Сильвио Морето

Bootstrap в примерах

Bootstrap By Example

Master Bootstrap 4's frontend framework and build your websites faster than ever before

Silvio Moreto

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Bootstrap в примерах

Освойте клиентский фреймворк Bootstrap 4 и создавайте веб-сайты быстрее, чем прежде

Сильвио Морето

ДМК Пресс
издательство
Москва, 2017

УДК 004.738.5:004.42Bootstrap
ББК 32.973.4
М79

М79 Сильвио Морето

Bootstrap в примерах. / Пер. с англ. Рагимов Р. Н. / Науч. ред. Киселев А. Н. – М.: ДМК Пресс, 2017. – 314 с.: ил.

ISBN 978-5-97060-423-6

Данная книга содержит различные примеры и пошаговое описание создания различных веб-приложений с помощью клиентского фреймворка Bootstrap. Рассматривается сеточная система, основные компоненты Bootstrap, HTML-элементы и настройка компонентов для адаптивной разработки. Описывается создание мониторинговой панели веб-приложения с помощью продвинутых возможностей Bootstrap, включая настройку компонентов, обработку событий и расширенную интеграцию библиотек.

Издание адресовано разработчикам внешних интерфейсов, не знакомым с Bootstrap. Тем не менее, подразумеваются базовые знания HTML, CSS и JavaScript, приветствуется знакомство с другими фреймворками, например, с JQuery.

УДК 004.738.5:004.42Bootstrap
ББК 32.973.4

ISBN 978-1-78528-887-6 (англ.)
ISBN 978-5-97060-423-6 (рус.)

ОГЛАВЛЕНИЕ

ОБ АВТОРЕ

Сильвио Морето (Silvio Moreto), разработчик с более чем 7-летним опытом работы в области веб-технологий, создал множество веб-сайтов и веб-приложений с использованием фреймворка Bootstrap. Постоянно применяет Bootstrap для создания простых и сложных страниц.

Он также является создателем плагина bootstrap-select (http://silviomoreto.github.io/bootstrap-select/), очень популярного среди членов сообщества. Этот плагин превращает элемент выбора в кнопку Bootstrap с раскрывающимся меню. Сильвио предугадал, что именно такого плагина не хватает фреймворку, и он пригодится другим членам сообщества. Поэтому он создал плагин, а члены сообщества помогают его поддерживать.

Кроме того, он очень активный член сообщества разработчиков программ с открытым исходным кодом, принимает участие в работе нескольких общедоступных репозиториев и сообществ взаимопомощи, таких как Stack Overflow. Занял третье место на всемирном ежегодном соревновании Django Dash 2013.

Прежде всего, я хочу поблагодарить мою жену, поддерживавшую меня на протяжении всего периода работы над книгой. Также хочу сказать спасибо моей собаке, находившейся рядом каждую ночь, когда я писал, и натолкнувшей меня на идеи для нескольких сценариев. Наконец, я хочу выразить признательность редакции издательства Packt за понимание и помощь в работе над книгой.

О ТЕХНИЧЕСКОМ РЕЦЕНЗЕНТЕ

Паула Барканте (Paula Barcante), 23-летняя проектировщица пользовательских интерфейсов с особой тягой к веб-интерфейсам. Обучается разработке, читая книги и посещая бесплатные онлайн-курсы. Начала пользоваться Bootstrap четыре года назад и продолжает применять его до сих пор. Паула с увлечением проектирует и разрабатывает красивые и удобные интерфейсы для пользователей по всему миру. Недавно поступила на работу в Amazon.com в качестве проектировщика пользовательских интерфейсов. Она всегда рада пообщаться с теми, кто увлечен дизайном или разработкой веб-интерфейсов. Ее работы можно найти на сайте paulabarcante.com.

ПРЕДИСЛОВИЕ

Разработку веб-приложений можно разделить на два периода, до и после появления Bootstrap. В 2011 году был выпущен величайший из всех клиентских фреймворков. В том же году значительно расширилась область применения фреймворка, охватив почти все сегменты рынка.

Причина проста: представьте, насколько сложно было создать просто красивую кнопку. Для этого требовалось объявить массу классов и стилей. Теперь всем этим занимается фреймворк Bootstrap, созданный разработчиками из Twitter. Он изменил саму парадигму разработки быстро меняющихся веб-интерфейсов.

Превосходство Bootstrap основывается на трех аспектах. Первый – таблица стилей, содержащая базовую CSS-разметку практически для каждого HTML-элемента, обеспечивающую их единообразный привлекательный вид.

Второй аспект – компоненты. Их можно использовать многократно простым копированием и вставкой кода. И последний аспект –подключаемые модули, или плагины на JavaScript, позволяющие создавать дополнительные элементы, которые больше нигде нельзя найти.

Разобраться в тонкостях клиентского фреймворка Bootstrap вам помогут примеры, демонстрирующие применение каждого элемента и компонента. Ознакомившись с примерами, вы лучшее поймете происходящее и определите свои цели.

Примеры, описанные в книге, позволят освоить фреймворк и научиться применять его в наиболее типичных ситуациях. К ним относятся целевые страницы, веб-приложения и панели мониторинга, созданием которых занимаются 10 из 10 веб-разработчиков. Разработчики имеют дело с такого рода страницами постоянно, и с помощью Bootstrap вы сможете сделать их привлекательнее, включить в них компоненты, анимационные эффекты, обработку событий и интеграцию с внешними библиотеками.

Мы начнем с основ, но не ограничимся ими и будем двигаться дальше, к полноценному освоению фреймворка. Самостоятельная проработка примеров из книги гарантировано сделает вас мастером Bootstrap.

Эта книга в первую очередь ориентирована на версию Bootstrap 4. Но в ней также предусматривается поддержка версии 3. Таким образом, она подготовит вас к встрече с любой ситуацией.

О чем рассказывается в этой книге

Глава 1, «*Начало*», знакомит с фреймворком Bootstrap и научит, как настроить окружение разработки.

Глава 2, «*Создание надежной основы*», начинает пример создания целевой страницы с применением приемов сеточной верстки.

Глава 3, «*Да, в первую очередь для мобильных устройств*», рассказывает о принципе разработки в первую для мобильных устройств и его реализации.

Глава 4, «*Применение стилей Bootstrap*», рассматривает применение тем оформления и нескольких элементов Bootstrap.

Глава 5, «*Что делает его фантастическим*», описывает добавление новых элементов Bootstrap в пример целевой страницы.

Глава 6, «*Можно ли создать веб-приложение?*», рассматривает процесс создания веб-приложения с помощью Bootstrap.

Глава 7, «*Конечно можно создать веб-приложение!*», посвящена созданию страницы веб-приложения с помощью только элементов и компонентов Bootstrap.

Глава 8, «*Работа с JavaScript*», начинает описание использования встроенных модулей на JavaScript в примере веб-приложения.

Глава 9, «*Введение в продвинутый режим*», начинает рассмотрение примера панели мониторинга с использованием современных компонентов и плагинов.

Глава 10, «*Оживление компонентов*», завершает пример панели мониторинга, охватывая окончательную настройку веб-страницы.

Глава 11, «*Переделка на свой лад*», содержит заключительный пример, демонстрирующий настройку существующих и создание новых плагинов Bootstrap.

Что потребуется для работы с книгой

Для работы с примерами из этой книги вам понадобится веб-браузер, предпочтительно Google Chrome, так как именно он был использо-

ван при подготовке примеров. Но можно использовать и другие браузеры.

Кроме того, вам потребуются базовые знания HTML, CSS и JavaScript. Несмотря на то, что сначала мы не торопясь пройдемся по этим технологиям, знание основных понятий очень поможет вам.

Еще одним плюсом станет знакомство с библиотекой JQuery, которая используется фреймворком Bootstrap. Хотя, библиотека JQuery будет использована в *главе 7, «Конечно можно создать веб-приложение!»*, и на очень простых примерах, навыки работы с JQuery будут кстати.

Кому адресована эта книга

Книга «*Bootstrap в примерах*» адресована программистам, интересующимся возможностями быстрой и адаптивной разработки, в первую очередь для мобильных устройств. Bootstrap – один из самых популярных клиентских фреймворков, с большим сообществом, которое радо будет принять вас в мир поддержки различных устройств, разрешений, браузеров и готовых компонентов. С помощью этой книги вы вступите в него и покажете себя профессионалом.

Соглашения

В этой книге используется несколько разных стилей оформления текста для выделения разных видов информации. Ниже приводятся примеры этих стилей и объясняется назначение.

Программный код в тексте, имена таблиц баз данных, имена папок и файлов, расширения файлов, пути к каталогам в файловой системе, фиктивные адреса URL, пользовательский ввод и ссылки в Twitter будут выглядеть так: «Затем, создадим тег `<div>` с классом `.navbar-right`, чтобы подключить правила CSS и сместить список вправо, расположив его в том же месте, где он находился раньше».

Блоки программного кода будут оформляться так:

```
<html>
  <head></head>
  <body>Hello World!</body>
</html>
```

Когда потребуется привлечь ваше внимание к определенному фрагменту в блоке программного кода, он будет выделяться жирным шрифтом:

```
<html>
  <head></head>
  <body>Hello World!</body>
</html>
```

Новые термины и **важные слова** будут выделены жирным. Текст, отображаемый на экране, например в меню или в диалогах, будет оформляться так: «На предыдущем скриншоте показан окончательный вид раздела **Features**».

Так будут оформляться предупреждения и важные примечания.

Так будут оформляться советы и рекомендации.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или может быть не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Загрузка исходного кода примеров

Загрузить файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф в разделе «Читателям – Файлы к книгам».

Список опечаток

Хотя мы приняли все возможные меры, чтобы удостовериться в качестве наших текстов, ошибки всё равно случаются. Если вы найдёте ошибку в одной из наших книг – возможно, ошибку в тексте или в коде – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдёте какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательства ДМК Пресс и Packt очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в Интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли принять меры.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, и помогающую нам предоставлять вам качественные материалы.

Вопросы

Вы можете присылать любые вопросы, касающиеся данной книги, по адресу dm@dmk-press.ru или questions@packtpub.com. Мы постараемся разрешить возникшие проблемы.

Загрузка цветных изображений для книги

Для вас также подготовлен файл в формате PDF с цветными скриншотами, рисунками и диаграммами, использованными в этой книге. Цветные изображения помогут вам отчетливее увидеть разницу в выводе. Загрузить этот файл можно по адресу: https://www.packtpub.com/sites/default/files/downloads/BootstrapByExample_ColorImages.pdf

ГЛАВА 1. Начало

С появлением и ростом популярности мобильных устройств, разработчики должны были приспособиться к решению новых проблем, таких как использование разных компоновок для разных разрешений, новый подход к работе с пользователем и оптимизация производительности для поддержки соединений с низкой пропускной способностью. При этом, оставалась еще масса старых проблем, связанных с совместимостью браузеров и отсутствием единых шаблонов проектирования.

Решение этих проблем и было поставлено во главу угла при проектировании фреймворка Bootstrap. Основной целью разработчиков из Twitter было создание клиентского фреймворка для адаптивной разработки, совместимого с основными браузерами. Результат получился превосходным! Веб-разработчики приняли его с восторгом и немедленно начали использовать.

Чтобы достичь главной цели этой книги – показать приемы использования фреймворка Bootstrap для быстрой разработки адаптивных веб-сайтов, в первую очередь предназначенных для мобильных устройств, – необходимо начать с создания рабочего окружения. Поэтому в данной главе мы рассмотрим следующие темы:

- получение Bootstrap;
- подключение Bootstrap к веб-странице;
- создание первого примера с использованием Bootstrap;
- тег элемента контейнера;
- источники поддержки;
- совместимость фреймворка.

Получение Bootstrap

Существует несколько версий фреймворка, но в этой книге, будет использоваться последняя версия Bootstrap 3 (то есть, версия 3.3.5), а

также новейшая версия Bootstrap 4 (версия 4.0.0-alpha). Когда в книге будут упоминаться элементы или компоненты, по-разному поддерживаемые этими версиями, это будет отмечаться особо.

Итак, перейдите на официальной веб-сайт http://getbootstrap.com/ и щелкните на кнопке **Download Bootstrap** (Загрузить Bootstrap), которую можно видеть на рис. 1.1.

Загрузка примеров кода

Загрузить файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф в разделе «Читателям – Файлы к книгам».

Кроме того, загрузить файлы с примерами кода для этой книги можно с помощью своей учетной записи по адресу: http://www.packtpub.com. Если вы приобрели эту книгу в другом месте, зарегистрируйтесь на странице http://www.packtpub.com/support и файлы будут высланы вам по электронной почте.

Чтобы загрузить файлы, выполните следующие действия:

- Войдите или зарегистрируйтесь на нашем веб-сайте с помощью вашего адреса электронной почты и пароля.
- Перейдите на вкладку **SUPPORT** (Поддержка).
- Щелкните на **Code Downloads & Errata** (Загрузка кода и опечатки).
- Введите название книги в поле **Search** (Поиск).
- Выберите книгу, для которой вы хотите загрузить файлы кодов.
- Выберите в раскрывающемся меню место, где вы приобрели эту книгу.
- Щелкните на кнопке **Code Download** (Загрузить код).

После загрузки файла архива, распакуйте его с помощью последней версии одной из перечисленных ниже программ:

- WinRAR / 7-Zip для Windows;
- Zipeg / iZip / UnRarX для Mac;
- 7-Zip / PeaZip для Linux.

Перед вами откроется другая страница со следующими кнопками:

- **Download Bootstrap** (Загрузить Bootstrap): для загрузки скомпилированной версии;

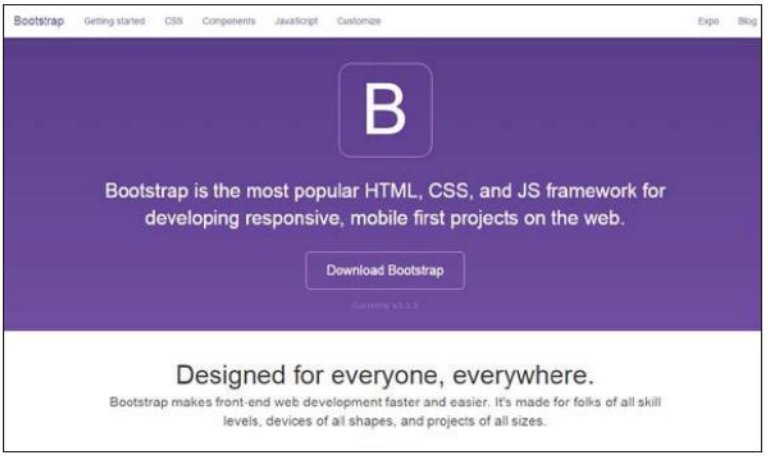


Рис. 1.1.

- **Download source** (Загрузить исходный код): для загрузки исходного кода, который вы сможете изменить и настроить под себя, но для этого потребуется знание языка Less;
- **Download Sass** (Загрузить код Sass): для загрузки исходного кода на языке Sass.

Щелкните на кнопке **Download Bootstrap** (Загрузить Bootstrap), так как мы будем рассматривать использование готового фреймворка, без Sass, только HTML, CSS и JavaScript. После загрузки распакуйте архив, и вы увидите, что фреймворк разбит на отдельные папки.

Другие версии и релизы

Загляните в официальный репозиторий https://github.com/twbs/bootstrap/, где вы найдете другие версии и новые разрабатываемые релизы. Там же вы найдете информацию о дополнительных возможностях и сведения о событиях в сообществе.

Если вы захотите опробовать версию 4, перейдите по адресу: http://v4-alpha.getbootstrap.com/ и загрузите ее, или войдите в репозиторий GitHub и выберите ветку, соответствующую версии 4.

После извлечения файлов из архива, вы увидите несколько папок. Первой по алфавиту будет папка `css`. В ней находится главный CSS-файл (`bootstrap.css`), несколько других файлов, имеющи отношение к минифицированной версии, а также файл `bootstrap-theme.`

css, содержащий определение простой темы оформления, используемой компонентами Bootstrap.

Имеется также папка fonts с файлами пиктограмм, которые мы рассмотрим в следующих главах. И, наконец, папка js, где можно найти файл bootstrap.js, его минифицированную версию и спецификацию для npm.

Что такое npm?

Утилита npm – популярный диспетчер пакетов для JavaScript. Она используется как диспетчер пакетов по умолчанию в среде Node.js.

Установка фреймворка

А теперь, после загрузки фреймворка и знакомства с его структурой, перейдем к подключению Bootstrap в веб-страницах.

Структура папок

Для начала, поясним структуру папок, которая будет использоваться для примеров в этой книге. Содержимое Bootstrap извлекается в папку main_folder и в ней же создается файл hello_world.html. Папки Bootstrap содержат файлы шрифтов, CSS и JavaScript. Структура папок фреймворка показана на рис. 1.2.

```
main_folder
-  hello_world.html
-  css
      -  bootstrap.css
-  fonts
      -  glyphicons-halflings-regular.eot
      -  glyphicons-halflings-regular.svg
      -  glyphicons-halflings-regular.ttf
      -  glyphicons-halflings-regular.woff

      -  glyphicons-halflings-regular.woff2
-  js
      -  bootstrap.js
```

Рис. 1.2.

