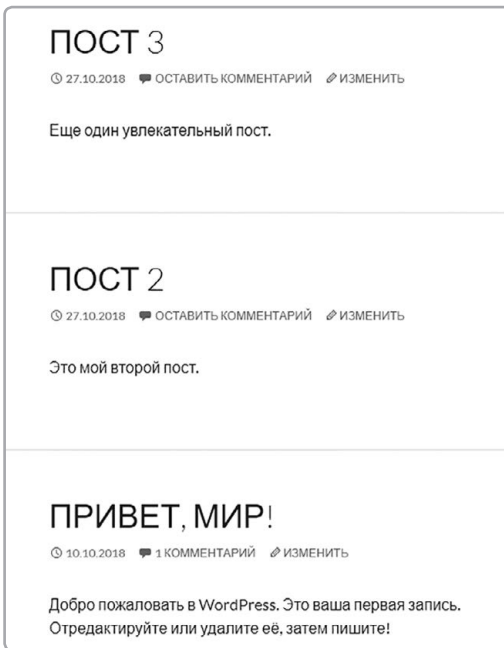


Рис. 4.40. На странице блога собраны все посты. В отличие от статичных страниц, они отображаются все вместе и сортируются по дате (самые новые наверху)

Задаем структуру: рубрики и метки постов

Твой блог можно структурировать при помощи меток и рубрик. Вернись в консоль и кликни **«Записи»**, чтобы увидеть список всех имеющихся постов, как на рис. 4.41.

Предположим, ты собираешься выкладывать в блог обзоры книг. Тогда посты с обзорами нужно сгруппировать, чтобы читатель мог легко найти все записи по этой теме. Кликни **«Добавить новую»**, чтобы создать новый пост. Дай ему название **«Великий Гэтсби»**. В правом нижнем углу редактора ты увидишь секцию рубрик и меток.

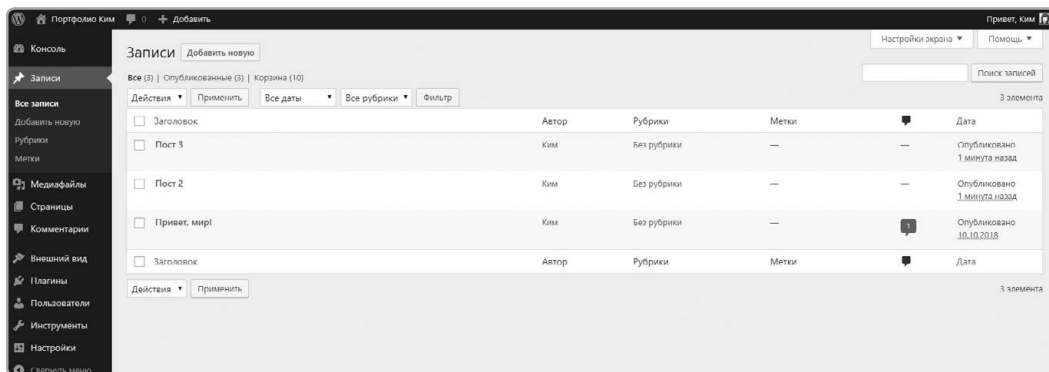


Рис. 4.41. Вкладка «Записи» позволяет увидеть все посты блога — как черновые, так и опубликованные

Это спецификаторы, которые можно создавать и использовать для группировки постов по темам. Рубрики предназначены для более широких категорий, а метки — для указания более узких тем. Для поста про книгу «Великий Гэтсби» нужна рубрика под названием «Обзоры книг», ведь мы планируем и дальше писать обзоры. Сейчас такой рубрики нет, а значит, ее пора добавить. Кликни по ссылке **«Добавить новую рубрику»**. Введи заголовок **«Обзоры книг»** и нажми на кнопку **«Добавить новую рубрику»**. Она появится в списке, и ее можно будет выбрать для нашего поста (см. рис. 4.42).

Метки обычно делают более конкретными. Введи в поле «Метки» слова **«Фитцджеральд»**, **«бутлегеры»**, **«эпоха джаза»** (без кавычек и через запятую), а затем кликни «Добавить» (см. рис. 4.43).

Одному посту может соответствовать несколько рубрик и меток. Как правило, меток бывает больше, чем рубрик, поскольку они более конкретны. Впрочем, это не правило, а скорее традиция.

Введи в качестве содержимого поста фразу **«Скоро здесь все будет»** и кликни **«Опубликовать»**. Наверху появится ссылка на пост (см. рис. 4.44). Нажми **«Просмотреть запись»**, чтобы проверить свою работу во фронтенде.

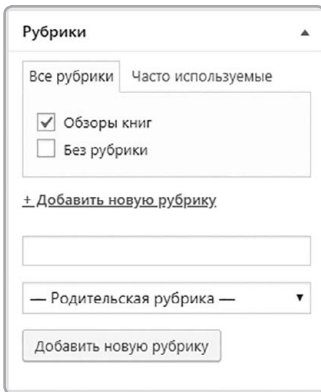


Рис. 4.42. Чтобы создать рубрику, достаточно ввести ее название и кликнуть «Добавить». Каждый пост должен относиться хотя бы к одной рубрике. После создания рубрика станет доступна для всех новых постов

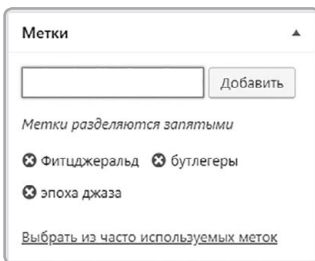


Рис. 4.43. Метки позволяют описать содержимое поста более конкретно. Когда ты добавишь метку к посту, она станет доступна для всех новых постов

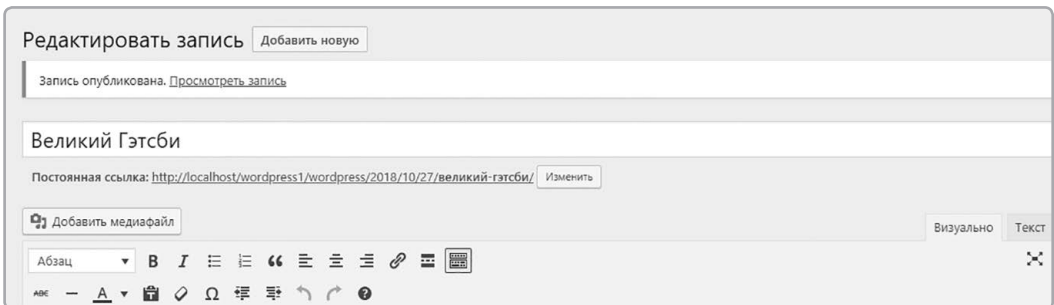


Рис. 4.44. После публикации поста наверху страницы появится ссылка на его просмотр

WordPress автоматически создает ссылки на рубрики и метки и показывает их под каждым постом (см. рис. 4.45). Переходя по ним, читатель легко отыщет другие посты на ту же тему.

Чем последовательнее ты будешь проставлять рубрики и метки, тем больше пользы они принесут. Сперва это может показаться не особо важным, но, когда в твоём блоге будет много постов, читателям пригодится возможность найти нужный контент за один клик.

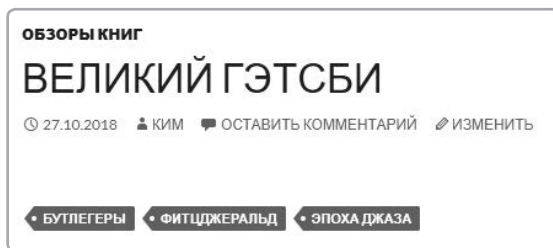


Рис. 4.45. В теме *Twenty Fourteen* рубрики отображаются над постом, а метки — под ним. И те и другие принимают вид ссылок

Верхнее изображение

В некоторых темах можно задать изображение, которое будет отображаться над каждым постом (см. рис. 4.46). Обычно это панорамное изображение в «шапке» страницы или поста, как на рис. 4.47, или миниатюра рядом с постом. Оно поможет сделать блог более привлекательным внешне. Поскольку мы активировали тему *Twenty Fourteen*, которая поддерживает эту функцию, под списком тегов будет еще одна настройка — **«Изображение записи»**. Кликни по ссылке, чтобы задать картинку для этого поста. При этом размер изображения установить вручную нельзя — его определяет тема.

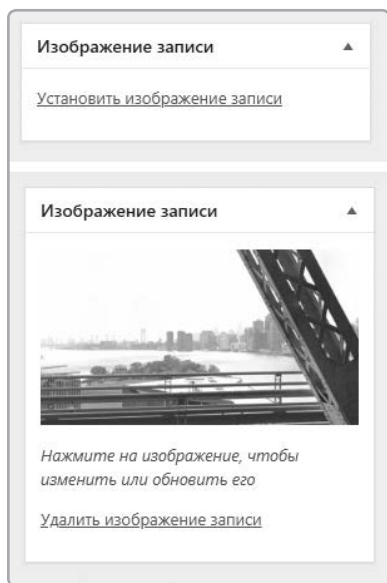


Рис. 4.46. Некоторые темы позволяют задать для поста верхнее изображение



Рис. 4.47. Пост с верхним изображением в теме Twenty Fourteen. Эта функция поддерживается и во многих других темах

Посты с видео, фотографиями и цитатами

Еще одна специфическая для тем функция, которая поддерживается в Twenty Fourteen, — *форматы постов*. Что же это значит? В некоторых темах «журнального типа» материалы вроде фотографий, видеороликов или цитат могут иметь особое оформление. Например, если выбрать для поста формат «Цитата», то тема отобразит его крупным шрифтом, чтобы он выделялся на фоне остальных. А для поста с видео есть особое оформление, фокусирующее внимание посетителей на видеоконтенте. Задать формат поста можно через панель «Формат», расположенную над панелью рубрик (см. рис. 4.48). Темы с поддержкой форматов помогают *выбрать оптимальный способ отображения данных*.

Каждому формату соответствует свое оформление (см. рис. 4.49).

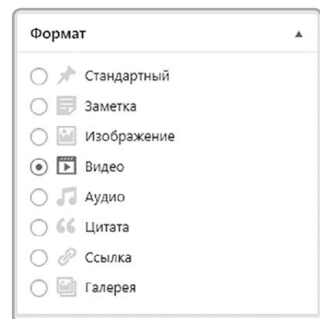


Рис. 4.48. Настройка формата в консоли позволяет выбрать оптимальный способ отображения данных

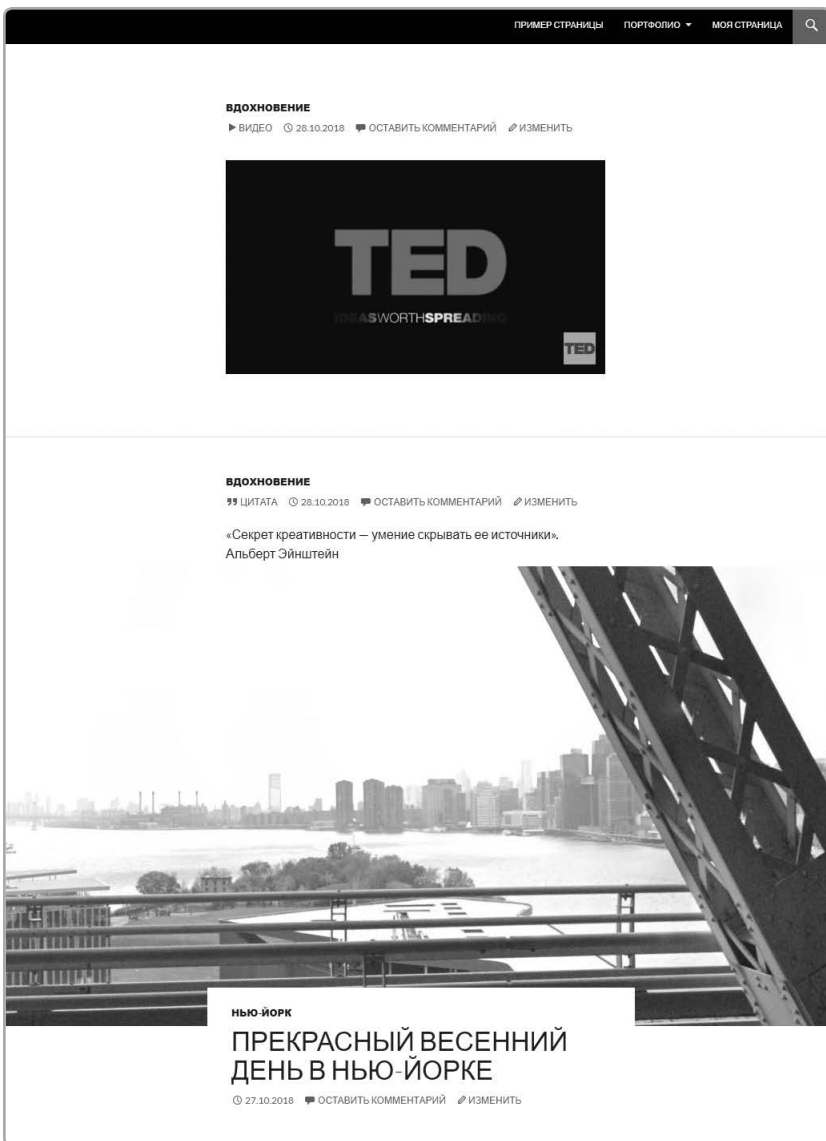


Рис. 4.49. Сверху вниз: пост с видео, пост-цитата и пост с обычным форматированием в теме Twenty Fourteen

Правка и удаление содержимого

Не стоит беспокоиться по поводу пустых страниц, которые мы создали для тренировки. С помощью консоли их легко обновить или удалить. В ней есть и функции для работы с другими типами контента, например фотографиями и постами. Открой консоль и кликни на **«Записи»**. Ты увидишь список всех твоих постов, как на рис. 4.50.

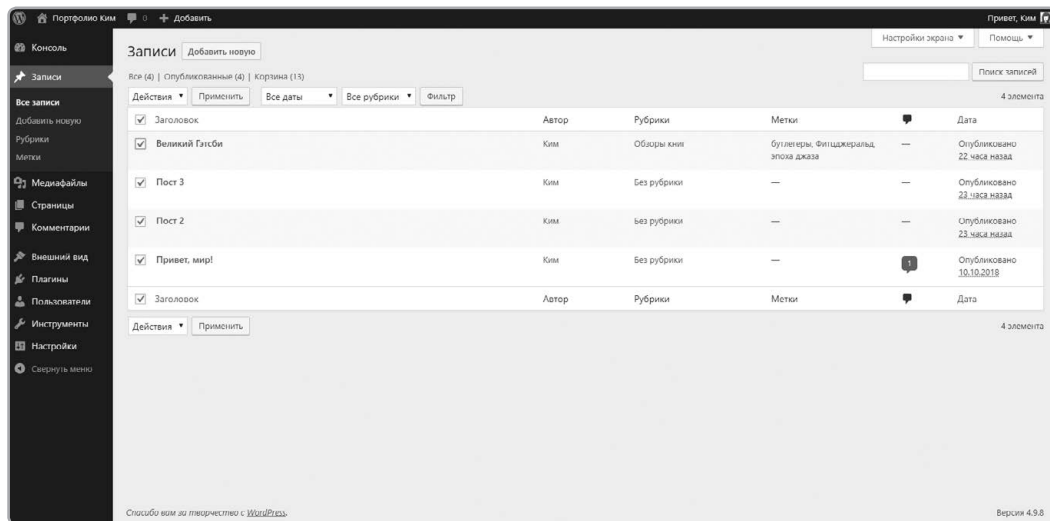


Рис. 4.50. В разделе «Все записи» на вкладке «Записи» показаны все посты, когда-либо добавленные в блог WordPress. Галочки слева от названий предназначены для групповых действий с постами

Над списком расположены инструменты для управления постами или их поиска. Выпадающее меню «Действия» позволяет редактировать и удалять данные. Просто поставь галочки напротив элементов, которые хочешь изменить, и выбери действие из меню. Например, можно отправить в корзину все тестовые страницы: отметить их галочками и выбери в меню «Удалить», как на рис. 4.51. Как и корзину на твоём компьютере, корзину WordPress время от времени нужно опустошать. Для этого кликни по вкладке **«Корзина»**, а затем **«Очистить корзину»** — и ее содержимое будет удалено безвозвратно.

Также можно удалить страницу или пост через визуальный редактор — используй для этого ссылку **«Удалить»**.

В этой главе мы разобрались, как управлять контентом — страницами, постами и изображениями. Но мы пока ничего не говорили о настройке шрифтов, цветов, компоновке страниц и других функциях WordPress. Давай посмотрим, куда дальше отправится Ким!

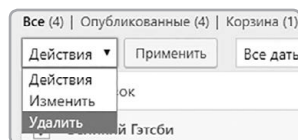
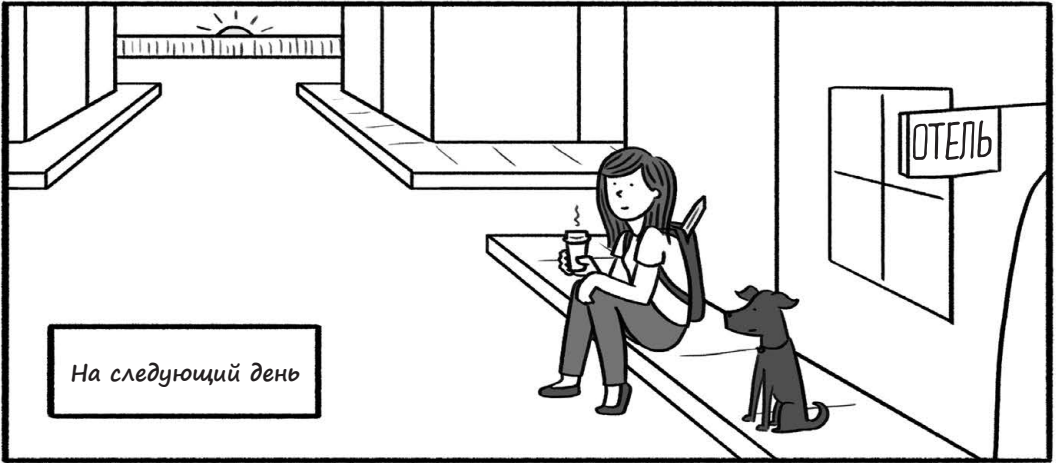


Рис. 4.51. Поставив галочки напротив всех постов, которые ты хочешь изменить, выбери в меню «Действия» команду «Редактировать» или «Удалить»

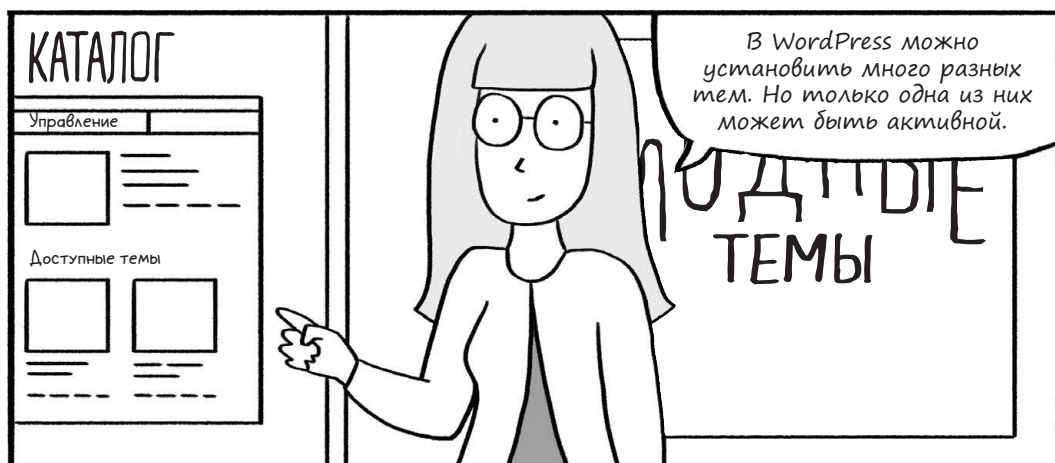
5

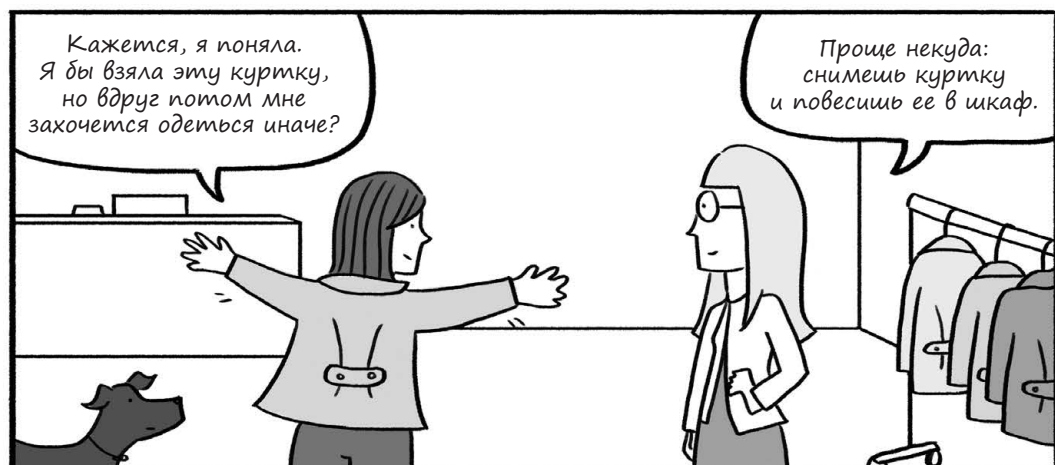
Индивидуальная настройка WordPress

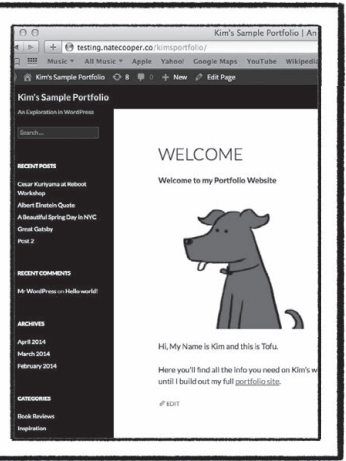
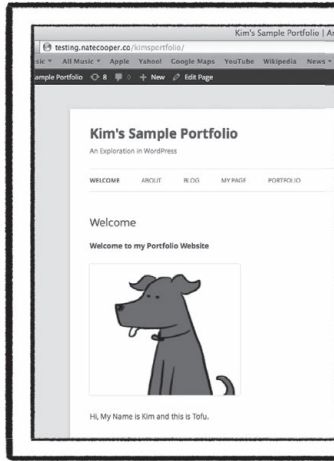


