Содержание

Об авторе 13

Введение 15
- Особенности книги 16
- Исходные предположения 17
- Пиктограммы, используемые в книге 17
- Что дальше 18
- Ждем ваших отзывов! 18

Часть I. Подготовка к программированию 19

Глава 1. Знакомство с программированием 21
- Исходный код 22
  - Инструкции и операции 22
  - Написание исходного кода с помощью Angry Birds 23
- Задачи, решаемые собственными программами 24
  - Роль программного обеспечения в мировой экономике 25
  - Программирование на рабочем месте 28
  - На пути к богатству и славе 29
- Краткий обзор языков программирования 30
  - Языки программирования низкого и высокого уровней 31
  - Компиляция и интерпретация исходного кода 32
  - Веб-программирование 32
- Веб-приложения 33
  - Назначение и цель веб-приложения 33
  - По следам первопроходцев 34

Глава 2. Программирование веб-приложений 36
- Отображение веб-страниц на экране настольного компьютера и мобильного устройства 37
  - Анализ кода популярного сайта новостей 37
  - Принципы функционирования Интернета 41
  - Интерфейсная и прикладная части приложения 43
  - Веб-приложение и мобильное приложение 45
- Разработка веб-приложения 46
  - Знакомство с HTML, CSS и JavaScript 46
  - Расширение функциональных возможностей с помощью Python, Ruby и PHP 47
- Написание мобильных приложений 48
  - Разработка мобильных веб-приложений 50
  - Создание “родных” мобильных приложений 51

Глава 3. Карьера программиста 54
- Написание кода как производственный процесс 54
  - Анализ функциональных возможностей и планирование приложения 57
  - Разработка структуры приложения 58

Написание исходного кода 59
Отладка исходного кода 60
Подбор инструментов разработки 61
Автономная работа 61
Работа в Codecademy.com 62

Часть II. Создание статических и интерактивных веб-страниц 65

Глава 4. Знакомство с HTML 67
Назначение языка HTML 67
Структура HTML-файла 68
Элементы HTML-документа 69
Добавление атрибута в HTML-элемент 71
Основные части веб-страницы: заголовок, название и тело 72
Общие задачи и теги языка HTML 74
Выделение заголовков 76
Структурирование текста 77
Связывание информационных элементов между собой 78
Добавление рисунков 79
Оформление HTML-документа 81
Курсив, полужирный текст, подчеркивание и зачеркивание 81
Изменение уровня расположения текста (верхний и нижний индексы) 82
Создание первой веб-страницы средствами языка HTML 83

Глава 5. Расширенные возможности языка HTML 86
Организация данных на веб-странице 86
Списки данных 90
Упорядоченные и неупорядоченные списки 90
Вложенные списки 91
Заполнение таблиц данными 92
Базовая структура таблицы 93
Объединение ячеек в столбцах и строках 95
Выравнивание таблиц и ячеек 96
Заполнение формы 99
Функциональные особенности формы 100
Создание простой формы 101
Практические занятия по языку HTML 102

Глава 6. Стилизация документа с помощью CSS 104
Функциональные возможности CSS 105
Структура CSS 107
Выбор форматируемого элемента 107
Установка значения свойства 109
Изменение правил CSS для веб-сайта 109
Общие задачи и селекторы CSS 112
Основные параметры шрифта: размер, цвет, начертание и оформление 113
Настройка гиперссылок 117
Добавление фона и оформление графических изображений 119

Правильная стилизация документа 124
Добавление правил CSS в HTML-код 125
Создание первой веб-страницы 127

Глава 7. Расширенные возможности CSS 128
Детальное оформление элементов веб-страницы 129
Форматирование списков 129
Настройка таблиц 132
Форматирование отдельных элементов 135
Оформление родственных элементов 135
Именование HTML-элементов 139
Размещение и выравнивание элементов 141
Организация данных на странице 141
Заключение элементов в контейнер 143
Блочная модель 145
Позиционирование блоков 147
Написание сложного CSS-кода 150

Глава 8. Упрощение разработки с Twitter Bootstrap 151
Функциональные возможности Bootstrap 151
Установка Bootstrap 154
Настройка макета документа 155
Выравнивание содержимого на координатной сетке 155
Перетаскивание элементов макета 158
Заранее заданные шаблоны 159
Адаптация макета к выводу на смартфоне, планшете и настольном компьютере 160
Создание базовых элементов веб-страницы 162
Кнопка 163
Панели навигации 164
Значки 166
Создание начальной страницы с помощью Bootstrap 167

Глава 9. Сценарии JavaScript 168
Назначение JavaScript 168
Структура кода JavaScript 171
Точки с запятой, кавычки и скобки 171
Выполнение простых операций 172
Сохранение данных в переменных 172
Принятие решений в условных операторах 173
Методы обработки строковых и числовых данных 177
Извещение пользователей и запрос данных 180
Выделение инструкций в функции 180
Добавление кода JavaScript на веб-страницу 182
Написание первой программы на JavaScript 183
Прикладной программный интерфейс 184
Назначение программного интерфейса 184
Обмен данными без программного интерфейса 186

Выбор программного интерфейса для работы 188
Библиотеки JavaScript 189
jQuery 189
D3.js 190
Поиск видеороликов с помощью YouTube API 190

Часть III. Построение веб-приложения 193

Глава 10. Создание собственных приложений 195
Создание приложений с географической привязкой 195
Анализ задачи 196
Дальнейшие размышления 197
Основные этапы разработки 197
Анализ и планирование 199
Общие принципы разработки 200
Совместная работа с другими разработчиками 203
Веб-дизайнер 203
Разработчики пользовательских и серверных компонентов 206
Менеджер по продукту 208
Тестирование и контроль качества 209

Глава 11. Планирование первого веб-приложения 211
Производственные этапы 212
Анализ функциональных возможностей 212
Планирование функциональных возможностей приложения. Авторская версия 213
Оформление веб-приложения 216
Дизайн приложения для сети ресторанов McDuck 220
Источники программного кода 222
Создание кода приложения для поддержки сети ресторанов McDuck 225
Выполнение задач каждого производственного этапа 228

Глава 12. Написание кода и отладка приложения 232
Подготовка к написанию кода 232
Написание кода 233
Среда разработки 233
Заранее написанный код 234
Дальнейший код 238
Отладка приложения 241

Часть IV. Дальнейшее знакомство с программированием 243

Глава 13. Язык программирования Ruby 245
Возможности языка программирования Ruby 246
Структура Ruby 247
Принципы разработки приложений на языке Ruby 247
Синтаксис и выделение блоков исходного кода 249

Типичные задачи и операции, возлагаемые на Ruby 250
Определение типов данных и объявление переменных 250
Математические вычисления 251
Строковые данные и специальные символы 252
Условные конструкции: `if`, `elsif`, `else` 253
Ввод и вывод данных 255
Управление строковыми данными 256
Управление строковыми данными: методы `.upcase`, `.downcase`, `.strip` 257
Вставка в строку значений переменных (оператор #) 257
Создание простой формы изменения форматирования текста 258

Глава 14. Знакомство с языком Python 259
Функциональные возможности Python 259
Структура языка программирования Python 261
Принципы разработки на языке Python 261
Синтаксис и выделение блоков исходного кода 262
Типичные задачи и операции, возлагаемые на Python 264
Определение типов данных и назначение переменных 264
Математические вычисления 265
Строковые данные и специальные символы 267
Условные конструкции `if`, `elif`, `else` 268
Ввод и вывод данных 270
Управление строковыми данными 271
Приведение строки к необходимому формату: методы `upper()`, `lower()`, `capitalize()` и `strip()` 271
Вставка в строку значения переменной с помощью оператора `%` 272
Создание простого приложения на языке Python 272