Простой пример

А теперь, следуя рекомендациям, подключим фреймворк Bootstrap к странице `hello_world.html`. Откройте файл страницы в текстовом редакторе и добавьте в него следующую разметку HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World!</title>
</head>
<body>
    Hello World
</body>
</html>
```

Затем внутрь тега `head` добавьте код для загрузки `css`-файла:

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World!</title>
    <link rel="stylesheet" href="css/bootstrap.css">
</head>
<body>
    Hello World
</body>
</html>
```

И в конце тега `body` загрузите JavaScript-файл:

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World!</title>
    <link rel="stylesheet" href="css/bootstrap.css">
</head>
<body>
    Hello World
    <script src="js/bootstrap.js"></script>
</body>
</html>
```

Откройте файл `hello_world.html` в браузере (в примерах к этой книге будет использоваться браузер Google Chrome) и в нем консоль JavaScript. В браузере Chrome, это можно сделать, щелкнув на кнопке

Options (Настройки) (кнопка с пиктограммой бутерброда в правом верхнем углу). Затем выберите в меню пункт **More Tools | Developer Tools** (Прочие инструменты | Инструменты разработчика), как показано на рис. 1.3, и в появившемся окне щелкните на вкладке **Console** (Консоль). Вы увидите сообщение **Bootstrap's JavaScript requires jQuery** (JavaScript-файлу Bootstrap требуется JQuery), как показано на рис. 1.3.

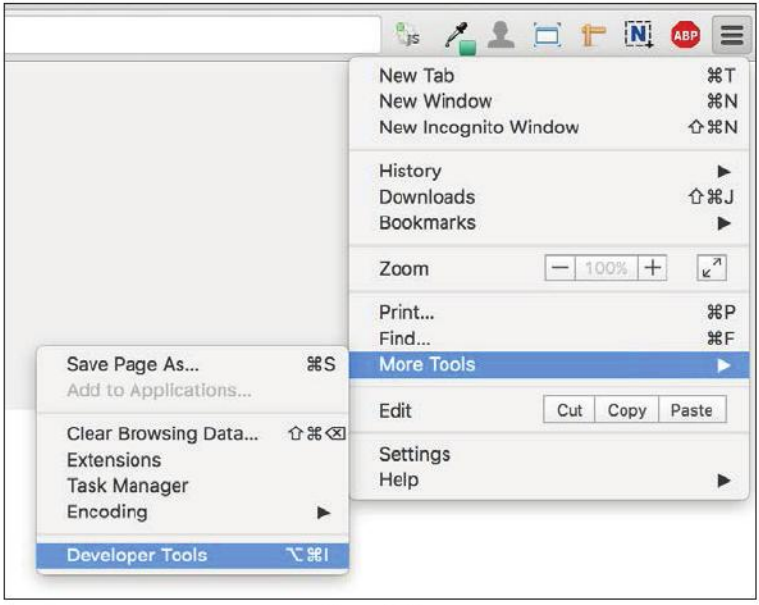


Рис. 1.3.

JQuery – кросс-платформенная библиотека JavaScript, и это единственная зависимость Bootstrap от сторонних поставщиков. Рекомендуем получить ее с официального сайта, загрузив последнюю версию (https://jquery.com/download/). Bootstrap требует версию 1.9 или выше.

Можете использовать версию 2.x и выше, если не требуется поддержка браузеров Internet Explorer 6, 7 и 8. В этой книге будет использоваться версия 1.11.3.

Скопируйте файл jQuery в папку `js` и добавьте его загрузку его в конце тега `body`, перед загрузкой файла `bootstrap.js`, как показано ниже:

```
<script src="js/jquery-1.11.3.js"></script>
<script src="js/bootstrap.js"></script>
```

Теги, необходимые Bootstrap

Для нормальной работы фреймворка Bootstrap в начало тега `<head>` необходимо добавить три тега. Они определяют кодировку текста и обеспечивают улучшенное отображение на мобильных устройствах:

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

Тег `viewport` связан с философией разработки в первую очередь для мобильных устройств. Он гарантирует надлежащее отображение в мобильных устройствах и масштабирование касанием.

Функцию масштабирования можно заблокировать, добавив параметр `user-scalable=no` в конец значения атрибута `content`. В этом случае пользователи смогут только прокручивать веб-страницу, как при использовании обычного мобильного приложения.

Если вы решили воспользоваться таким тегом, убедитесь, что пользователям не понадобится функция масштабирования и это не затруднит их работу. Пользуйтесь им с осторожностью.

Также, если потребуется предусмотреть поддержку старых версий браузера Internet Explorer (IE) (предшествовавших версии 9), добавьте библиотеки, обеспечивающие обратную совместимость элементов HTML5 и CSS3, посредством CDN, как это рекомендуется в документации к Bootstrap. Для этого вставьте следующие строки в тег `<head>`:

```
<!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.
min.js"></script>
    <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.
js"></script><![endif]-->
```

На этом этапе файл должен выглядеть следующим образом:

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

```
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible"
    content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <title>Hello World!</title>

    <link rel="stylesheet" href="css/bootstrap.css">

    <!--[if lt IE 9]>
       <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.
min.js">
       </script>
       <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js">
      </script>
    <![endif]-->
  </head>
  <body>
    Hello World!

    <script src="js/jquery-1.11.3.js"></script>
    <script src="js/bootstrap.js"></script>
  </body>
</html>
```

Это и есть пример базовой страницы! Сохраните его, поскольку он станет основой для всех примеров в книге и прочих ваших веб-страниц.

Вы знаете, что такое CDN?

CDN это аббревиатура от **Content Delivery Network** (Сеть доставки контента) – термин, определяющий сеть компьютеров, которые совместно обеспечивают доставку контента. Задачей CDN является обеспечение высокой надежности и производительности.

Следует подчеркнуть, что фреймворк Bootstrap требует наличия тега `doctype` перед тегом `<html>` в стиле HTML5:

```
<!DOCTYPE html>
<html>
    ... <!-- остальная разметка HTML -->
</html>
```

Создание первого примера с использованием Bootstrap

На текущий момент мы подготовили все необходимое для использования фреймворка. Замените строку `Hello World!` в теге `body` следующим образом:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial- scale=1">
    <title>Hello World!</title>

    <link rel="stylesheet" href="css/bootstrap.css">

    <!--[if lt IE 9]>
      <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js">
      </script>
      <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js">
      </script>
    <![endif]-->
  </head>
  <body>

    <div class="jumbotron">
      <h1>Hello, world!</h1>
      <p>This is our first sample example that will be more
awesome in the next chapters!</p>
      <a class="btn btn-primary btn-lg" href="#" role=
"button"> Bootstrap by Example, Chapter 1

      </a>
    </div>

    <script src="js/jquery-1.11.3.js"></script>
    <script src="js/bootstrap.js"></script>
  </body>
</html>
```

Откройте файл `hello_world.html` в браузере и вы увидите страницу, изображенную на рис. 1.4.

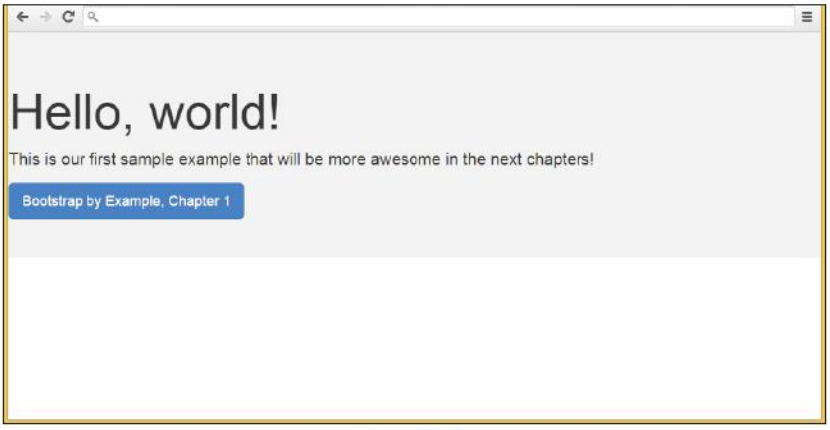


Рис. 1.4.

Примите поздравления! Вы создали первую страницу с использованием Bootstrap. Это очень простая страница, но здесь важно было понять, как правильно настроить фреймворк.

Кроме того, в этот пример были добавлены несколько компонентов, которые будут рассмотрены в следующих главах, но уже сейчас дают вам возможность познакомиться с CSS-классами и размещением элементов.

Тег контейнера

Наверное вы обратили внимание, что содержимое страницы в этом примере слишком близко расположено к левому краю, без полей и отступов. Так получилось потому, что в пример не был включен обязательный элемент Bootstrap – `container`.

Тег `container` должен обертывать содержимое сайта и вложенную сеточную систему (сеточная система будет рассмотрена в следующей главе). Существует два варианта использования элемента контейнера.

Первый вариант используется при создании веб-страниц с фиксированной шириной. В этом случае контейнер добавит адаптивные поля, учитывающие ширину области просмотра устройства:

```
<div class="container">
    ...
</div>
```

Если нужен контейнер, занимающий всю ширину области просмотра, используйте класс `.container-fluid`:

```
<div class="container-fluid">
    ...
</div>
```

В нашем примере создается адаптивный веб-сайт с фиксированной шириной. Поэтому его реализация выглядит, как показано ниже:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <title>Hello World!</title>

    <link rel="stylesheet" href="css/bootstrap.css">

    <!--[if lt IE 9]>
        <script src="https://oss.maxcdn.com/html5shiv/
 3.7.2/html5shiv.min.js"></script>
        <script src="https://oss.maxcdn.com/respond/
 1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <div class="container">
        <div class="jumbotron">
          <h1>Hello, world!</h1>

          <p>This is our first sample example that will
be more awesome in the next chapters!</p>
          <a class="btn btn-primary btn-lg" href="#" role=
"button">
              Bootstrap by Example, Chapter 1
          </a>
        </div>
    </div>

    <script src="js/jquery-1.11.3.js"></script>
    <script src="js/bootstrap.js"></script>
  </body>
</html>
```

Рис. 1.5 демонстрирует, как выглядит та же страница после добавления класса `.container`. Я рекомендую для практики и лучшего понимания заменить класс `.container` на `.container-fluid` и оценить

происшедшие при этом изменения. Измените размеры области просмотра, изменив размеры окна браузера, и посмотрите, как Bootstrap адаптирует отображение страницы:

Рис. 1.5 демонстрирует различия при использовании классов `container` и `container-fluid`. Обратите внимание на различия в ширине боковых полей.

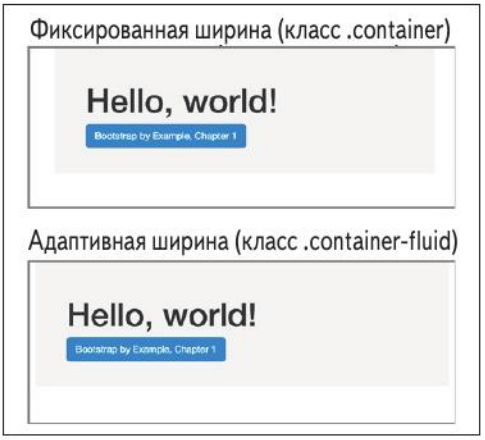


Рис. 1.5.

Ниже в этой книге, будет описано, как создавать более сложные и привлекательные веб-сайты, используя наиболее современные компоненты Bootstrap, например, такие как целевая страница на рис. 1.6.

Не волнуйтесь. Изучать основы Bootstrap и приемы его использования в веб-страницах мы будем постепенно. Следующий пример – наша первая цель – разработка целевой страницы. Просто имейте в виду, что во всех примерах будет использована одна и та же основа, представленная в этой главе.

Необязательность использования CDN

Bootstrap поддерживает возможность загрузки фреймворка из сети CDN. Это наиболее простой вариант установки, но требует определенных пояснений. Загрузка CSS в теге `<link>` в этом случае должна осуществляться из CDN, как показано ниже:

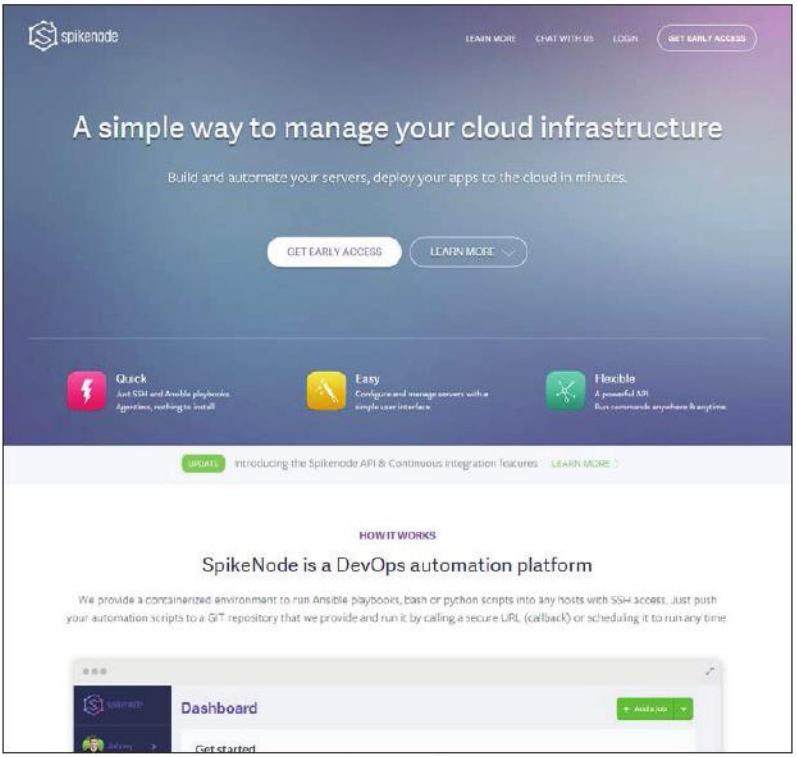


Рис. 1.6.

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/
css/bootstrap.min.css">
```

Аналогично, загрузка JavaScript-файла в теге `<script>` должна выполняться иначе:

```
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/
bootstrap.min.js"></script>
```

Существует разные мнения о том, следует или не следует использовать CDN. Мы не будем вдаваться в обсуждение этого вопроса, но главным доводом «за» является обеспечение высокой доступности. А главный довод «против» заключается в потере прямого контроля над предоставляемым контентом и зависимости от ненадежного импортированного кода.

Решение использовать или не использовать CDN зависит от конкретного случая. Нужно рассмотреть разные аргументы и сделать выбор, оптимальный для вашей веб-страницы. Не существует однозначного правильного или неправильно выбора, имеются только различные точки зрения.

Деятельность сообщества

Обсуждение фреймворка Bootstrap можно найти на нескольких сайтах в Интернете. Важно иметь доступ к заинтересованному обществу, которое занимается развитием фреймворка и его поддержкой. Вы можете получить помощь и узнать больше с помощью, например, следующих ресурсов:

- Официальный репозиторий Bootstrap. Здесь (https://github.com/twbs/bootstrap) вы найдете планы дальнейшей разработки и новейшие релизы, сообщить о проблемах и попросить помощь в их разрешении.
- Официальная документация Bootstrap (http://getbootstrap.com/) – содержит дополнительную информацию об использовании и поддержке фреймворка.
- Блог Bootstrap (http://blog.getbootstrap.com/) – лучшее место для получения новостей о Bootstrap и чтения примечаний к ним.
- Сайт Bootstrap Expo (http://expo.getbootstrap.com/) – демонстрационный стенд с примерами веб-сайтов, использующих фреймворк и такие ресурсы, как плагины, темы оформления и так далее.
- Вопросы на сайте Stack Overflow, связанные с Bootstrap (http://stackoverflow.com/questions/tagged/twitter-bootstrap). Одно из лучших средств получения помощи в решении возникших проблем. Поищите вопросы, близкие к вашим, и если вы не найдете готовый ответ, задайте свой вопрос и вы очень быстро получите на него ответ.

В Интернете существует множество других подходящих ресурсов. Используйте их с пользой для себя, и вы оцените возможность быстрой разработки, основанной на лучшем современном клиентском фреймворке.

Инструменты

Для Bootstrap имеется официальный инструмент проверки синтаксиса HTML: Bootlint (https://github.com/twbs/bootlint). Он проверяет дерево DOM на наличие ошибок использования элементов и компонентов Bootstrap. Подключите Bootlint, чтобы избежать ошибок, мешающих разработке. Инструкции по установке и использованию можно найти в репозитории.

Bootstrap и веб-приложения

Bootstrap – один из лучших фреймворков для создания веб-приложений. Он позволяет использовать единый шаблон компоновки для всего веб-приложения. С помощью предварительно подготовленных классов и тем оформления, предоставляемых фреймворком, можно значительно увеличить скорость разработки, обеспечив при этом согласованность элементов.

После выхода фреймворка, разработчики Twitter адаптировали под него множество других веб-приложений. Скриншот на рис. 1.7 демонстрирует отличный пример «резиновой верстки» веб-приложения с использованием Bootstrap и его контейнера `.container-fluid`:

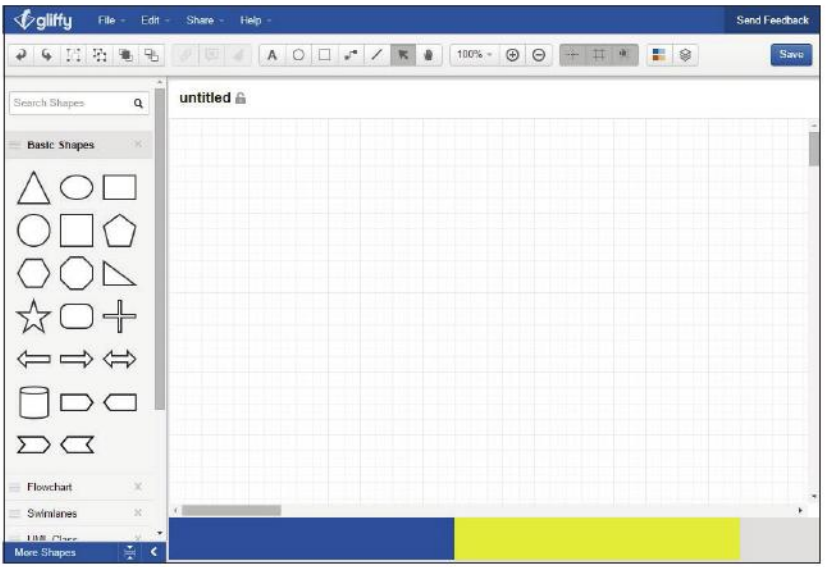


Рис. 1.7.

Совместимость с браузерами

Фреймворк Bootstrap поддерживает самые последние версии популярных браузеров. Однако, в разных браузерах отображаемые элементы не всегда выглядят одинаково, например, в версиях браузеров Chrome и Firefox для Linux.

Старые версии Internet Explorer (IE) не поддерживают некоторых свойств CSS3 и HTML5, используемых фреймворком, поэтому следует учесть это, обеспечивая поддержку этих браузеров. Определить совместимость с браузерами можно по табл. 1.1 ниже.

Кроме того, в новой версии 4 фреймворка будет убрана совместимость с некоторыми версиями браузеров. Например, было решено отказаться от поддержки IE8, так как она тормозит добавление новых возможностей, зато теперь Bootstrap получит возможность использовать новые функции CSS.

В связи с этим, версия 4 перешла от измерения размеров в пикселях к единицам rem и em, чтобы упростить адаптивность и изменение размеров, но при этом была исключена поддержка iOS 6, как следует из табл. 1.1.

Таблица 1.1.