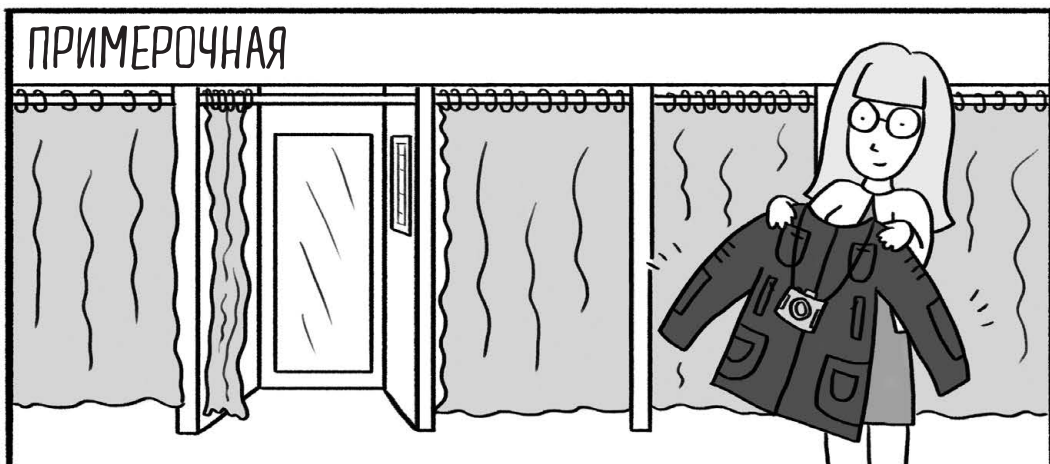


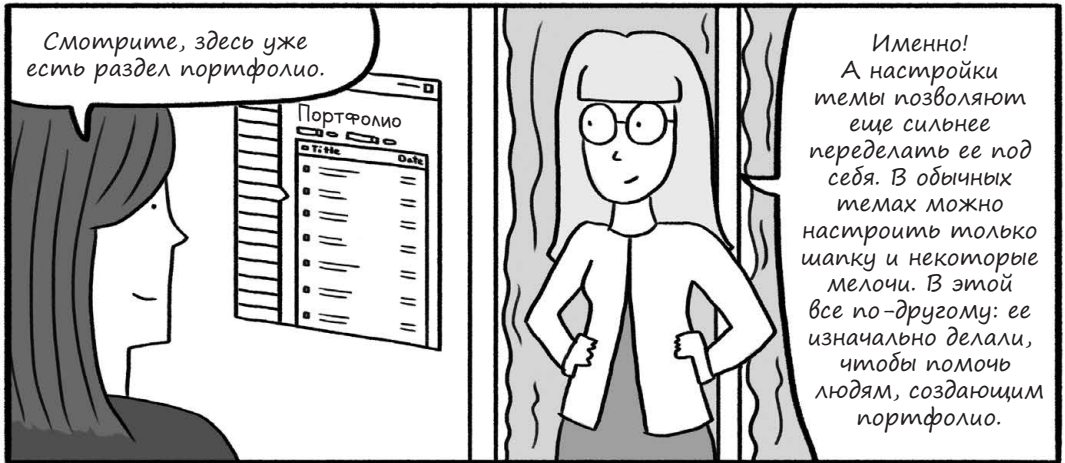
Панель «Внешний вид»





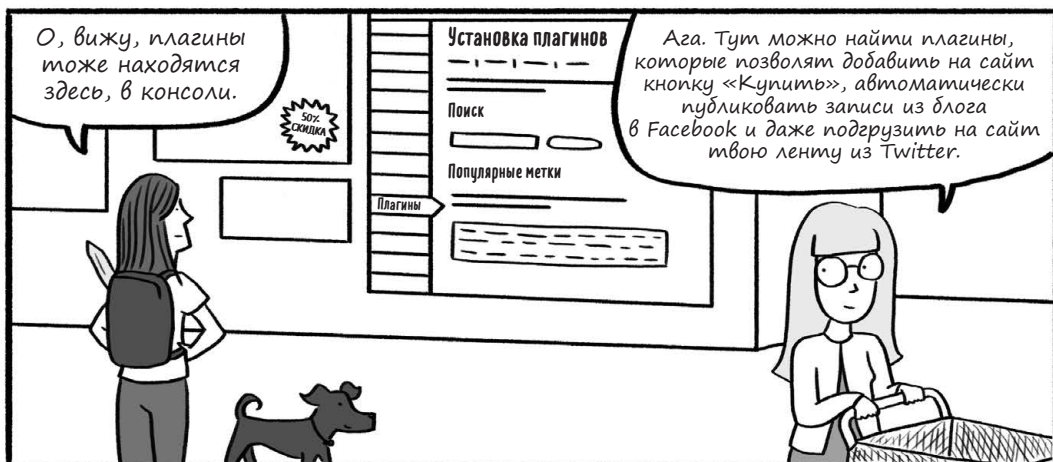


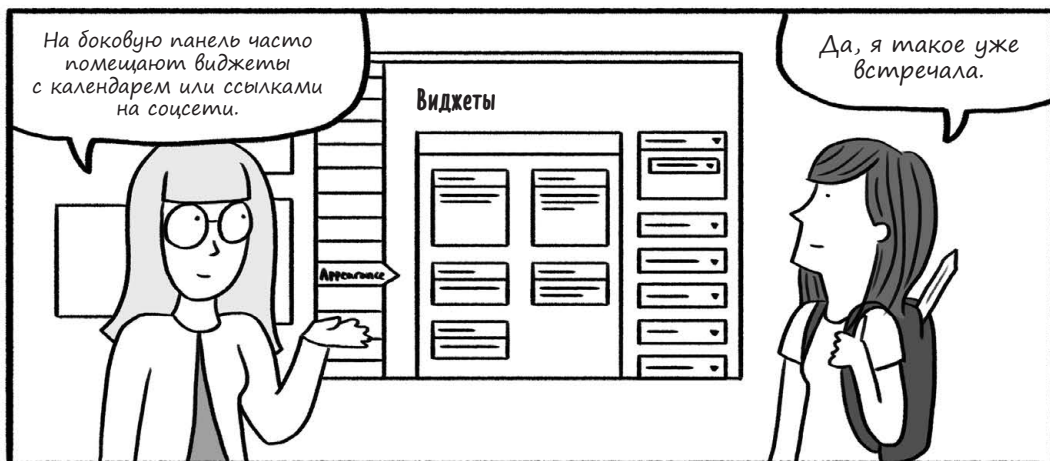






Ким устанавливает на сайт плагины



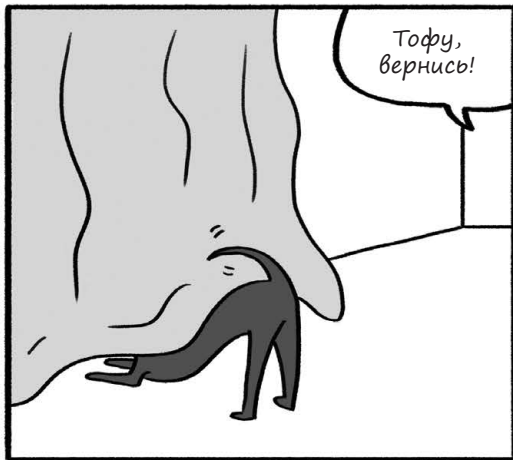






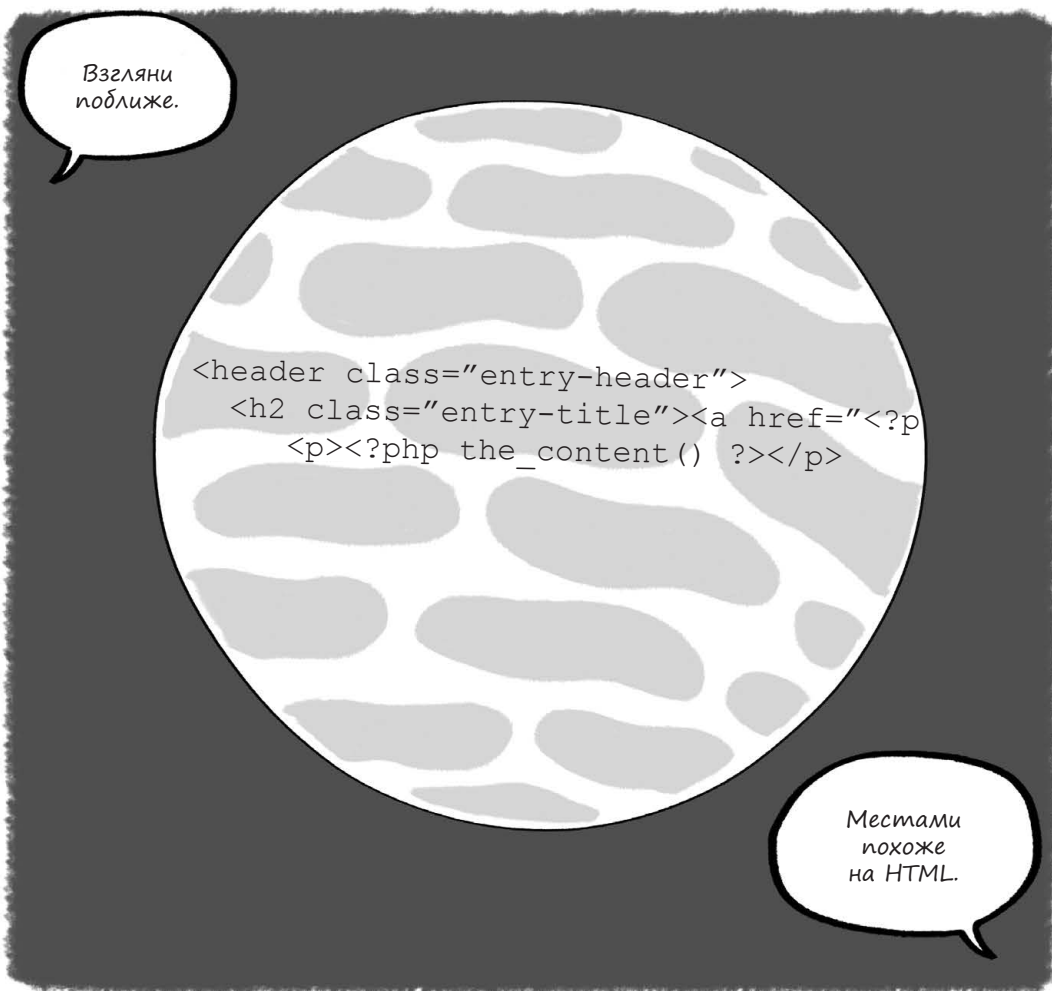
Ким заглядывает за занавеску











Верно. HTML и CSS — основа всех тем WordPress. Зная их, ты можешь самостоятельно менять свою тему.



А все остальное?



В WordPress используется скриптовый язык под названием PHP. Он удобен тем, что его можно комбинировать с HTML, а еще он позволяет использовать команды, называемые функциями, чтобы изменять чужие темы или создавать свои.



Значит, этим и занимался портной?



Ага. Он создал дочернюю тему, что позволило ему изменить оформление с помощью HTML, CSS и PHP.





Меняем оформление: главное о темах

Вплоть до этого момента мы в основном говорили о контенте: словах, ссылках и картинках на страницах и в постах. Если не считать простейшего форматирования, вроде выравнивания картинок и переносов строк, мы никак не меняли внешний вид сайта. Это сделано специально: все, что относится к оформлению, в WordPress связано с *темой* — шаблоном, задающим внешний вид сайта. При этом в каждый момент времени активной может быть только *одна* тема. Существуют миллионы тем для WordPress, и очень здорово, когда есть такой выбор, хотя поначалу просто глаза разбегаются. Кроме того, ты можешь создать собственную тему с нуля.

Находясь в консоли, открой панель «Внешний вид», затем перейди в раздел «Темы», и ты увидишь каталог доступных тем, как на рис. 5.1. Одно из главных отличий сайта на WordPress.com от сайта на собственном хостинге — это количество доступных тем. На WordPress.com их очень много: одни бесплатные, а за другие надо платить (их еще называют *премиальными*). Однако ты не сможешь установить тему, которой нет в каталоге.

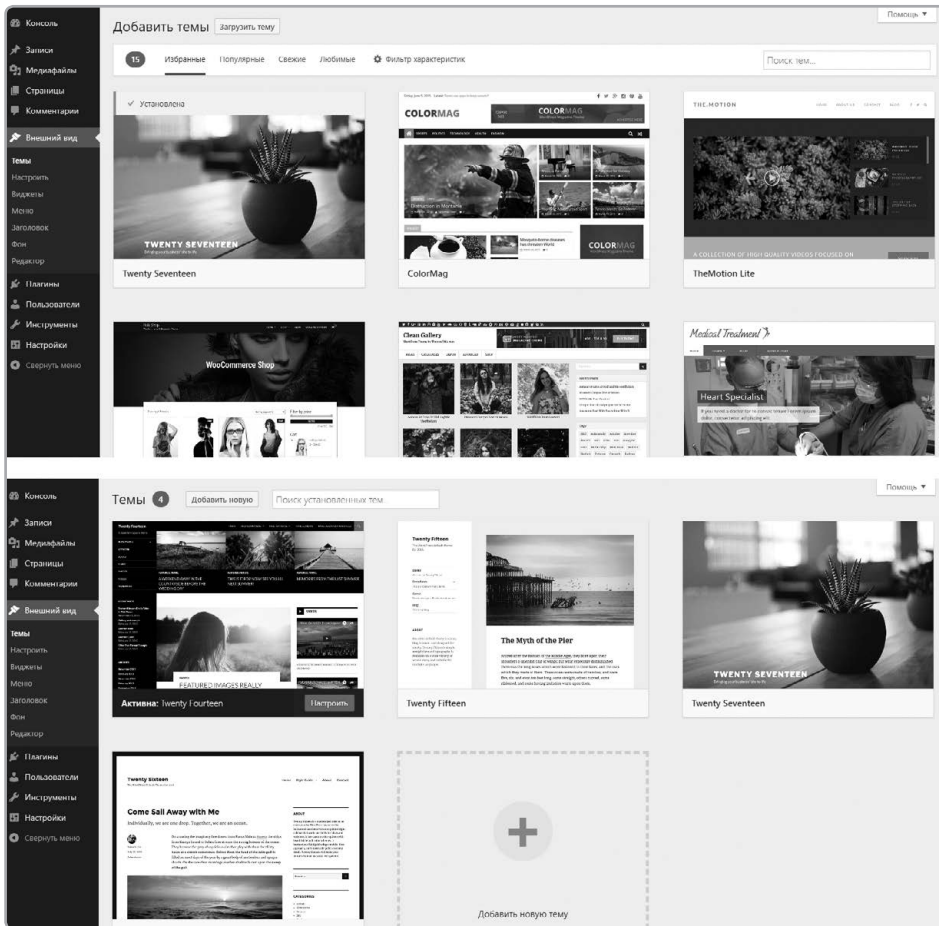


Рис. 5.1. Список тем на портале WordPress.com (вверху) и на отдельном хостинге (внизу). Выбрать тему можно в соответствующем пункте раздела «Внешний вид»

Чтобы изменить оформление сайта, достаточно активировать новую тему. При этом предыдущая тема автоматически отключится. Активация новой темы не повлияет на контент (посты, страницы и картинки), а лишь поменяет способ его отображения. Нажми на «Добавить новую тему», чтобы открыть бесплатный каталог на сайте WordPress.org (см. рис. 5.2). Здесь показан поиск темы с гибким дизайном (responsive). Чтобы выбрать приглянувшуюся тему, кликни по ссылке «Установить». Тема будет загружена с сайта WordPress.org и установлена на твой хостинг.

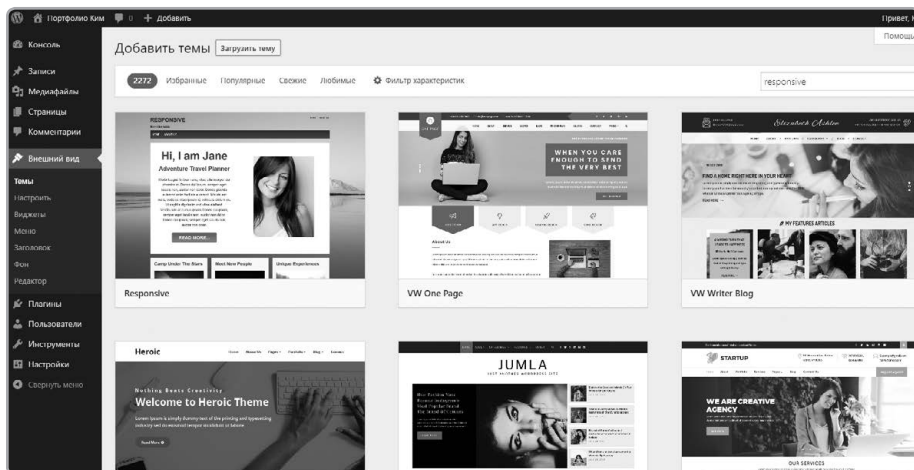


Рис. 5.2. Кликни «Добавить новую тему» и воспользуйся поиском, чтобы найти в каталоге подходящую бесплатную тему

После установки темы на хостинг нужно активировать ее (см. рис. 5.3). Имей в виду, что темы могут кардинально менять оформление сайта. На рис. 5.4 показано одинаковое содержимое при использовании двух разных тем. Обрати внимание на различия в оформлении поста-цитаты.



Рис. 5.3. После установки темы на хостинг ее нужно активировать

ВДОХНОВЕНИЕ

Цитата

 ИЗМЕНИТЬ

- Ким | Опубликовано 28.10.2018

”

Секрет креативности — умение скрывать ее источники

— Альберт Эйнштейн

ВДОХНОВЕНИЕ

 ЦИТАТА  28.10.2018  КИМ  ОСТАВИТЬ КОММЕНТАРИЙ 
ИЗМЕНИТЬ

«Секрет креативности — умение скрывать ее источники».
Альберт Эйнштейн

Рис. 5.4. Два одинаковых поста при включенной теме Twenty Fourteen (вверху) и Customizr (внизу)

Темы можно искать не только в каталоге WordPress.org. Есть и другие сайты с темами, как бесплатными, так и платными. Чтобы установить тему, скачанную со стороннего сайта, проще всего загрузить ZIP-файл с темой в WordPress и активировать, как было показано выше. Старайся использовать только темы от проверенных разработчиков. К ним, например, относятся ThemeForest (<http://themeforest.net/>), WooThemes (<http://woothemes.com/>) и StudioPress (<http://studiopress.com/>).

Всего за несколько кликов можно активировать новую тему, поменяв дизайн всего сайта. И поскольку контент в WordPress хранится отдельно от тем, после этого тебе не придется пересоздавать страницы или заново загружать фотографии.

Индивидуальная настройка темы

Вряд ли ты найдешь тему, которая сразу же после установки сделает сайт таким, как тебе хочется. К счастью, тему можно подогнать под себя, меняя ее настройки. Для начала открой панель «Внешний вид» и перейди в раздел «Заголовок» (см. рис. 5.5). Здесь можно поменять шрифт и цвета шапки и даже добавить в шапку изображение.

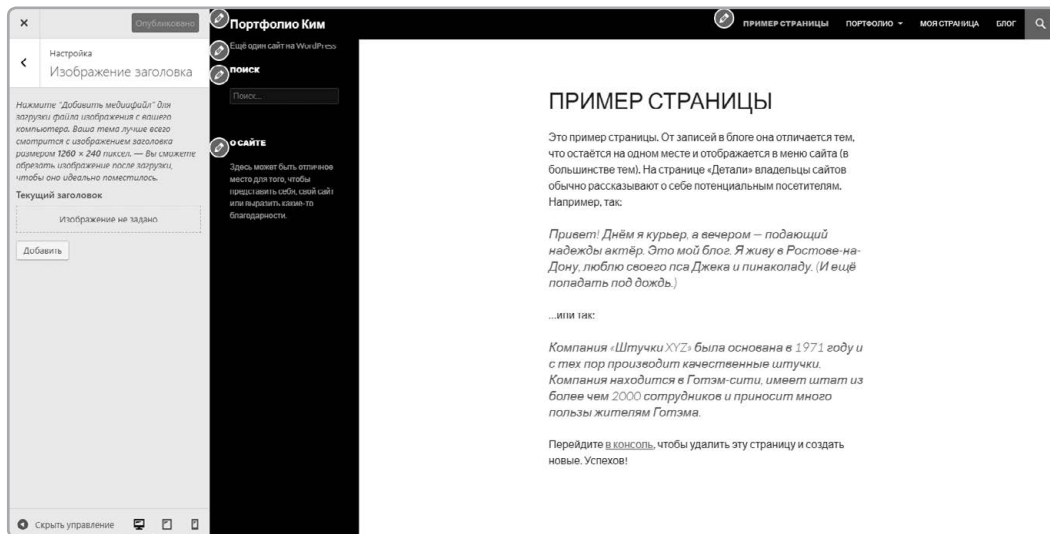


Рис. 5.5. Настройка шапки для темы *Twenty Fourteen*. Раздел «Заголовок» на панели «Внешний вид» позволяет загрузить в шапку изображение или поменять шрифт и цвет надписи

Можно нарисовать логотип и загрузить его в шапку, чтобы придать своему сайту уникальный вид.