Часть V. Великолепные десятки 275

Глава 15. Десять полезных ресурсов для программистов 277
Обучение разработке веб-сайтов 277
Codecademy 277
Coursera и Udacity 278
Hackdesign.org 279
Code.org 280
Справочные ресурсы 280
htmlbook.ru 281
Mozilla Developer Network 282
Stack Overflow 282
Новости технологий и сообщества разработчиков 283
Хабрахабр 283
Тостер 284
Meetup 285

Глава 16. Десять советов начинающим программистам 286
Выбор языка программирования 286
Мотивация 287
Разделение производственной задачи на этапы 288
Расстановка акцентов 289
Google — лучший друг разработчика 290
Отладка кода 291
Преодоление трудностей 292
Сбор отзывов 292
Улучшение приложения 293
Признавайте ошибки и делитесь успехами с другими 294
Предметный указатель 295

Глава 2

Программирование веб-приложений

В этой главе...

- Отображение исходного кода ежедневно просматриваемых веб-страниц
- Языки программирования, используемые для разработки веб-сайтов
- Написание приложений для мобильных устройств

Мысль о том, что в комнате студенческого общежития можно создать то, что станет частью жизни многих миллионов других людей, кажется невероятной. Но она вдохновляет.

Марк Цукерберг

Разработка веб-приложений открывает перед вами невероятные возможности по вовлечению в свой проект аудитории просто немыслимых масштабов. Всего спустя четыре года после запуска в 2004 году социальная сеть Facebook насчитывала сто миллионов пользователей, а на начало 2012 года их количество превысило миллиард. В качестве сравнения стоит заметить, что устанавливаемому в настольные компьютеры программному обеспечению требуются многие годы, чтобы оно распространялось всего миллионным тиражом. На сегодняшний день мобильные устройства как никогда ранее зависимы от функциональных возможностей веб-приложений. Статистика утверждает, что ежегодно во всем мире продается около 300 миллионов настольных компьютеров, но за этот же срок продается более 2 миллиардов смартфонов и других мобильных устройств, и их количество продолжает неуклонно увеличиваться. В этой главе вы узнаете о том, как и в каком виде содержимое веб-сайтов отображается на экранах компьютеров и мобильных устройств. Вы познакомитесь с языками программирования, на которых создаются веб-сайты, и узнаете принципы разработки приложений, запускаемых в браузере.

Отображение веб-страниц на экране настольного компьютера и мобильного устройства

На мониторы настольных компьютеров и экраны мобильных устройств содержимое веб-страниц выводится с помощью специального программного обеспечения, известного как *браузер*. К наиболее распространенным современным браузерам относят Google Chrome, Mozilla Firefox (продолжатель Netscape Navigator), Microsoft Internet Explorer и Apple Safari. До настоящего момента все вы взаимодействовали с содержимым веб-сайта как обычные пользователи, поэтому для получения необходимой информации в точности следовали правилам, предложенным его разработчиком, щелкая на всем, что только возможно. Для того чтобы стать разработчиком сайтов и успешным программистом веб-приложений, вам нужно познакомиться с их структурой, а также понять, как программный код предопределяет их функциональные возможности, внешний вид и содержимое.

Анализ кода популярного сайта новостей

На каком сайте вы обычно смотрите новости? Следуя приведенным ниже инструкциям, вы сможете не только просмотреть, но и изменить исходный код, применяемый для отображения информации на веб-страницах сайта. (Не волнуйтесь, следуя приведенным ниже инструкциям, вы не нарушите ничьих и никаких прав.)

Несмотря на то что для просмотра исходного кода, лежащего в основе любого веб-сайта, подходит практически любой современный браузер, приведенные ниже инструкции в точности справедливы только для Google Chrome. Установить его очень просто, достаточно посетить сайт производителя по следующему адресу:

`www.google.com/chrome/browser`

Для просмотра содержимого популярного сайта новостей выполните описанные ниже действия.

1. Отобразите в окне браузера Chrome любимый новостной сайт. (Для примера мы любезно воспользуется сайтом `http://postnauka.ru`.)
2. Наведите указатель мыши на любой статический заголовок начальной страницы сайта и щелкните на нем один раз правой кнопкой мыши, чтобы отобразить контекстное меню. Теперь щелкните левой кнопкой мыши на команде Просмотр кода элемента (Inspect Element), как показано на рис. 2.1. Браузер немедленно свяжется с веб-сайтом и отобразит в нижней части окна исходный код, лежащий в основе страницы, которую вы видите в текущий момент на экране.

Рис. 2.1. Щелкните правой кнопкой мыши на заголовке и выполните команду Просмотр кода элемента

В компьютерах Macintosh вместо щелчка правой кнопкой мыши нажмите и удерживайте нажатой клавишу <Control>, а затем щелкните на необходимой команде.

Панель инструментов разработчика (Developer Tools) добавляется в нижнюю или правую часть окна браузера. На этой панели показан исходный код, используемый для отображения на странице всех ее элементов и содержимого. Синим цветом выделяется код, применяемый для вывода на странице заголовка, на который вы навели указатель мыши перед отображением в окне панели (рис. 2.2.)

Взгляните на левый край строки, содержащей исходный код текущего элемента страницы. Если вы видите стрелку, направленную вправо, то щелкните на ней один раз, чтобы раскрыть вложенный блок кода.