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	Есть	Есть	НЕ ПРИМЕНЯЕТСЯ	Нет	НЕ ПРИМЕНЯЕТСЯ
iOS	Есть	НЕ ПРИМЕНЯЕТСЯ	НЕ ПРИМЕНЯЕТСЯ	Нет	Есть (iOS 7+ для v4)
OS X	Есть	Есть	НЕ ПРИМЕНЯЕТСЯ	Есть	Есть
Windows	Есть	Есть	Есть (IE 9+ для v4)	Есть	Нет

Единицы em и rem

Единицы em и rem превратились из тенденции в реальность! Они необходимы, поскольку уже присутствуют в наших контекстах, а сейчас стали поддерживаться и Bootstrap. Главная особенность единиц em и rem в том, что они являются относительными единицами, в отличие от пикселей. Причем единица em определена относительно размера родительского шрифта, а единица rem относительно размера корневого элемента, что наилучшим образом подходит для разработки адаптивного контекста.[1]

[1] Дополнительное описание этих единиц с наглядными примерами можно найти по адресу: https://learn.javascript.ru/css-units. – *Прим. ред.*

Итоги

В этой главе вы познакомились с основными идеями использования фреймворка Bootstrap. Они являются ключевыми в создании высококачественных веб-сайтов. Углубленное их понимание даст вам огромное преимущество и поможет решать любые задачи в будущем.

Целью этой главы было показать рекомендованные настройки фреймворка Bootstrap, размещение тегов, необходимые библиотеки и пример создания очень простой веб-страницы. Помните, что согласованность веб-сайта – вот главная особенность Bootstrap, которая сэкономит вам драгоценное время.

Кроме того, приступая к работе над новой веб-страницей, удостоверьтесь в правильном размещении основных тегов и компонентов, независимо от того как она создана (вручную, из шаблона или как-то еще). Использование фундамента с изъянами повлечет за собой множество проблем.

Здесь также было перечислено несколько ресурсов, где можно узнать дополнительную информацию или получить необходимую помощь. Вы стали членом большого открытого сообщества, на которое всегда можете рассчитывать.

А теперь можно переходить к решению практических задач! В следующей главе, будет начата разработка часто встречающегося примера целевой страницы, в которой будут использованы несколько компонентов Bootstrap, HTML-элементы и сеточные системы.

ГЛАВА 2. Создание надежной основы

В этой главе рассматривается несколько новых элементов и компонентов Bootstrap. При этом вы познакомитесь с понятием сеточной верстки и несколькими базовыми компонентами. Здесь мы начнем разработку примера адаптивной целевой страницы, для оформления которой сначала будет использована базовая тема фреймворка, а затем мы создадим свой собственный стиль.

Эта глава имеет следующую структуру:

- сеточная система Bootstrap;
- оформление;
- таблицы;
- кнопки.

Суть сеточной системы

Фундаментом фреймворка Bootstrap является собственная сеточная система. В составе Bootstrap имеются компоненты, позволяющие выстроить структуру веб-сайта на основе адаптивной сеточной компоновки.

Например, представьте электронную таблицу с множеством строк и столбцов, как показано на рис. 2.1. В таблице можно создать необходимое количество строк посредством слияния клеток. Но, что если попытаться изменить компоновку хотя бы одной из строк? Это может оказаться весьма непростой задачей.

Сеточная система Bootstrap работает по-другому. Она дает возможность определить набор строк, каждая из которых содержит набор независимых столбцов, что позволяет создать надежную сеточную компоновку веб-страницы. Кроме того, каждый столбец может иметь различные размеры, чтобы идеально соответствовать шаблону.

	A	B	C	D	E
1	Sauce year selling				
2			2014	2015	Total
3	Ketchup	Restaurants	12000	18000	30000
4		Black market	5000	7500	12500
5		Reatilers	48500	72750	121250
6	Mustard	Black market	6590	9885	16475
7		Retailers	1576	2364	3940
8	Relish	Black market	589600	884400	1474000
9			663266	994899	1658165
10					

Рис. 2.1.

И это еще не все, сеточная система Bootstrap автоматически подстраивается под любую область просмотра и произвольное разрешение, что и позволяет называть ее адаптивной.

Чтобы освоить сеточную систему, построим на ее основе пример целевой страницы. Как вы увидите ниже, Bootstrap позволяет определить компоновку, которая автоматически управляет содержимым, адаптируясь к любой области просмотра.

Построение основы

Для целевой страницы будет использована сетка, представленная на рис. 2.2. Как видите, она содержит семь строк, каждая из которых делится на разное число столбцов. В этом первом примере мы не будем предусматривать возможность отображения на мобильных устройствах, эта задача будет рассматриваться в следующей главе.

Настройка

Для начала воспользуемся компоновкой по умолчанию, рассмотренной в *главе 1, «Начало»*. Добавим в тег `div.container` другой тег `div` с классом `.row`:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
```

Рис. 2.2.

```
initial-scale=1">
    <title>Landing page</title>

    <link rel="stylesheet" href="css/bootstrap.css">

    <!--[if lt IE 9]>
      <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.
min.js"></script>

      <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.
js"></script>
    <![endif]-->
  </head>
  <body>
    <div class="container">
      <div class="row"></div>
    </div>

    <script src="js/jquery-1.11.3.js"></script>
```

```
    <script src="js/bootstrap.js"></script>
  </body>
</html>
```

Иерархия сеточной системы всегда представляет собой последовательность контейнеров, обертывающих строки с несколькими столбцами. Имейте в виду, что для правильного отображения всегда следует использовать такую последовательность.

Теперь, имея контейнер с первой строкой, создадим первый столбец. Каждая строка делится на 12 частей, поэтому в одной строке можно разместить до 12 столбцов.

Для идентификации столбцов используется шаблон `.col-*-*`. Первая звездочка определяет вид области просмотра, а вторая – ширину столбца. Мы еще вернемся к этому, а пока создадим первый столбец внутри строки с помощью класса `.col-md-12`:

```
<div class="container">
  <div class="row">
  header class="col-md-12"> HEADER
    </header>
  </div>
</div>
```

Для этого столбца, с классом `.col-md-12`, используется область просмотра среднего размера (как определяет идентификатор `md`), а в ширину этот столбец занимает все 12 столбцов. Другими словами, этот столбец заполняет строку целиком. Он и должен занимать всю ширину, поскольку это заголовок, который, как это показано на рис. 2.2, состоит из одной строки.

То есть, чтобы создать столбец в сеточной системе Bootstrap, нужно определить для него класс `.col-*-*`. При этом требуется указать целое число от `1` до `12` для выбора ширины и правильный префикс класса области просмотра. В табл. 2.1 перечислены префиксы классов и соответствующие им разрешения:

Что получится если создать строку с более чем 12 столбцами?

Попытайтесь добавить столбцы с общей шириной больше 12, например, пять столбцов с классом `.col-md-3`. Поскольку каждая строка может содержать не более 12 столбцов, избыточные будут перенесены в новую строку.

Таблица 2.1.

	Сверх-малые устройства (телефоны < 544 px / 34 em)	Малые устройства (планшеты ≥ 544 px / 34 em и < 768 px / 48 em)	Средние устройства (настольные ≥ 768 px / 48 em и < 900 px / 62 em)	Большие устройства (настольные ≥ 900 px / 62 em и < 1,200 px / 75 em)	Очень большие устройства (настольные ≥ 1,200 px / 75 em)
Особенности сетки	Только горизонтальные строки	Сворачивание при запуске и подгонка под столбцы сетки			
Фиксированная ширина контейнера	Авто	544 px или 34 rem	750 px или 45 rem	970 px или 60 rem	1170 px или 72.25 rem
Префикс класса	.col-xs-*	.col-sm-*	.col-md-*	.col-lg-*	.col-xl-*
Количество столбцов	12				
Ширина фиксированных столбцов	Авто	~ 44 px или 2.75 rem	~ 62 px или 3.86 rem	~ 81 px или 5.06 rem	~ 97 px или 6.06 rem

Смещение столбцов

Вторая строка разделена на три равных по размеру столбца, но первый столбец играет компенсирующую роль, то есть, это пустой столбец, образующий поле смещения слева. Поэтому вторую строку следует определить следующим образом:

```
<div class="row">
  <div class="col-md-offset-4 col-md-4">1/3</div>
  <div class="col-md-4">1/3</div>
</div>
```

Как видите, добавление класса `.col-md-offset-4` привело к созданию внутри строки левого поля с шириной в четыре столбца. Так как все строки независимы, можно определить любую желаемую компоновку.

Что получится, если указать в одном столбце несколько смещений?

Вы окажетесь в забавной ситуации. Но, если честно, будет применено только одно смещение, но какое из них? Ответ: наименьшее!

Завершение строк сетки

А теперь перейдем к третьей строке основы. Если вы поняли суть, у вас не должно возникнуть никаких проблем с этой строкой. Для тренировки, попробуйте сделать это самостоятельно и сверьте свое решение с приведенным ниже! Я верю, что вы справитесь.

Итак, эта строка состоит из двух колонок. Первый столбец занимает 4 части из 12, а второй – всю остальную ширину. HTML-код строки должен выглядеть следующим образом:

```
<div class="row">
  <div class="col-md-4"></div>
  <div class="col-md-8"></div>
</div>
```

Что касается четвертой строки: она состоит из четвертой части, за ней следует половина, за которой идет завершающая четвертая часть. Воспользовавшись базой из 12 частей, определим следующую сетку:

```
<div class="row">
  <div class="col-md-3">1/4</div>
  <div class="col-md-6">1/2</div>
  <div class="col-md-3">1/4</div>
</div>
```

Вложение строк

Пятую и шестую строки можно определить двумя способами. Определим пятую строк так же, как другие, а при определении шестой строки используем идею вложения строк.

Итак, создадим пятую строку, как уже делали это раньше, с четырьмя столбцами одинакового размера, что означает, что каждый столбец будет иметь класс `.col-md-3`:

```
<div class="row">
  <div class="col-md-3">1/4</div>
```

```
    <div class="col-md-3">1/4</div>
    <div class="col-md-3">1/4</div>
    <div class="col-md-3">1/4</div>
</div>
```

А для шестой используем прием вложения строк. Итак, создадим сначала строку с тремя столбцами:

```
<div class="row">
    <div class="col-md-3">1/4</div>
    <div class="col-md-6"></div>
    <div class="col-md-3">1/4</div>
</div>
```

Как видите, первый и последний столбцы используют один и тот же класс, что и столбцы в пятой строке: `.col-md-3`, а для среднего столбца с двойной шириной указан класс `.col-md-6`.

Теперь вложим внутрь среднего столбца другой блок `.row`. При создании вложенной строки, столбцы внутри нее инициализируются и рассматриваются как еще один набор для размещения 12 частей. Поэтому, внутри вновь созданной строки, определим два столбца с классом `.col-md-6`, чтобы получить два столбца, занимающих четверть строки:

```
<div class="row">
    <div class="col-md-3">1/4</div>
    <div class="col-md-6">
        <div class="row">
            <div class="col-md-6">1/4</div>
            <div class="col-md-6">1/4</div>
        </div>
    </div>
    <div class="col-md-3">1/4</div>
</div>
```

Идея вложенных строк весьма сложна, так как дает возможность бесконечного деления строки, но она отлично подходит для создания небольших компонентов сетки, которые могут использоваться в других местах.

Завершение сетки

Чтобы получить последнюю строку, нужно определить столбец `.col-md-2`, а затем столбцы `.col-md-7` и `.col-md-3`. Для этого просто

создайте строку с помощью тега <footer>, содержащую эти столбцы. Ниже показана законченная основа страницы:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <title>Landing page</title>
    <link rel="stylesheet" href="css/bootstrap.css">
    <!--[if lt IE 9]>
      <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.
min.js"></script>
      <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.
js"></script>
    <![endif]-->
  </head>
  <body>
    <div class="container">
      <!-- строка 1 -->
      <div class="row">
        <header class="col-md-12">
          HEADER
        </header>
      </div>

      <!-- строка 2 -->
      <div class="row">
        <div class="col-md-offset-4 col-md-4">1/3</div>
        <div class="col-md-4">1/3</div>
      </div>

      <!-- строка 3 -->
      <div class="row">
        <div class="col-md-4"></div>
        <div class="col-md-8"></div>
      </div>
      <!-- строка 4 -->
      <div class="row">
```

```
        <div class="col-md-3">1/4</div>
        <div class="col-md-6">1/2</div>
        <div class="col-md-3">1/4</div>
      </div>

      <!-- строка 5 -->
      <div class="row">
        <div class="col-md-3">1/4</div>
        <div class="col-md-3">1/4</div>
        <div class="col-md-3">1/4</div>
        <div class="col-md-3">1/4</div>
      </div>

      <!-- строка 6 - вложение строк -->
      <div class="row">
        <div class="col-md-3">1/4</div>
        <div class="col-md-6">
          <div class="row">
            <div class="col-md-6">1/4</div>
            <div class="col-md-6">1/4</div>
          </div>
        </div>
        <div class="col-md-3">1/4</div>
      </div>

      <!-- строка 7 -->
      <footer class="row">
        <div class="col-md-2">1/2</div>
        <div class="col-md-7">7/12</div>
        <div class="col-md-3">1/4</div>
      </footer>
    </div>

    <script src="js/jquery-1.11.3.js"></script>
    <script src="js/bootstrap.js"></script>
  </body>
</html>
```

Контейнер container-fluid

Пример сетки выше легко можно перевести на «резиновую» компоновку, занимающую всю ширину области просмотра. Попробуйте

сделать это, заменив класс обрамляющего блока `.container` на `.container-fluid`:

```
<div class="container-fluid">
...
</div>
```

Нам не хватает оформления!

А теперь, используем несколько CSS-элементов из числа предлагаемых Bootstrap, чтобы сделать компоненты адаптивными и более элегантными. Наша главная задача – реализовать сеточную верстку, изображенную на рис. 2.3.

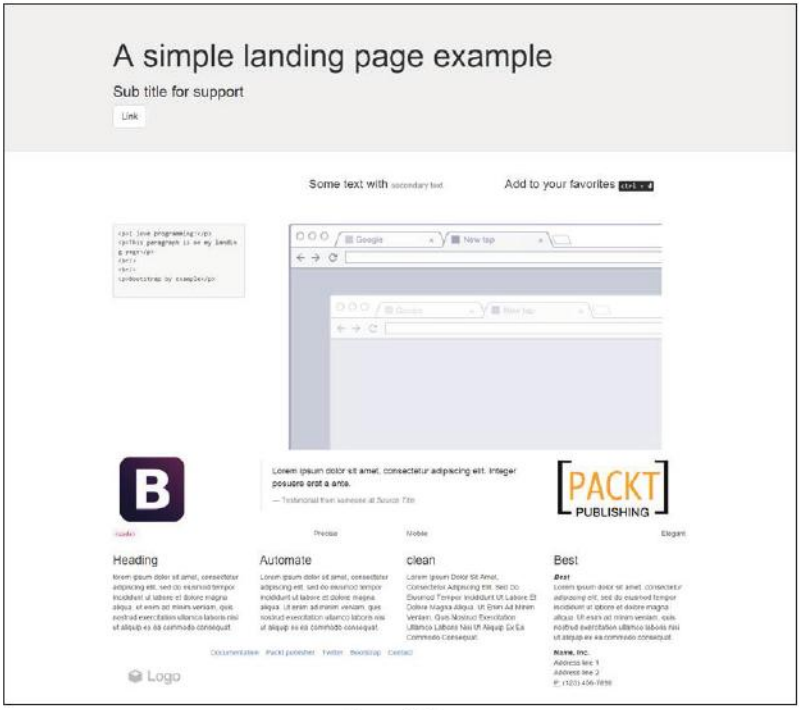


Рис. 2.3.

Давайте разберемся с каждой строкой отдельно, а затем познакомимся с оформлением и еще несколькими компонентами. При этом мы не напишем ни единой строки CSS-кода!

Начнем с первой строки, имеющей серый фон, который отсутствует в прочих элементах макета. Чтобы создать ее, внесем изменения в сетку, добавив новый блок `.container`. Итак, создаем еще один блок `.container` и помещаем в него первую строку:

```
<div class="container">
  <!-- строка 1 -->
  <div class="row">
    <header class="col-md-12">
    </header>
  </div>
</div>
<div class="container">
  <!-- другие строки (2 - 7) -->
</div>
```

Теперь, чтобы залить область серым фоном, применим класс `.jumbotron` из фреймворка Bootstrap. Класс `jumbotron` определяет гибкий компонент Bootstrap, который может расширяться, занимая всю область просмотра для отображения некоторого контента, в нашем случае – шапки страницы. Итак, поместим контейнер внутрь `div.jumbotron`:

```
<div class="jumbotron">
  <div class="container">
    <!-- строка 1 -->
    <div class="row">
      <header class="col-md-12">
      </header>
    </div>
  </div>
</div>
```

Внутри шапки, как показано на рис. 2.3, следует создать заголовок, подзаголовок и кнопку. Используем для заголовка и подзаголовка теги `<h1>` и `<h2>`. Чтобы получить кнопку, добавим ссылку с классами `.btn`, `.btn-default` и `.btn-lg`. В следующих разделах эти компоненты будут рассмотрены более подробно:

```
<div class="jumbotron">
  <div class="container">
    <!-- строка 1 -->
    <div class="row">
```

```
        <header class="col-md-12">
           <h1>A simple landing page example</h1>

           <h2>Sub title for support</h2>
           <a class="btn btn-default btn-lg" href="#" role=
"button">Link
           </a>
        </header>
      </div>
    </div>
</div>
```

Заголовки повсюду

Bootstrap автоматически оформляет заголовки с тегами `h1` до `h6`. Вы должны указывать их в порядке значимости, от `<h1>` до `<h6>` (наименее важный).

Почему заголовки так важны?

Заголовки важны для **поисковой оптимизации** (Search Engine Optimization, SEO). Они позволяют поисковым системам обнаружить наиболее важные моменты в контексте страницы. Старайтесь сохранять согласованность их иерархии в странице и не пропускать теги (то есть, не допускать перепрыгивания от заголовка 3 к заголовку 5). В противном случае, структура будет нарушена и средства SEO не станут принимать ее во внимание.

Для определения стилей заголовков можно использовать классы. То есть, если самая важная для вас фраза выглядит недостаточно крупной, вы можете изменить ее размеры, добавив класс заголовка, как это сделано в следующем примере, вывод которого показан на рис. 2.4.