Темы нередко добавляют свои опции в раздел «Настроить» на панели «Внешний вид». На рис. 5.6 показан раздел «Настроить» для темы *Twenty Fourteen*. Здесь можно выбрать цвета и фоновое изображение для сайта. Более сложные (и зачастую платные) темы могут добавить в консоль раздел «Настройки темы» (см. рис. 5.7) с широким диапазоном опций. Так, во многих современных темах не надо править код, а можно просто выбрать нужный шрифт в меню.

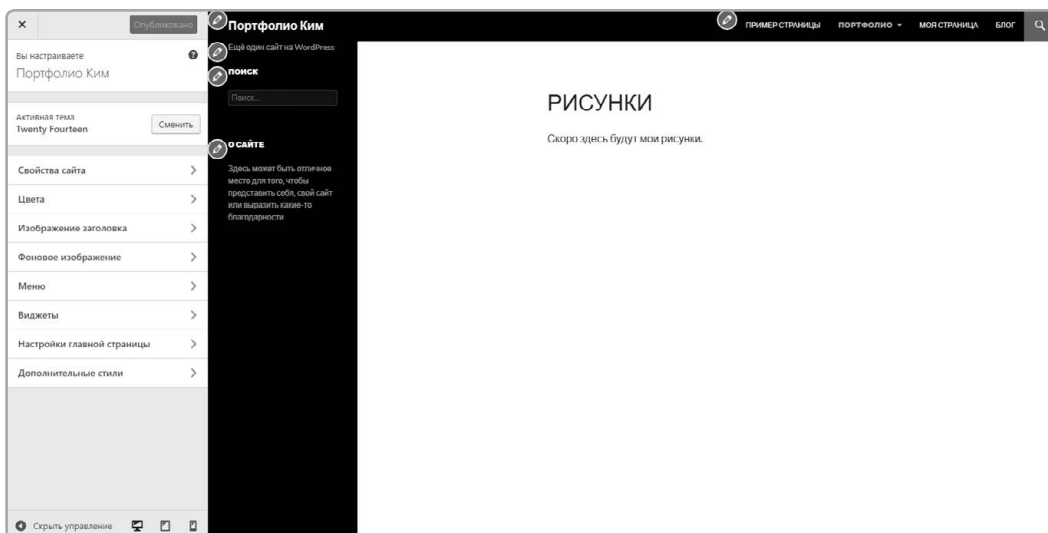


Рис. 5.6. Раздел «Настроить» в панели «Внешний вид» для бесплатной темы *Twenty Fourteen*, идущей в комплекте с *WordPress*. Здесь можно установить цвета или задать фоновое изображение. У разных тем разделы «Настроить» могут отличаться друг от друга. При активации новой темы загляни в этот раздел и изучи доступные опции

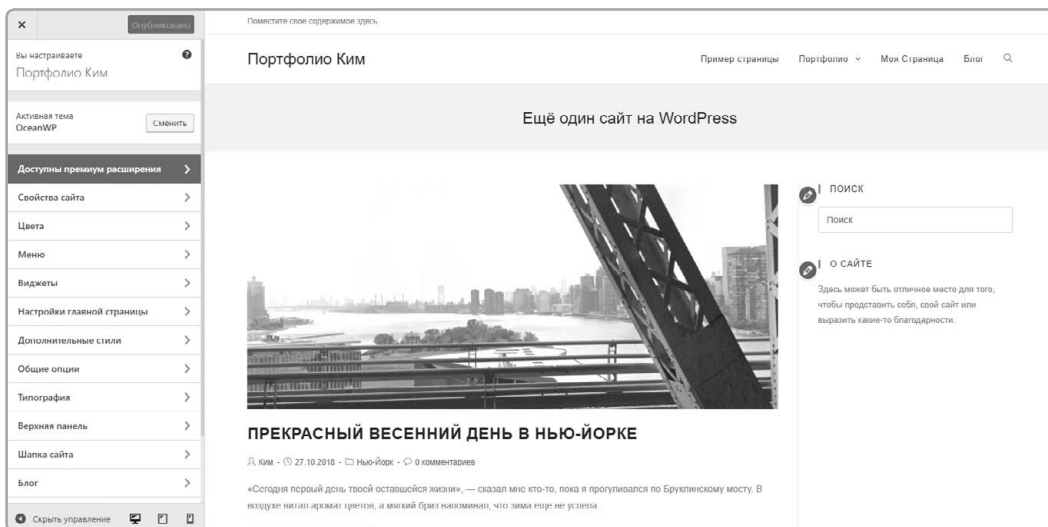


Рис. 5.7. Бесплатная тема *OceanWP*. Оформлением она похожа на тему по умолчанию (*Twenty Fourteen*), но предлагает больше опций настройки

В комплекте с новыми темами идут шаблоны оформления, позволяющие менять дизайн отдельных страниц (см. рис. 5.8). В теме Twenty Fourteen таких шаблонов три. Например, если выбрать шаблон «Страница на всю ширину», у нее не будет боковой панели справа, как на рис. 5.9. В Twenty Fourteen эти шаблоны нельзя поменять без правки кода, но некоторые темы позволяют настраивать их через разделы «Настройки темы» и «Настроить». Чтобы назначить странице шаблон, нужно выбрать его в панели визуального редактора «Атрибуты страницы» — там же, где мы задавали дочерние страницы в главе 4.

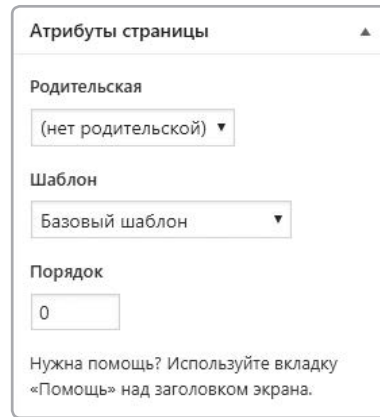


Рис. 5.8. Можно изменить оформление страницы, выбрав для нее другой шаблон

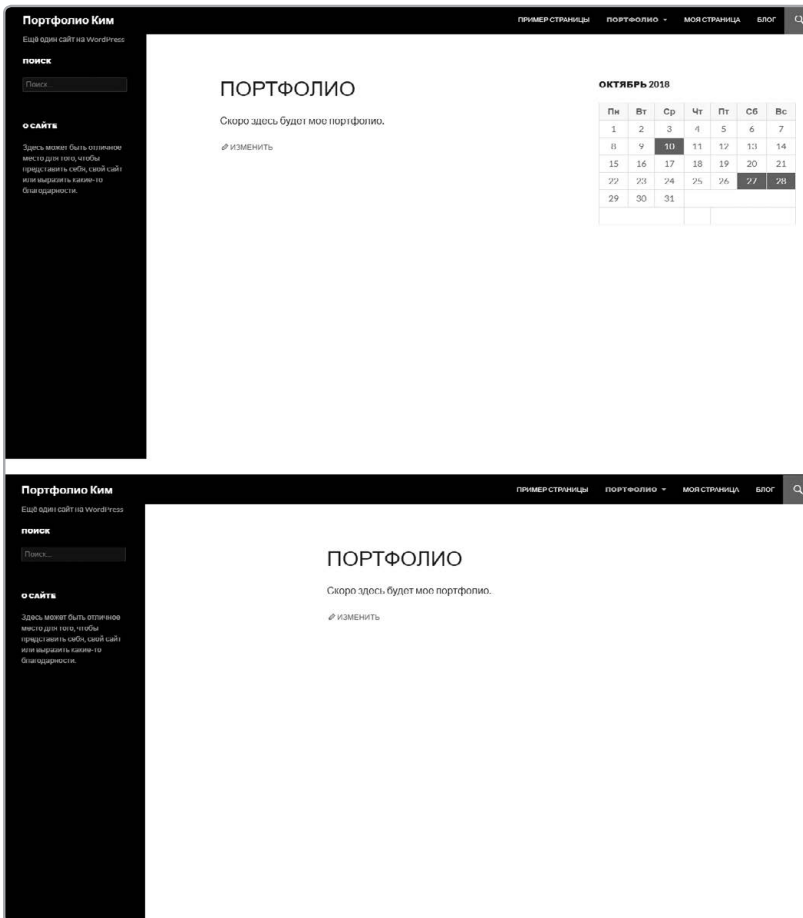


Рис. 5.9. Базовый шаблон (вверху) и шаблон «Страница на всю ширину» (внизу) для темы Twenty Fourteen

Для блога или простого сайта может хватить и бесплатной темы. Но если тебе нужны портфолио, слайд-шоу или встроенный интернет-магазин, лучше найти премиальную тему, которая за единоразовую умеренную плату предоставит тебе расширенные функции.

Настройка меню навигации

Большинство тем также позволяет настраивать меню навигации через раздел «Меню» в панели «Внешний вид». Здесь можно указать, какие страницы, посты или иные материалы должны отображаться в навигационном меню твоего сайта. Посмотрим, как это работает.

ПРИМЕЧАНИЕ

Если поля с галочками в левой панели неактивны, как на рис. 5.10, сперва нужно создать меню. Для этого введи слово «Меню» в поле «Название меню» и кликни «Сохранить меню».

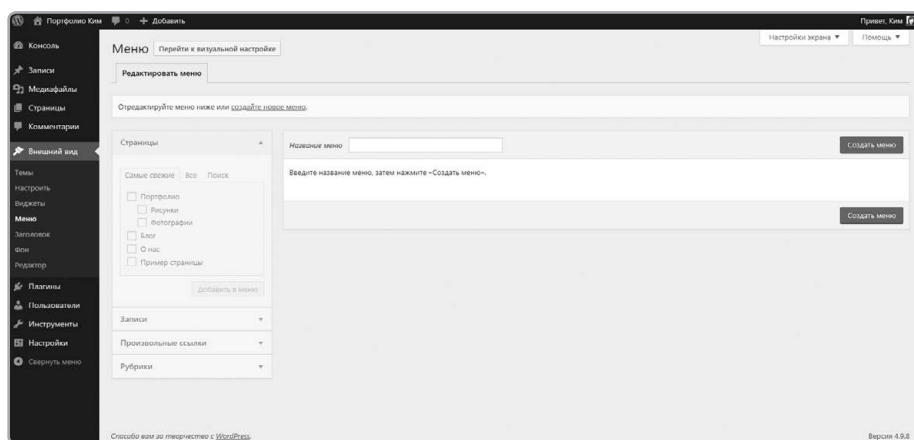


Рис. 5.10. Перед тем как добавлять в меню элементы, его нужно создать

Индивидуальная настройка меню позволяет выбрать, какие ссылки попадут в навигационную панель сайта. Мы уже видели, что после создания страницы она автоматически отображается в меню. Но возможности этого раздела позволят нам взять управление в свои руки.

Чтобы добавить страницы в навигационное меню, поставь галочки слева от их названий и нажми на кнопку «Добавить в меню» (см. рис. 5.11).

Возможно, тебе захочется упростить для посетителей переход к другим областям сайта, например к одной из рубрик в блоге. Добавлять в навигационное меню рубрики не сложнее, чем обычные страницы.

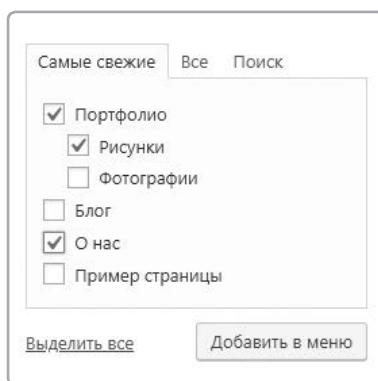


Рис. 5.11. Чтобы добавить страницы в меню, поставь галочки рядом с их названиями и кликни «Добавить в меню»

На рис. 5.12 мы видим в меню рубрику «Обзоры книг». Теперь посетитель может кликнуть по этой ссылке, и WordPress покажет все посты, которые относятся к данной рубрике.

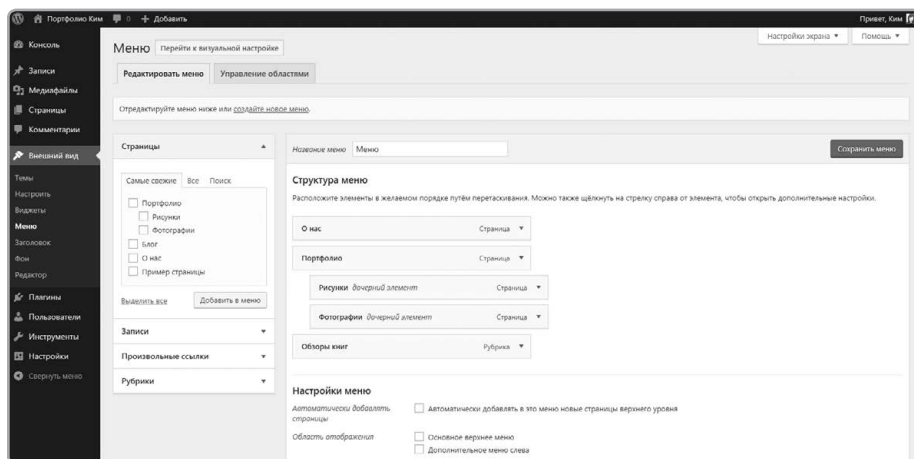


Рис. 5.12. При ручной настройке меню иерархические отношения между страницами не учитываются, но можно создавать подпункты, перетаскивая одни элементы меню под другие

Можно добавлять в меню ссылки на сторонние сайты. Например, если ты что-то продаешь на Etsy или в Instagram и хочешь, чтобы посетители могли быстро перейти в магазин, просто добавь ссылку на него в меню и дай ей подходящее название (см. рис. 5.13).

Некоторые темы поддерживают несколько меню: например, одно вверху страницы и второе — в нижней части. Нужно положение можно выбрать с помощью опции «Область темы». Области темы — это особые места на странице, в которых тема позволяет создавать меню. Twenty Fourteen предлагает две такие позиции: в шапке сайта («Основное верхнее меню») и в левой боковой панели («Дополнительное меню слева»).

Чтобы проверить, как это работает, мы выберем вариант **«Основное верхнее меню»**. Можно поставить галочку напротив опции «Автоматически добавлять в это меню новые страницы верхнего уровня» (см. рис. 5.14). В этом случае все новые страницы будут сами собой попадать в меню. Независимо от того, выбрал ты эту опцию или нет, нажми на кнопку «Сохранить меню», чтобы применить новые настройки.

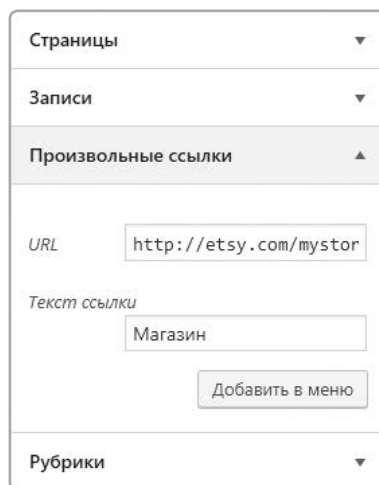


Рис. 5.13. Добавляем в меню ссылку на внешний сайт



Рис. 5.14. Выбирая области темы для меню, ты сообщаешь WordPress, где хочешь видеть навигационную панель. Twenty Fourteen предлагает выбор из двух позиций: основное меню в шапке или вспомогательное в левой боковой панели

Если ты еще этого не сделал, промотай страницу до панели «Страницы», отметь галочками пункты «Портфолио», «Фотографии» и «Рисунки» и кликни **«Добавить в меню»**, как показано на рис. 5.15.

После того как ты кликнешь **«Добавить в меню»**, страницы окажутся в правой части соответствующей панели. Перетаскивая элементы, можно менять их порядок. Сделай страницу «Обо мне» первой, дальше пусть идет «Портфолио», затем «Рисунки» и наконец «Фотографии», как на рис. 5.16.

Теперь кликни **«Сохранить меню»** и перейди во фронтенд. Обрати внимание, что первый пункт меню в консоли соответствует первому пункту меню фронтенда, а «Рисунки» и «Фотографии» являются подпунктами «Портфолио».

Получилось? Здорово, правда? Теперь вернись в консоль и попробуй добавить в меню рубрику «Обзоры книг» и ссылку на магазин. Если у тебя получится это сделать — отлично, но, скорее всего, разделы «Рубрики» и «Произвольные ссылки» будут недоступны. Дело в том, что сначала их надо включить во вкладке «Настройки экрана». Давай поговорим об этом подробнее.

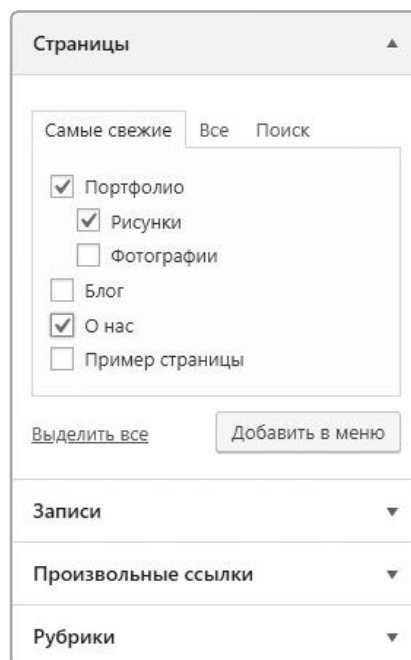


Рис. 5.15. При ручной настройке меню ты указываешь самостоятельно, какие элементы должны в нем отображаться, а не оставляешь это на откуп WordPress. Некоторые страницы лучше не включать в меню, чтобы сделать навигацию более гибкой и понятной, особенно для крупных сайтов

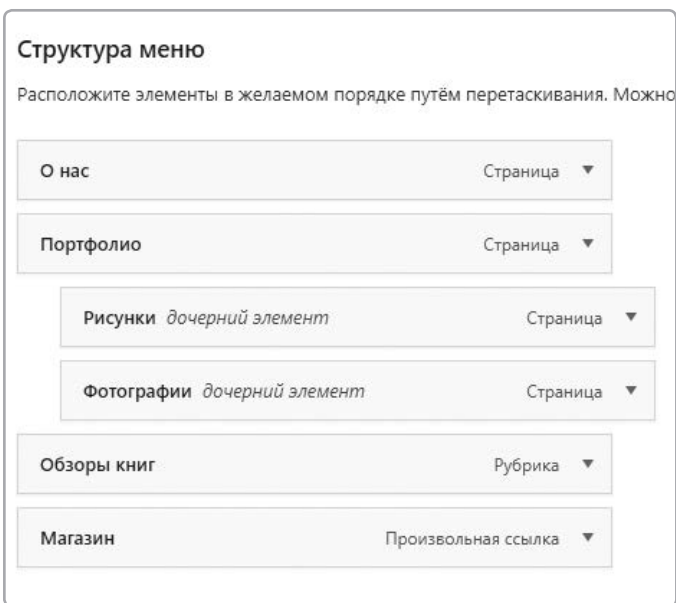


Рис. 5.16. Можно перетаскивать элементы меню, меняя их порядок

Вкладка «Настройки экрана»

Порой работа в консоли может вызвать затруднения. Тебе кажется, что можно выполнить то или иное действие, но ты не видишь соответствующей опции. В предыдущем разделе мы добавили в меню ссылку на Etsy, но как сделать, чтобы она открывалась в отдельном окне? Такая возможность должна быть, однако неясно, какая опция этим управляет.

Вернись в раздел **«Меню»** панели **«Внешний вид»**, найди справа вверху вкладку **«Настройки экрана»** и кликни по ней (см. рис. 5.17).

Эта вкладка есть на каждом экране консоли, и она таит немало полезных возможностей. Если кликнуть по ней во время редактирования навигационного меню, откроются опции, которых нет в основном разделе **«Меню»** панели **«Внешний вид»**.

В данном случае ты увидишь набор опций с галочками (см. рис. 5.18). Причем если мы откроем эту вкладку из редактора постов или раздела «Все страницы», то набор галочек будет другим. Когда тыставишь галочку напротив опции, в навигационном меню появляется соответствующая функция. Убедись, что у тебя включены следующие опции: «Страницы», «Произвольные ссылки», «Рубрики» и «Цель ссылки».

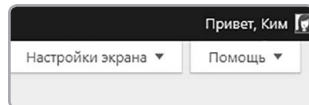


Рис. 5.17. Вкладка «Настройки экрана» в правом верхнем углу консоли

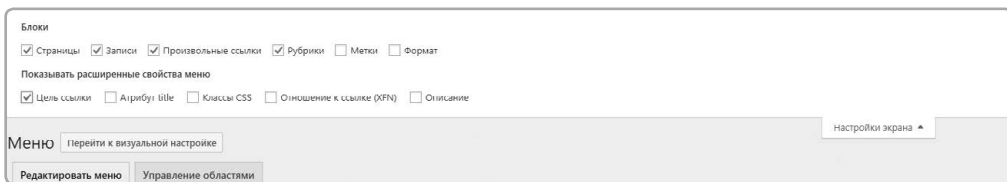


Рис. 5.18. Панель «Настройки экрана» позволяет включать и выключать функции текущего экрана

Если раньше у тебя не получалось добавить в меню ссылку или рубрику, сейчас эти опции должны появиться слева, как на рис. 5.13. Попробуй добавить в меню ссылку: в поле URL панели «Произвольные ссылки» введи <http://etsy.com/mystore>, а в поле «Навигационная метка» укажи «Магазин». Еще ниже должна быть галочка с подписью «Открывать в новой вкладке» (см. рис. 5.19). Когда ты в следующий раз задумаешься, как выполнить то или иное действие в консоли, вспомни о вкладке «Настройки экрана».