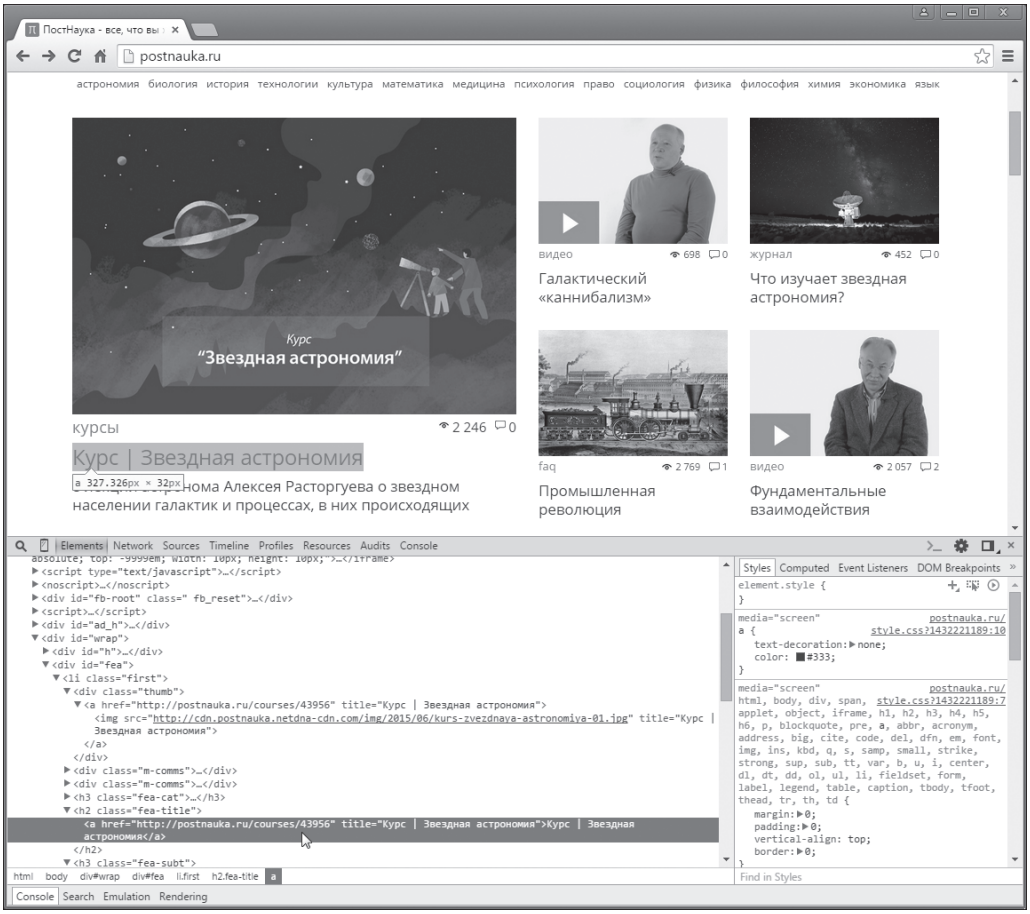

Рис. 2.2. Исходный код, выделенный цветом, используется для отображения на странице заголовка, на который наведен указатель мыши перед добавления в окно браузера панели разработчика

3. Исследуйте выделенную серым цветом часть исходного кода в поиске текста заголовка, отображенного на странице. Дважды щелкните на соответствующем коде. Как показано на рис. 2.3, тем самым вы сможете изменить заголовок.

   Будьте внимательны и не щелкайте на тексте, начинающемся с ключевого слова `http`, с которого начинается ссылка на заголовок. Щелчок на ссылке, привязанной к заголовку, приводит к открытию нового окна или вкладки с последующей загрузкой страницы, на которую указывает эта ссылка.

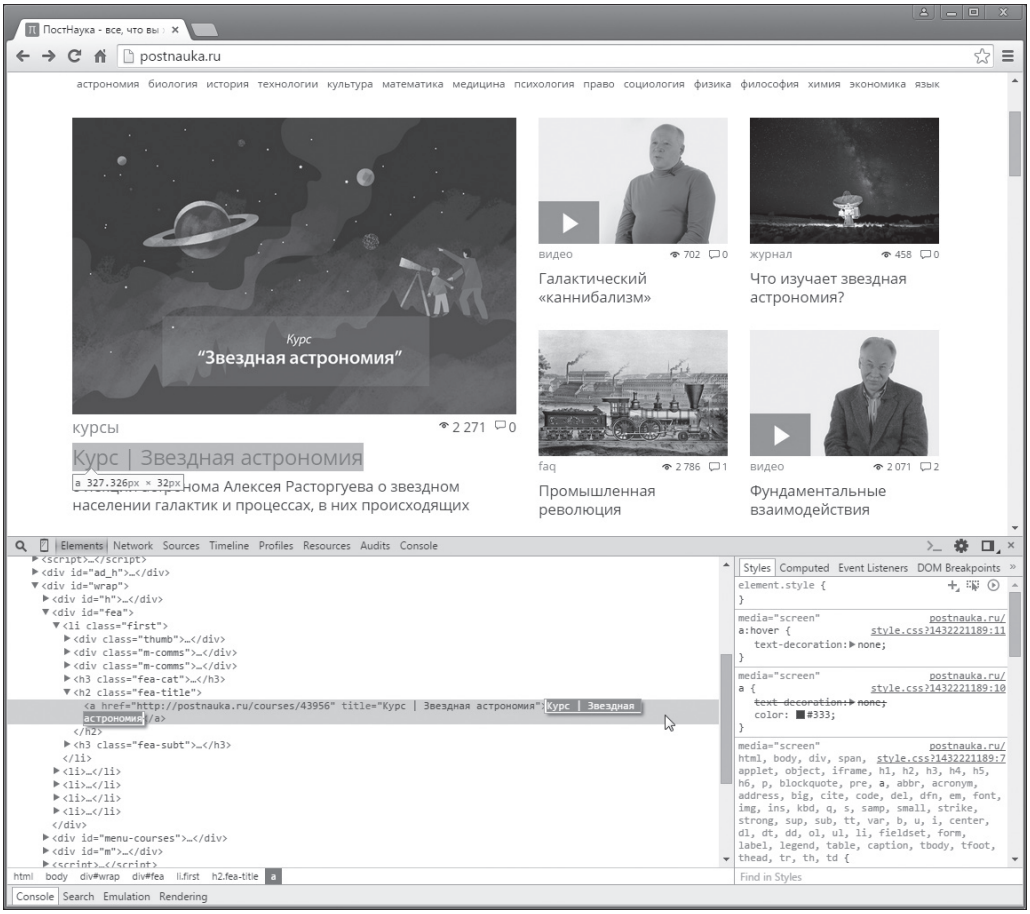

Рис. 2.3. Двойной щелчок на тексте заголовка позволяет отредактировать его

4. Введите новый текст заголовка (например, свое имя) и нажмите клавишу <Enter>.

   Новый заголовок будет отображаться на действующей веб-странице (рис. 2.4). Вот он, момент славы!

Если у вас не получилось изменить заголовок, следуя приведенным выше инструкциям, то посетите сайт `http://goggles.webmaker.org`, на котором найдете более подробное описание необходимых действий. На нем представлены исчерпывающие рекомендации по редактированию исходного кода страницы, загруженной в браузер. Используя приведенные инструкции как пособие, вы узнаете, как изменить исходный код любой страницы, загружаемой через Интернет. Щелкнув на желтой кнопке Активируйте Рентген-очки (Activate X-Ray Goggles), вы сможете изучить и отредактировать исходный код текущей страницы сайта `webmaker.org`. Попробуйте заново изменить код страницы вашего любимого новостного сайта, воспользовавшись полученными инструкциями.