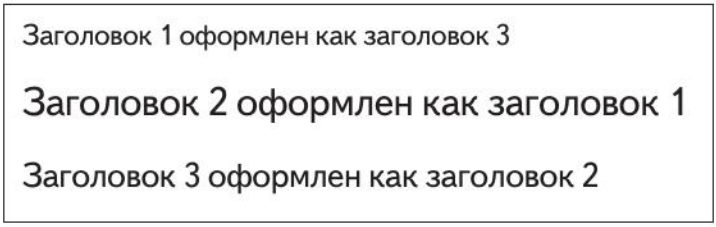


Рис. 2.4.

```
<h1 class="h3">Заголовок 1 оформлен как заголовок 3</h1>
<h2 class="h1">Заголовок 2 оформлен как заголовок 1</h2>
<h3 class="h2">Заголовок 3 оформлен как заголовок 2</h3>
```

Поиграем с кнопками

Еще один элемент первой строки – кнопка! Классы кнопок можно применять к гиперссылкам, собственно кнопкам и элементам ввода. Чтобы преобразовать один из этих элементов в кнопку, достаточно добавить класс `.btn` вместе с классом, определяющим вид кнопки, в данном случае использован класс `.btn-default`, окрашивающий кнопку синий цвет. В табл. 2.2. перечислены классы для получения кнопок всевозможных цветов:

Таблица 2.2.

Класс кнопки	Отображение
`.btn-default`	Default
`.btn-primary`	Primary
`.btn-success`	Success
`.btn-info`	Info
`.btn-warning`	Warning
`.btn-danger`	Danger
`.btn-link`	Link

Также добавим к кнопке в первой строке класс `.btn-lg` – он увеличивает размер кнопки. Bootstrap предлагает несколько классов кнопок других размеров, например, `.btn-sm` для небольших кнопок и `.btn-xs` для еще меньших.

Вы также можете заставить кнопку заполнить всю ширину родительского элемента, добавив класс `.btn-block`, отображающий кнопку в виде блока.

Подробнее об оформлении и тегах

Что касается второй строки: она содержит заголовок, за которым следует мелкий текст.

Чтобы вставить в заголовок уменьшенный и малозначительный текст, можно внутрь заголовка добавить тег `<small>` или какой-либо другой, с классом `.small`. HTML-код первого столбца второй строки будет выглядеть так:

```
<div class="row">
  <div class="col-md-offset-4 col-md-4">
    <h3>
      Some text with <small>secondary text</small>
    </h3>
  </div>
  <div class="col-md-4">
    <h3>
      Add to your favorites
      <small>
        <kbd><kbd>ctrl</kbd> + <kbd>d</kbd></kbd>
      </small>
    </h3>
  </div>
</div>
```

Отметим, что внутрь тега `small` был добавлен тег `<kbd>` для наглядного обозначения ввода с клавиатуры. Обновите страницу в веббраузере и вы увидите эту строку, как показано на рис. 2.5.

Рис. 2.5.

В третьей строке присутствует фрагмент кода и изображение. Для отображения фрагментов кода используется тег `<pre>`. Тег `<pre>`

предназначен для вывода неформатированного текста, такого как фрагмент кода. В определение столбца можно также добавить класс `.pre-scrollable`, который обеспечит прокрутку фрагмента кода, если для его отображения потребуется пространство с высотой, превышающей максимальное значение, равное 350 px (или 21.8 em).

В этой же строке в правый столбец требуется поместить изображение. Для этого создадим тег `<img>` и добавим в него класс `.img-responsive`, который автоматически подгоняет размеры изображения к области просмотра. HTML-код для третьей строки:

```
<div class="row">
  <div class="col-md-3">
    <pre>&lt;p&gt;I love programming!&lt;/p&gt;
&lt;p&gt;This paragraph is on my landing page&lt;/p&gt;
&lt;br/&gt;
&lt;br/&gt;
&lt;p&gt;Bootstrap by example&lt;/p&gt;
    </pre>
  </div>
  <div class="col-md-9">
    <img src="imgs/center.png" class="img-responsive">
  </div>
</div>
```

Обновите страницу в браузере и вы должны увидеть эту строку, как показано на рис. 2.6.

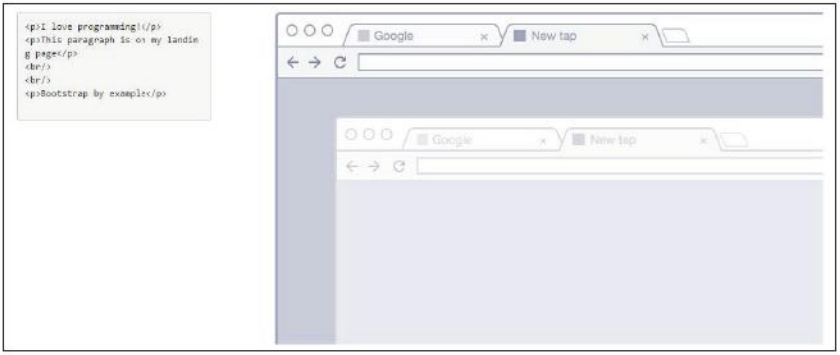


Рис. 2.6.

В четвертой строке, в левом и правом столбцах размещены изображения, а в среднем – рекомендация, как показано на рис. 2.7.

Bootstrap поддерживает оформление цитат, поэтому просто создадим тег <blockquote>. Внутрь добавим тег <footer>, чтобы указать источник, и заключим название в тег <cite>:

```
<div class="row">
  <div class="col-md-3">
    <img src="imgs/bs.png" class="img-responsive">
  </div>
  <div class="col-md-6">
    <blockquote>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Integer posuere erat a ante.</p>
      <footer>Testimonial from someone at <cite title=
"Source Title">Source Title</cite></footer>
    </blockquote>
  </div>
  <div class="col-md-3">
    <img src="imgs/packt.png" class="img-responsive">
  </div>
</div>
```

Рис. 2.7.

Перейдем дальше, к пятой строке. При определении этой строки будут продемонстрированы различные способы оформления и кодирования элементов с помощью тегов Bootstrap. Рассмотрим использование каждого из них.

В первый столбец поместим однострочный фрагмент кода. Для этого заключим фрагмент кода в тег <code>. К столбцам с первого по четвертый этой строки применим классы выравнивания. С их помощью можно легко выровнять содержимое текста в теге абзаца. Определение строки выглядит следующим образом:

```
<div class="row">
  <div class="col-md-3">
    <p class="text-left"><code>&lt;Left&gt;</code></p>
  </div>
  <div class="col-md-3">
    <p class="text-center">Center</p>
  </div>
```

```
        <div class="col-md-3">
          <p class="text-justify">Jusitfy</p>
        </div>
        <div class="col-md-3">
          <p class="text-right">Right</p>
        </div>
      </div>
```

Для правильного выравнивания достаточно просто использовать нужные классы. Результат показан на рис. 2.8.

Рис. 2.8.

Шестая строка состоит из четырех одинаковых по ширине столбцов, но в данном случае используется вариант с вложенными строками. К первым трем столбцам применяются классы Bootstrap, преобразующие текст в нижний регистр, в верхний регистр или выводящие каждое слово с заглавной буквы, соответственно:

```
<div class="row">
      <div class="col-md-3">
        <h3>Lowercase</h3>
        <p class="text-lowercase">
          Lorem ipsum dolor ... consequat.
        </p>
      </div>
      <div class="col-md-6">
        <div class="row">
          <div class="col-md-6">
            <h3>Uppercase</h3>
            <p class="text-uppercase">
              Lorem ipsum dolor ... consequat.
            </p>
          </div>
          <div class="col-md-6">
            <h3>Capitalize</h3>
            <p class="text-capitalize">
              Lorem ipsum dolor ... consequat.
            </p>
          </div>
        </div>
      </div>
      <div class="col-md-3">
        <h3>Strong and italic</h3>
```

```
        <p>
          <strong>Lorem ipsum</strong> dolor ...
          <em>consequat</em>.
        </p>
      </div>
    </div>
```

Обратите внимание на последний столбец, где тег `<strong>` используется для вывода текста жирным шрифтом и тег `<em>` – курсивом. Обновите страницу в браузере, результат должен выглядеть так, как показано на рис. 2.9.

Рис. 2.9.

Альтернативное использование элементов вывода жирным и курсивом

Для вывода текста жирным шрифтом и курсивом, можно использовать теги `<b>` и `<i>`. Однако в HTML5 тег `<b>` сейчас используется для стилистического отступа, подобно ключевым словам в абзацах, а тег `<i>` – для альтернативной голосовой разметки.

Наконец мы добрались до нижнего колонтитула, который является последней строкой. Если посмотреть на изображение полного макета (представленного в начале раздела), можно увидеть, что нижний колонтитул состоит из трех столбцов. Первый содержит изображение логотипа, средний – выстроенный в линию список, и последний – контактный адрес компании.

Для первого столбца создадим тег `<img>` с классом `.img-responsive`. Для второго столбца, со списком, выстроенным в линию, создадим тег `<ul>`. По умолчанию, каждый элемент `<li>` списка `<ul>` будет иметь слева круглый маркер. Чтобы удалить их, применим класс `.unstyled`. Кроме того, тег `<ul>` создает элементы `<li>` в виде блоков. В нашем случае нужно, чтобы элементы `<li>` располагались рядом, для этого воспользуемся классом `.list-inline`.

Для вывода информации о контактах в последнем столбце используем тег <address>. В Bootstrap имеется CSS-тема для этого тега, чтобы воспользоваться ею следует определить форматирование с помощью тега
, как показано ниже:

```
<footer class="row jumbotron">
  <div class="col-md-2">
    <img src="imgs/logo.png" class="img-responsive">
  </div>
  <div class="col-md-7">
    <ul class="list-inline list-unstyled">
      <li><a href="#">Documentation</a></li>
      <li><a href="#">Packt publisher</a></li>
      <li><a href="#">Twitter</a></li>
      <li><a href="#">Bootstrap</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </div>
  <div class="col-md-3">
    <address>
      <strong>Name, Inc.</strong><br> Address line 1<br>
      Address line 2<br>
      <abbr title="Phone">P:</abbr> (123) 456-7890
    </address>
  </div>
  </footer>
```

Обратите внимание на тег <footer>. В него добавлен класс .jumbotron, чтобы закруглить углы и залить фон серым. Результат показан на рис. 2.10:

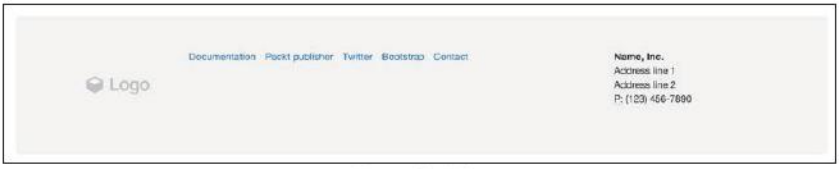
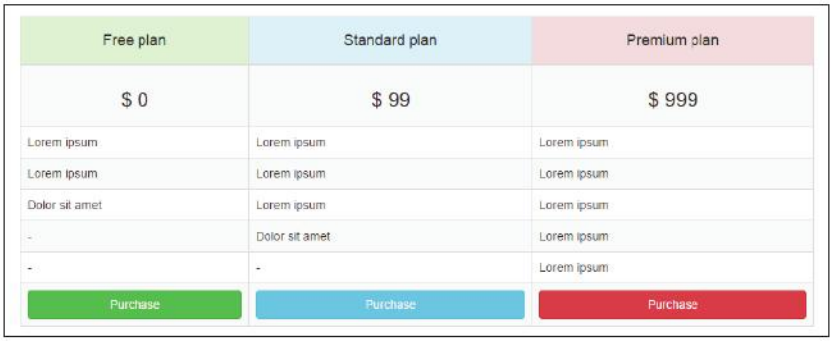


Рис. 2.10.

Управление таблицами

Фреймворк Bootstrap предоставляет широкие возможности для работы с таблицами. Для их демонстрации создадим перед тегом <footer> новую строку и поместим в нее таблицу цен, изображенную на рис. 2.11.



Free plan	Standard plan	Premium plan
$ 0	$ 99	$ 999
Lorem ipsum	Lorem ipsum	Lorem ipsum
Lorem ipsum	Lorem ipsum	Lorem ipsum
Dolor sit amet	Lorem ipsum	Lorem ipsum
-	Dolor sit amet	Lorem ipsum
-	-	Lorem ipsum
Purchase	Purchase	Purchase

Рис. 2.11.

Для этого создадим с помощью тегов `<table>`, `<thead>`, `<tbody>`, `<tr>`, `<th>` и `<td>` обычную таблицу с тремя столбцами и восемью строками:

```
<div class="row">
  <div class="col-md-10 col-md-offset-1">
    <table>
      <thead>
        <tr>
          <th>
            <h4>Free plan</h4>
          </th>
          <th>
            <h4>Standard plan</h4>
          </th>
          <th>
            <h4>Premium plan</h4>
          </th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>
            <h3>$ 0</h3>
          </td>
          <td>
            <h3>$ 99</h3>
          </td>
          <td>
```

```
                <h3>$ 999</h3>
              </td>
            </tr>
            <tr>
              <td>Lorem ipsum</td>
              <td>Lorem ipsum</td>
              <td>Lorem ipsum</td>
            </tr>
            <tr>
              <td>Lorem ipsum</td>
              <td>Lorem ipsum</td>
              <td>Lorem ipsum</td>
            </tr>
            <tr>
              <td>Dolor sit amet</td>
              <td>Lorem ipsum</td>
              <td>Lorem ipsum</td>
            </tr>
            <tr>
              <td>-</td>
              <td>Dolor sit amet</td>
              <td>Lorem ipsum</td>
            </tr>
            <tr>
              <td>-</td>
              <td>-</td>
              <td>Lorem ipsum</td>
            </tr>
            <tr>
              <td><a href="#">Purchase</a></td>
              <td><a href="#">Purchase</a></td>
              <td><a href="#">Purchase</a></td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
```

Сейчас в этой таблице нет ничего необычного. А теперь начнем добавлять в нее CSS-стили Bootstrap! Прежде всего, добавим класс `.table` в тег `<table>` (а как же!). Это кажется излишним, но фреймворк предоставляет его для большей наглядности.

Затем применим несколько специальных стилей. Для чередования цветов строк добавим класс `.table-striped` в тег `<table>`. Чтобы создать в таблице границы, добавим класс `.table-bordered`. И последнее, но не менее важное, добавим класс `.table-hover` для изменения внешнего вида строк в теле таблицы `<tbody>`, когда над ними находится указатель мыши.

Теперь займемся тегом `<tr>` внутри `<thead>`. Чтобы сделать фон зеленым, добавим класс `.success`. Не только кнопкам, но так же ячейкам, строкам или таблице целиком внутри тега `<table>` можно присваивать классы окрашивания в заданный цвет, официально называемые в Bootstrap контекстными классами.

Контекстные классы определяют те же цвета, которые используются для кнопок. Поэтому ко второму столбцу применим класс `.info`, чтобы окрасить фон в голубой цвет, и к последнему столбцу применим класс `.danger`, чтобы окрасить в красный цвет.

Фреймворк также предлагает класс `.active`, придающий цвет, применяемый при наведении указателя мыши, и класс `.warning`, придающий желтый цвет.

Внутрь каждого тега `<th>` помещен тег оформления `<h4>`. Если посмотреть на изображение таблицы (рис. 2.11) , можно заметить, что текст в заголовках выровнен по центру. Вы уже знаете, что для этого нужно просто добавить в заголовки класс `.text-center`.

После применения стилей получаем следующий тег `<thead>`:

```
<thead>
  <tr>
    <th class="success">
      <h4 class="text-center">Free plan</h4>
    </th>
    <th class="info">
      <h4 class="text-center">Standard plan</h4>
    </th>
    <th class="danger">
      <h4 class="text-center">Premium plan</h4>
    </th>
  </tr>
</thead>
```

А теперь перейдем к первой строке таблицы в теге `<tbody>` – строке цен. Выравнивание по центру текста тега `<h4>` выполняется так же, как в строке `<thead>` – добавлением класса `.text-center`:

```
<h3 class="text-center">$ 0</h3>
```

Следующие пять строк не требуют применения стилей, но последняя должна содержать кнопки и к ней будет применен хитрый прием!

Стилизация кнопок

Вы помните, как применять цветовое оформление к кнопкам? Для этого достаточно использовать те же стили, что использовались для окрашивания столбцов `<thead>`, добавив к именам контекстных классов префикс `.btn-*`. Например, первой кнопке следует присвоить класс `.btn-success`, определяющий зеленую кнопку.

Кроме того, кнопка должна заполнить ячейку во всю ширину. Чтобы кнопка растянулась на всю ширину родительского элемента, добавим класс `.btn-block`, а все остальное сделает закулисная магия! Код последней строки приводится ниже:

```
<tr>
   <td><a href="#" class="btn btn-success btn-block">Purchase</a></td>
   <td><a href="#" class="btn btn-info btn-block">Purchase</a></td>
   <td><a href="#" class="btn btn-danger btn-block">Purchase</a></td>
</tr>
```

Как босс!

Мы закончили первую версию целевой страницы! Она должна выглядеть, как показано на рис. 2.12. Обратите внимание, что при ее создании не было написано не единой строчки CSS-кода!

В этом и заключается мощь Bootstrap. Почувствовали ее? Вы сможете очень быстро создавать красивые страницы. Bootstrap является отличным инструментом компоновки!

Измените размеры области просмотра для вашей страницы, изменив размеры окна, и вы увидите, насколько хорошо Bootstrap адаптируется к любым разрешениям. После создания тщательно продуманной сетки и размещения элементов, вам не придется беспокоиться о размерах.