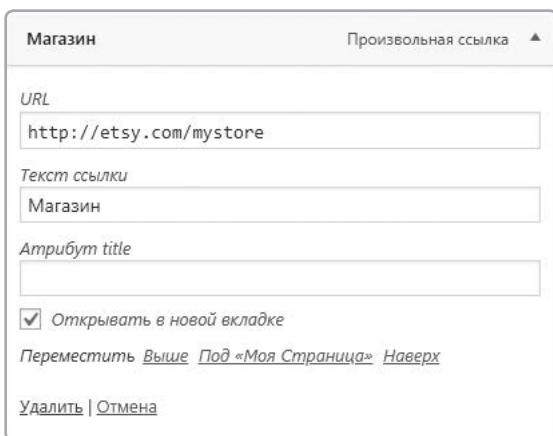


Рис. 5.19. Ссылка будет открываться в новом окне при включенной опции «Цель ссылки»

Настройки для всего сайта

Еще ниже в левой боковой панели находится вкладка «Настройки» (см. рис. 5.20). Она содержит опции, относящиеся ко всему сайту целиком. Их очень много, поэтому мы рассмотрим только самые популярные из них.

В разделе «Общие» на панели «Настройки» можно изменить название сайта и его краткое описание. Обычно оно отображается в шапке страницы под ее заголовком, хотя это зависит от темы. Если краткое описание тебе не нужно, оставь поле пустым (но имей в виду, что краткое описание может повысить поисковый рейтинг сайта). Изменив настройки, не забудь нажать на «Сохранить изменения» внизу страницы.

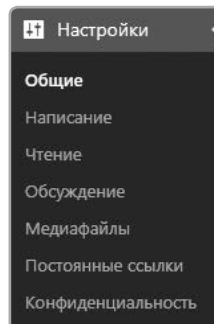


Рис. 5.20. Настройки WordPress

ПРИМЕЧАНИЕ

Многие не утруждаются изменить краткое описание, установленное по умолчанию: «Еще один сайт на WordPress» (или на английском — *Just another WordPress site*). Попробуй ввести в Google «*Just another WordPress site*», и ты увидишь миллионы сайтов, владельцы которых не позаботились о кратком описании. Ты же не хочешь стать одним из них?

В разделе «Чтение» можно изменить структуру сайта (рис. 5.21). Чтобы понять, о чем я говорю, перейди в раздел «Добавить новую» на панели «Страницы» и назови страницу «Добро пожаловать».

Введи в окне редактора короткую фразу, например «Добро пожаловать на мой сайт!», и кликни «Опубликовать». Затем создай еще одну страницу и назови ее «Блог». Заглянув в раздел «Чтение» на панели «Настройки», ты увидишь, что по умолчанию на главной странице отображаются последние посты.

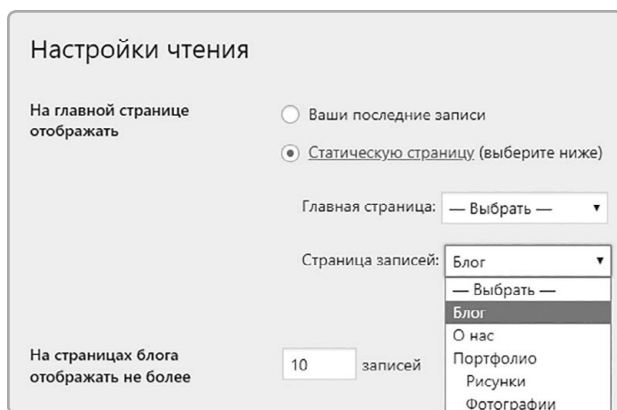


Рис. 5.21. В разделе «Чтение» на панели «Настройки» можно сделать так, чтобы посетители начинали знакомиться с сайтом не с блога, а с какой-то другой страницы

А если тебе хочется, чтобы, зайдя на сайт, посетитель видел страницу-заставку, а не наш свежесозданный блог? Тогда выбери пункт «Статическая страница» и найди в выпадающем меню «Домашняя страница» пункт «Добро пожаловать». Затем выбери в поле «Страница записей» значение «Блог».

После того как ты **сохранишь изменения**, сайт будет автоматически реорганизован. Вернись в раздел «Меню» панели «Внешний вид» и добавь в меню страницы «Добро пожаловать» и «Блог». Теперь перейди во фронтенд. Ты увидишь, что первой стала показываться пустая страница «Добро пожаловать», которую мы только что создали, а кликнув в меню на «Блог», ты попадешь на страницу с постами.

Во вкладке «Настройки» есть еще много всего полезного, однако я не могу подробно рассказать обо всем. Ты можешь менять размер картинок по умолчанию, управлять комментариями, задавать максимальное количество постов на странице блога и так далее. Если тебя мучает любопытство, поэкспериментируй с настройками и посмотри, как будет меняться сайт. Не бойся запутаться: в интернете есть подробная документация по всем опциям (см. раздел «Дополнительная информация» на стр. 227).

Вот краткий обзор некоторых разделов панели «Настройки»:

- **«Написание»** — настройки редактирования постов, включая исправление ошибок (только для WordPress.com и плагина Jetpack), а также рубрики по умолчанию.
- **«Обсуждение»** — здесь можно включать и выключать комментирование, а также настроить почтовые уведомления о новых комментариях.
- **«Медиафайлы»** — тут настраивают размеры картинок («Миниатюра», «Средний», «Крупный»).
- **«Ссылки»** — настройка формата URL для страниц и постов (это пригодится для поисковой оптимизации и улучшения читаемости).

Расширенные настройки

Когда в качестве хостинга используется WordPress.com, возможности по его настройке ограничены. Если тебе недостаточно тех настроек, с которыми мы познакомились ранее, установи платное расширение Custom Design (за его использование необходимо раз в год вносить абонентскую плату). Среди прочего оно позволит изменять код твоего сайта и встраивать в тему собственный CSS. Иначе придется довольствоваться базовыми настройками выбранной темы в консоли.

Если же ты установишь WordPress на собственном хостинге, сайт можно будет настраивать без ограничений. Обрати внимание на дополнительную опцию «Редактор» панели «Внешний вид» — она дает полный доступ к коду темы. Если ты захочешь создать свою тему или изменить что-либо помимо CSS, установи WordPress на свой хостинг, и у тебя будут почти неограниченные возможности. Можно даже воспользоваться FTP для загрузки собственного кода — еще одно подтверждение тому, как полезно знать HTML и CSS!

Например, попробуй изменить шрифт абзацев в твоей теме. Для этого добавь в конец таблицы стилей следующий код:

```
p { font-family: times, serif; }
```

В результате все абзацы станут отображаться шрифтами Times (а не Helvetica или Arial). В целом требуется лишь понять, какой элемент нужно изменить, и добавить для него соответствующие CSS-свойства. Чтобы потренироваться, попробуй поменять цвет ссылок.

Хотя код, исправленный вручную при помощи редактора, и работает, этот подход считается нежелательным. Если ты хочешь изменить имеющуюся тему, лучше использовать *наследование тем*. Этот метод позволяет модифицировать тему, не трогая ее изначальный код. Возможно, тебе также стоит установить WordPress на своем компьютере, чтобы безопасно тестировать сайт. Эти вопросы выходят за рамки нашей книги, но ты можешь обратиться к документации: https://codex.wordpress.org/Child_Themes/ и https://codex.wordpress.org/Installing_WordPress_Locally_on_Your_Mac_With_MAMP/.

Плагины

Термин *плагин* обозначает программу, которую устанавливают на сайт, чтобы расширить его функционал. Например, для онлайн-магазина пригодится плагин для работы с банковскими картами. Есть плагины, повышающие быстродействие сайта, — так называемые *плагины кэширования*. К слову, именно широкое разнообразие доступных плагинов — одна из причин популярности WordPress.

ПРИМЕЧАНИЕ

Плагины будут доступны только при использовании собственного хостинга (а не портала WordPress.com). То есть на сайт в домене WordPress.com установить какие-либо плагины не выйдет. Впрочем, там уже установлены многие дополнительные функции, что отчасти компенсирует нехватку плагинов.

Плагины позволяют делать немало вещей, которые невозможны на обычном сайте на WordPress. Но их следует регулярно обновлять: устаревшие плагины могут привести к уязвимости в системе безопасности, и тогда сайт станет легкой добычей для взломщиков. Впрочем, если ты будешь пользоваться только стандартными плагинами, регулярно их обновляя, WordPress останется вполне надежным инструментом управления сайтом.

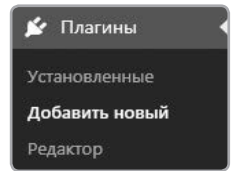


Рис. 5.22. Установка нового плагина на сайт

Прокрутив вниз панель в левой части консоли, ты обнаружишь вкладку «Плагины», как на рис. 5.22. Если этой вкладки нет (а сайт работает на собственном хостинге), убедись, что ты вошел в консоль с логином и паролем администратора.

Давай установим классный плагин под названием Meta Slider. Он позволяет создавать галереи изображений с поддержкой слайд-шоу. Клики по вкладке «Добавить плагин» на панели «Плагины» в консоли и введи в строке поиска «slider» (см. рис. 5.23). Должен открыться большой список плагинов с функцией слайд-шоу. Все плагины в нем бесплатны, поэтому если хочешь поэкспериментировать, то можешь установить любой плагин, а затем удалить его, если он придется тебе не по вкусу.

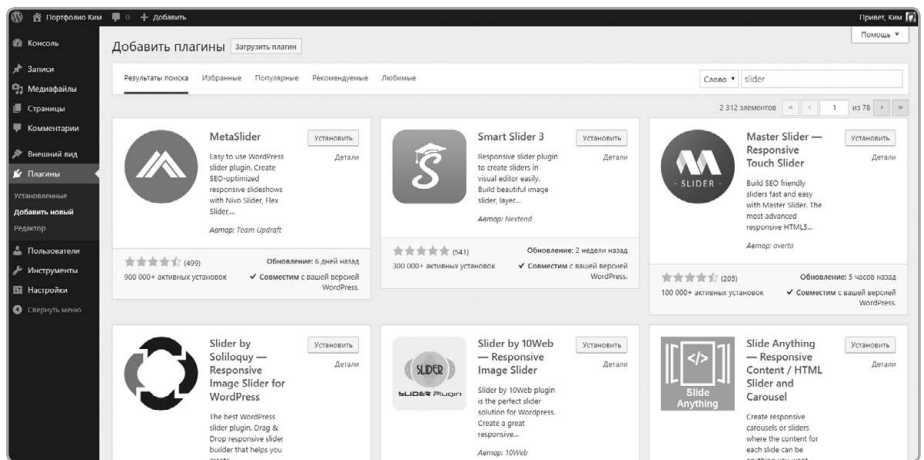


Рис. 5.23. Поиск в каталоге плагинов по запросу «slider»

Чтобы понять, стоит ли связываться с очередным плагином, имеет смысл кликнуть по ссылке «Детали» чуть ниже его названия. Откроется экран, где, помимо общих сведений, можно узнать, как оценили этот плагин другие люди (см. рис. 5.24).

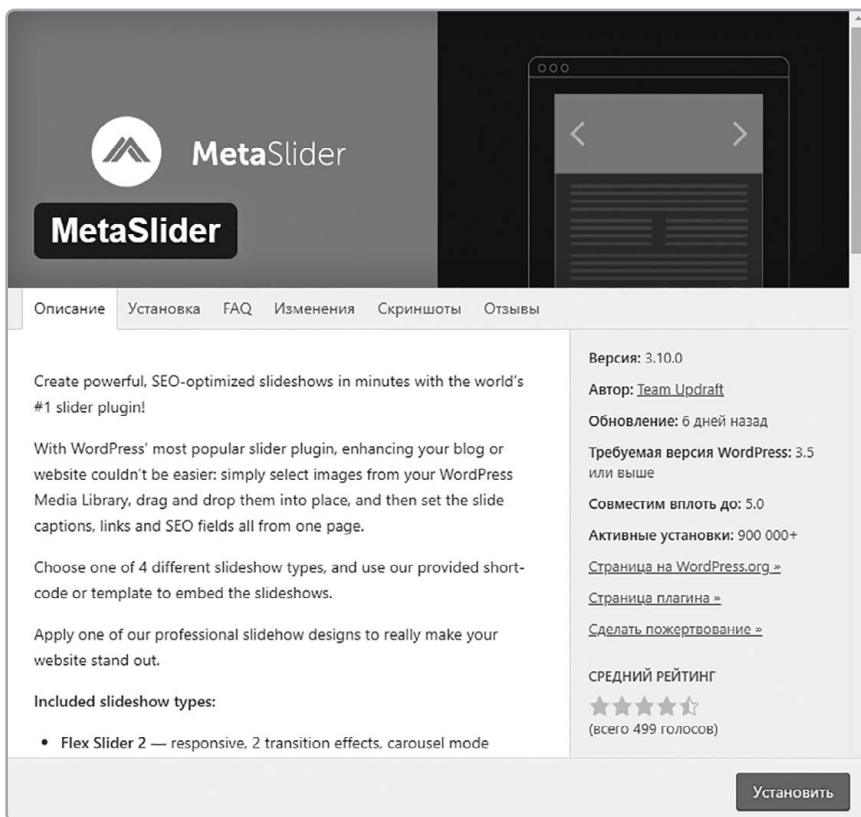


Рис. 5.24. Кликни по ссылке «Детали», чтобы открыть окно с подробной информацией о плагине. Там ты сможешь узнать его рейтинг, посмотреть на скриншоты и прочитать ответы на часто задаваемые вопросы

Просмотри список плагинов. Обнаружив Meta Slider, кликни **«Установить»**. Как и при установке темы WordPress, начнется скачивание файла на твой хостинг, после чего плагин станет доступным для использования. Нажми **«Активировать плагин»**. В отличие от тем, на сайте может быть *много* активных плагинов, выполняющих различные функции.

После того как ты активируешь Meta Slider, в панели «Настройки» в консоли появится вкладка с его настройками. Имей в виду, что не все плагины создают вкладки в этом разделе. Чтобы узнать, как пользоваться тем или иным плагином, обратись к его документации.

Кликни по вкладке Meta Slider, чтобы перейти к экрану управления плагином. Чтобы создать свое первое слайд-шоу, нажми на кнопку «+» вверху экрана. Установи курсор в поле **«Новое слайд-шоу»** и введи его название.

Предположим, нам хочется добавить слайд-шоу на страницу «Добро пожаловать». Назовем его «Приветственные слайды». Теперь можно добавлять в слайд-шоу картинки. Нажми на кнопку «Новый слайд» вверху окна. Можно выбрать картинку из медиатеки, а можно загрузить новые. Добавь два или три слайда, выделив подходящие картинки и кликнув «Добавить в слайдер».

Мы только что создали слайд-шоу! Чтобы увидеть его в действии, нажми на кнопки «Сохранить» и «Просмотр». Настройки, расположенные ниже, позволяют задать скорость анимации и эффект смены слайдов. Тут есть с чем поиграться! Но как поместить слайд-шоу на страницу «Добро пожаловать»? Для этого в Meta Slider используется так называемый *шорткод*. Промотав правую часть страницы в самый низ, ты увидишь панель «Использование», а на ней вкладку «Шорткод» (см. рис. 5.25).

```
<?php echo do_shortcode('[metaslider id="168"]'); ?>
```

Рис. 5.25. Шорткод для Meta Slider

Шорткод — это строка кода, обозначающая плагин, виджет (см. «Виджеты» на стр. 222) или элемент темы. Ты можешь скопировать этот набор символов и вставить его на любую страницу сайта, тем самым добавив нужный элемент. В некоторых темах есть шорткоды для быстрого создания кнопок и других визуальных элементов без помощи HTML или CSS.

Скопируй шорткод, указанный на соответствующей вкладке, затем открой список страниц сайта, выбрав вкладку **«Все страницы»** в разделе **«Страницы»** на панели слева, и перейди к редактированию страницы «Добро пожаловать». Впрочем, если хочешь, можешь добавить слайд-шоу на любую другую страницу. Кстати, Meta Slider добавляет в панель визуального редактора кнопку «Добавить слайдер», которая автоматически вставляет на страницу шорткод. Поскольку это не HTML, для его вставки незачем открывать вкладку «Текст» — все можно сделать прямо в визуальном редакторе. Вставив шорткод, добавь ниже еще одну строчку: **«Добро пожаловать на мой сайт!»**

Нажми на кнопку **«Обновить»**, а затем **«Просмотреть изменения»**. Вместо шорткода на странице должно появиться слайд-шоу, как на рис. 5.26.

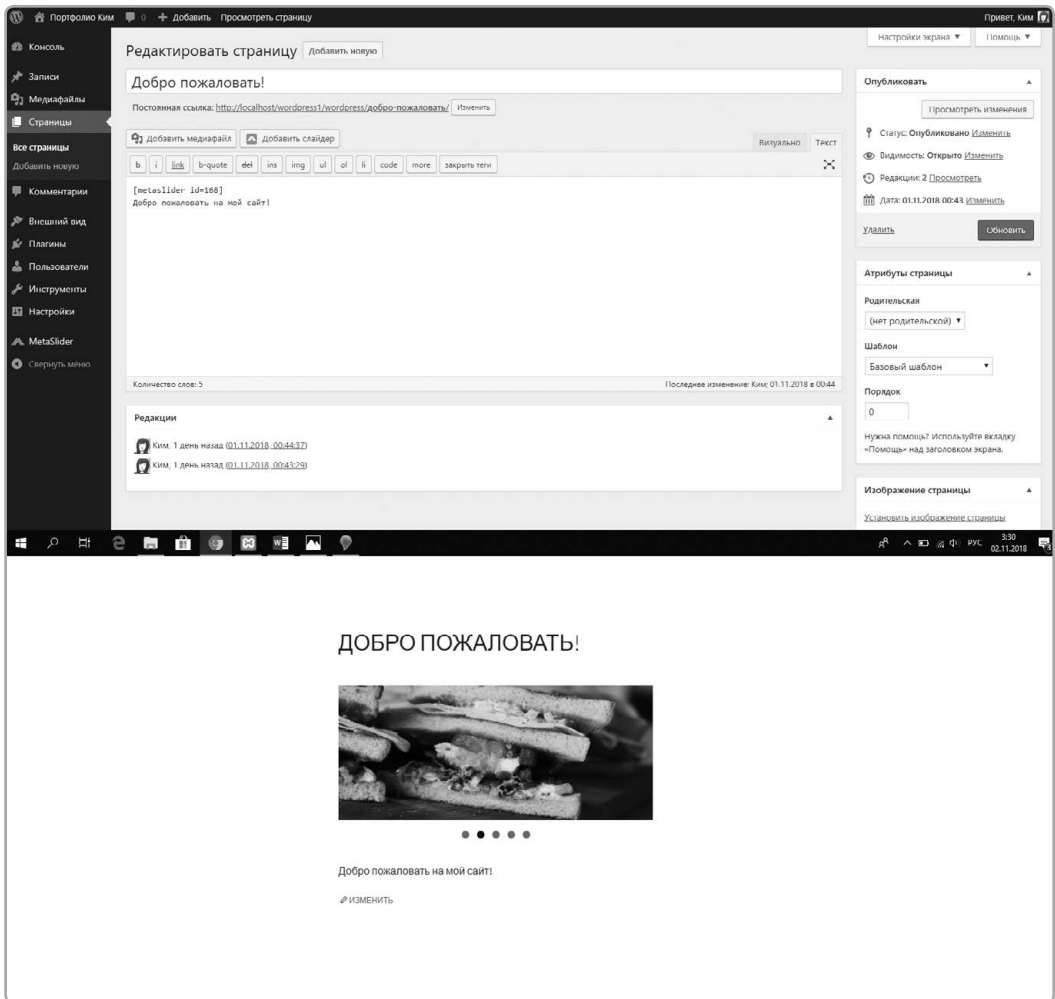


Рис. 5.26. Шорткод плагина Meta Slider в консоли и готовое слайд-шоу во фронтенде сайта

Виджеты

Виджеты — это разновидность плагинов, которые можно устанавливать на сайт, чтобы добавить ему новые возможности. Но функциональность виджетов, как правило, ограничена, и добавлять их можно только в особые области страниц. Предположим, ты хочешь показывать в боковой панели блога посты из Twitter. Установи виджет — эта процедура ничем не отличается от установки плагина: кликни по вкладке «Добавить плагин» на панели «Плагины», а затем введи в поисковую строку «Twitter widget». Если у тебя нет Twitter-аккаунта, попробуй какой-нибудь другой виджет, например отображающий картинку.

Как и любые плагины, виджет нужно сперва установить, а потом активировать. Выполнив эти шаги, открой в консоли раздел «Виджеты» (см. рис. 5.27). Справа от него в панели «Внешний вид» показан список областей страницы, куда можно добавлять виджеты в текущей теме. В большинстве тем есть хотя бы одна колонка, пригодная для размещения виджетов. В некоторых есть и другие подходящие области, например дополнительная боковая колонка или подвал.