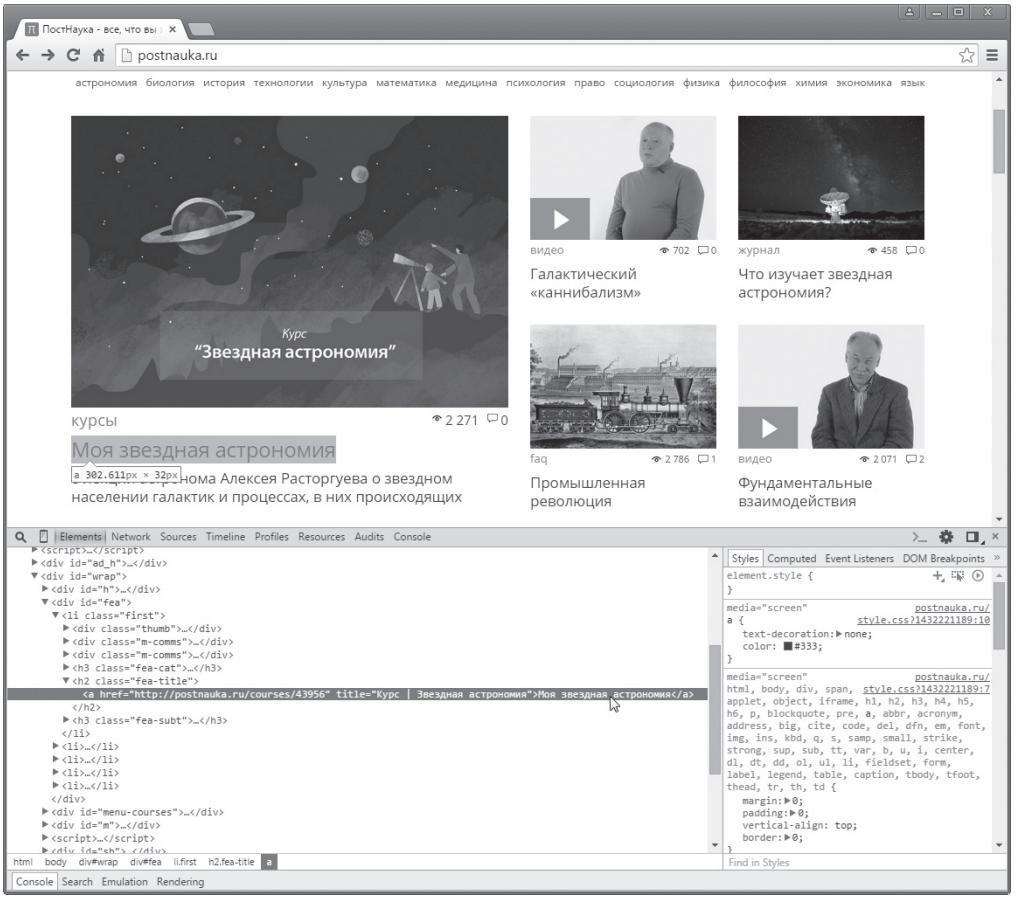

Рис. 2.4. Успешная замена заголовка веб-страницы в браузере не требует с вашей стороны больших усилий

Если вы выполнили действия, описанные в приведенных выше инструкциях, и у вас получилось изменить произвольный заголовок веб-страницы, то не спешите радоваться больше положенного. Насладившись минутой славы, перезагрузите страницу, и вы увидите, что на экране отображается исходный заголовок. Как же так? Разве вы только что не изменили исходный код страницы? Почему же после обновления страница приобрела прежний вид?

Чтобы ответить на все эти вопросы, вам сначала нужно понять, как веб-страницы из Интернета попадают в ваш компьютер.

Принципы функционирования Интернета

При вводе URL-адреса в строке адреса браузера, например `http://youtube.com`, перед началом загрузки данных веб-страницы происходит несколько важных событий, а также выполняются действия, скрытые от пользователей (рис. 2.5).

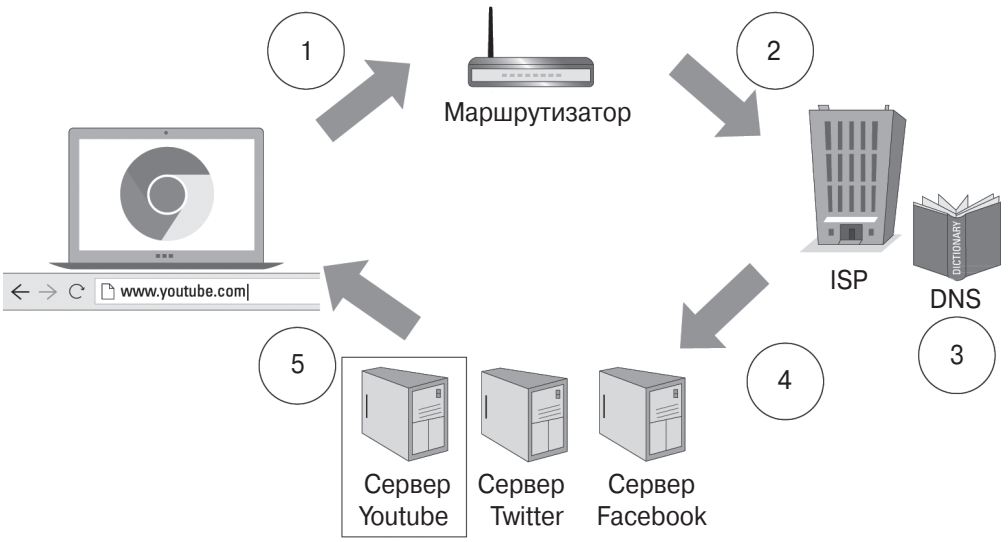

Рис. 2.5. Схема доставки страницы с веб-сайта в браузер

1. Ваш компьютер передает запрос на загрузку веб-страницы маршрутизатору. Маршрутизатор отвечает за предоставление доступа к Интернету всем подключенным к нему устройствам, находящимся в вашем доме или офисе.
2. Маршрутизатор передает ваш запрос провайдеру (ISP — Internet Service Provider), обеспечивающему установку соединения с Интернетом. В зависимости от региона провайдеров может быть несколько с разными пакетами предоставляемых услуг.
3. На стороне провайдера символы и слова, введенные в поле адреса браузера (URL-адрес, например `http://youtube.com`), преобразуются в числовой адрес, больше известный как IP-адрес. В общем случае IP-адрес представляет собой четыре трехзначных числа, разделенных точками (например, 192.168.143.213). Подобно любому физическому адресу, такой числовой адрес уникальный, т.е. свой для каждого устройства или компьютера, подключенного к Интернету. У вашего провайдера имеется специальный цифровой инструмент (работающий во многом подобно телефонной книге), известный как *DNS* (Domain Name System — система доменных имен), который преобразует текстовые URL-адреса, вводимые в браузере, в IP-адреса.
4. Определив точный IP-адрес, ваш провайдер устанавливает положение сервера в Интернете, к которому необходимо перенаправить запрос. Теперь в такой запрос также включается IP-адрес вашего компьютера.
5. В итоге запрос поступает на сервер, на котором находится веб-сайт, содержащий необходимую вам информацию, и сервер отправляет копию запрашиваемой страницы в целевой браузер.
6. После получения данных браузеру остается только отобразить содержимое страницы на экране.

При изменении исходного кода страницы на панели инструментов разработчика (в окне браузера) вы работаете только с копией веб-страницы, переданной сервером в ваш компьютер. Изменения будут отображаться только до момента перезагрузки страницы, которая заключается в повторной подаче запроса на сервер с выполнением всех вышеописанных действий (пп. 1–6). Как только браузер получит свежую копию запрашиваемой веб-страницы, ее локальная копия, измененная вами, будет заменена исходным вариантом.

Вы уже могли слышать о специальном программном обеспечении, известном под названием *блокировщик рекламы*. Приложения, отвечающие за блокировку рекламы, редактируют локальную копию исходного кода, передаваемого с веб-сервера (во многом подобно тому, как это делали мы с вами), чтобы удалить в нем упоминание о рекламе. Блокировка рекламы — это противоречивое действие, поскольку прибыль от предоставления рекламы позволяет владельцам веб-сайтов снизить расходы на их содержание и поддержку. Если блокировщики рекламы станут использоваться повсеместно, то прибыль от рекламы сильно снизится, и бо́льшая часть операционных расходов на содержание веб-сайтов будет возложена на их владельцев, что немедленно поднимет вопрос о предоставлении информации в Интернете за отдельную плату.

Интерфейсная и прикладная части приложения

Теперь, когда вы знаете о том, как браузер получает данные с веб-сервера, углубимся в суть процесса и узнаем, как структурно организован веб-сайт. Как показано на рис. 2.6, исходный код самих веб-сайтов и приложений, применяемых для управления данными на них, функционально обеспечивается несколькими уровнями интеграции.