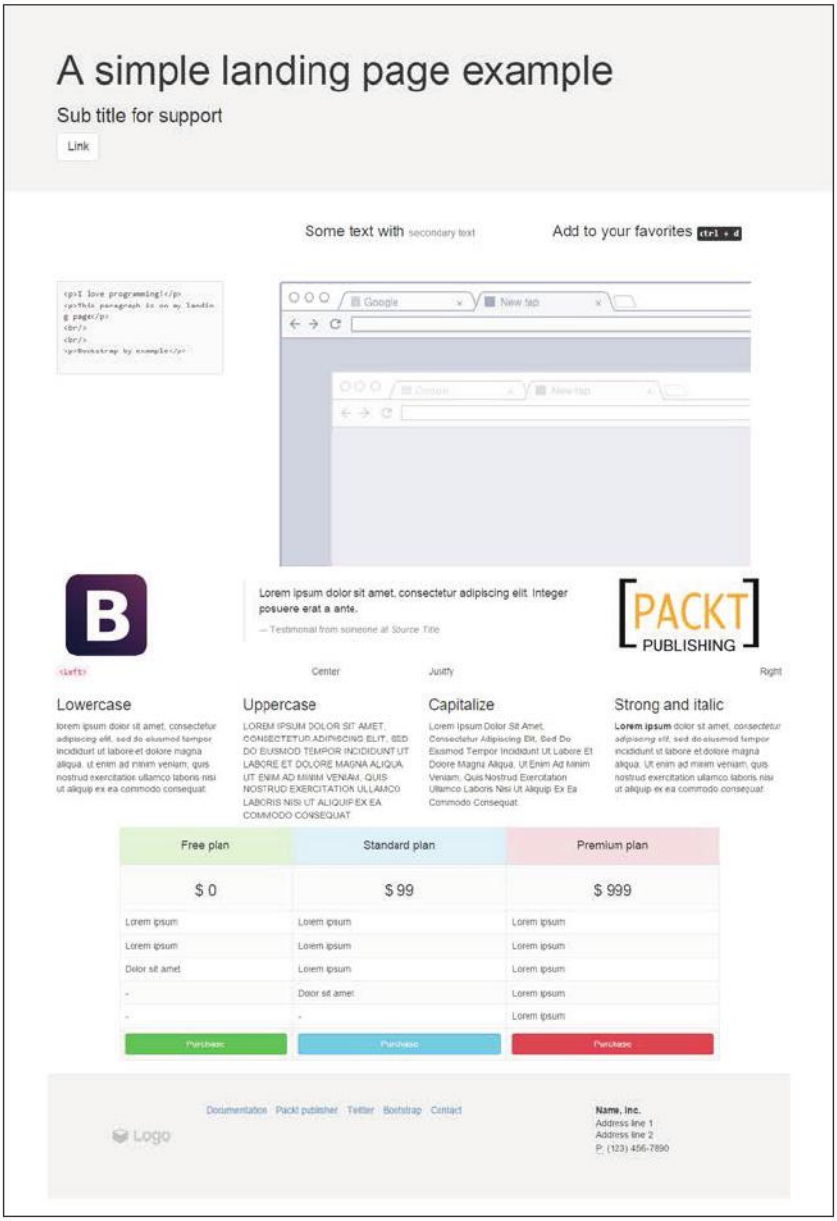

A simple landing page example
Sub title for support
Link
Some text with secondary text
Add to your favorites ctrl + d
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.
— Testimonial from someone at Source Title
PACKT PUBLISHING
Center
Justify
Right
Lowercase
Uppercase
Capitalize
Strong and italic
Free plan
Standard plan
Premium plan
$ 0
$ 99
$ 999
Documentation Packt publisher Twitter Bootstrap Contact
Logo
Name, Inc.
Address line 1
Address line 2
P: (123) 456-7890

Рис. 2.12.

Заключительные соображения

Перед тем как закончить главу необходимо кое-что прояснить. Bootstrap предлагает несколько вспомогательных классов примесей для поддержки совместимости браузеров.

Расчет размеров

Начиная с версии Bootstrap 3, для всех элементов и псевдо-элементов начал использоваться расчет размеров `box-sizing: border-box`. Если этот параметр включен, ширина и высота включают отступы и границы, но не поля.

Это упрощает расчет правильных размеров элементов, потому что внесение любых изменений в элемент, например, при корректировке отступов в `.row`, будет отражаться на ширине и высоте всего элемента. На рис. 2.13 показано, чем отличаются разные режимы определения размеров.

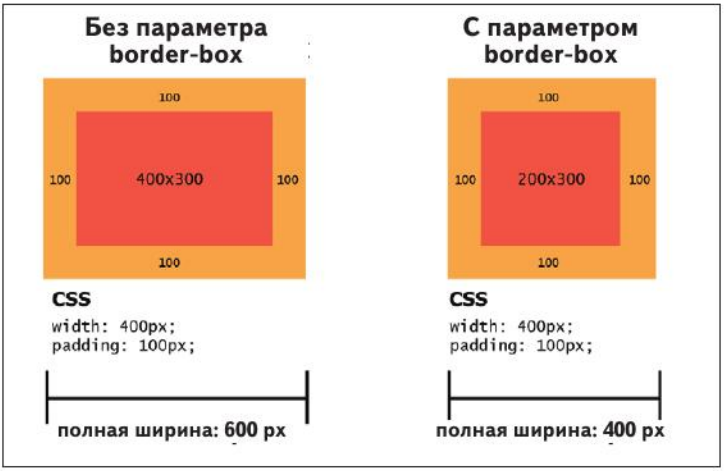


Рис. 2.13.

Псевдоэлементы – это текст в определении стиля CSS, следующий после знака `:`. Они используются для определения стилей частей элементов, например, `:after` и `:before`, которые являются наиболее типичными псевдоэлементами.

Выравнивание плавающих элементов

Bootstrap предоставляет классы для создания смещающихся элементов. Добавление класса `.pull-left` или `.pull-right` заставит элементы смещаться влево или вправо, соответственно. Имейте в виду, что оба класса применяют модификатор `!important`, исключающий переопределение:

```
<div class="pull-left"></div>
<div class="pull-right"></div>
```

В следующей главе будут рассмотрены компоненты для создания навигационных панелей. Запомните, что для выравнивания элемента в навигационной панели следует использовать `.navbar-left` и `.navbar-right` вместо `.pull-left` и `.pull-right`.

Метод clearfix

Clearfix – это способ отмены обтекания дочерних плавающих элементов. Столбцы в Bootstrap всегда смещаются влево, а к строке-родителю применяется класс `clearfix`. В результате столбцы располагаются друг за дружкой, слева направо, не перекрывая другие строки. Действие класса `clearfix` демонстрирует рис. 2.14:

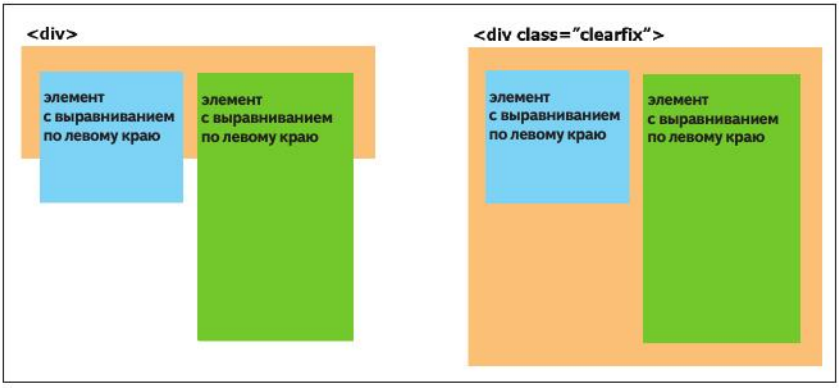


Рис. 2.14.

То есть, если понадобится применить метод clearfix к новому элементу или псевдо-элементу, просто добавьте класс `.clearfix`.

Определение шрифтов для оформления

В табл. 2.3 перечислены размеры шрифтов для текста по умолчанию и заголовков. Она включает заголовок, семейство шрифтов и высоту строки. Глубокое знание конфигурации Bootstrap по умолчанию пригодится вам в будущем, если вы захотите изменить ее.

Таблица 2.3.

Варианты CSS	Значения по умолчанию
font-family	"Helvetica Neue", Helvetica, Arial и sans-serif
line-height	1.42857143 (почти 20 px)
font-size	14 px (0.875 em)
h1 font-size	36 px (2.25 em)
h2 font-size	30 px (1.875 em)
h3 font-size	24 px (1.5 em)
h4 font-size	18 px (1.125 em)
h5 font-size	14 px (0.875 em)
h6 font-size	12 px (0.75 em)
Heading line-height	16 px (1 em)

Итоги

В этой главе начата разработка примера целевой страницы. Теперь мы знаем, как создавать привлекательные страницы, не написав ни одной строки кода CSS или JavaScript!

Основное внимание было уделено раскрытию секретов, составляющих основу Bootstrap, и разъяснению приемов ее использования. Был описан порядок размещения строк в контейнере, столбцов в строке, а также настройки классов для определенной области просмотра. Мы познакомились с некоторыми хитрыми особенностями столбцов, а также использовали вложенные строки, смещение столбцов и несколько контейнеров.

Базовые понятия сеточной системы будут применяться во всех примерах книги и являются основой фреймворка Bootstrap. Возможность управления ею является ключевым фактором освоения фреймворка. Советую попробовать по-другому скомпоновать строки целе-

вой страницы и посмотреть, что при этом будет происходить. Далее в этой книге будут описаны другие сеточные комбинации и специальные компоновки.

Мы также поэкспериментировали с кнопками, которым в Bootstrap также придается большое значение. Познакомились с базовыми параметрами настройки кнопок. В оставшейся части книги будет показано еще несколько разных подходов, но основа останется той же.

Таблицы очень часто используются в веб-страницах, и Bootstrap предлагает для них широкий выбор настроек. Мы рассмотрели пример, демонстрирующий все основные возможности оформления таблиц, которые наверняка понадобятся вам для решения повседневных задач.

И, наконец, были рассмотрены некоторые хитрые приемы, поддерживаемые фреймворком. Как я уже упоминал, важно знать и понимать основу Bootstrap, чтобы осознать его магическую силу. Фреймворк Bootstrap включает большое количество вспомогательных элементов, значительно облегчающих и ускоряющих разработку.

Следующая глава будет посвящена принципу разработки «в первую очередь для мобильных устройств» и поддержке различных конфигураций областей просмотра, что сделает целевую страницу отлично подходящей для любого устройства. Также будет описан способ, который поможет отладить поддержку любых виртуальных устройств.

ГЛАВА 3. Да, в первую очередь для мобильных устройств

У вас наверняка возник вопрос: «Я полагал, что нужно сначала разработать компоновку для мобильных устройств, а затем уже переходить к настольной версии. Почему же двигаемся в обратном порядке?»

И вы совершенно правы! Всегда следует начинать с разработки для мобильных устройств. Противоположное направление было выбрано исключительно в целях обучения и прямо сейчас это упущение будет исправлено.

Эта глава посвящена разработке для мобильных устройств и созданию адаптивных сайтов с помощью фреймворка Bootstrap, описанию компоновки страниц для различных областей просмотра, изменению контента и многому другому.

Основные темы этой главы:

- разработка в первую очередь для мобильных устройств;
- отладка для любых устройств;
- сеточная система Bootstrap для разных разрешений.

Для иллюстрации всего этого, продолжим разработку целевой страницы, начатую в предыдущей главе.

Что делает его великим

Возможно, вы задавались вопросом о смысле разработки в первую очередь для мобильных устройств. Ответ на него прост и заключается в ускорении разработки.

Главный довод в пользу парадигмы разработки в первую очередь для мобильных устройств заключается в том, что идти таким путем проще, чем впоследствии пытаться уменьшить страницу. Другими словами, если начать с разработки веб-страницы для настольных

устройств (так называемая адаптивная разработка или разработка для мобильных устройств в последнюю очередь), а затем попытаться выполнить настройку веб-сайта для работы на мобильных устройствах, с 99-процентной вероятностью случится нарушение компоновки, для исправления которого потребуется внести массу изменений.

С другой стороны, если сначала создать мобильную версию, вебсайт будет использовать (или выводить) меньше информации, чем версия для настольных устройств. Как следствие, будет проще добавить информацию, поместив ее в нужные места, и создать полноценный адаптивный набор.

Эту идею иллюстрирует рис. 3.1. При разработке для мобильных устройств в последнюю очередь, вы получите скудную, корявую и подпорченную компоновку, а начав с разработки для мобильных устройств – постепенно расширяемую, поддерживающую дальнейшие изменения и прекрасно выглядящую веб-страницу. Рис. 3.1. демонстрирует процесс разработки для каждого из подходов. Посмотрите, что происходит с бедным слоном... При разработке в первую очередь для мобильных устройств, слон вполне естественно, увеличивается в размерах, а не подстраивается под область просмотра.

Рис. 3.1.

Bootstrap и разработка в первую очередь для мобильных устройств

На начальном этапе развития фреймворка Bootstrap, в нем отсутствовала идея разработки в первую очередь для мобильных устройств. Впервые этот подход был использован для разработки адаптивных веб-страниц. Начиная с версии 3 фреймворка, эта концепция прочно утвердилась в сообществе.

Чтобы поддержать разработку в первую очередь для мобильных устройств, был полностью переписан код сеточной системы. Разработчики решили пересмотреть правила настройки сетки, вместо простого добавления стилей для мобильных устройств. Это оказало большое влияние на совместимость с версиями ниже v3, но привело к росту популярности фреймворка.

Как упоминалось в первой главе, для правильного отображения страницы следует установить корректную область просмотра в теге `<head>`:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Отладка для различных областей просмотра в браузере

Рассмотрим отладку для различных областей просмотра в веб-браузере Google Chrome. Даже если эта тема вам знакома, прочитайте этот раздел, чтобы освежить в памяти необходимые действия.

Прежде всего, откройте в браузере Google Chrome проект целевой страницы, с который мы продолжим работать в этой главе. Откройте на этой странице инструменты разработчика. Это можно сделать несколькими способами:

- щелкните правой кнопкой мыши на любой части страницы и выберите в контекстном меню последний пункт **Element inspector** (Просмотреть код);
- откройте настройки (кнопка с пиктограммой бутерброда в верхнем правом углу адресной панели), щелкните на пункте **More tools** (Дополнительные инструменты) и выберите **Developer tools** (Инструменты разработчика);

- нажмите комбинацию клавиш *Ctrl* + *Shift* + *I* (*cmd* – для пользователей OS X);
- нажмите клавишу *F12* в Windows (получено в наследство от Internet Explorer).

В открывшейся панели инструментов разработчика щелкните на пиктограмме мобильного телефона слева от лупы, как показано на рис. 3.2.

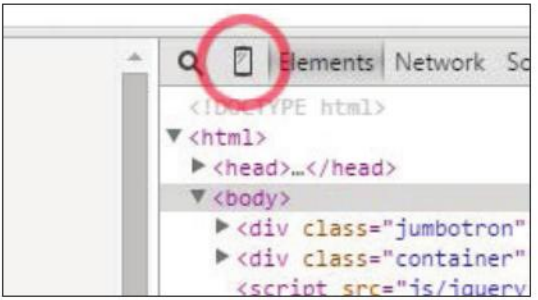


Рис. 3.2.

Это приведет к изменению окна просмотра на соответствующее определенному устройству, а кроме того появится возможность ограничить пропускную способность канала передачи данных (рис. 3.3). В Chrome появится сообщение, что для правильного отображения потребуется перезагрузить страницу.

Рис. 3.3.

На рис. 3.4 показано окно браузера с активированным режимом отображения для устройства iPhone 5. При установке этого режима возникли проблемы, поскольку страница не была разработана в первую очередь для мобильных устройств.

Первая проблема наблюдается во второй строке макета. Как видите, надпись *Ctrl+D* разделилась на две строки. Так не должно быть.

Другая проблема в том, что по неизвестной причине на этом устройстве появилась горизонтальная полоса прокрутки. Дело дрянь! Работы будет больше, чем при использовании альтернативного подхода,

когда разработка начинается со страницы для мобильных устройств. Пусть это станет уроком, чтобы в будущем не допускать такую ошибку.

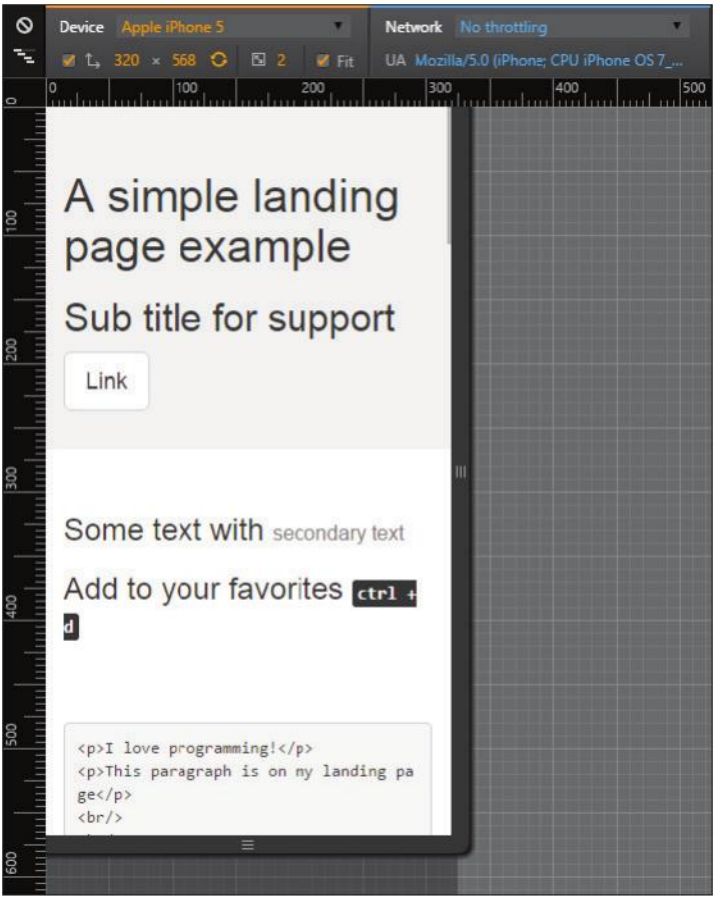


Рис. 3.4.

Теперь можно приступать к отладке веб-сайта на различных устройствах с различными разрешениями. Как видите, указатель мыши изменился на серый круг. Кроме того, события щелчка заменили события касания. Благодаря всему этому, можно полностью протестировать веб-сайт, не имея физического устройства.

Перед тем, как применить несколько хитрых приемов, приведем в порядок компоновку.

Наведение порядка

Первым делом, устраним перенос текста *Ctrl + D* во второй строке страницы. Для этого напишем первую строку CSS-кода. Добавим в тег `<head>` загрузку пользовательского CSS-файла. Запомните, что она должна выполняться после загрузки файла `bootstrap.css`:

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

В файле `base.css` определим вспомогательный класс `.nowrap`:

```
.nowrap {
white-space: nowrap;
}
```

В HTML-файле добавим созданный класс в элемент `<kbd>`:

```
<kbd class="nowrap"><kbd>ctrl</kbd> + <kbd>d</kbd></kbd>
```

Обновим страницу и убедимся, что одна проблема решена. А теперь разберемся с полосой горизонтальной прокрутки. Сможете ли вы определить причину ее появления? *Подсказка*: проблема заключается в таблице!

Назначение CSS-свойства white-space.

Свойство `white-space` определяет порядок обработки пробелов внутри элемента. По умолчанию по пробелам выполняется перенос строк, если есть такая необходимость. Правило `nowrap` запрещает перенос строк, и только тег `pre` позволяет разрывать строки.

Проблема вызвана кнопками, которые отображаются в виде блоков и делают ширину столбца больше, чем предполагалось. Чтобы решить ее создадим первый медиа-запрос CSS:

```
@media (max-width: 48em) {
    table .btn {
        font-size: 0.75rem;
        font-size: 3.5vw;
    }
}
```

Разберем по отдельности каждую из строк этого медиа-запроса. Правило предназначено для устройств со значением `max-width` до

`48em` (определяет в Bootstrap малые устройства). Если область просмотра больше `48em`, правило не применяется.