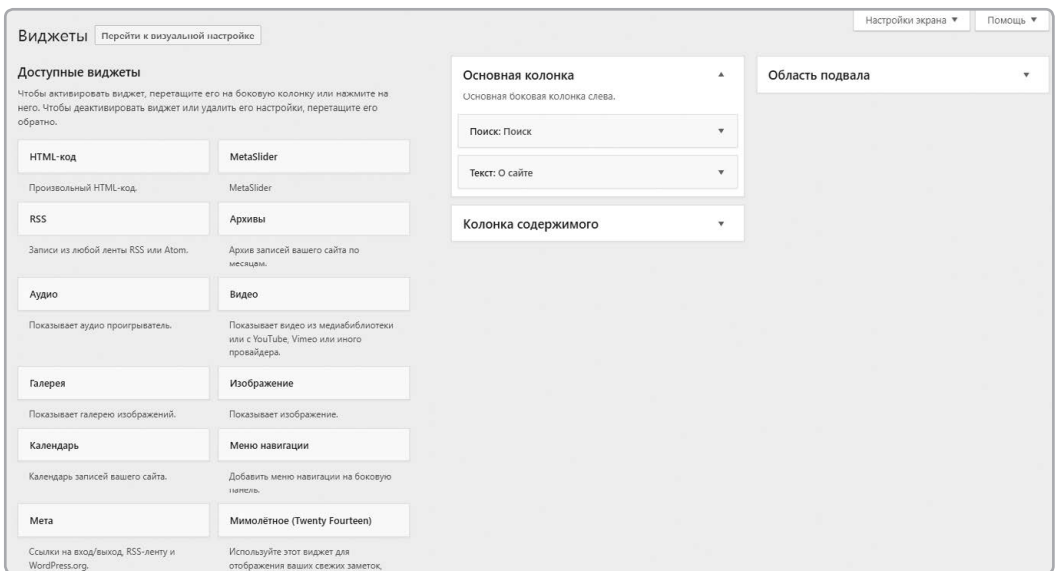


Рис. 5.27. Виджеты позволяют добавлять функционал в специальные области сайта, например в боковую колонку. Здесь показан список областей, доступных для размещения виджетов, в теме *Twenty Fourteen*

Слева перечислены доступные виджеты, которые можно добавить на сайт. Справа показаны области, где эти виджеты могут находиться. Чтобы установить виджет «Изображение», прокручивай список до тех пор, пока не увидишь его. Затем перетащи этот виджет внутрь области «Основная колонка», как на рис. 5.28. Если страница бэкенда достаточно велика, перетащить виджет будет непросто. Вместо этого можно кликнуть по виджету и выбрать для него область (см. рис. 5.29).

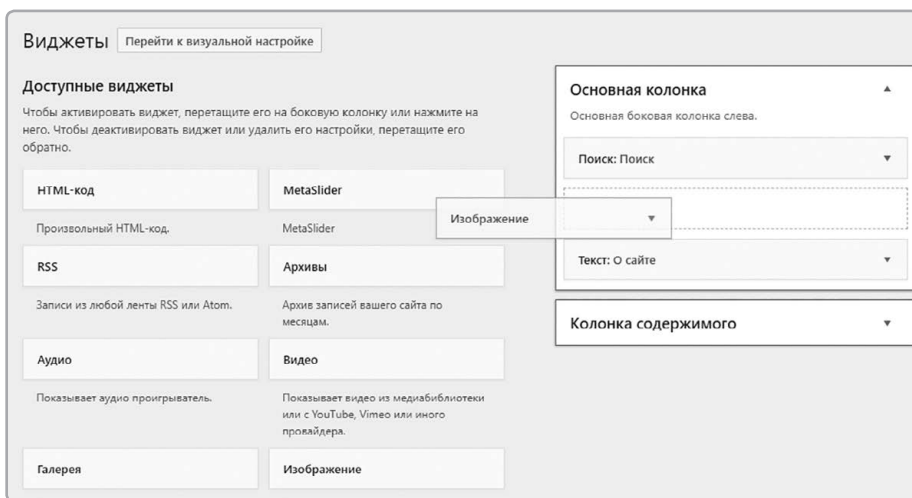


Рис. 5.28. Перетаскивание виджета в поле «Основная колонка»

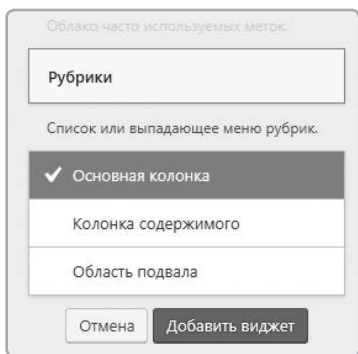


Рис. 5.29. Виджет также можно установить, кликнув по нему и выбрав область размещения

Дальше нужно задать настройки, необходимые для работы виджета (см. рис. 5.30). Кликни по треугольничку в правом верхнем углу виджета, выбери нужные опции, а затем нажми «Сохранить».

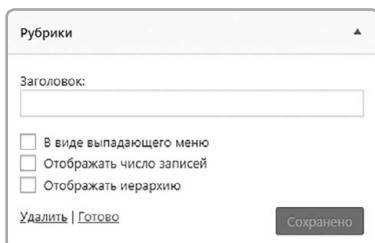


Рис. 5.30. Настройка виджета «Рубрики»

Перейди во фронтенд своего сайта. На боковой панели должен появиться список твитов, как на рис. 5.31.

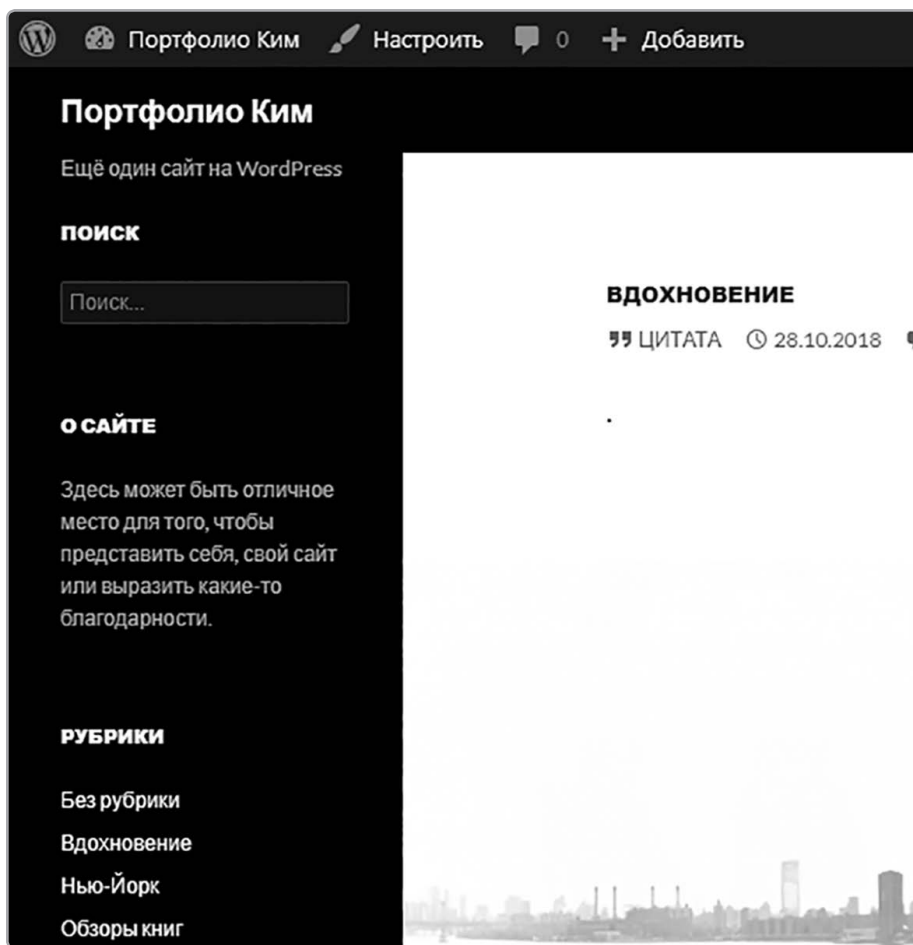


Рис. 5.31. Работающий виджет «Рубрики»

Предположим, теперь ты хочешь удалить виджет. Вернись в раздел **«Виджеты»** на панели «Внешний вид» и перетащи виджет из области, соответствующей боковой колонке фронтенда, наружу и влево.

Виджет должен исчезнуть с панели. Если хочешь его вернуть, снова найди его в списке и перетащи в область боковой колонки. Настраивая сайт, ты можешь попробовать разные виджеты, чтобы понять, какие возможности тебе доступны.

Изучение виджетов

Для знакомства с виджетами и дополнительными плагинами отлично подходит плагин Jetpack. Это сборник полезных виджетов и плагинов, по умолчанию доступных только пользователям WordPress.com. Чтобы поэкспериментировать с ними, просто подключи бесплатный аккаунт на WordPress.com.

Обновления

Все дополнения, которые ты установишь на свой сайт на WordPress, нужно обновлять. Чтобы не возникало проблем с безопасностью, нужно ставить последние обновления тем, плагинов и самого WordPress. Устаревшие версии могут сделать сайт уязвимым для хакеров и спамеров.

Начиная с версии 3.7, WordPress устанавливает небольшие обновления автоматически. Однако крупные обновления WordPress, а также обновления тем и плагинов ставятся вручную. Если около раздела «Обновления» в консоли стоит число, это значит, что в сети появились обновления, которые нужно установить. Для этого открой раздел **«Обновления»** (см. рис. 5.32).

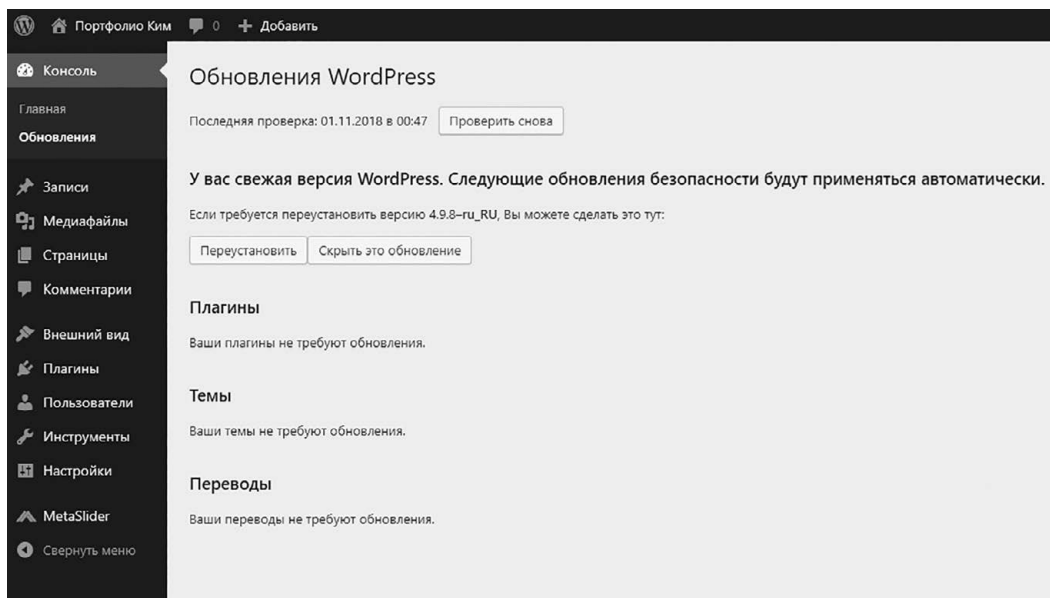


Рис. 5.32. В разделе «Обновления» ты найдешь все доступные последние версии как самого WordPress, так и установленных тем и плагинов

WordPress покажет все доступные обновления. Отметь их галочками и нажми на кнопку «Обновить». Стоит взять привычку делать это регулярно.

Если ты используешь качественную тему и стандартные плагины, проблем возникнуть не должно. Однако если на сайте стоят нестандартные плагины или темы, стоит убедиться в их совместимости с обновлениями, иначе они могут перестать работать. Не допускай ситуации, когда плановое обновление приводит к поломке сайта.

Использование старых версий WordPress ставит под угрозу безопасность сайта. Если ты не уверен в совместимости какой-либо темы или плагина с последней версией WordPress, уточни это у разработчиков. Разработчики с хорошей репутацией обычно следят за обновлениями WordPress, проверяя свои продукты (темы или плагины) на совместимость. Впрочем, сайт очень редко полностью перестает работать из-за одного устаревшего плагина или темы.

Перенос сайта на новый хостинг

Одно из достоинств WordPress в том, что ты не привязан к одному конкретному домену или хостинг-провайдеру. Неважно, находится ли твой сайт на портале WordPress.com или на другом хостинге. Инструменты из соответствующего раздела консоли позволяют экспортировать контент (посты, страницы, картинки), а затем импортировать его на другой сайт (см. рис. 5.33).

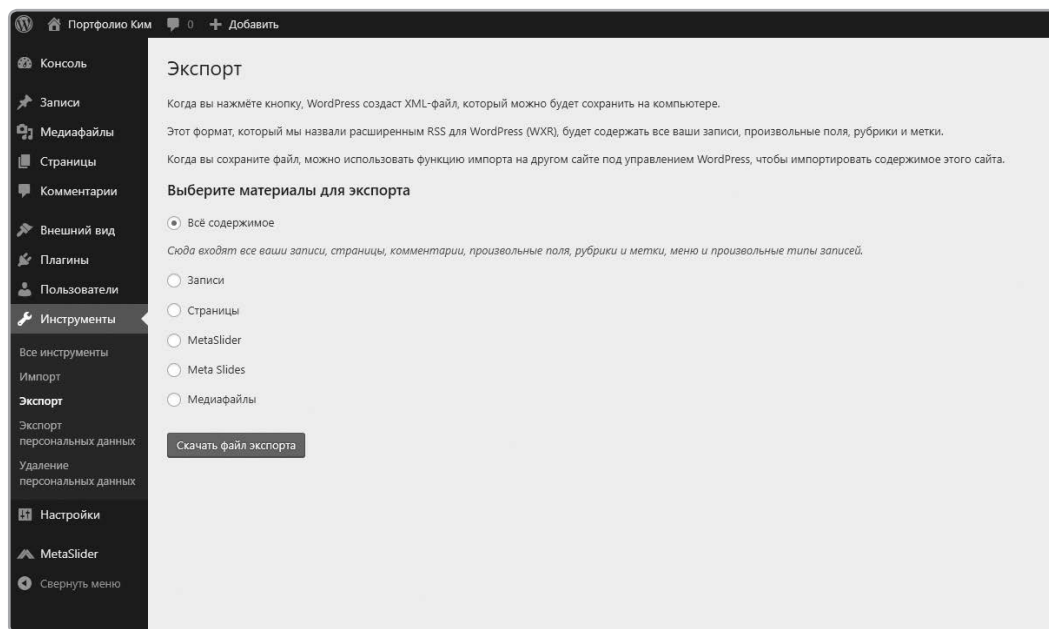


Рис. 5.33. В разделе «Инструменты» есть утилита экспорта, облегчающая перенос сайта и создание резервных копий

Кроме того, ты можешь импортировать наполнение из ряда других популярных CMS (см. рис. 5.34), что упрощает переход на WordPress. Воспользовавшись функцией экспорта, ты получишь один большой файл со всем контентом сайта. Затем, настроив новый сайт, ты просто запускаешь импорт, и все материалы снова оказываются на месте. Помни, что темы так переносить нельзя: на новом сайте придется устанавливать нужную тему вручную. Зато посты, на которые было потрачено так много времени и сил, импортируются все сразу, автоматически.

Импорт

Если у вас есть записи и комментарии в другой системе, WordPress может импортировать их на этот сайт. Для начала выберите систему из списка:

Blogger Установить Детали	Импорт записей, комментариев и пользователей из Blogger.
LiveJournal Установить Детали	Импорт записей из LiveJournal с помощью их API.
Movable Type и TypePad Установить Детали	Импорт записей и комментариев из Movable Type или TypePad.
RSS Установить Детали	Импорт записей из RSS-ленты.
Tumblr Установить Детали	Импорт записей и медиафайлов из Tumblr с помощью их API.
WordPress Установить Детали	Импорт записей, страниц, комментариев, произвольных полей, рубрик и меток из файла экспорта WordPress.
Конвертер рубрик и меток Установить Детали	Выборочное преобразование рубрик в метки или меток в рубрики.
Ссылки Установить Детали	Импорт ссылок в формате OPML.

Если нужного вам скрипта нет в списке, [поищите его в каталоге плагинов](#).

Рис. 5.34. WordPress позволяет импортировать материалы из ряда популярных блогинг-платформ

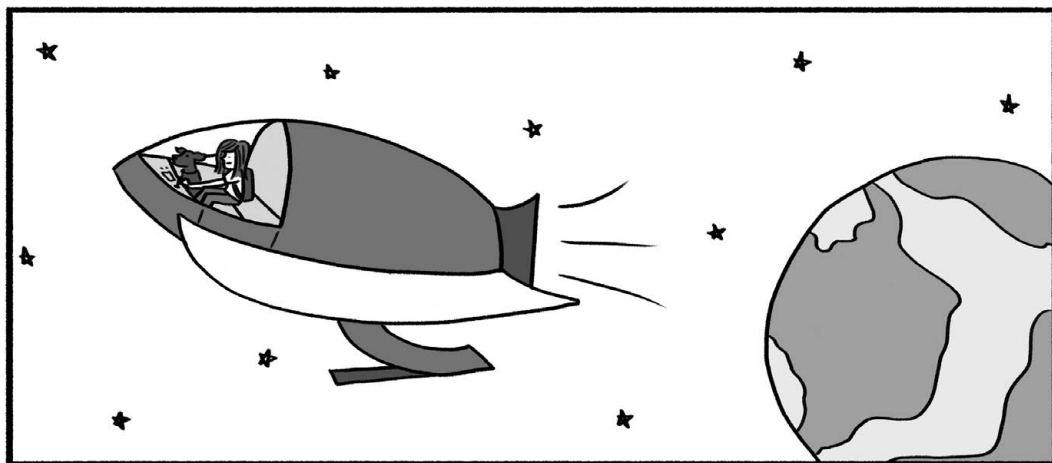
Дополнительная информация

Если ты хочешь больше узнать об использовании WordPress и настройке сайтов, посмотри документацию по адресу <http://codex.wordpress.org/>. Поиск сведений о WordPress в сети чаще всего приводит именно сюда. Помимо стандартной документации, там есть форум, где можно изучить вопросы и ответы других людей. Существует и много других хороших сайтов о WordPress, например <http://digwp.com/>, <http://torquemag.io/> и <http://wp.smashingmagazine.com/>. Самое приятное в изучении WordPress — это обилие открытой информации и множество людей, готовых прийти тебе на помощь.

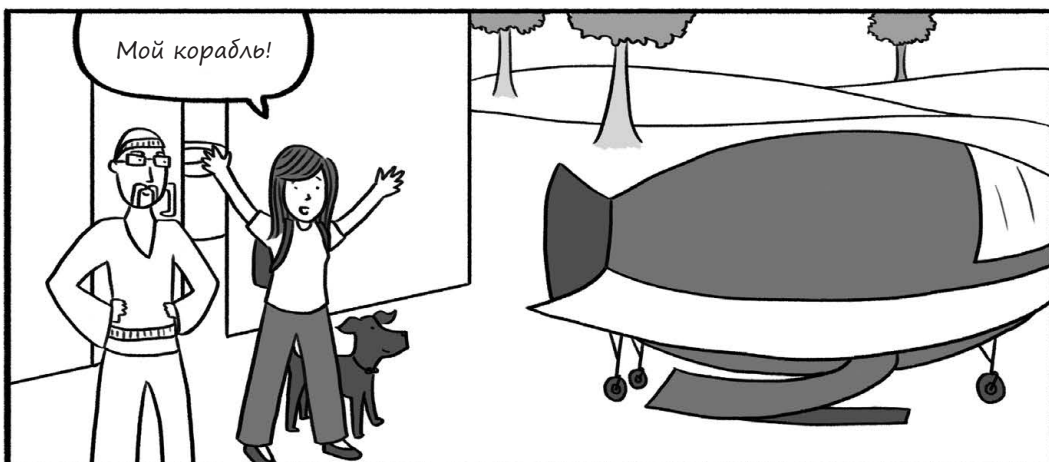
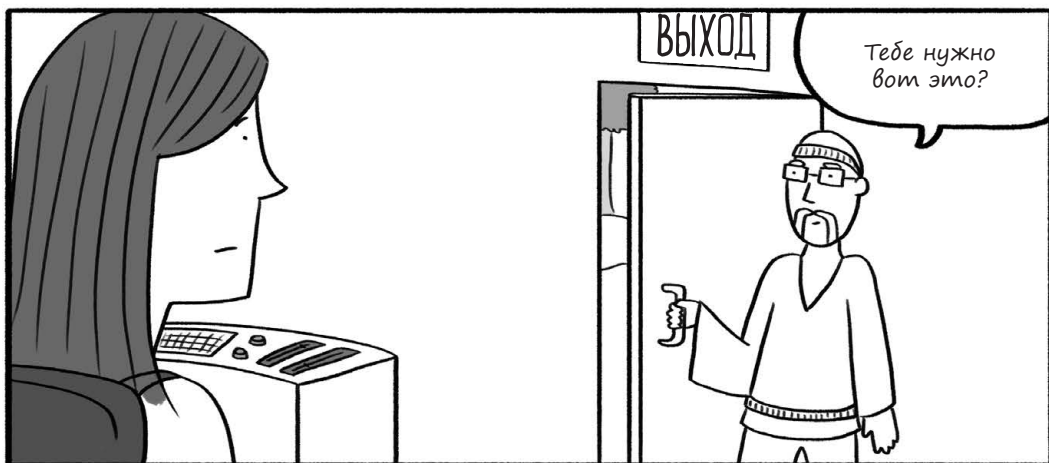
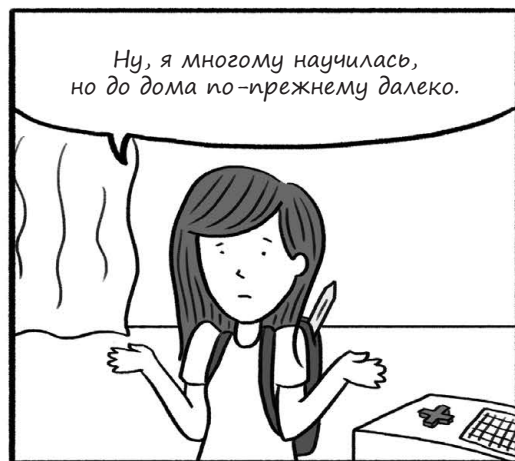
Теперь, когда ты представляешь себе, на что действительно способен WordPress, самое время составить замечательный план сайта. Что и где там должно быть? Наметь структуру страниц и разделов — и вперед! Нас ждут великие дела!