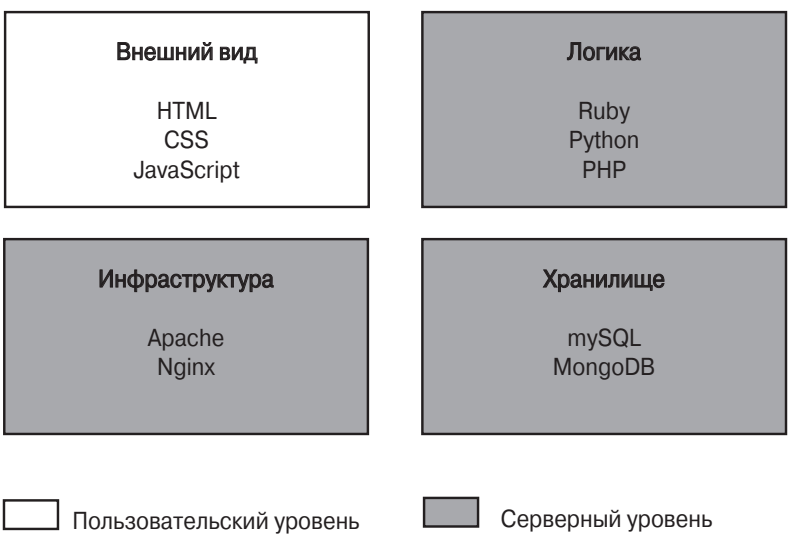

Рис. 2.6. Любой веб-сайт состоит из четырех структурных уровней

- ✓ **Вид.** Внешний вид — это то, что пользователи видят на экране мониторов настольных компьютеров и мобильных устройств. Внешний вид данных определяется применяемыми стилями, в которых среди всего прочего устанавливаются типы, размеры и начертания шрифтов, а также размеры загружаемых изображений. Этот уровень веб-приложения называют *интерфейсным* или *пользовательским*, а реализуется он посредством таких инструментов, как HTML, CSS и JavaScript.
- ✓ **Логика.** На логическом уровне устанавливается поведение всего, что отображается на страницах веб-сайта. Например, жители Санкт-Петербурга и Стамбула при посещении погодного веб-сайта должны видеть на экране разные прогнозы — каждый для своего региона, но никак не для чужого. Этот структурный уровень веб-приложения называют *серверным*, и создается он с использованием языков программирования Ruby, Python и PHP. Они позволяют изменять исходный код, написанный на HTML, CSS и JavaScript, а потому влияют на информацию, отображаемую на экране в необходимый момент времени.
- ✓ **Хранилище.** В хранилище сохраняются любые данные, генерируемые как программным образом, так и конечными пользователями. Содержимое, создаваемое на стороне пользователя (например, настройки и учетные сведения), сохраняются с целью дальнейшего использования в процессе функционирования приложения. Являясь частью серверного решения, подобная информация хранится в специальных базах данных, таких как MongoDB и MySQL.
- ✓ **Инфраструктура.** Инфраструктурная часть веб-сайта отвечает за передачу данных с сервера в клиентский компьютер или мобильное устройство. Если инфраструктурная часть выполнена безупречно, то никто не замечает этого; заметными становятся только ошибки, возникающие при передаче данных, или регулярные сбои, делающие просмотр информации некомфортным и затруднительным. Чаще всего такие ситуации наблюдаются при чрезмерно высокой посещаемости сайта, возникающей в случае форс-мажорных обстоятельств (например, на президентских выборах, рейтинговых спортивных соревнованиях или во время стихийных бедствий).

Как правило, каждый разработчик специализируется на разработке одного или нескольких структурных уровней веб-сайтов, но не всех сразу. Например, каждый отдельно взятый программист идеально разбирается в создании интерфейсных решений или разработке баз данных, а с остальными уровнями знаком только в рамках своей основной деятельности. Разработчики веб-сайтов, как и любые другие специалисты, обладают определенной квалификацией и имеют свою специализацию; вне сферы своей деятельности их познания весьма расплывчаты. Для аналогии лучше всего подходит кинематограф, в котором операторская и актерская работа требует своих, отличных от других, навыков, но при этом актер не сможет провести на должном уровне съемку, а оператор не сыграет даже самую простую роль. Но при ответственном выполнении своих обязанностей всеми участниками процесса на экраны выходит прекрасный фильм.

Очень немногие разработчики программного обеспечения достигают той вершины мастерства, с высоты которой создание веб-приложений на всех его производственных уровнях не вызывает для них затруднений. Эти разработчики гордо называются *универсальными*. Чаще всего такие специалисты востребованы в небольших компаниях, а вот крупные производители программного обеспечения больше нуждаются в узкоспециализированных кадрах, обладающих только ограниченным количеством навыков.

Веб-приложение и мобильное приложение

К *веб-приложениям* относятся сайты, которые вы открываете в окне браузера, ежедневно просматривая Интернет в поисках интересующей вас информации. Веб-сайты, оптимизированные под работу на мобильных устройствах, таких как смартфоны и планшеты, называются *мобильными приложениями*. В противоположность им "родные" мобильные приложения выполняются без участия браузера. Напротив, загружаясь из магазина приложений, подобного Apple App Store или Google Play, они предназначены для выполнения только в определенных типах устройств, например iPhone или планшетах, работающих под управлением Android. Исторически сложилось так, что настольные компьютеры появились раньше планшетов и мобильных устройств, хотя сегодня они продаются почти в одинаковых количествах.

- ✓ В 2014 году количество пользователей мобильных устройств превысило количество людей, работающих за настольными компьютерами. Разрыв продолжает уверенно увеличиваться, что демонстрирует график на рис. 2.7.
- ✓ Пользователи мобильных устройств более 80% времени проводят за работой с "родными" мобильными приложениями и только 20% времени посвящают мобильным веб-приложениям, запускаемым в браузере.

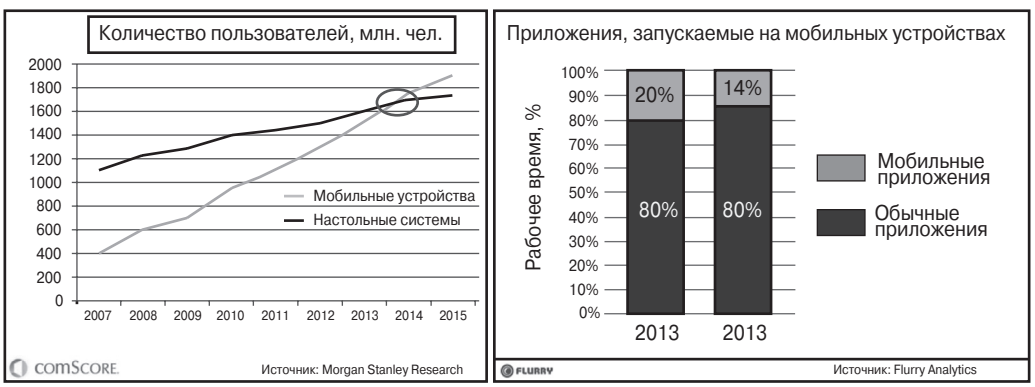

Рис. 2.7. Объемы продаж мобильных устройств уже несколько лет подряд растут быстрее, чем настольных компьютеров

Популярность мобильных устройств за последние десять лет увеличилась настолько, что многие компании, производящие программное обеспечение, ориентируются в первую очередь на выпуск мобильных приложений, откладывая разработку версий для настольных компьютеров “на потом”. Такие невероятно популярные проекты, как WhatsApp и Instagram, исходно создавались как загружаемые мобильные приложения, которые обладают большей функциональностью, чем их аналоги, представленные веб-приложениями на соответствующих сайтах в Интернете.