Для элементов `.btn`, помещенных внутрь элемента `table`, изменяется размер шрифта (именно в этом причина переполнения по горизонтали). Здесь используется новый подход к установке размера шрифта равным значению `3.5vw`, основанный на ширине области просмотра. Каждой единице `1vw` соответствует 1 процент от общей ширины области просмотра. При изменении области просмотра размер шрифта будет изменяться динамически, не нарушая компоновки.

Из-за новизны это свойство в настоящее время поддерживается только в Chrome 20 – 34 и Safari 6+. Поэтому была добавлена еще одна строка `font-size: 0.75rem`, как запасной вариант. Если браузер не сможет обработать определение размера шрифта, привязанное к окну просмотра, шрифт будет уменьшен до `12px`, что позволит не сломать компоновку.

Создание целевой страницы для разных устройств

Теперь, исправив обе проблемы и познакомившись с медиа-запросами и свойствами CSS3, займемся компоновкой и слегка изменим ее для разных устройств. Начнем с мобильных устройств и пойдем дальше вплоть до больших настольных устройств.

Для этого нужно применить к столбцам классы для конкретной области просмотра, как это было сделано для среднеразмерных дисплеев с помощью класса `.col-md-*`. В табл. 3.1, которая была уже представлена в предыдущей главе, перечислены различные классы и разрешения, применимые к определенным классам:

Таблица 3.1.

	Сверх-малые устройства (телефоны < 544 px / 34 em)	**Малые устройства (планшеты ≥ 544 px / 34 em и < 768 px / 48 em)**	**Средние устройства (настольные ≥ 768 px / 48 em и < 900 px / 62 em)**	**Большие устройства (настольные ≥ 900 px / 62 em и < 1,200 px / 75 em)**	**Очень большие устройства (настольные ≥ 1,200 px / 75 em)**
Особенности сетки	Только горизонтальные строки	Сворачивание при запуске и подгонка под столбцы сетки			

	Сверх-малые устройства (телефоны < 544 px / 34 em)	Малые устройства (планшеты ≥ 544 px / 34 em и < 768 px / 48 em)	Средние устройства (настоль-ные ≥ 768 px / 48 em и < 900 px / 62 em)	Большие устройства (настольные ≥ 900 px / 62 em и < 1,200 px / 75 em)	Очень большие устройства (настольные ≥ 1,200 px / 75 em)
Фиксированная ширина контейнера	Авто	544 px или 34 rem	750 px или 45 rem	970 px или 60 rem	1170 px или 72.25 rem
Префикс класса	.col-xs-*	.col-sm-*	.col-md-*	.col-lg-*	.col-xl-*
Количество столбцов	12				
Ширина фиксированных столбцов	Авто	~ 44 px или 2.75 rem	~ 62 px или 3.86 rem	~ 81 px или 5.06 rem	~ 97 px или 6.06 rem

Мобильные и сверхмалые устройства

Чтобы адаптировать нашу целевую страницу для мобильных устройств, воспользуемся инструментом отладки в Chrome с выбранным устройством iPhone 5, но без наложения ограничений на пропускную способность сети.

Можно заметить, что при отображении на небольших устройствах, Bootstrap просто укладывает столбцы друг за другом без учета строк. Некоторые строки выглядят отлично, например, заголовок и вторая строка. Третья строка выглядит несколько странно, поскольку фрагмент кода и изображение оказались в разных строках, как показано на рис. 3.5.

Чтобы исправить этот недостаток, следует добавить класс столбцов для сверхмалых устройств `.col-xs-*`, где `*` обозначает размер в строке от 1 до 12. Добавим классы `.col-xs-5` и `.col-xs-7` в столбцы соответствующей строки (примерно строка 49), обновим страницу и убедимся, что столбцы расположились рядом друг с другом:

```
<div class="row">
  <!-- строка 3 -->
  <div class="col-md-3 col-xs-5">
    <pre>&lt;p&gt;I love programming!&lt;/p&gt;
&lt;p&gt;This paragraph is on my landing page&lt;/p&gt;
```

```
&lt;br/&gt;
&lt;br/&gt;
&lt;p&gt;Bootstrap by example&lt;/p&gt;
    </pre>
  </div>
  <div class="col-md-9 col-xs-7">
    <img src="imgs/center.png" class="img-responsive">
  </div>
</div>
```

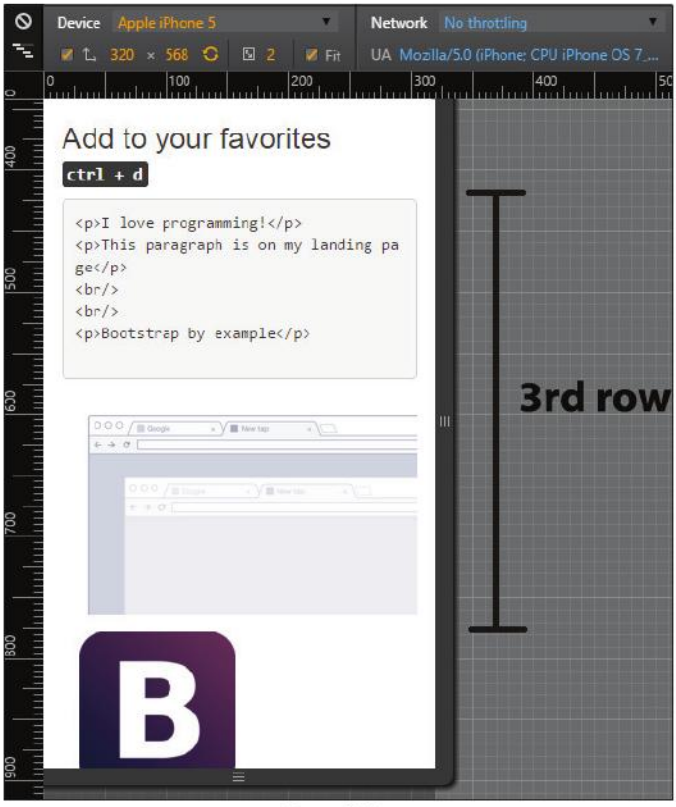


Рис. 3.5.

Несмотря на то, что изображение окна веб-браузера справа очень мало, было бы лучше, если бы оно было вытянуто по вертикали, как, например, изображение мобильного телефона (вот совпадение!). Для этого нужно скрыть существующее изображение окна браузера на

сверхмалых устройствах и вывести изображение мобильного устройства. Добавим изображение мобильного телефона ниже существующего:

```
<img src="imgs/mobile.png" class="img-responsive">
```

Оба изображения будут выводиться в правой колонке друг над другом. Теперь используем новую идею – классы доступности. Итак, на сверхмалых устройствах требуется скрыть изображение браузера и вывести изображение мобильного телефона. Для этого добавим класс `.hidden-xs` в тег с изображением браузера, и класс `.visible-xs` в тег с изображением мобильного телефона:

```
<div class="row">
  <!-- строка 3 -->
  <div class="col-md-3 col-xs-5">
    <pre>&lt;p&gt;I love programming!&lt;/p&gt;
&lt;p&gt;This paragraph is on my landing page&lt;/p&gt;
&lt;br/&gt;
&lt;br/&gt;
&lt;p&gt;Bootstrap by example&lt;/p&gt;
    </pre>
  </div>
  <div class="col-md-9 col-xs-7">
    <img src="imgs/center.png" class="img-responsive hidden-xs">
    <img src="imgs/mobile.png" class="img-responsive visible-xs">
  </div>
</div>
```

Теперь эта строка прекрасно смотрится! Изображение браузера не выводится на сверхмалых устройствах, а изображение мобильного телефона отображается только в них. На рис. 3.6 показан окончательный вид этой строки:

Перейдем к следующей, четвертой строке, с цитатой, окруженной двумя изображениями. Было бы лучше, если бы цитата выводилась первой, а оба изображения оказались ниже, в той же строке. Для этого используем те же приемы, что применялись в предыдущей строке. Проделаем это снова для тренировки.

Сначала скроем изображение Bootstrap с помощью класса `.hidden-xs`. После этого создадим еще один тег изображения для логотипа Bootstrap в одном столбце с логотипом PACKT. В результате реализация строки должна выглядеть так:

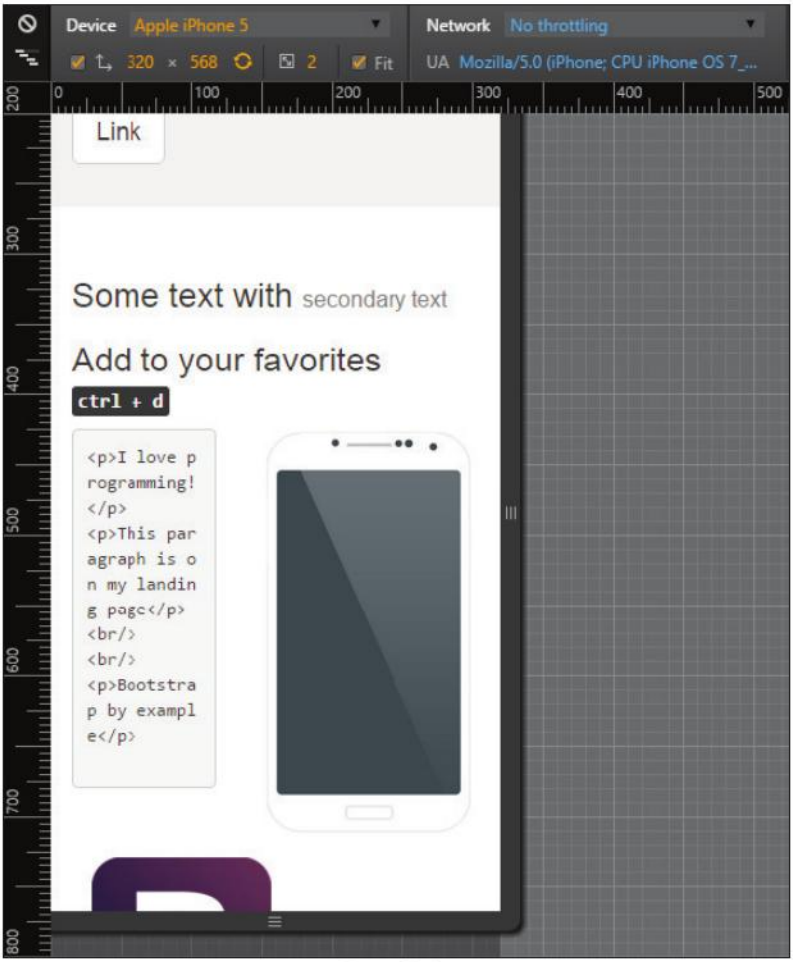


Рис. 3.6.

```
<div class="row">
  <!-- строка 4 -->
  <div class="col-md-3 hidden-xs">
    <img src="imgs/bs.png" class="img-responsive">
  </div>
  <div class="col-md-6 col-xs-offset-1 col-xs-11">
    <blockquote>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Integer posuere erat a ante.</p>
```

```
        <footer>Testimonial from someone at <cite title=
"Source Title">Source Title</cite></footer>
      </blockquote>
    </div>
    <div class="col-md-3 col-xs-7">
      <img src="imgs/packt.png" class="img-responsive">
    </div>
    <div class="col-xs-5 visible-xs">
      <img src="imgs/bs.png" class="img-responsive">
    </div>
  </div>
```

Здесь сделано много изменений, они выделены жирным шрифтом. Прежде всего, в первый столбец с логотипом Bootstrap добавлен класс `.hidden-xs`, чтобы скрыть его при выводе в ограниченной области просмотра.

Затем, измена сетка с цитатой для мобильных устройств, путем добавления смещения столбца на 1 и класса `.col-xs-11`, чтобы цитата заполнила всю оставшуюся часть столбца.

И, наконец, как уже было сказано, оба логотипа – Packt и Bootstrap – помещены в одну строку. Для этого, с помощью класса `.col-xs-7`, был создан первый столбец, занимающий семь столбцов.

Второй столбец с логотипом чуть сложнее. Так как он видим только на мобильных устройствах, в него добавлен класс `.col-xs-5`. В результате на сверхмалых устройствах элемент займет пять столбцов . Кроме того, с помощью класса `.visible-xs` столбец сделан скрытым в других областях просмотра.

Классы доступности

Подобно классам `.hidden-*`, для каждой разновидности областей просмотра и ширины столбца от 1 до 12 существуют классы `.visible-*-*`. Кроме того, поддерживается возможность изменения CSS-свойства `display` с помощью классов `.visible-*-*`, где вместо последнего символа `*` можно подставить `block`, `inline` или `inline-block`. Это свойство используется для выбора нужного вида отображения.

Как видите, эта строка содержит более 12 столбцов (1 смещение, 11 для цитаты, 7 для изображения Packt и 5 для изображения Bootstrap). Такой способ вызывает перенос столбцов, который происходит, когда

строка содержит более 12 столбцов, то есть не поместившиеся столбцы переносятся на следующие строки.

Результат изменений показан на рис. 3.7: сначала выводится цитата со сдвигом на один столбец, а ниже, рядом друг с другом, оба логотипа:

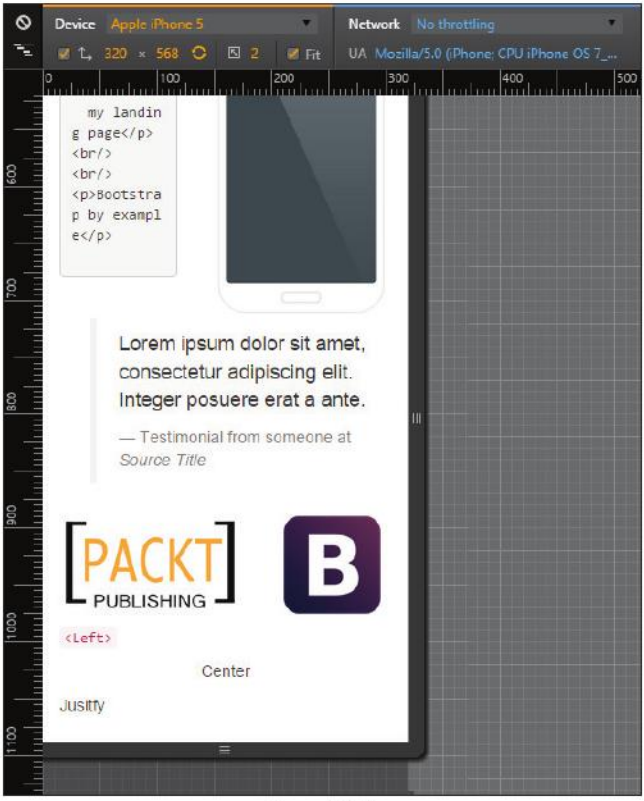


Рис. 3.7.

Планшеты и малые устройства

Завершив работу над внешним видом страницы для мобильных устройств перейдем к планшетам и малым устройствам с размерами области просмотра от 48 em до 62 em. Большую часть этих устройств составляют планшеты или устаревшие настольные мониторы. Используем в этом примере IPad Mini в портретной ориентации с разрешением 768 × 1024 пикселей.

В этом разрешении Bootstrap обрабатывает строки так же, как для сверхмалых устройств, просто выстраивая столбцы друг под другом так, что каждый из них будет заполнять всю ширину страницы. Если это нежелательно, нужно вручную установить ширину элемента с помощью класса `.col-sm-*`.

В настоящий момент наблюдаются две основные проблемы. Первая относится ко второй строке, где заголовки оказались в разных строках, хотя вполне могут уместиться в одной. Чтобы исправить ее, достаточно применить сеточные классы для малых устройств `.col-sm-6`, придав столбцам равные размеры:

```
<div class="row">
  <div class="col-md-offset-4 col-md-4 col-sm-6">
    <h3>
      Some text with <small>secondary text</small>
    </h3>
  </div>
  <div class="col-md-4 col-sm-6">
    <h3>
      Add to your favorites
      <small>
        <kbd class="nowrap"><kbd>ctrl</kbd> + <kbd>d</
kbd></kbd>
      </small>
    </h3>
  </div>
</div>
```

Результат показан на рис. 3.8.

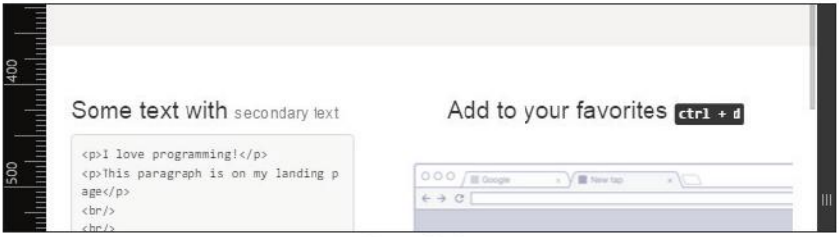


Рис. 3.8.

Вторая проблема для этой области просмотра снова возникла в строке с цитатой! Из-за классов поддержки мобильных устройств, цитата получила сдвиг и разную ширину столбцов. Нужно добавить

классы для малых устройств и разместить в этой строке логотип Bootstrap слева, цитату в центре и логотип Packt справа:

```
<div class="row">
  <div class="col-md-3 hidden-xs col-sm-3">
    <img src="imgs/bs.png" class="img-responsive">
  </div>
<div class="col-md-6 col-xs-offset-1 col-xs-11
            col-sm-6 col-sm-offset-0">
    <blockquote>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Integer posuere erat a ante.</p>
      <footer>Testimonial from someone at <cite title=
"Source Title">Source Title</cite></footer>
    </blockquote>
  </div>
  <div class="col-md-3 col-xs-7 col-sm-3">
    <img src="imgs/packt.png" class="img-responsive">
  </div>
  <div class="col-xs-5 hidden-sm hidden-md hidden-lg">
    <img src="imgs/bs.png" class="img-responsive">
  </div>
</div>
```

Как видите, понадобилось отключить смещение для столбца цитаты, которое было добавлено для сверхмалых устройств. Кроме того, для колонок с логотипами потребовалось выделить точно по три столбца. Результат показан на рис. 3.9.