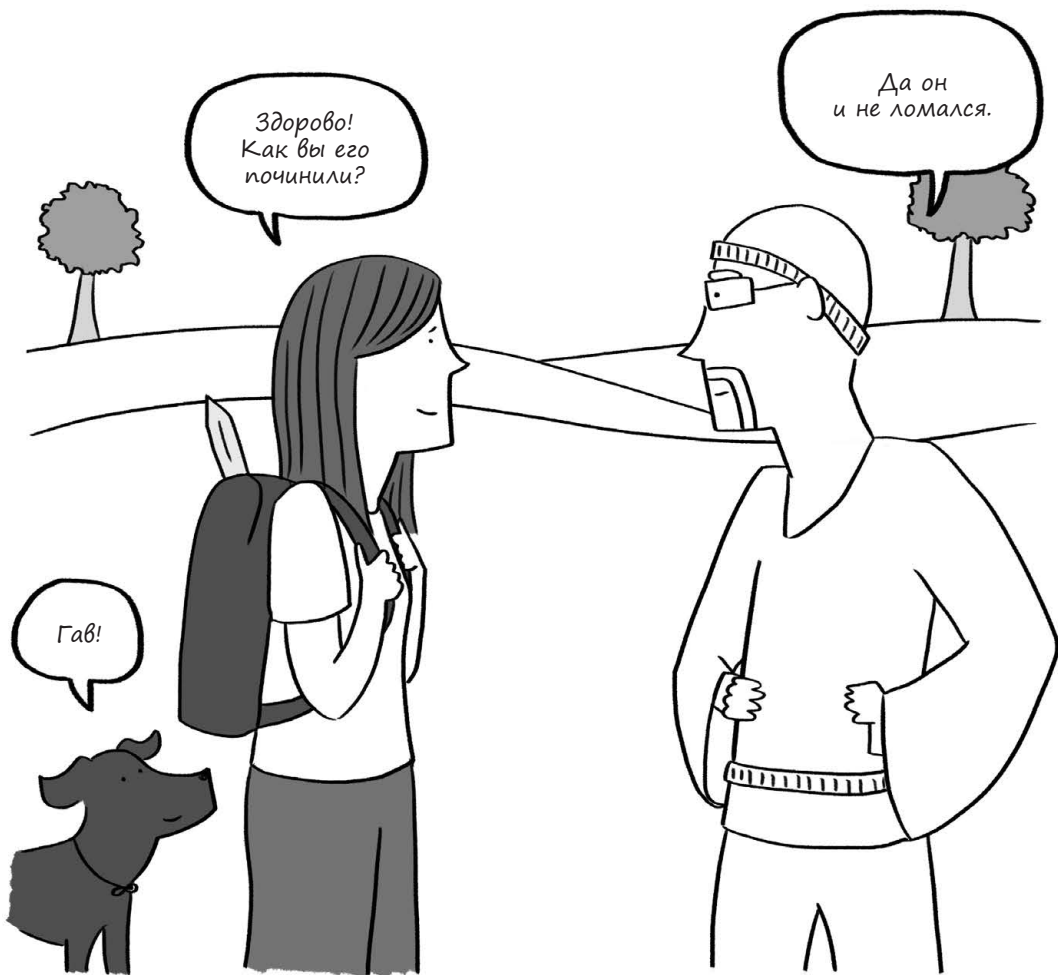
6

Запуск сайта

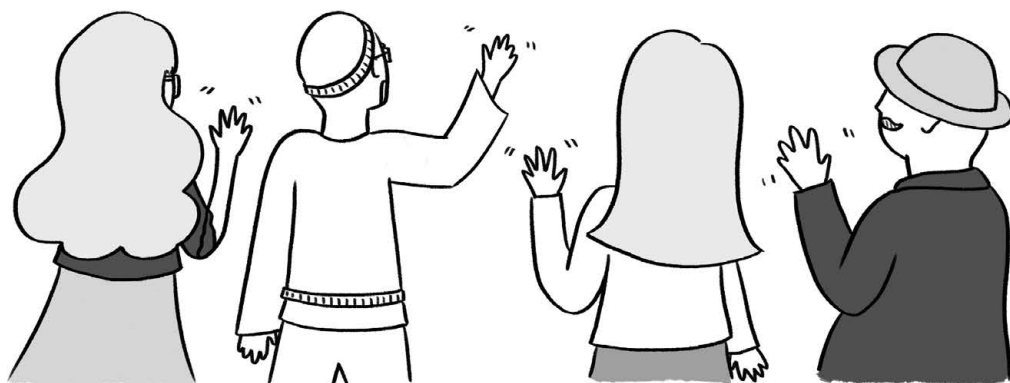


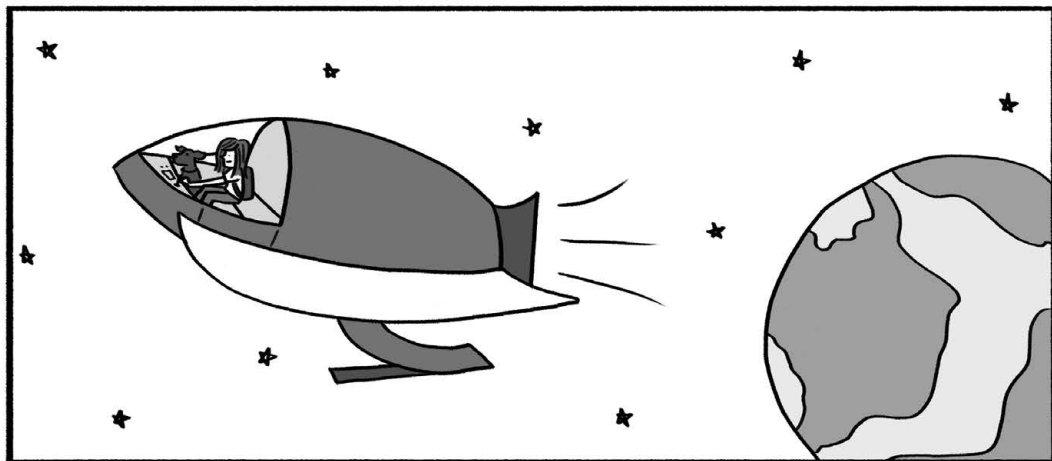
Нет ничего лучше хостинга

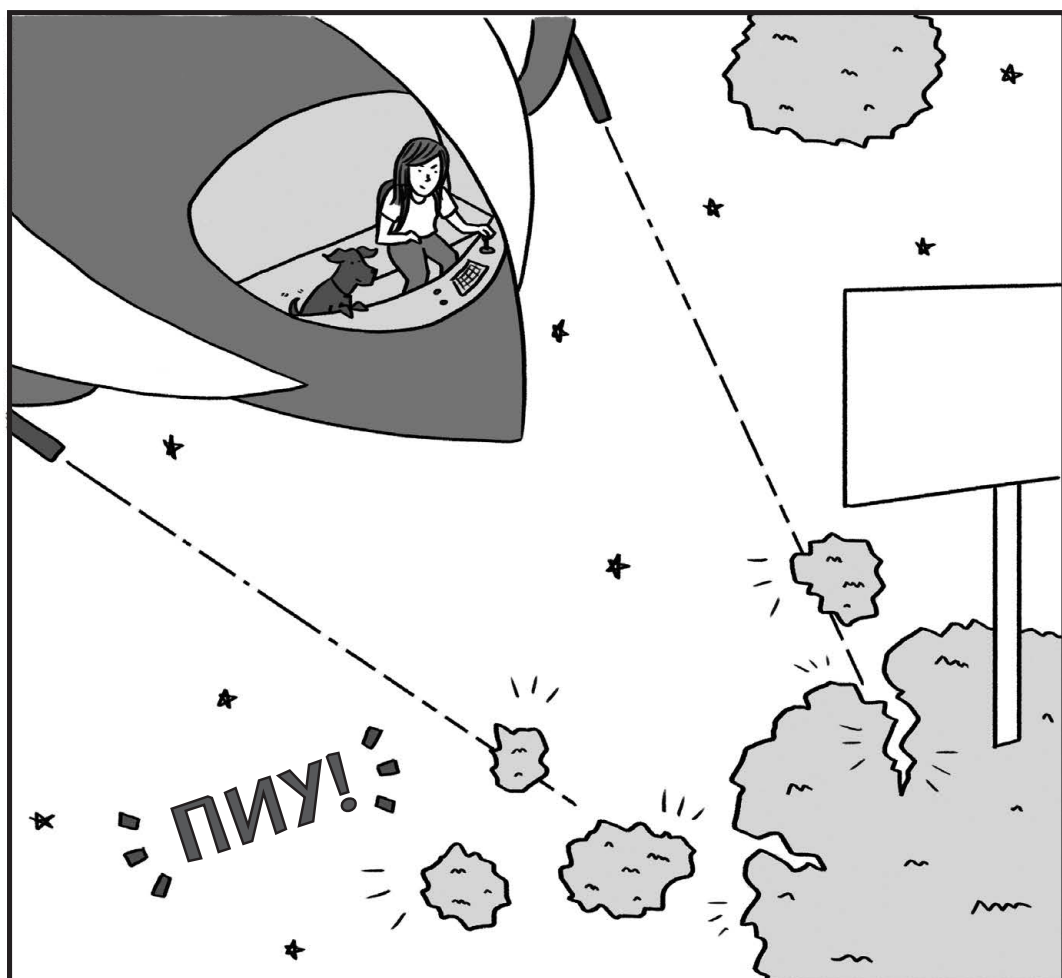
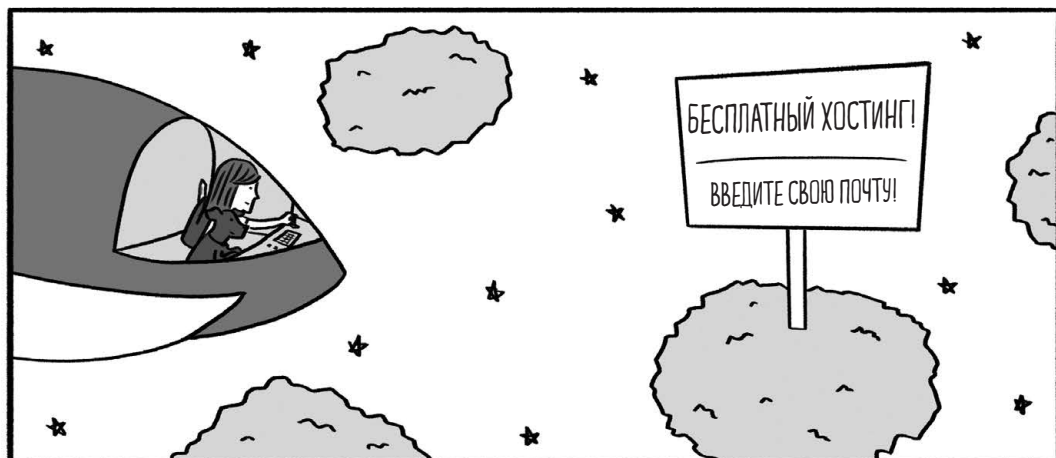


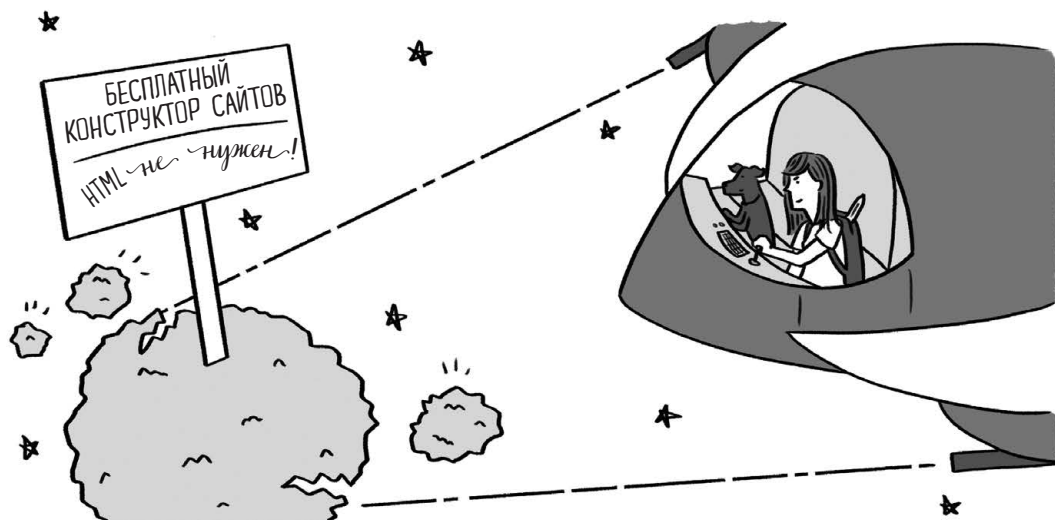


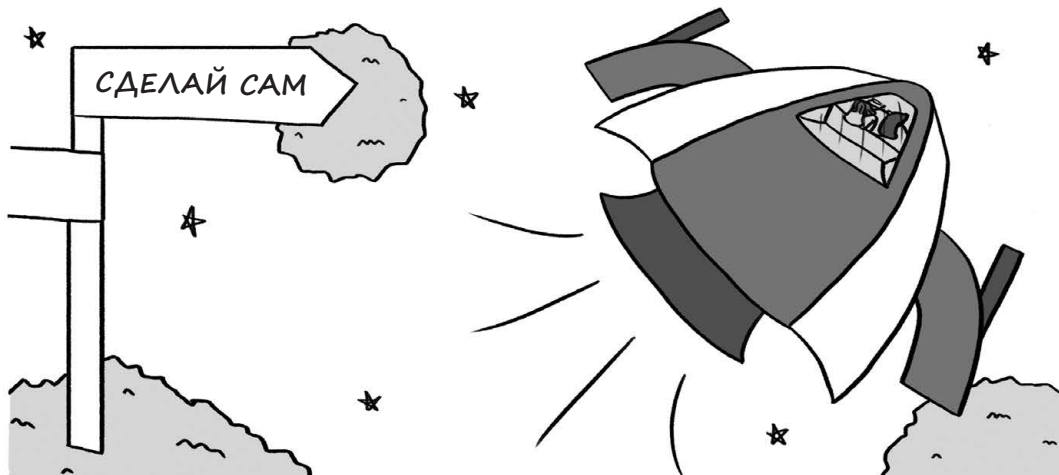














Дом для портфолио Ким



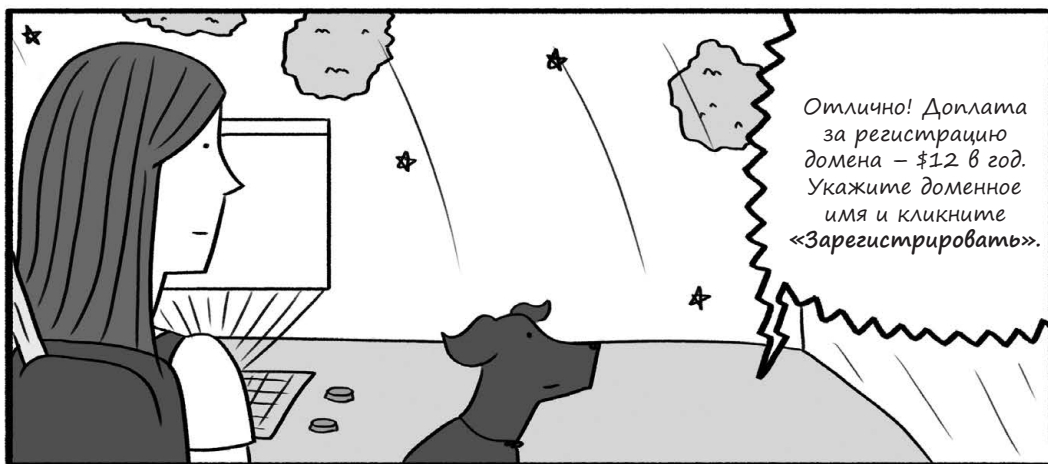
Доменное имя, доменное имя...
Ах да, это же адрес сайта.



Нет. Я бы хотела его
зарегистрировать.



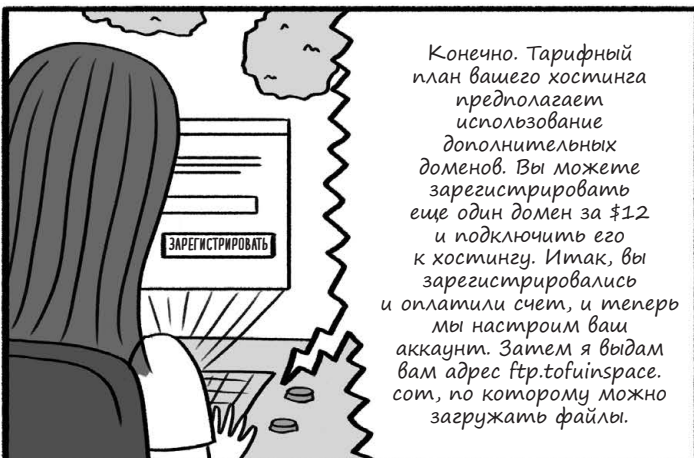
Отлично! Доплата
за регистрацию
домена – \$12 в год.
Укажите доменное
имя и кликните
«Зарегистрировать».

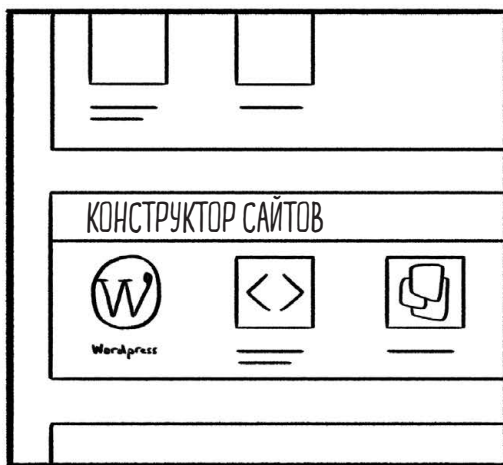
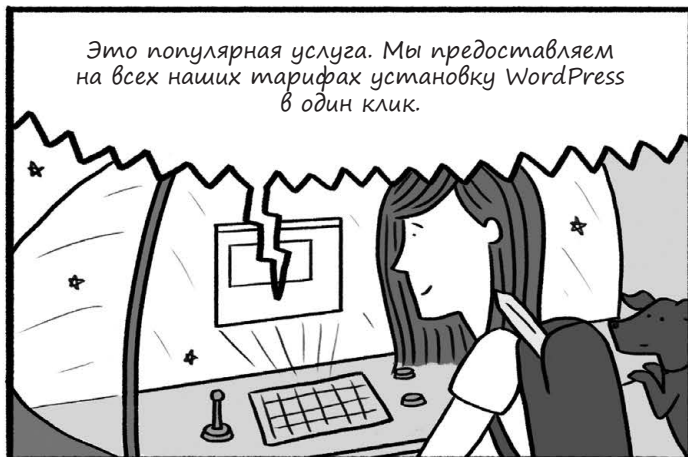


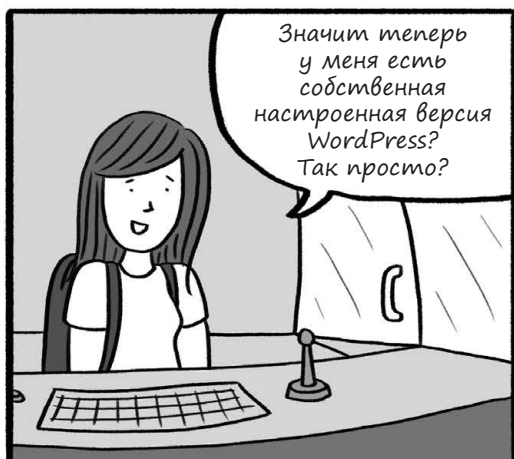
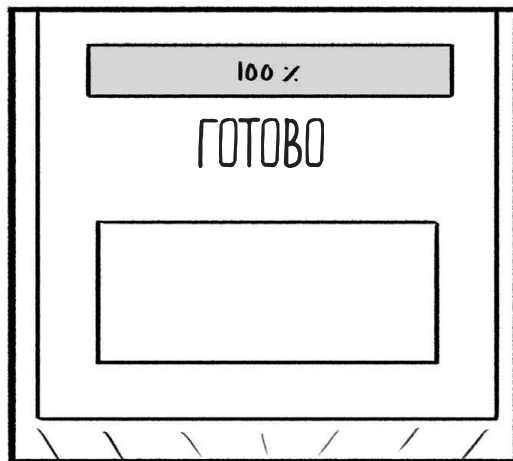
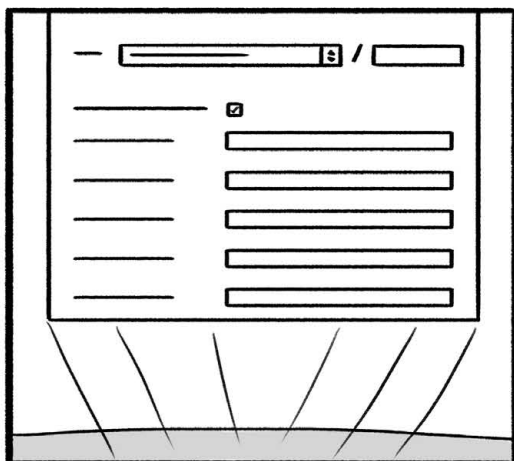
Это просто
тестовый домен.
Возможно, потом мне
понадобится еще один.