Разработка веб-приложения

Веб-приложения создаются проще, чем мобильные приложения, для их разработки и тестирования не требуется дополнительное программное обеспечение, а запускаются они на всех типах пользовательских компьютерных устройств, включая настольные системы, ноутбуки, планшеты и смартфоны. Несмотря на то что мобильные приложения решают большое количество повседневных, ставших привычными, задач (например, отправка электронной почты), некоторые действия все же проще выполнять с помощью веб-приложений. В частности, планирование путешествия — это задача, больше подходящая для веб-приложений, поскольку в процессе ее решения вам следует быть в курсе цен на перелеты, номера в отелях, а также аренду автомобилей в самых разных странах мира. Добавьте к этому еще несколько окон — календарь, ввод персональных данных и проведение оплаты, — и вы поймете, что с помощью мобильного приложения выполнить все необходимые действия по планированию отпуска будет весьма затруднительно.

В следующих разделах мы рассмотрим основные языки, которые используются для написания исходного кода веб-приложений: HTML (Hypertext Markup Language — язык гипертекстовой разметки), CSS (Cascading Style Sheets — каскадные таблицы стилей) и JavaScript. Дополнительные элементы управления и пользовательские инструменты добавляются на веб-сайты с помощью таких языков программирования, как Python, Ruby и PHP.

Знакомство с HTML, CSS и JavaScript

Простые веб-сайты, подобные показанным на рис. 2.8, создаются с помощью языков HTML, CSS и JavaScript. Язык HTML применяется для правильного размещения на странице текста, CSS — для стилизации текста, а JavaScript — для добавления на страницу интерактивных элементов управления. В качестве примера интерактивного элемента управления достаточно привести кнопку Поделиться в Твиттер или Facebook, позволяющую поделиться информацией с другими участниками социальной сети и обновить сведения о количестве пользователей, которые уже поделились ею с другими. Сайты преимущественно хранят статическую, мало изменяемую информацию, которая обычно создается с помощью трех основных языков программирования. Детально с возможностями каждого из них вы познакомитесь в следующих главах.

Расширение функциональных возможностей с помощью Python, Ruby и PHP

Для создания веб-сайтов, снабженных расширенными функциональными возможностям (например, средствами регистрации пользователей, инструментами загрузки файлов и электронной торговли), вам понадобится прибегнуть к языкам программирования, обладающим соответствующим инструментарием. Несмотря на то что Python, Ruby и PHP далеко не единственный возможный вариант, они применяются для описанных целей чаще других. Популярность делает их наиболее часто применяемыми при разработке веб-приложений, что предполагает существование огромного сообщества программистов, обменивающихся опытом и готовыми программными решениями, которые впоследствии используются в большинстве стандартных приложений.

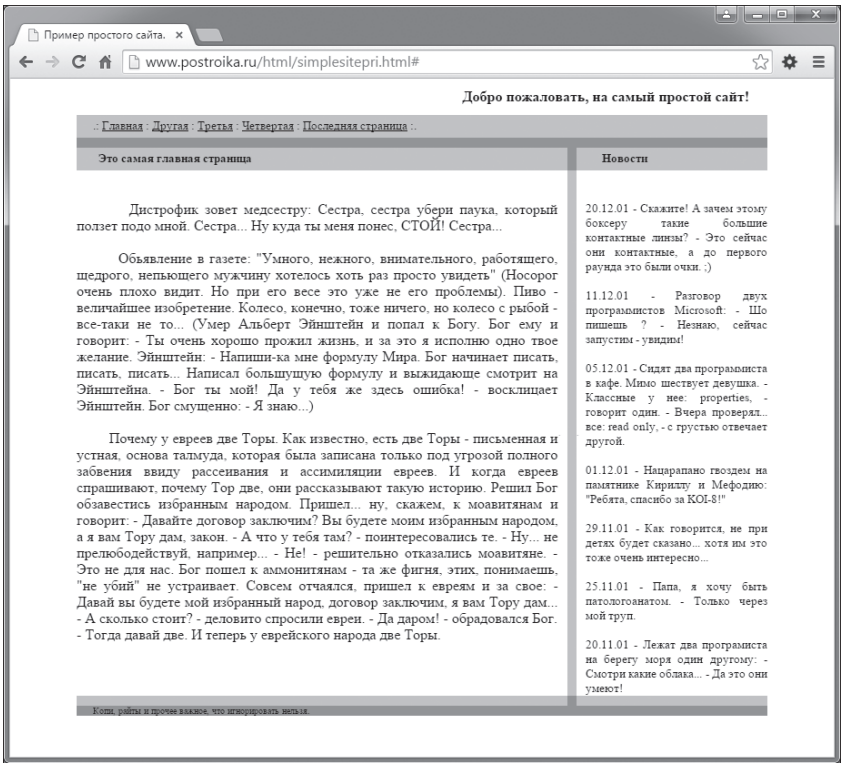

Рис. 2.8. Веб-сайт, созданный с помощью только HTML, CSS и JavaScript

Каждый из упомянутых языков программирования не в последнюю очередь стал популярным благодаря существованию хорошо задокументированных базовых средств разработки. *Базовое средство разработки*, или *фреймворк*, представляет собой набор стандартных компонентов, например, средств управления учетными записями пользователей или инструментов аутентификации, настолько часто используемых в новых проектах, что повторная их разработка, отладка, тестирование с нуля, а также последующее добавление на веб-сайт лишены здравого смысла. Базовые средства

разработки в наборе инструментов программиста играют такую же роль, как шаблоны в любом текстовом процессоре. Вы можете самостоятельно разработать дизайн резюме, поздравительной открытки или календаря, но намного удобнее и быстрее воспользоваться одним из встроенных шаблонов для каждого из упомянутых документов, не обременяя себя работой, уже проделанной другими людьми. К базовым средствам веб-разработки можно отнести следующие инструменты:

- ✓ Django и Flask для Python;
- ✓ Rails и Sinatra для Ruby;
- ✓ Zend и Laravel для PHP.

Написание мобильных приложений

На сегодняшний день создание мобильных приложений популярно как никогда ранее, и не в последнюю очередь благодаря высокой прибыльности. Так, например, прибыль от таких сервисов, как WhatsApp и Instagram, превысила миллиард долларов, а компании, специализирующиеся на создании мобильных приложений, например Rovio (создатели эпической Angry Birds) и King Digital (придумавшие Candy Crush), ежегодно получают прибыль в несколько десятков миллионов долларов.

При разработке программного обеспечения перед вами открываются следующие очевидные перспективы.

- ✓ Создать мобильное веб-приложение, написанное на HTML, CSS и JavaScript.
- ✓ Написать "родное" мобильное приложение с помощью языка программирования, характерного для выбранной платформы. Например, программное обеспечение для устройств компании Apple пишется на Objective-C или Swift, а приложения, рассчитанные на устройства под управлением Android, создаются на языке программирования Java.