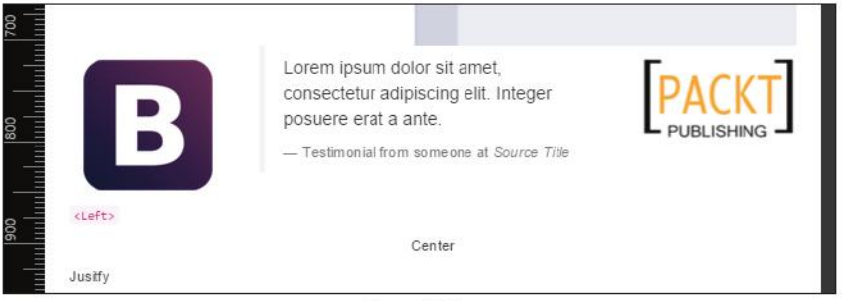


Рис. 3.9.

Все остальное выглядит отлично! Настройка для этой области просмотра далась проще. Убедились, насколько полезен Bootstrap? А теперь перейдем к последней области просмотра: настольные и большие устройства.

Настольные и большие устройства

В заключение перейдем к разработке сетчатой компоновки для настольных компьютеров и больших устройств. Мы пропустим средние устройства, потому что с них был начат процесс кодирования.

Отключите *режим устройств* в Chrome и установите для страницы область просмотра с шириной большей или равной 1200 пикселям или 75 em.

Будем использовать префикс `.col-lg-*`. Просмотрев целевую страницу, убедитесь, что с размещением все в порядке и в изменениях нет особой необходимости! И тем не менее, мы могли бы применить несколько интересных приемов, чтобы улучшить компоновку и познакомиться с некоторыми особенностями поведения сетки Bootstrap.

Обсудим порядок следования столбцов в строке. Его можно изменять с помощью классов `.col-lg-push-*` и `.col-lg-pull-*` (Обратите внимание, что хотя здесь используется префикс для больших устройств, все сказанное применимо к любому другому префиксу класса сетки).

Префикс `.col-lg-push-*` указывает, что столбец будет сдвинут вправо на `*` столбцов. С другой стороны, префикс `.col-lg-pull-*` указывает, что столбец будет сдвинут влево на * столбцов. Проверим этот прием во второй строке, изменив порядок столбцов:

```
<div class="row">
  <div class="col-md-offset-4 col-md-4 col-sm-6 col-lg-push-4">
    <h3>
      Some text with <small>secondary text</small>
    </h3>
  </div>
  <div class="col-md-4 col-sm-6 col-lg-pull-4">
    <h3>
      Add to your favorites
      <small>
        <kbd class="nowrap"><kbd>ctrl</kbd> + <kbd>d</kbd></kbd>
      </small>
    </h3>
  </div>
</div>
```

Чтобы получить нужный результат, достаточно было добавить класс `.col-lg-push-4` в первый столбец и класс `.col-lg-pull-4` –

во второй. Порядок столбцов во второй строке изменился, как показано на рис. 3.10.

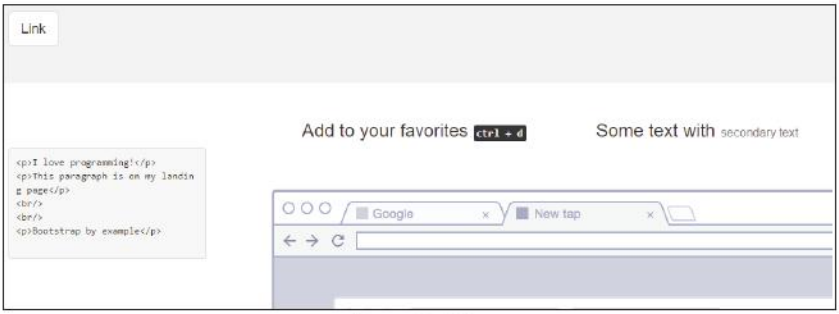


Рис. 3.10.

Итоги

Завершена еще одна глава. В ней рассказывалось, почему нужно начинать с мобильных устройств, если веб-страница должна поддерживать экраны всех размеров, от мобильных телефонов, до больших настольных дисплеев. Работать с большими экранами всегда проще, так как они вызывают меньше проблем, поэтому всегда следует начать с небольших мобильных устройств, а затем дорабатывать веб-страницы, пока не будут реализована поддержка больших областей просмотра настольных решений.

Также была рассмотрена отладка для различных устройств с помощью браузера и установка соответствующих классов для каждой области просмотра. Теперь целевая страница приобрела полноценную адаптивность и способна работать на любом устройстве.

Были описаны параметры сетки для устройств с различными разрешениями, применяемые в методологии «сначала для мобильных»: от версий для мобильных устройств до версий для больших настольных устройств.

Главный вывод этой главы в том, что всегда следует начинать разработку с компоновки для мобильных устройств. Мы не с самого начала следовали этому подходу, поэтому столкнулись с несколькими проблемами, которых можно было бы избежать, если начать с разработки для мобильных устройств.

Кроме того, разработка в первую очередь для мобильных устройств удобна всей команде разработчиков. Дизайнер с самого начала имеет более ясную картину, чего он должен достичь, и какая информация

наиболее важна. Разработчик серверной части может сосредоточиться на основных функциях и оптимизировать их для мобильных устройств, прежде, чем перейти к разработке доставки содержимого страницы. Методология «сначала для мобильных» является важной частью стратегии разработки.

На данный момент, у нас имеется целевая страница полностью настроенная под все разрешения. Использование Bootstrap позволяет получить прямой доступ к адаптивности, сводя все возникающие при этом сложности к нескольким дополнительным строкам HTML- и CSS-кода.

В следующей главе, будет рассмотрено несколько настраиваемых стилей, позволяющих сделать страницу менее похожей на страницу Bootstrap. Также будет описан процесс создания целевых страниц для различных целей с помощью настройки компонентов.

ГЛАВА 4. Применение стилей Bootstrap

После создания целевой страницы с применением методологии «сначала для мобильных», полностью адаптивной для разных устройств, пришло время еще больше погрузиться в возможности фреймворка Bootstrap и добавить новые компоненты и стили.

Главной целью этой главы станет улучшение компоновки и использование компонентов Bootstrap.

Основные темы этой главы:

- улучшение компоновки;
- формы Bootstrap;
- использование изображений в Bootstrap;
- вспомогательные стили Bootstrap.

К концу главы целевая страница будет почти готова, а вы научитесь настраивать любые HTML-компоненты с помощью Bootstrap.

Изменение компоновки сетки

Сетка, использованная для верстки текущей целевой страницы, лишь в общих чертах демонстрирует потенциал и возможности Bootstrap. В этой главе нашей целью станет создание более интересной и красивой сетки. Для этого мы изменим сетку так, чтобы она стала похожей на макет, представленный на рис. 4.1.

На этот раз мы будем двигаться вперед чуть быстрее, так как вы уже знаете, как создать сетку с помощью Bootstrap. Кроме того, разработка будет вестись в первую очередь для мобильных устройств, как было оговорено в предыдущей главе, но скриншоты будут сделаны для больших экранов из-за их лучшей наглядности.

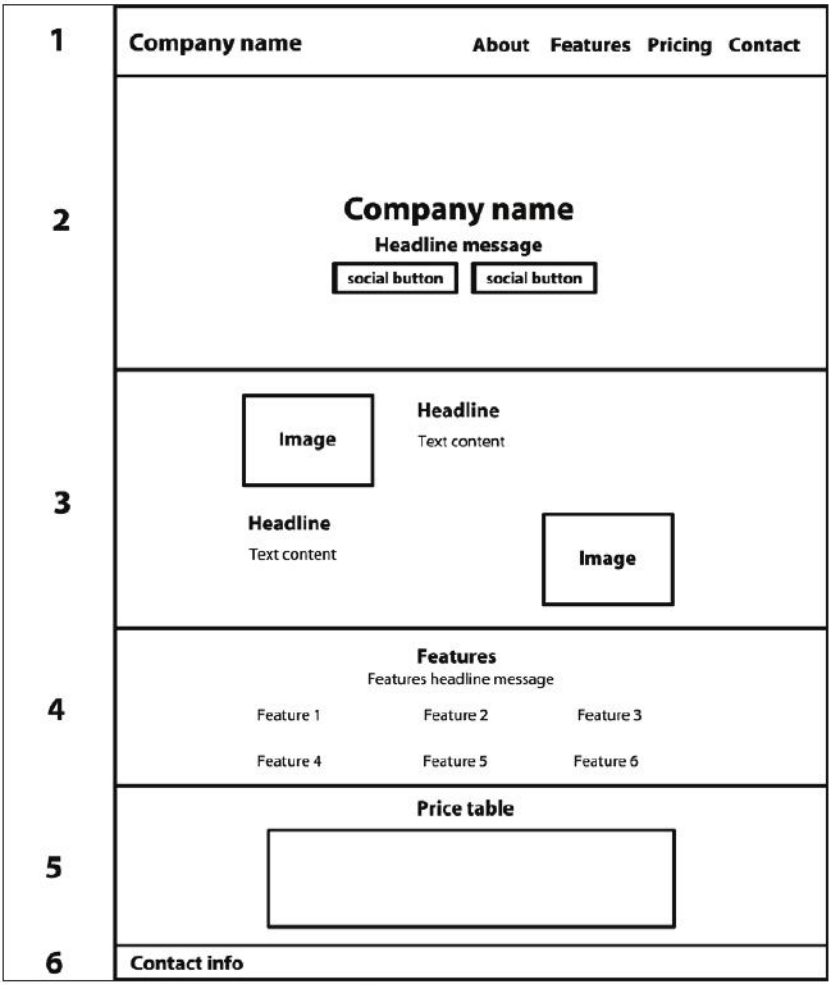


Рис. 4.1.

Начало создания сеточной системы

Как видно на рис. 4.1, сетка макета разделена на шесть частей. Для описания каждой части будет выделен отдельный раздел. Если вы начинаете работу над примером с нуля, не забудьте сохранить шаблон, созданный ранее.

Заголовок

Итак, начнем с заголовка. Следующий код, определяющий сетку, необходимо поместить сразу за открывающим тегом `<body>`:

```
<header>
  <div class="container">
    <!-- строка 1 -->
    <div class="row">
      <a class="brand pull-left" href="#">Company name</a>
      <ul class="list-inline list-unstyled pull-right">
        <li><a href="#about">About</a></li>
        <li><a href="#features">Features</a></li>
        <li><a href="#pricing">Pricing</a></li>
        <li><a href="#contact">Contact</a></li>
      </ul>
    </div>
  </div>
</header>
```

Как видите, тег `<header>` обертывает блок `.container`, делая его похожим на раздел. Просто обратите внимание, что для расположения названия компании слева и списка справа к ним были добавлены классы `.pull-left` и `.pull-right`, соответственно. В терминологии Bootstrap они называются вспомогательными стилями.

А теперь, внесем изменения в CSS-код, чтобы изменить стиль заголовка. Не забудьте импортировать пользовательский CSS-файл в теге `<head>`:

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

В этой части мы изменим цвет фона и откорректируем выравнивание для улучшения размещения ссылки и элементов списка, настроив и переопределив некоторые стили Bootstrap:

```
header {
  background-color: #F8F8F8;
}

header ul {
  margin: 0;
}
header a, header li {
  padding: 1.4rem 0;
```

```
    color: #777;
    font-weight: bold;
}
```

Заголовок будет выглядеть, как показано на рис. 4.2.

Рис. 4.2.

Представительный заголовок

Раздел 2 сетки мы будем называть *представительным* заголовком. В этом разделе находится название компании, напечатанное крупным шрифтом, а также подзаголовок и несколько кнопок. Определение этой строки следует ниже:

```
<section id="intro-header">
  <div class="container">
    <!-- строка 2 -->
    <div class="row">
      <div class="wrap-headline">
        <h1 class="text-center">Company name</h1>
        <h2 class="text-center">Tagline message</h2>
        <hr>
        <ul class="list-inline list-unstyled text-center">
          <li>
            <a class="btn btn-default btn-lg" href="#"
role="button">Sign in</a>
          </li>
          <li>
            <a class="btn btn-primary btn-lg" href="#"
role="button">Sign up</a>
          </li>
        </ul>
      </div>
    </div>
  </div>
</section>
```

Мы завернули весь контейнер в тег `section`. Здесь нет никаких секретов: для названия компании использован тег `<h1>`, для подзаголовка тег `<h2>`. Кнопки помещены в отцентрированный список, подобный использованному в заголовке, которому присвоен вспомо-

гательный класс `.text-center`, а кнопки настроены так же, как это делалось выше.

Определим в качестве фона раздела `#intro-header` большое изображение. Воспользуемся для этого CSS-файлом, добавив в него следующий код:

```
section#intro-header {
  background-image: url(../imgs/landscape.jpg);
  background-size: cover;
}
```

Правило `background-size: cover` требует растянуть изображение на всю ширину, хотя в настоящий момент размер раздела слишком мал для этого. Чтобы избежать искажений, применим хитрый прием и увеличим размеры раздела с помощью `.wrap-headline`:

```
  ection#intro-header .wrap-headline { position: relative;
  padding-top: 20%;
  padding-bottom: 20%;
}
```

Как вы могли заметить, были определены отступы в `20%` сверху и снизу относительно текущей позиции. При этом высота раздела станет адаптивной для любой области просмотра.

Теперь добавим еще несколько CSS-правил для форматирования:

```
section#intro-header {
  background-image: url(../imgs/landscape.jpg);
  background-size: cover;
}
section#intro-header .wrap-headline {
  position: relative;
  padding-top: 20%;
  padding-bottom: 20%;
}
section#intro-header h1,
section#intro-header h2 {
  color: #FFF;
}
section#intro-header h2 {
  font-size: 1.5rem;
}
section#intro-header hr {
  width: 10%;
```

```
}

section#intro-header .btn-default {
  background-color: rgba(255, 255, 255, 0.5);
  border: none;
}
```

Окончательный вид этих двух разделов представлен на рис. 4.3. Выглядит фантастично, не так ли?

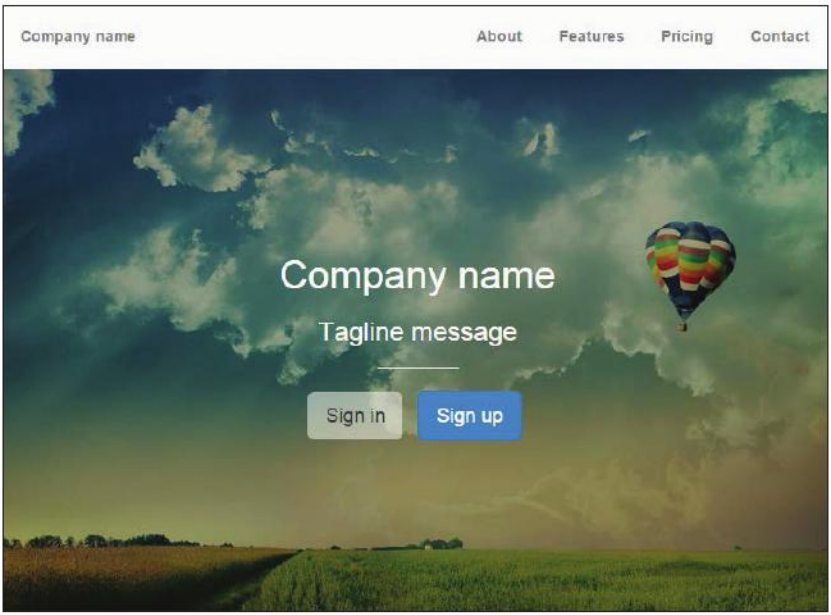


Рис. 4.3.

Раздел описания

Как и для прочих разделов, создадим для раздела *описания* обертывающий контейнер и в нем – две одинаковых по высоте строки, в каждой из которых будут выводиться изображение и текст в разном порядке:

```
<section id="about">
  <div class="container">
    <!-- строка 3 -->
    <div class="row">
      <div class="col-sm-6">
        <img src="imgs/mock_ipad.jpg" class="img-responsive">
```

```
        </div>
        <div class="col-sm-6">
          <h3>Lorem ipsum dolor sit amet</h3>
          <p>
            Lorem ipsum dolor...
          </p>
        </div>
      </div>
      <hr>

      <!-- строка 4 -->
      <div class="row">
        <div class="col-sm-6">
          <h3>Lorem ipsum dolor sit amet</h3>
          <p>
            Lorem ipsum dolor...
          </p>
        </div>
        <div class="col-sm-6">
          <img src="imgs/mock_nexus.jpg" class="img-responsive">
        </div>
      </div>
    </div>
  </section>
```

Этот раздел состоит из двух строк, по два столбца в каждой. Так как столбцы делят строку пополам, им присвоен класс `.col-sm-6`. Изображениям присвоен класс `.img-responsive`, обеспечивающий пропорциональность их размеров размерам области просмотра, а в столбце рядом с изображением разместили некоторый текст.

Добавим несколько CSS-правил для создания полей между содержимым и верхней частью страницы:

```
section#about img {
  margin-top: 6.5rem;
  margin-bottom: 5rem;
}
section#about h3 {
  margin-top: 10rem;
}
```

На рис. 4.4 показан окончательный вид этого раздела. Проверьте, выглядит ли точно так же страница у вас, а затем переходите к разделу с описанием *свойств*:

Рис. 4.4.

Раздел свойств

Раздел *свойств* состоит из двух строк по три столбца, хотя создается только один элемент `.row` и используется метод переноса столбцов. Вы его еще не забыли?

Метод переноса столбцов заключается в использовании более 12 частей для столбцов в одной строке. Не поместившиеся столбцы переносятся в строку ниже, благодаря чему создается тот же эффект, что дают два элемента `.row`:

```
<section id="features">
  <div class="container">

    <!-- строка 5 -->
    <div class="row">
      <div class="col-sm-12">
        <h3 class="text-center">Features</h3>
        <p class="text-center">Features headline message</p>
```

```
        </div>
      </div>

      <!-- строка 6 -->
      <div class="row">
        <div class="col-sm-2 col-md-4">
          <div class="feature">Feature</div>
        </div>
        <div class="col-sm-2 col-md-4">
          <div class="feature">Feature</div>
        </div>
        <div class="col-sm-2 col-md-4">
          <div class="feature">Feature</div>
        </div>
        <div class="col-sm-2 col-md-4">
          <div class="feature">Feature</div>
        </div>
        <div class="col-sm-2 col-md-4">
          <div class="feature">Feature</div>
        </div>
        <div class="col-sm-2 col-md-4">
          <div class="feature">Feature</div>
        </div>
        </div>
      </div>
    </div>
  </section>
```

В этом разделе созданы две строки. Первая содержит заголовок и подзаголовок раздела внутри тегов `<h3>` и `<p>`, соответственно. Вторая объединяет шесть равных по ширине столбцов с классами `.col-sm-2` и `.col-md-4`. Использование класса `.col-sm-2` позволяет разместить элементы `.feature` в одной строке при отображении в малой области просмотра.