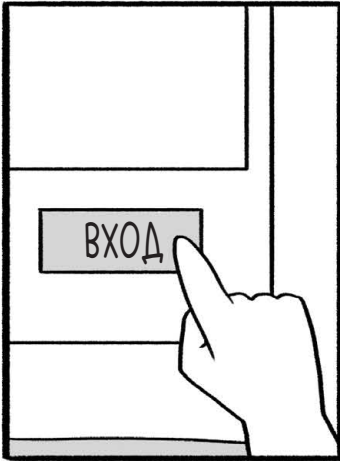


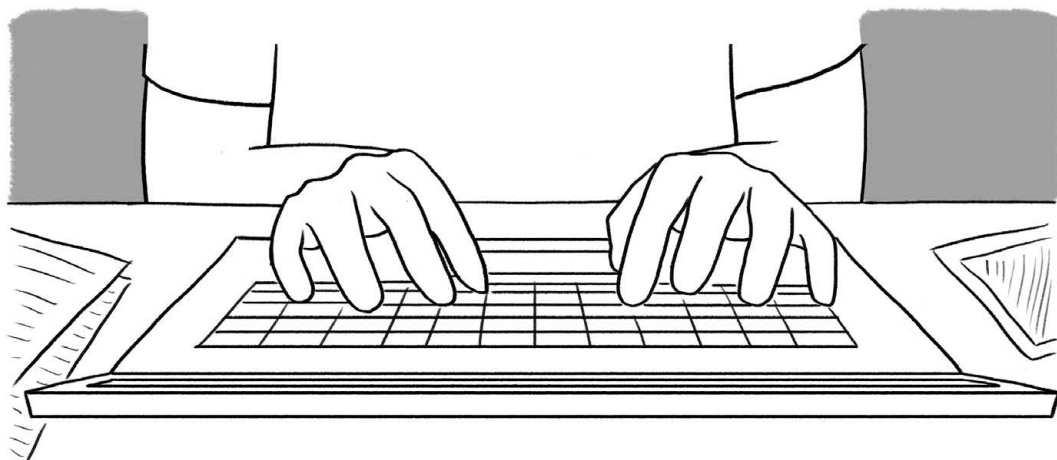
Конечно. Тарифный
план вашего хостинга
предполагает
использование
дополнительных
доменов. Вы можете
зарегистрировать
еще один домен за \$12
и подключить его
к хостингу. Итак, вы
зарегистрировались
и оплатили счет, и теперь
мы настроим ваш
аккаунт. Затем я выдам
вам адрес ftp.tofiginspase.
com, по которому можно
загружать файлы.



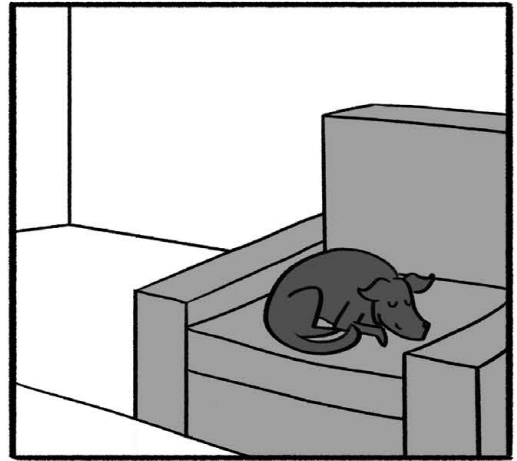


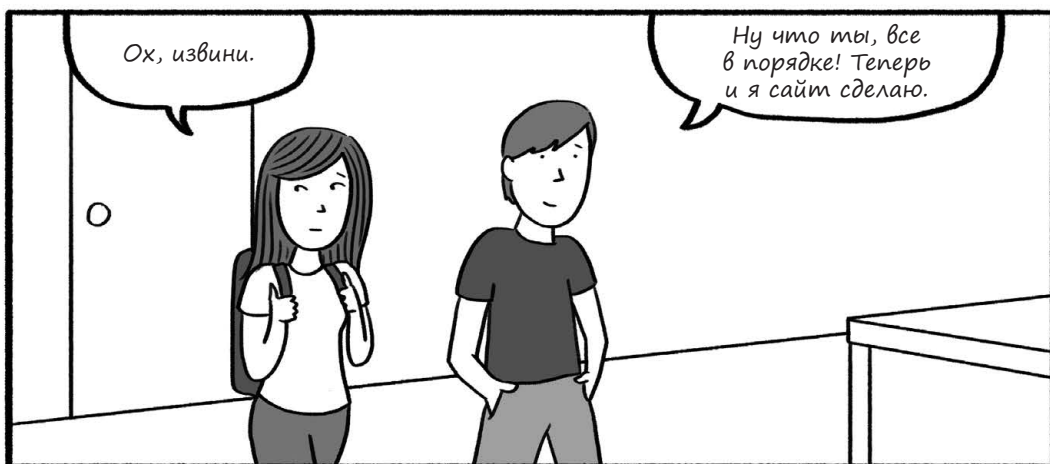


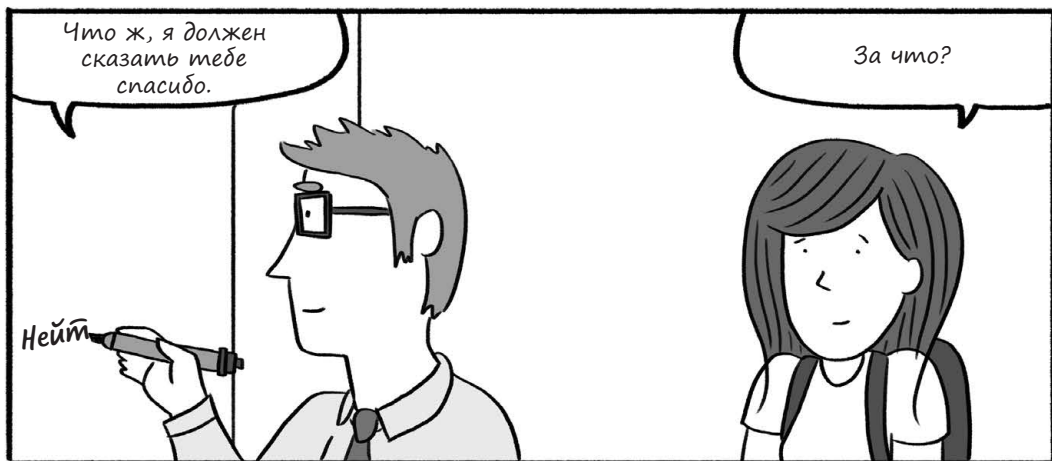




Дружеская сеть







Итак, ты готов запустить сайт

Сайт начинается с имени. Прежде чем обзавестись хостингом (где будут храниться все файлы сайта), тебе нужно зарегистрировать доменное имя, то есть адрес, по которому сайт можно будет найти в интернете.

Компании, которые выдают доменные имена, называются *регистраторами*. В сети их несколько тысяч. У большинства из них можно получить имя в одном из популярных доменов первого уровня: *.com*, *.org*, *.net* или *.ru*. Чтобы это сделать, зайти на сайт компании, которая предоставляет подобные услуги, например GoDaddy, HostGator, Ru-Center или Reg.ru (см. рис. 6.1). Там ты сможешь проверить, не занято ли выбранное имя, и арендовать его, если оно доступно. Плата за регистрацию не единовременная, а периодическая. Можно заплатить за один год или больше — как хочешь. В конце этого периода регистрацию нужно будет продлить.

Домены верхнего уровня используют уникальные имена. Если ты хочешь зарегистрировать имя *kimswebsite.com*, но оно уже занято, проверь другие домены, например *kimswebsite.ru* или *kimswebsite.org*. Подходи к выбору имени творчески!

ПРИМЕЧАНИЕ

Когда срок регистрации заканчивается, доменное имя опять становится доступным, то есть арендовать его может кто угодно. Если не хочешь лишиться имени, продлевай регистрацию вовремя, не откладывая на последний момент. Многие регистраторы предлагают услугу автопродления.

Не знаешь, в какой доменной зоне выбрать имя? Разные домены предназначены для разных целей (хотя никто за этим строго не следит). Домен *.org*, как правило, используют некоммерческие организации, *.com* — коммерческие фирмы, а *.net* (от английского network — «сеть») подходит для разных целей. Есть и зарезервированные домены для особых случаев, например *.edu* для учебных учреждений, *.mil* для военных и *.gov* для правительства. Существуют и уникальные домены для разных стран, но часто арендовать их может кто угодно. Так, домены первого уровня *.co* (Колумбия), *.tv* (Тувалу) и *.io* (Британская территория в Индийском океане) сперва предназначались для конкретных стран, но затем их прибрали к рукам компании, желающие заполучить звучное доменное имя. Часто подобные домены выдаются лишь конкретными регистраторами, и это стоит дороже, чем аренда имени в более распространенной доменной зоне.

Ты можешь зарегистрировать доменное имя до того, как обзавелся хостингом, чтобы оно точно никому больше не досталось. Арендовав домен, ты можешь делать с ним что угодно. Например, если понадобится, можно привязать его к другому хостингу.

Как правило, регистрация домена стоит недорого. Но чтобы завести сайт, одного доменного имени недостаточно. Тебе также потребуется хостинг, то есть сервер, на котором сайт будет работать. Хостинг и доменное имя не связаны между собой, но для удобства и экономии средств многие компании предлагают арендовать сразу и то и другое.

Начальный	Эконом	Люкс	Максимум
Для одного веб-сайта.	Базовые ресурсы для сайтов, которые запущены недавно.	Больше пространства и широкие возможности для разных сайтов.	Больше мощности специально для сложных сайтов и высокого трафика.
Всего за 129,00 руб в МЕСЯЦ	Всего за 299,00 руб в МЕСЯЦ	Всего за 349,00 руб в МЕСЯЦ	Всего за 609,00 руб в МЕСЯЦ
Распродажа: Скидка 35% 199,00 руб в месяц при продлении 4	Распродажа: Скидка 42% 519,00 руб в месяц при продлении 4	Распродажа: Скидка 47% 659,00 руб в месяц при продлении 4	Распродажа: Скидка 44% 1 099,00 руб в месяц при продлении 4
Добавить в корзину	Добавить в корзину	Добавить в корзину	Добавить в корзину
1 веб-сайт Место на диске: 30 ГБ Неограниченная полоса пропускания ②	Начальные функции, а также Профессиональная поддержка Место на диске: 100 ГБ Бесплатная деловая почта в течение первого года ②	Функции плана "Эконом", а также Неограниченное количество веб-сайтов ⁶⁰ Неограниченное дисковое пространство Неограниченное число субдоменов	Функции плана "Люкс", а также В два раза больше памяти и вычислительной мощности ② Бесплатный сертификат SSL на 1 год ¹¹ ② Бесплатно: Премиум-DNS ② Неограниченные базы данных
Стандарт план	Бизнес план	Эконом план	
<ul style="list-style-type: none"> 5 домена Безлимитный дисковое пространство Безлимитный Трафик Безлимитный email-аккаунты 	<ul style="list-style-type: none"> 10 домена Безлимитный дисковое пространство Безлимитный Трафик Безлимитный email-аккаунты 	<ul style="list-style-type: none"> Личный undefined Безлимитный дисковое пространство Безлимитный Трафик Безлимитный email-аккаунты 	
3 года за 355.00 руб/мес	3 года за 485.00 руб/мес	3 года за 260.00 руб/мес	
Купить	Купить	Купить	
Виртуальный хостинг 100 Простые статичные сайты	РЕКОМЕНДУЕМ Виртуальный хостинг 200 Блоги и простые сайты	Виртуальный хостинг 300 Сайты компаний	
от 129 руб./мес.	от 199 руб./мес.	от 299 руб./мес. от 199 руб./мес.	
Диск 4 Гб SSD Количество сайтов 10	Диск 10 Гб SSD Количество сайтов 15 PHP, Perl, Python Да	Диск 15 Гб SSD Количество сайтов 25 PHP, Perl, Python Да	
✓ Домен .ru, .рф, .site, .online, .store в подарок	✓ SSL-сертификат в подарок	✓ Домен .ru, .рф, .site, .online, .store в подарок ✓ SSL-сертификат в подарок ✓ Бесплатный перенос до 20 сайтов	
Заказать	Заказать	Заказать	

Рис. 6.1. Сравнение тарифов компаний GoDaddy (вверху), HostGator (в середине) и Ru-Center (внизу)

Покупка хостинга: твой дом в сети

Часто выбор хостинга пугает новичков: столько разных вариантов! Не волнуйся: ошибиться тут сложно. А если хостинг тебе не понравится, его всегда можно сменить. Первым делом подумай, что тебе нужно. WordPress? Тогда выбирай хостинг с установкой WordPress в один клик. А может, для начала ты хочешь сделать сайт на чистом HTML?

Вот некоторые вопросы, которые стоит обдумать при выборе хостинга.

Сколько доменных имен поддерживает хостинг?

Для каждого сайта нужен отдельный домен. Если у тебя сразу несколько проектов или ты хочешь создать отдельно личный сайт и сайт для учебы, тебе потребуется несколько доменных имен.

Сколько данных можно будет хранить?

В основном это касается хранения файлов. Если ты делаешь портфолио, где будет много фотографий или видеороликов в высоком разрешении, ищи хостинг с подходящим объемом дискового пространства. Сейчас многие хостинги предлагают тарифы с неограниченным дисковым пространством — возможно, тебе подойдет именно такой.

Какой лимит пропускной способности?

Пропускной способностью (или *ограничением трафика*) называют суммарный объем данных, который хостинг может передать посетителям в течение месяца. Если у сайта много посетителей и они скачивают объемные файлы, нужен хостинг с повышенной пропускной способностью. Есть и тарифы без ограничений по скачиванию данных — если сомневаешься, рассмотри такой вариант.

Доступна ли на хостинге электронная почта?

Хотя электронная почта и не обсуждается в этой книге, отмечу, что большинство хостинговых компаний предлагают и почтовые сервисы. Но почта не всегда идет в комплекте с хостингом: часто ее подключают за отдельную плату.

Какие услуги службы поддержки включены в тариф?

Можно ли обратиться в службу поддержки по телефону? Доступна ли поддержка через онлайн-чат или по электронной почте? Техподдержка — один из главных факторов, которыми хостинговые компании отличаются друг от друга.

После того как ты найдешь хостинг с необходимым функционалом, подыщи тариф, соответствующий твоему бюджету. Если понадобится, в будущем его можно будет сменить на более дорогой. Имеет смысл платить больше, только если у тебя есть особые требования, вроде поддержки двух разных доменов. Например, ты можешь зарегистрировать одно доменное имя для личного сайта, а другое — для учебного проекта. Многие хостинговые тарифы включают бесплатные дополнительные адреса. Оплатив хостинг, ты сможешь подключать дополнительные адреса к разным папкам на сервере.

Запуск простого сайта на HTML/CSS

После аренды домена с хостингом тебе предоставят данные для входа в аккаунт. Например, их могут выслать по электронной почте. Это значит, что ты не увидишь серверы своими глазами, а будешь работать с ними по сети, используя FTP или панель администрирования в окне браузера.

Следуй инструкциям хостинговой компании, чтобы зайти в аккаунт. После этого ты окажешься в панели администрирования — такой как cPanel (см. рис. 6.2) или Plesk.

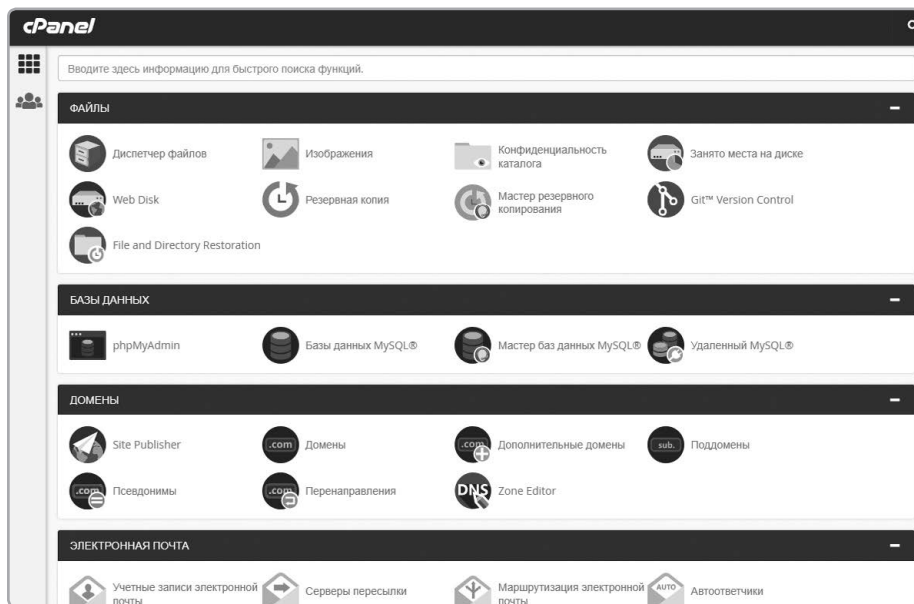


Рис. 6.2. Экран cPanel на хостинге HostGator

Также тебе должны выдать адрес, логин и пароль для FTP. Часто эта информация указана и в панели администрирования. FTP-доступ понадобится для загрузки файлов с компьютера на сервер.

Как ты уже знаешь из главы 1, чтобы вручную создавать страницы, нужны две программы: редактор кода и FTP-клиент. Если у тебя установлен клиент FileZilla (<https://filezilla-project.org/>), введи туда FTP-адрес, логин и пароль (полученные от хостинг-провайдера), чтобы подключиться к своей папке на сервере (см. рис. 6.3).

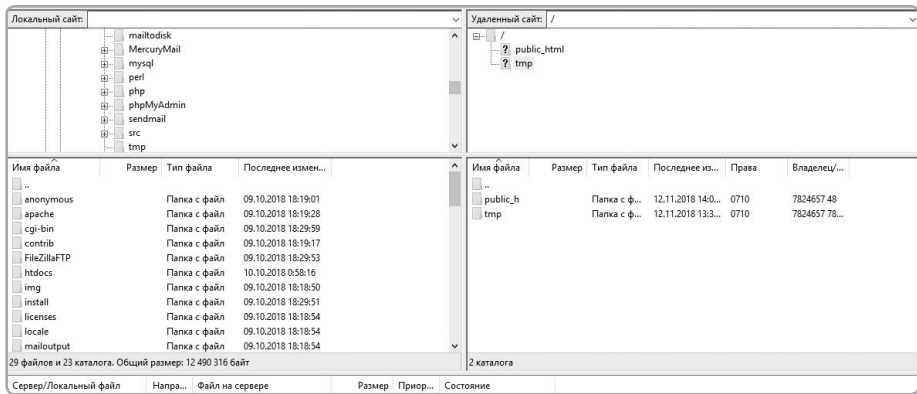


Рис. 6.3. FileZilla показывает файлы, хранящиеся и на твоём компьютере, и на подключенном по FTP хостинге. Чтобы загрузить файлы с компьютера на сервер, просто перетащи их с панели на панель

Поскольку ты работаешь со своего компьютера, советуем создать на жестком диске папку для каждого сайта. Она должна полностью дублировать содержимое сайта на сервере. Во-первых, это будет резервная копия, а во-вторых, так ты сможешь тестировать все изменения *локально* — до того как выложить их на сайт. Убедившись, что все работает, с чистой душой отправляй файлы на сервер с помощью FileZilla.

Запусти FileZilla, введи адрес, логин и пароль от FTP и кликни **«Быстрое соединение»**. После успешного подключения слева будет отображаться панель локальных файлов (твой компьютер), а справа — панель серверных файлов (твой хостинг).

Открой в локальной панели папку со своими HTML- и CSS-файлами, а в серверной панели — папку, куда эти файлы нужно загрузить. Помни, что у сайта всегда есть *корневая* папка, в которой можно создавать подпапки, как это делала Ким. Убедившись, что открыта нужная папка сайта, перетащи файлы из левой (локальной) панели в правую (серверную). В результате файлы будут загружены на хостинг, а у тебя на диске останутся их локальные копии.

Корнем твоего сайта (местом, где хранятся все страницы) обычно является папка под названием *www* или *public_html*. Если не знаешь, какая из папок корневая, уточни это у своего хостинг-провайдера. Если ты используешь WordPress, локальные копии обычно не нужны: можешь создавать сайт прямо на хостинге.

Установка WordPress

Как ты знаешь из главы 4, WordPress — это система управления контентом, которая работает на сервере, позволяя нам создавать веб-страницы и посты в блоге, а также загружать медиафайлы (картинки, видео и так далее) через браузер. В системах управления контентом есть много инструментов, которые помогают... как бы это сказать... управлять контентом. С помощью WordPress тебе будет проще создать сайт и сделать его привлекательным, причем для этого не придется писать код или загружать файлы по FTP.

Сегодня на многих хостингах есть поддержка WordPress по умолчанию и с мгновенной установкой. Сам по себе WordPress бесплатный, но для работы ему нужен сервер, то есть хостинг, за который придется платить. Сервер хранит текст, картинки и прочий контент сайта, а также выдает эту информацию посетителям.

О покупке хостинга для WordPress

Если ты собираешься арендовать хостинг для сайта на WordPress, выбирай один из тех хостингов, где есть установка WordPress «в один клик» (см. рис. 6.4). Скрипт мгновенной установки упрощает настройку WordPress до предела. Он создает отдельную папку, например `<твой-сайт>.com/wordpress` или `<твой-сайт>.com/blog`. Когда ты запустишь скрипт установки, он предложит выбрать этот адрес (см. рис. 6.5). Можешь указать там корень сайта, и тогда адресом будет просто `<твой-сайт>.com`. Установка «в один клик» берет на себя всю тяжелую работу: скачивает свежую версию WordPress, создает для него базу данных и автоматически ее подключает. Если на хостинге нет мгновенной установки, обратись к подробным инструкциям в онлайн-документации WordPress: http://codex.wordpress.org/Установка_WordPress.

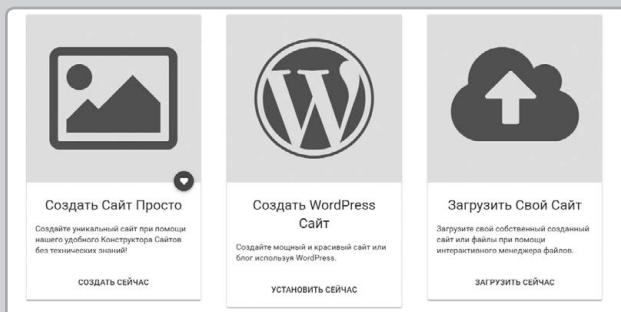


Рис. 6.4. Панель cPanel со скриптом установки WordPress

The image shows a form titled 'Установить WordPress' (Install WordPress). It contains several input fields: 'Имя пользователя' (Username) with 'admin' entered, 'Администратора' (Administrator), 'Пароль Администратора' (Administrator Password), 'Адрес' (Address), and 'Язык' (Language) with 'Russian' selected in a dropdown menu. A 'Установить' (Install) button is located at the bottom right of the form.