Выбор в пользу одного из предложенных вариантов может показаться вам очевидным, но не спешите с окончательным решением. Примите во внимание такие факторы.

- ✓ Компаниям, создающим мобильные веб-приложения, необходимо удостовериться в том, что их детище запускается в большинстве современных браузеров, рассчитано на вывод информации на экраны самых разных размеров, а также поддерживается всеми известными брендами современности, такими как Apple, Samsung, RIM и Microsoft. Как нетрудно догадаться, существует несколько тысяч всевозможных комбинаций оборудования и операционных платформ, которым должно удовлетворять создаваемое мобильное приложение, что не может не сказаться на сложности выполняемых работ и длительности тестирования проекта перед выходом на рынок. "Родные" мобильные приложения запускаются только на одной платформе, что значительно упрощает их отладку и тестирование.

- ✓ Несмотря на запуск только на одной платформе, создание “родных” мобильных приложений стоит дороже и занимает больше времени, чем создание мобильных веб-приложений.
- ✓ Некоторые производители программного обеспечения утверждают, что в мобильных веб-приложениях чаще возникают неполадки, а загружаются они в устройство дольше, чем “родные” мобильные приложения.
- ✓ Как упоминалось выше, в среднем пользователи больше времени проводят за использованием “родных” мобильных приложений, чем веб-приложений, запускаемых в браузере.
- ✓ “Родные” мобильные приложения распространяются через специальный электронный магазин программного обеспечения, что требует специальной их регистрации у владельца сервиса. Мобильные веб-приложения запускаются непосредственно в окне браузера и не требуют наличия специальных разрешений. Например, компания Apple внедрила весьма строгую систему допуска приложений в свой магазин App Store, в которой каждое новое приложение получает разрешение на распространение через сервис в течение целых шести дней. Требования компании Google, выдвигаемые к приложениям, намного мягче: они добавляются в магазин в течение двух часов с момента подтверждения регистрации.

Ярчайший пример запрета на публикацию приложения в магазине случился на фоне традиционного противостояния двух крупнейших IT-гигантов. Компания Apple запретила компании Google включить в свой магазин приложение Google Voice, поскольку его функциональные возможности во многом повторяют таковые в телефонах производства Apple. Ответ Google не заставил себя долго ждать, и вскоре она выпустила мобильное веб-приложение, включающее все спорные функции и запускаемое в браузере, — в подобной ситуации Apple не осталось ничего другого, как смириться со своей участью.

Принимая окончательное решение о выборе типа приложения, в первую очередь оценивайте его сложность. Простые приложения, например, отвечающие за создание рабочих расписаний и меню, проще всего реализовать в виде браузерных решений, а вот сложные проекты, подобные мессенджерам и социальным сетям, вне всяких сомнений, относятся к “родным” мобильным приложениям. Даже именитые компании, давно занимающиеся разработкой программного обеспечения, часто не могут сделать однозначный выбор в пользу одного из вариантов. Пример тому — Facebook и LinkedIn, исходно созданные как веб-приложения. Спустя некоторое время политика их продвижения на рынке была пересмотрена, и они были дополнены известными вам “родными” мобильными приложениями. Окончательное решение в пользу одного из вариантов часто принимается на основе второстепенных факторов, таких как производительность, управление памятью, доступность инструментов разработки и т.п.

Разработка мобильных веб-приложений

Несмотря на возможность просмотра веб-сайтов в любых браузерах, запускаемых в мобильных устройствах, без специальной оптимизации их вид в большинстве случаев будет вызывать только разочарование (рис. 2.9). Как видите, только оптимизированные под мобильные устройства веб-сайты содержат сбалансированные шрифты, масштабируемые изображения и вертикальную разбивку, более подходящую для небольших вертикально ориентированных экранов.

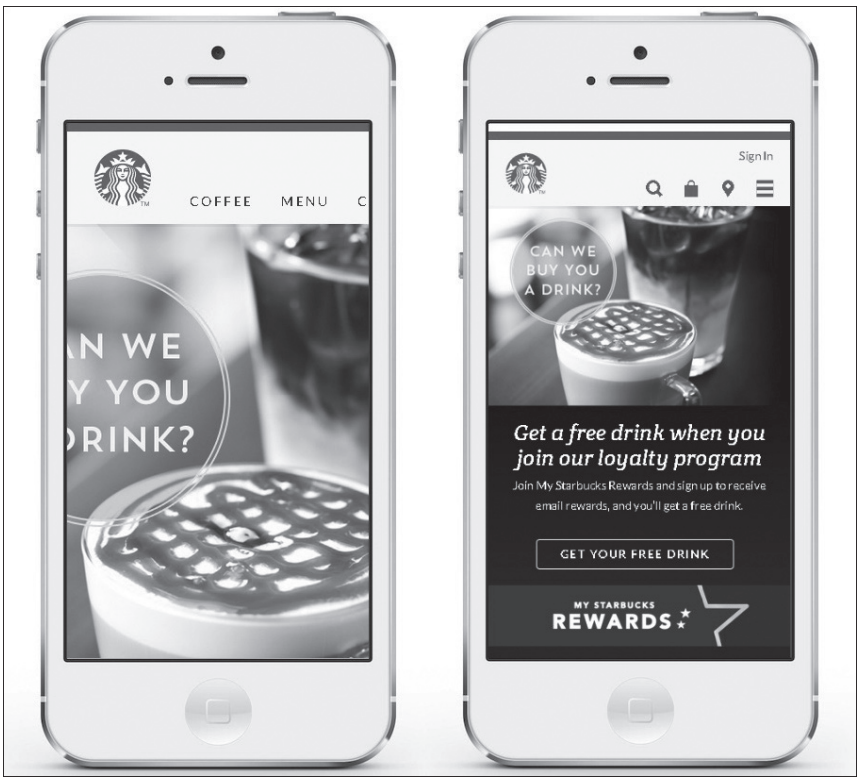

Рис. 2.9. Не оптимизированный для мобильных устройств сайт `www.starbucks.com` *(слева). Он же, но после проведения оптимизации (справа)*

Вы уже знаете, что написание мобильных веб-приложений требует использования языков программирования HTML, CSS и JavaScript. С помощью технологии CSS устанавливается внешний вид веб-сайта, который отображается в устройствах любых типов, оснащенных экранами разной ширины. Для экранов небольшой ширины, характерных для смартфонов, больше подходит *вертикальная разбивка* (vertically-based layout), а для широких экранов, которыми снабжаются планшеты, — *горизонтальная разбивка* (horizontally-based layout). Но поскольку веб-приложения запускаются в браузере, а не устанавливаются в конечное устройство, их нельзя снабдить функциями отправки извещающих уведомлений, выполнения в фоновом режиме при сворачивании, а также взаимодействия с другими приложениями, запускаемыми в устройстве.

Имея достаточно знаний, вы можете создавать веб-приложения с нуля, воспользовавшись уже упомянутыми выше средствами HTML, CSS и JavaScript. Но, чтобы сэкономить рабочее время и эффективнее распорядиться всеми имеющимися в вашем распоряжении ресурсами, обратитесь к базовым средствам разработки, которые представляют собой уже написанные кем-то блоки исходного кода. Чаще всего они представляют набор общих компонентов, которые применяются разработчиками в подавляющем количестве проектов, не только на этапе написания приложения, но и отладки, а также последующего тестирования. К таким средствам относится набор инструментов разработчика Twitter Bootstrap, с которым вы познакомитесь в главе 8.

Создание “родных” мобильных приложений

“Родные” мобильные приложения производительнее, надежнее и имеют более привлекательный внешний вид, чем мобильные веб-приложения (рис. 2.10). Написанные на языке программирования Java для устройств под управлением Android и на языке Objective-C или Swift для устройств компании Apple (с операционной системой iOS), “родные” приложения перед запуском нужно загрузить из электронного магазина, что требует специального одобрения его владельца. Главное преимущество такого магазина в централизованном распространении программного обеспечения, поэтому добавление в него приложения собственного написания — всегда хорошая идея. К тому же, поскольку “родные” мобильные приложения устанавливаются в целевое устройство, для обеспечения нормальной работоспособности им чаще всего не нужно подключение к Интернету. Наконец, самое главное преимущество устанавливаемых приложений перед запускаемыми в браузере — бо́льшая распространенность, и вряд ли эта ситуация поменяется в ближайшем будущем.

“Родные” мобильные приложения обеспечивают нормальную работоспособность даже при сворачивании окна и переходе в фоновый режим. Это позволяет им, например, принимать извещения и обмениваться данными с другими приложениями в течение всего времени работы устройства, независимо от его состояния. К тому же графически насыщенные программы, такие как игры, как правило, представляются исключительно “родным” программным обеспечением. Подведя итог, можно утверждать, что такие приложения производительнее и имеют более широкие функциональные возможности, но на их разработку уходит намного больше времени, а стоимость таких программ несказанно выше, чем веб-приложений.

Существует альтернативный способ создания “родного” мобильного приложения — заключить вручную написанный на языках HTML, CSS и JavaScript код в специальный контейнер или “оболочку” и впоследствии запускать мобильное приложение из него. Один из самых известных контейнеров на сегодняшний день — это PhoneGap. Этот невероятно полезный инструмент распознает отдельные инструкции JavaScript, что позволяет ему получать доступ к функциям устройства, которые обычно невозможно реализовать в мобильных веб-приложениях. Создав всего одну версию приложения, вы сможете запустить ее на нескольких платформах — Apple, Android, Blackberry и Windows Phone, воспользовавшись для этого девятью предлагаемыми PhoneGap контейнерами. Преимущество такого решения очевидно: достаточно написать приложение

всего один раз, и его версии под другие платформы будут создаваться автоматически и с минимальными временны*ы*ми затратами.

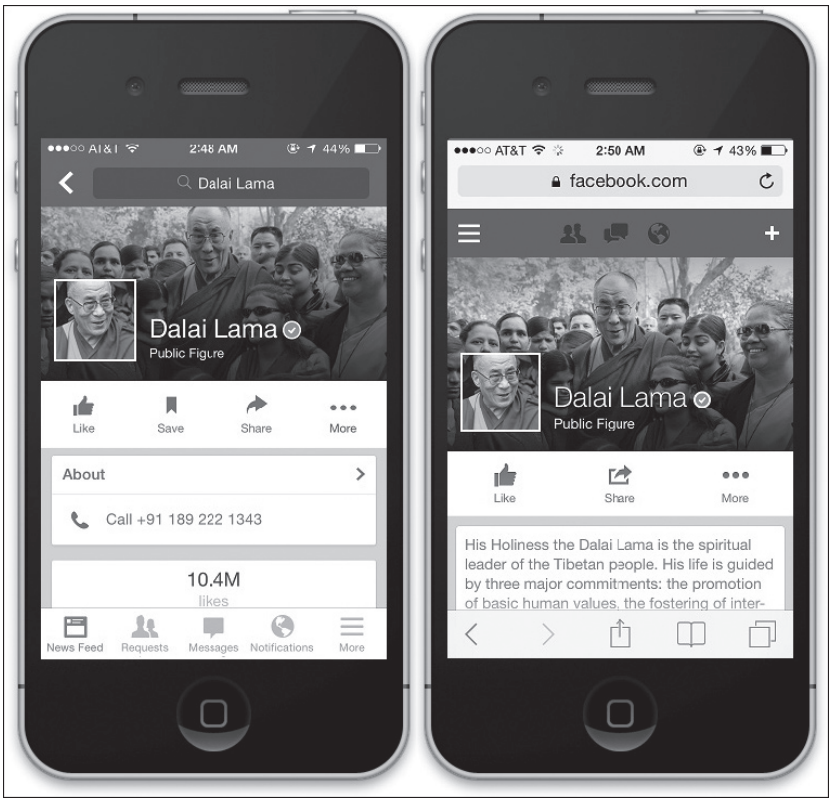

Рис. 2.10. Социальная сеть `www.facebook.com`*, представленная "родным" мобильным приложением (слева). Она же, но в виде мобильного веб-приложения (справа)*

Представьте себе, что, умея играть на рояле, вы решили научиться выполнять партии на виолончели. Один из способов решения этой задачи заключается в приобретении виолончели и обучении игре на ней. Второй способ более хитроумный: вам необходимо приобрести клавишный синтезатор, настроить его на тональность виолончели и сыграть на нем, как на рояле. Такой подход к игре на музыкальных инструментах подобен альтернативному программированию мобильных приложений, достаточно лишь сопоставить рояль с HTML, CSS и JavaScript, виолончель — с разрабатываемым программным обеспечением для определенной операционной системы, а клавишный синтезатор — с программным контейнером, подобным PhoneGap. Подобно тому, как клавишный синтезатор умеет воспроизводить мелодии в тональности виолончели, скрипки или гитары, "оболочка" PhoneGap позволяет запускать ваше приложение в устройствах Apple, Android и на других платформах.

Разработка приложений на других языках программирования (C, Java и др.)

Вас может заинтересовать, почему мобильные приложения разрабатываются только на нескольких языках программирования, несмотря на то, что их существует огромное количество. Все дело в том, что новые языки программирования создаются только тогда, когда уже существующие инструменты разработки не могут полностью удовлетворить нужды программистов. Например, не так давно компания Apple разработала язык программирования Swift для того, чтобы упростить создание приложений для iPhone и iPad, которые исходно писались на более старом языке Objective-C. Каждый следующий создаваемый язык программирования все больше подобен разговорному английскому или латинскому языку. В случае успешной разработки новый язык программирования приобретает быструю популярность, подобно тому, как английский язык стал языком международного общения. В противном случае новый язык программирования ожидает участь латинского языка, с течением времени ставшего мертвым.

Вы наверняка слышали о таких языках программирования, как C++, Java и FORTRAN. Они все еще активно используются программистами для создания популярных приложений, зачастую находя эффективное применение там, где вы этого и сами не ожидали. В частности, язык программирования C++ преимущественно используется при создании высокопроизводительного программного обеспечения — браузеров Chrome, Firefox и Safari, а также многих графически насыщенных игр (Call of Duty и Counter Strike). А вот язык программирования Java лучше всего подходит для написания многих полномасштабных бизнес-приложений, а также разработки программ под платформу Android. Последним стоит упомянуть язык программирования FORTRAN, который давно утратил былую популярность, но все еще активно используется научным сообществом, а также в финансовом секторе, особенно в крупнейших банках мира, сохраняющих приверженность старым традициям и не спешащих переходить на новый исходный код.

Основная же тенденция сохраняется неизменной: до тех пор пока программисты продолжают искать новые способы решения возникающих перед ними задач, их набор инструментов пополняется новыми языками программирования, а старые языки постепенно отмирают и забываются.