Рис. 6.5. Скрипт установки спросит, куда ты хочешь установить WordPress

Заключение

Итак, наше приключение подошло к концу. Сколько новой информации! Теперь ты владеешь древней мудростью HTML и утонченным искусством CSS. Знаешь, как справиться с драконами 404 и выбрать истинный путь. Ориентируешься в WordPress-сити и понимаешь, как работают современные системы управления контентом. Тебе даже довелось примерить самые модные темы. Теперь ты можешь создать собственный сайт.

Но это вовсе не конец обучения — это его начало. Старт был хорошим, но что именно ты создашь, зависит от тебя. Не надейся, что путь будет гладким, как шелк. Когда мы беремся за что-то новое, задача всегда оказывается сложнее, чем нам казалось вначале. Секрет в том, чтобы не бросать все на полпути. Никогда не опускай руки! А если запутаешься, снова открой эту книгу, перечитай нужный раздел и возвращайся на свой путь. Гуру, Ким и ее друзья всегда будут здесь, чтобы дать тебе подсказку, но с этого момента начинается новое, твое собственное приключение!

Предметный указатель

Числа и символы

.com, доменная зона 24, 249
.css, расширение 106
.html, расширение 60
.net, доменная зона 249
.org, доменная зона 249
<a> (ссылка), тег HTML 36–37, 49, 63, 69
<body>, тег HTML 34, 59, 69

 (перенос), тег HTML 35, 62, 69
<div>, тег HTML 102–103, 115–118, 122–125
, тег HTML 33, 62, 69
<h1>, тег HTML 63, 69, 109–110
<head>, тег HTML 59, 66–70, 79–81, 105–106
<html>, тег 59, 69
, тег HTML 47–50, 53–57, 64–65, 69
<meta> (метаданные), тег HTML 67–68, 70
<p> (абзац), тег HTML 33–35, 60–61, 69, 86
, тег HTML 62, 69
<title>, тег HTML 66–67, 70, 79–80
404, ошибка 40–42, 55

А

Абзац (<p>), тег HTML 33–35, 60–61, 69, 86
«Абзац», выпадающее меню в WordPress 166
Абзацы, оформление в CSS 86–88, 107–109
Абсолютные URL 48–49, 56, 64, 65, 82
Администраторы, WordPress 156, 157
Адрес сайта. См. также Доменные имена; URL
Активные плагины, WordPress 196, 219
Активные темы, WordPress 160, 188, 205–206
«Атрибут alt», поле в панели «Свойства

изображения», WordPress 169
«Атрибуты страницы», раздел WordPress 174–175

Атрибуты, HTML

alt 50, 70
class 98, 100, 113, 118
href 36, 63, 69
id 98–100, 111
src 49, 70

Б

Без засечек, шрифты 88–89, 107
Бэкенд, WordPress 158, 160
Бесплатный аккаунт, WordPress 157
Библиотека файлов, WordPress 150–154, 168–169
Браузеры 23, 60

В

Веб-хостинг
WordPress 26, 157, 242–243, 253–254
загрузка файлов 253
запуск простого сайта 252–254
обзор 25–26, 235–243
перенос при помощи панели инструментов 226–227
установка 251–252
Верхнего уровня домены 249
Верхнего уровня страницы, WordPress 174–176
Видео в постах, WordPress 181–182
Виджеты, WordPress 195, 222–225
«Видимость», раздел WordPress 173

Визуальный редактор, WordPress
добавление изображений 167–171
обзор 141–144
создание постов 176–177
создание страниц 163–167
«Википедия» 25
«Внешний вид», панель WordPress.
См. также Настройка WordPress
«Заголовок», раздел 208
«Меню», раздел 211
«Настроить», раздел 208–209
«Настройки темы», раздел 192, 208, 209
обзор 188–189
темы 205
«Все записи», раздел WordPress 183
«Вставить в страницу», кнопка WordPress 169
Встроенный CSS 105
Вход и выход, WordPress 140, 157–159, 160
Выравнивание изображений, WordPress 170, 171

Г

Гипертекстовый язык разметки (HTML)
404, ошибка 40–42, 55
<head>, секция 66–68
CSS внутри документа 92–93
в WordPress 144, 165–167, 201–204, 217
изображения в файловой структуре 49–58
, тег 47–50, 53–57, 64–65, 69
карта сайта 45–46
обзор 64–65
запуск простого сайта 252–254
обзор 30–33, 59
отличие от CSS 103–104
основные теги 33–37, 60–64, 69–70
подключение CSS-документов 78–83, 105
правила именования 39–44
«Просмотреть код», опция 91

пути 38–39, 106–107
создание нового документа 59–60
теги. См. Теги HTML

Д

Дата публикации, WordPress 173
«Действия», выпадающее меню WordPress 183
«Детали», ссылка для плагинов WordPress 219
Дефисы в имени HTML-документов 43, 59
Дисковое пространство
доступное на хостинге 251
хранение картинок и HTML 49–58
«Добавить в меню», кнопка WordPress 211, 213
«Добавить медиафайл», кнопка WordPress 167–168
«Добавить новую тему», кнопка WordPress 206
Добавление картинок в HTML
, тег 47–50, 53–57, 64–65, 69
карта сайта 45–46
обзор 64–65
Домашняя страница, WordPress 162
Доменные имена 24, 241, 249–250, 251–252
Дополнительные домены 251–252
«Дополнительные опции форматирования»,
кнопка WordPress 166
Дочерние страницы, WordPress 146, 174–176

Е

Единый указатель ресурсов (URL)
абсолютные ссылки 48–49, 56, 64, 65, 82
обзор 24
относительные ссылки 56–57, 64, 65,
82–83

Ж

Жирный шрифт, HTML 62

З

Заглавные буквы, в HTML-документах 42, 59
Заголовки, создание в WordPress 166
Заголовки, теги HTML 63, 69, 109–110
«Заголовок», поле в панели «Свойства изображения», WordPress 169
Загрузка файлов на хостинг 253
Заккрытие тегов, HTML 35, 37, 62
Записи, вкладка WordPress 178, 183
Защищенный протокол передачи гипертекста (HTTPS) 24

И

Идентификаторы, CSS 84–88, 98–102, 103, 105
Изображения
 встраивание в HTML
 , тег 47–50, 53–57, 64–65, 69
 карта сайта 45–46
 обзор 64–65
 файловая структура 49–58
 добавление на страницу WordPress 167–171
Имена HTML-документов 39–44, 59
Импорт контента в WordPress 226–227
Имя HTML-файла по умолчанию 43–44, 52, 57
Информационные ресурсы по WordPress 227
Исходный код веб-страницы, просмотр 60, 91

К

Карта сайта 150–151
Картинки. См. изображения
Каскадные таблицы стилей (CSS)
 <div>, тег 102–103, 115–118, 122–125
 базовые свойства 125–126
 в WordPress 155, 167, 201–204, 217
 внутри HTML-документов 92–93
 запуск простого сайта 252–254
 идентификаторы 96–100, 110–114, 115, 117

классы 100–101, 113–114, 115
наследование 93–94, 111–112
обзор 75–78, 105
отличие от HTML 103–104
подключение к HTML-файлам 78–83, 105
поля и отступы 119–121
«Просмотреть код», опция 91
таблица стилей, создание 105–107
синтаксис 107–110
цвета 94–95, 114–118
шрифты и текст, вид 85–90, 107

Классы, CSS 100–101, 113–114, 115

Клиенты 24–25

Колонки, в CSS 101–103, 122–125

Консоль, WordPress

 «Настройки», вкладка 215–217

 «Настройки экрана», вкладка 213, 214–215

 обзор 140–141, 157–1581

 «Обновления», вкладка 224

 переключение между бэкендом и фронтендом 160

Контейнеры 117

Корень сайта

 CSS-файлы в корне 81, 106

 FTP 252

 определение 40, 47

 относительные пути 64, 82–83

Корзина, WordPress 183

Краткое описание 216

Курсив, HTML 33, 62, 69

Л

Логотипы 65, 83

М

Медиафайлы, WordPress

 добавление на страницу 167–171

 форматы постов для 181–182

Меню навигации, WordPress

настройка 211–214

обзор 161

Меню, настройка в WordPress 211–214

Метки, WordPress 148–149, 178–180

Мягкий перенос 62

Н

«Написание», раздел вкладки «Настройки», WordPress 217

Наследование тем, WordPress 217

Наследование, CSS 93–94, 111–112

Настройка WordPress

меню навигации 211–214

опции 215–217

расширенная 217

темы 192–193, 198–204, 208–211

«Настройки темы», раздел WordPress 192, 208–209

«Настройки экрана», вкладка WordPress 213, 214–215

«Настройки», вкладка в WordPress 215–217

Неверные ссылки 42, 55–56, 65

О

Область темы, WordPress 212–213

Обновление страницы в браузере 35, 60

Обновления, в WordPress 218, 224–226

«Обсуждение», раздел вкладки «Настройки», WordPress 217

«Общие», раздел вкладки «Настройки», WordPress 215–216

Одиночные теги, HTML 62, 69

«Описание», поле в панели «Свойства изображения» WordPress 169

Оптимизация для поисковых систем (SEO) 50, 169

Отмена публикации, в WordPress 172–173

Относительные URL 56–57, 64, 65, 82–83

Отступы, HTML 60–62

П

Панель администратора, WordPress 160

Панель администрирования, хостинг 252–253

Панель форматирования, WordPress 164

Перевод строк

в CSS 107

в HTML 59

Перенос строки, HTML 35, 62–63, 69

Плагины, WordPress

обзор 193–196

обновления 224

общие сведения 218–221

План сайта 145–149, 162–163

«Подпись», поле в панели «Свойства изображения», WordPress 169

Поисковые системы и тег <meta> 67–68

Посты в блоге, создание в WordPress 176–177

верхнее изображение 180–181

и страницы 161

метки и рубрики 148–149, 178–180

обзор 137–139

сортировка 147–149

управление и удаление, контент 183

форматы постов для медиаконтента 181–182

Правила именования HTML-файлов 39–44

Предпросмотр страниц, WordPress 163, 164–165, 166, 170–171

Премиальные темы, WordPress 191–192, 205, 211

Пробелы

в CSS 107

в HTML 43, 59

Продление регистрации доменного имени 249

Пропускная способность, хостинг 251

Просмотр исходного кода, опция 60

«Просмотр», кнопка плагина Meta Slider 220

«Просмотреть запись», опция в WordPress 178–179

«Просмотреть код», опция 91

Протокол передачи гипертекста (HTTP) 24

Протокол передачи файлов (FTP) 24, 26, 253–254

Процент (%), размер шрифта в CSS 89

Псевдоэлементы, в CSS 124

Публикация страниц в WordPress 163, 172–173

Пути, HTML 38–39, 106–107

Р

«Размер», поле в панели «Свойства изображения», WordPress 169

Разметка, HTML 59

Регистр букв, HTML 42, 43, 59

Регистраторы доменных имен 249–250

Редакторы кода 23–24

Решетка (#), в CSS 99–100, 111

Родительские страницы, WordPress 146, 174–176

Рубрики, WordPress

добавление в меню навигации 211–212

обзор 148–149, 178–180

С

С засечками, шрифты 86, 107

«Свойства изображения», панель WordPress 169

Свойства, CSS 108

Селекторы, CSS 87, 107, 108

Серверы 24–25

Синтаксис, CSS 108–110

Система управления контентом (CMS).
См. также WordPress 157

слайд-шоу, плагин для WordPress 218–221

Слеш (/), HTML 57, 65, 82–83, 106–107

Служба поддержки, хостинг 251

Собственный хостинг и WordPress 205, 217, 218

Соглашения, HTML 61

Сохранение изменений в HTML 35

Ссылка (<a>), тег HTML 36–37, 49, 63, 69

«Ссылка на файл», поле в панели «Свойства изображения», WordPress 169

«Ссылки», раздел вкладки «Настройки», WordPress 217

Ссылки. См. также Единый указатель ресурсов (URL)

добавление в меню навигации WordPress 212, 214–215

на рубрики и метки WordPress 180

создание в WordPress 165–166

чтение 24

Сторонние веб-сайты с темами WordPress 207

Страница блога, WordPress 147, 176–177

«Страницы», вкладка WordPress 163, 172

Страницы, WordPress

добавление в меню навигации 211, 213–214

и посты в блоге 161

медиафайлы, добавление 167–171

обзор 139

правка и удаление содержимого 183

публикация 172–173

создание первой 163–167

упорядочивание 145–147, 174–176

Структура папок, HTML 49–57, 64–65

Структура, темы WordPress 210

Т

Теги, HTML

атрибуты 49

заголовки 63, 69, 109–110

закрывающие 35, 37, 62

одиночные 62, 69

поиск описаний 104
ссылки 36–37, 63–64, 69
часто используемые 33–37, 60–64, 69–70
«Текст», вкладка визуального редактора WordPress 144, 166–167
Текст, оформление в CSS 85–90, 107
Текстовые редакторы 23–24
Темы, WordPress
 активные 160, 188, 205–206
 верхнее изображение 180–181
 меню навигации, настройка 211–214
 настройка 192–193, 198–204, 208–211
 «Настройки экрана», вкладка 214–215
 обзор 188–193
 обновления 224
 основы 205–207
 премиальные 191–192, 205, 211
 расширенная настройка 217
 форматы постов 181–182
Точка (.), CSS 101, 114
Точка с запятой (;), CSS 88, 108
У
Угловые скобки (<>), HTML 59
Удаление контента, WordPress 170, 183
Установка
 плагинов WordPress 219
 тем WordPress 206
Установка «в один клик», WordPress 254
Ф
Файловая иерархия, HTML 49–58
Файловые менеджеры 24
Файлы, загрузка на хостинг 253
Фигурные скобки ({}), CSS 87, 107
Форматы постов, WordPress 181–182
Фото в постах, WordPress 181

Фронтенд, WordPress 159, 160
Функции, HTML 203

Х

Хостинг. См. веб-хостинг

Ц

Цвета, в CSS 94–95, 114–116
Цитаты в постах, WordPress 181–182

Ч

Черновики, WordPress 172–173
«Чтение», раздел вкладки «Настройки», WordPress 216

Ш

Шаблоны страниц, темы WordPress 210
Шапка, раздел, WordPress 208
Шестнадцатеричные значения 94, 114–116
Шорткод, плагин Meta Slider 220–221
Шрифты в CSS 85–90, 107–110

Э

Экран выбора темы, WordPress 160
Экспорт контента из WordPress 226
Электронная почта на хостинге 251

А–Z

alt, атрибут HTML 50, 64, 70
background-color, свойство CSS 117, 124
background-image, свойство CSS 124
border-color, свойство CSS 124
border-style, свойство CSS 124
border-width, свойство CSS 124
border, свойство CSS 124
Chrome, браузер 23
class, атрибут HTML 98, 113
clear, свойство CSS 125
CMS (система управления контентом).
См. также WordPress 157

Codex, документация WordPress 217, 227, 254
color, свойство CSS 94, 115, 125
cPanel 252
CSS. См. Каскадные таблицы стилей (CSS) 105
Custom Design, расширение WordPress 217
DOCTYPE, команда 59
em, размер шрифтов в CSS 89, 108
FileZilla 26, 252
Firefox, браузер 23
float, свойство CSS 117, 124–125
font-family, свойство CSS 125
font-size, свойство CSS 125
font-weight, свойство CSS 125
FTP (протокол передачи файлов) 26, 252–253
GoDaddy 249
height, свойство CSS 116, 125
HostGator 26, 249, 250
href, атрибут HTML 49, 63, 69
HTTP (протокол передачи гипертекста) 24
HTTPS (защищенный протокол передачи гипертекста) 24
id, атрибут 98, 101, 111
index.html 43
Internet Explorer, браузер 23
Jetpack, плагин 217, 224
Mac, компьютер
 FTP-клиент для 26
 веб-браузер для 23
 текстовый редактор для 23
margin, свойство CSS 121, 125
Meta Slider, плагин WordPress 218
Microsoft Word 23
NotePad++ 23
overflow, свойство CSS 125
p, селектор в CSS 107
padding, свойство CSS 120–121, 125
PHP, скриптовый язык 24, 203, 239
pt, размер шрифта в CSS 89,
px, размер шрифта в CSS 89, 108, 125
RGB, значения 114
Safari, браузер 23
SEO (оптимизация для поисковых систем) 50, 169
src, атрибут HTML 49, 64, 70
Sublime Text, редактор кода 24
TextWrangler, редактор кода 23
text-align, свойство CSS 126
text-decoration, свойство CSS 126
Twenty Fourteen, тема WordPress
 категории и метки 180
 меню навигации 161
 «Настройка», раздел 208, 209
 обзор 160, 210
 области для виджетов 222
 области темы 212–213
 форматы постов 181–182
 шаблоны страниц 210
 «Шапка», раздел 208
URL. См. Единообразный указатель ресурсов (URL) 24
width, свойство CSS 116, 126
Windows, компьютер
 FTP-клиент для 26
 веб-браузер для 23
 текстовый редактор для 23
WordPress
 Codex 227
 администраторы 156, 157
 библиотека файлов 150–154, 167–169
 блог
 и страницы 161
 верхнее изображение 180–181
 метки и рубрики 148–149, 178–180
 обзор 137–139

создание 176–177
сортировка 147–149
форматы постов для медиафайлов 181–182
бэкенд 158, 160
виджеты 195, 222–225
визуальный редактор
 добавление изображений 167–171
 обзор 141–144
 создание постов 176–177
 создание страниц 163–167
«Внешний вид», панель 188–189, 205
вход и выход 140, 157–159, 160
импорт контента 226–227
информационные ресурсы 227
консоль 140–141, 157–158
меню навигации 161, 211–214
метки 148–149, 178–180
настройки сайта 215–219
«Настройки экрана», вкладка 214–215
обзор 134–136, 157
обновления 218, 224–225
панель администрирования 160
плагины 193–196, 218–221
план сайта 145–149, 162–163
расширенная настройка 217
смена хостинга при помощи панели

инструментов 226–227
собственный хостинг 205, 217, 218
ссылки, создание 165–166
страницы
 и посты 161
 медиафайлы, добавление 167–171
 обзор 139
 публикация 163, 172–173
 создание первой 163–167
 упорядочивание 145–147, 174–176
темы
 активная 160, 188, 205–206
 меню навигации, настройка 211–214
 настройка 192–193, 198–204, 208–211
 обзор 155, 157, 188–193
 основы 205–207
 премиальные 191–192, 205, 211
 расширенная настройка 217
 форматы постов 181–182
 Twenty Fourteen. См. Twenty Fourteen, тема WordPress
удаление содержимого 170, 183
управление контентом 183
фронтенд 159, 160
хостинг 26, 157, 242–243, 254–255
экспорт контента 226
WYSIWYG-инструменты 165

Психология

Умные книжки

Истории для души

Помощь в учебе

Творчество

Родителям

Игры и квесты



МИФ Подростки

Все книги
на одной странице:
mif.to/teen

Подписывайтесь на полезную
рассылку: книги, скидки и подарки
mif.to/d-letter



mif.podrostki

Справочное издание
Для среднего и старшего школьного возраста

Нейт Купер

КАК СОЗДАТЬ САЙТ **Комикс-путеводитель по HTML, CSS и WordPress**

С иллюстрациями Ким Джи

Руководитель редакции *Анастасия Троян*
Шеф-редактор *Юлия Петропавловская*
Ответственный редактор *Валерия Важнова*
Верстка *Ольга Булатова*
Дизайн обложки *Сергей Хозин*
Корректурa *Надежда Болотина, Олег Пономарев*

ООО «МАНН, ИВАНОВ И ФЕРБЕР»
www.mann-ivanov-ferber.ru
www.facebook.com/mifdetstvo
<http://www.vk.com/mifdetstvo>
<http://www.instagram.com/mifdetstvo>

НАУЧИТ ДЕЛАТЬ САЙТЫ С НУЛЯ



ДВА САМЫХ РАСПРОСТРАНЕННЫХ ЯЗЫКА РАЗМЕТКИ — HTML И CSS

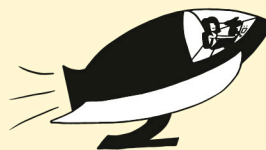
Ты хочешь сделать собственный сайт? Тогда открой эту книгу и отправляйся в увлекательное путешествие вместе с ее героями — веб-гуру, художницей Ким и ее песиком Тофу! Вместе вы найдете дорогу в дремучем лесу, сразитесь с драконами неверных ссылок и разберетесь, что за непонятные дела происходят в WordPress-сити, а заодно освоите два языка разметки, которые используются на всех без исключения сайтах, — HTML и CSS. Готовься к приключениям!

Ты узнаешь, как:

- создать красивый и удобный сайт;
- применять основные HTML-теги;
- задать оформление сайта с помощью CSS;
- использовать все возможности конструктора сайтов WordPress;
- избежать ошибок при выборе хостинга.

Книга подойдет детям от 10 лет и старше, а также взрослым, которые любят приключения и мечтают создать собственный сайт.

Автор книги Нейт Купер не только делает крутые сайты, но и передает свои знания ученикам. Он начал карьеру в Apple, а затем основал собственную компанию Simple Labs, которая проводит тренинги по работе с WordPress.



Детские книги на сайте
mann-ivanov-ferber.ru

[facebook.com/mifdetstvo](https://www.facebook.com/mifdetstvo)
vk.com/mifdetstvo
[instagram.com/mifdetstvo](https://www.instagram.com/mifdetstvo)



ISBN 978-5-00100-914-6



9 785001 009146 >