А теперь изменим цвета текста и добавим отступы между списками столбцов свойств:

```
section#features {
  background-color: #eef2f5;
  border-top: 0.1rem solid #e9e9e9;
  border-bottom: 0.1rem solid #e9e9e9;
}
section#features * {
```

```
    color: #657C8E;
  }
  section#features .feature {
    padding-top: 2rem;
    padding-bottom: 4rem;
    text-align: center;
  }
```

На рис. 4.5 представлен окончательный вид раздела свойств. А теперь приступим к изменению таблицы цен. Это будет несложно, поскольку ее основа уже готова.

Рис. 4.5.

Раздел таблицы цен

Для раздела *таблицы цен* используем таблицу из раздела «*Работа с таблицами*» в *главе 2*, «*Создание надежной основы*», но с некоторыми чисто внешними улучшениями. Изменения незначительные, как это видно ниже:

```
<section id="pricing">
  <div class="container">

    <!-- строка 7 -->
    <div class="row">
      <div class="col-sm-12">
        <h3 class="text-center price-headline">Price table</h3>
      </div>
    </div>

    <!-- строка 8 -->
    <div class="row">
      <div class="col-sm-10 col-sm-offset-1">
```

```
<table class="table table-striped table-hover">
  <thead>
    <tr>
      <th class="success">
        <h4 class="text-center white-text">Free plan</h4>
      </th>
      <th class="info">
        <h4 class="text-center white-text">Standard
plan</h4>
      </th>
      <th class="danger">
        <h4 class="text-center white-text">Premium
plan</h4>
      </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td class="success">
        <h3 class="text-center white-text">$ 0</h3>
      </td>
      <td class="info">
        <h3 class="text-center white-text">$ 99</h3>
      </td>
      <td class="danger">
        <h3 class="text-center white-text">$ 999</h3>
      </td>
    </tr>
    <tr>
      <td>Lorem ipsum</td>
      <td>Lorem ipsum</td>
      <td>Lorem ipsum</td>
    </tr>
    <tr>
      <td>Lorem ipsum</td>
      <td>Lorem ipsum</td>
      <td>Lorem ipsum</td>
    </tr>
    <tr>
      <td>Dolor sit amet</td>
      <td>Lorem ipsum</td>
      <td>Lorem ipsum</td>
```

```
                </tr>
                <tr>
                  <td>-</td>
                  <td>Dolor sit amet</td>
                  <td>Lorem ipsum</td>
                </tr>
                <tr>
                  <td>-</td>
                  <td>-</td>
                  <td>Lorem ipsum</td>
                </tr>
                <tr>
                  <td><a href="#" class="btn btn-success
                         btn-block">Purchase</a></td>
                  <td><a href="#" class="btn btn-info
                         btn-block">Purchase</a></td>
                  <td><a href="#" class="btn btn-danger
                         btn-block">Purchase</a></td>
                </tr>
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </section>
```

Во-первых, в первую строку раздела был добавлен заголовок `<h3>`. Кроме того, в первый тег `<tr>` в `<tbody>` добавлены классы `.success`, `.info` и `.danger` (выделены жирным).

Также из элемента `<table>` убран класс `.table-bordered`, чтобы выключить границы. И, наконец, изменились некоторые цвета в CSS-файле и создан класс `.white-text`, который также выделен в следующем коде:

```
section#pricing h3.price-headline {
  margin-top: 5rem;
  margin-bottom: 3rem;
}
section#pricing .white-text {
  color: #FFF;
}
section#pricing thead .success {
  background-color: #78CFBF;
```

```
}
section#pricing thead .info {
  background-color: #3EC6E0;
}
section#pricing thead .danger {
  background-color: #E3536C;
}
section#pricing tbody .success {
  background-color: #82DACA;
}
section#pricing tbody .info {
  background-color: #53CFE9;
}
section#pricing tbody .danger {
  background-color: #EB6379;
}
```

На рис. 4.6 показана законченная таблица цен. И в завершение перейдем к созданию нижнего колонтитула с контактной информацией:

Рис. 4.6.

Нижний колонтитул

Нижний колонтитул будет содержать пять столбцов, в первом из которых, с классом `.col-sm-2`, будет находиться логотип. За ним следуют три информационных столбца с классом `.col-sm-2`. И последний столбец, с классом `.col-sm-4`, будет содержать адрес:

```
<footer>
  <div class="container">
```

```
        <div class="col-sm-2">
          <img src="imgs/logo.png" class="img-responsive">
        </div>
        <div class="col-sm-2">
          <h5>The company</h5>
          <ul class="list-unstyled">
            <li><a href="#">Documentation</a></li>
            <li><a href="#">Packt publisher</a></li>
            <li><a href="#">About us</a></li>
            <li><a href="#">Contact</a></li>
          </ul>
        </div>
        <div class="col-sm-2">
          <h5>Social</h5>
          <ul class="list-unstyled">
            <li><a href="#">Facebook</a></li>
            <li><a href="#">Twitter</a></li>
            <li><a href="#">Blog</a></li>
          </ul>
        </div>
        <div class="col-sm-2">
          <h5>Support</h5>
          <ul class="list-unstyled">
            <li><a href="#">Contact</a></li>
            <li><a href="#">Privacy police</a></li>
            <li><a href="#">Terms & conditions</a></li>
            <li><a href="#">Help desk</a></li>
          </ul>
        </div>
        <div class="col-sm-4">
          <address>
            <strong>Name, Inc.</strong> Address line 1<br>
             Address line 2<br>
            <abbr title="Phone">P:</abbr> (123) 456-7890
          </address>
        </div>
      </div>
    </footer>
```

А теперь, для улучшения внешнего вида нижнего колонтитула добавим несколько CSS-правил:

```
footer {
  background-color: #191919;
  color: #ADADAD;
  margin-top: 3em;
}
footer h5, footer img {
  margin-top: 5em;
  font-weight: bold;
}
footer address {
  margin-top: 5em;
  margin-bottom: 5em;
  color: #5A5A5A;
}

footer ul {
  margin-bottom: 5em;
}
footer address strong {
  color: #ADADAD;
  display: block;
  padding-bottom: 0.62em;
}

footer a {
  font-weight: 300;
  color: #5A5A5A;
}

footer a:hover {
  text-decoration: none;
  color: #FFF;
}
```

Здесь мы сменили цвет фона на более темный, добавили поля для увеличения размера нижнего колонтитула и изменили цвет ссылок. Внесение изменений в компоновку завершено! Окончательный вид нижнего колонтитула показан на рис. 4.7.

Измените размер окна просмотра, и вы убедитесь, что страница корректно адаптируется к любому разрешению. Итак, с помощью Bootstrap мы вновь создали целевую страницу, но на этот раз с использованием методики «сначала для мобильных устройств». Отлично сделано!

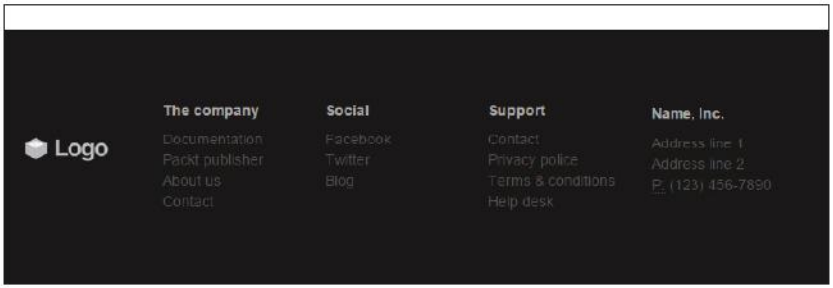


Рис. 4.7.

Формирование форм

Всемирная паутина была бы совсем другой без форм. Формы – основной способ взаимодействия пользователя с веб-страницами и отправки данных. С самого начала оформление и отображение форм в Веб было источником неприятностей, потому что они по-разному отображались разными браузерами и возникали проблемы с их компоновкой.

Одной из причин разработки Bootstrap было стремление добиться одинакового отображения веб-страниц любыми браузерами и устройствами. Это же относится и к формам. Существуют стили практически для всех элементов. В этом разделе будет начато рассмотрение форм, но их обсуждение продолжится в следующих главах, поскольку они являются важным элементом в веб-разработке.

Форма информационного бюллетеня

Для простоты начнем с создания формы, встроенной в целевую страницу, содержащую элементы ввода, выстроенные в одну линию. Вставим новую строку между таблицей цен и нижним колонтитулом, добавив следующий HTML-код:

```
<section id="newsletter" class="text-center">
  <h4>Stay connected with us. Join the newsletter to receive fresh
info.</h4>
  <form class="form-inline" method="POST">
    <div class="form-group">
      <input class="form-control" placeholder="You name">
    </div>
    <div class="form-group">
      <input class="form-control" placeholder="Your email">
```

```
      </div>
      <button type="submit" class="btn btn-primary">Join now!</button>
    </form>
  </section>
```

Отлично, а теперь рассмотрим реализацию, разбив ее на части. В первой части создается новый раздел `<section>` и его содержимое центрируется с помощью вспомогательного класса `.text-center`.

Первый тип форм, с которым вы познакомитесь, тип `.form-inline`, содержит все управляющие элементы внутри встроенного блока. Именно поэтому содержимое формы можно отцентрировать с помощью класса `.text-center`. Кроме того, форма выстраивает в линию элементы управления для малых областей просмотра, группируя их в блоки, занимающие одну строку.

Внутри формы `.form-inline` имеется два блока `div.form-group`. Каждый элемент внутри блока `<form>`, содержащий класс `.form-group`, будет автоматически отображаться как блочный элемент. Элементы формы почти всегда помещаются внутрь блока `.form-group`, поскольку он является оболочкой для надписей и элементов управления, позволяющей оптимизировать занимаемое место.

В данном случае, так как, форма определена как вытянутая в линию (поскольку указан класс `.form-inline`), элементы `.form-group` также будут выстраивать свое содержимое в линию.

В двух тегах `<input>` нет ничего загадочного, просто используйте их, как показано в примере. То же самое сделайте с кнопками, окрасив их в синий цвет с помощью класса `.btn-primary`.

CSS-код для этого раздела очень прост: несколько простых настроек для улучшения отображения:

```
section#newsletter {
  border-top: 1px solid #E0E0E0;
  padding-top: 3.2em;
  margin-top: 2em;
}
section#newsletter h4 {
  padding: 1em;
}
section#newsletter form {
  padding: 1em;
  margin-top: 2em;
  margin-bottom: 5.5em;
}
```

Наша первая форма готова! Окончательный ее вид показан на рис. 4.8:

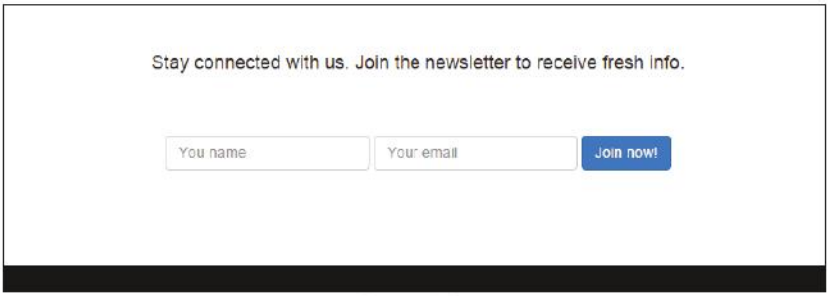


Рис. 4.8.

Это одна из простейших форм. А теперь можно замахнуться на создание других форм с помощью Bootstrap.

Форма обратной связи

Для реализации формы *обратной связи* необходимо создать отдельный HTML-файл. Назовите его `contact.html` и используйте в нем тот же заголовок и нижний колонтитул, что прежде использовался в целевой странице. Окончательный вид формы показан на рис. 4.9. Разберем все части формы, чтобы понять, как получить его.

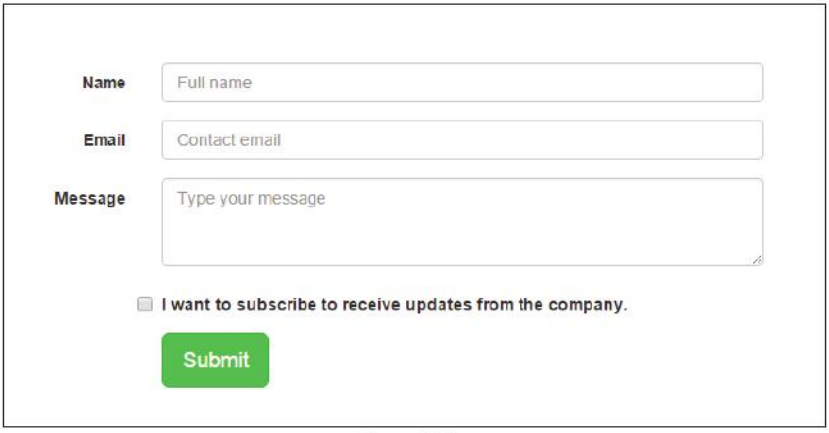


Рис. 4.9.

Прежде всего, нужно создать сетку для формы. Форма должна находится в центре страницы, поэтому понадобится следующий HTML-код:

```
<section id="contact" class="container">
  <div class="row">
    <div class="col-sm-offset-2 col-sm-8">
        ...
    </div>
  </div>
</section>
```

Этот код создает сетку контейнера. Внутрь столбца нужно добавить элемент <form>:

```
<form class="form-horizontal">
  <div class="form-group">
    <label class="col-sm-2 control-label" for="contact-name">
Name</label>
    <div class="col-sm-10">
      <input class="form-control" type="text" id="contact-name"
placeholder="Full name">
    </div>
  </div>
  <div class="form-group">
    <label class="col-sm-2 control-label" for="contact-email">
Email</label>
    <div class="col-sm-10">
      <input class="form-control" type="text" id="contact-
email" placeholder="Contact email">
    </div>
  </div>
  <div class="form-group">
    <label class="col-sm-2 control-label" for="contact-
email">Message</label>
    <div class="col-sm-10">
      <textarea class="form-control" rows="3" placeholder="
Type your message"></textarea>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <label class="checkbox">
        <input type="checkbox" value="">
        I want to subscribe to receive updates from the
company.
      </label>
```

```
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
        <button class="btn btn-success btn-lg" type="submit">
Submit</button>
      </div>
    </div>
</form>
```

На первый взгляд, форма выглядит как обычно: она имеет два поля ввода, область ввода текста, флажок и кнопку *submit*. Класс `.form-horizontal` отвечает за выстраивание меток и полей ввода рядом по горизонтали. Отметим, что для меток и полей ввода были использованы классы `.col-sm-*` в сетке с 12 частями внутри `.form-group`, как при вложении столбцов.

В теге `.form-group` флажка создан элемент `<div>` со смещением `2` для заполнения части, которая в данном случае не нужна. Обратите внимание, что внутри формы можно использовать те же классы сетки для получения тех же результатов. Чтобы применить тему Bootstrap к флажку, необходимо добавить класс `.checkbox` в метку, обертывающую поле ввода.

CSS-код для этого раздела не особенно нужен, лишь отступы для придания форме разреженности:

```
section#contact form {
  padding-top: 9rem;
  padding-bottom: 3rem;
}
```

Добавление JavaScript

Пришло время включить в игру JavaScript! Создайте файл `main.js` в папке `js`, где уже имеется JavaScript-файл с реализацией фреймворка Bootstrap и библиотеку jQuery. В JavaScript-файл поместите код, который будет выполняться только после полной загрузки документа `document`:

```
$(document).ready(function() {
    // документ готов, поместите сюда ваш код
});
```

Реализуем проверку формы перед ее отправкой. Для этого используем следующий обработчик события отправки формы:

```
$(document).ready(function() {
  $('#contact form').on('submit', function(e) {
    e.preventDefault();
  });
});
```

Как вы, наверное, знаете, строка кода `e.preventDefault()` содержит метод, предотвращающий реакцию по умолчанию, в данном случае это отправка формы.

Затем, создадим нужные переменные и добавим проверку формы:

```
$(document).ready(function() {
  $('#contact form').on('submit', function(e) {
    e.preventDefault();
    var $form = $(e.currentTarget),
        $email = $form.find('#contact-email'),
        $button = $form.find('button[type=submit]');

    if($email.val().indexOf('@') == -1) {
       vaca = $email.closest('form-group')
       $email.closest('.form-group').addClass('has-error');
    } else {
       $form.find('.form-group').addClass('has-success')
.removeClass('has-error');
       $button.attr('disabled', 'disabled');
       $button.after('<span>Message sent. We will contact
you soon.</span>');
    }
  });
});
```

Итак, сначала были созданы переменные для поля `email` и элемента `button`. После этого была реализована очень простая проверка присутствия символа `@` в поле `email`. Если этот символ отсутствует, в родительский блок `.form-group` поля добавляется класс `.has-error`. Это приведет к окрашиванию элементов формы внутри группы в красный цвет, как показано на рис. 4.10.

Добавьте загрузку JavaScript-файла в файл `contact.html`, сразу после загрузки `bootstrap.js`:

```
<script src="js/bootstrap.js"></script>
<script src="js/main.js"></script>
```

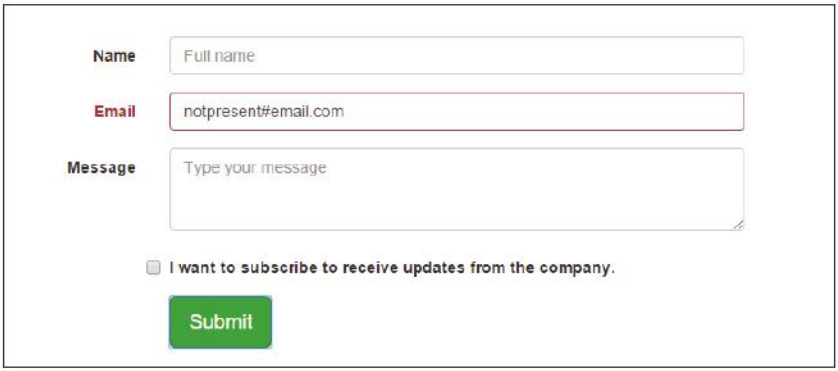


Рис. 4.10.

Если символ `@` присутствует в поле, проверка считается успешной и выполняется эмуляция отправки формы. При этом во все группы `.form-group` добавляется класс `.has-success`, окрашивающий их в зеленый цвет. Затем в кнопку добавляется атрибут `disabled`, вынуждающий Bootstrap изменить ее поведение и внешний вид.

И, наконец, после кнопки вставляется небольшое сообщение, уведомляющее, что сообщение было успешно отправлено. На рис. 4.11 показана форма после успешной отправки сообщения:

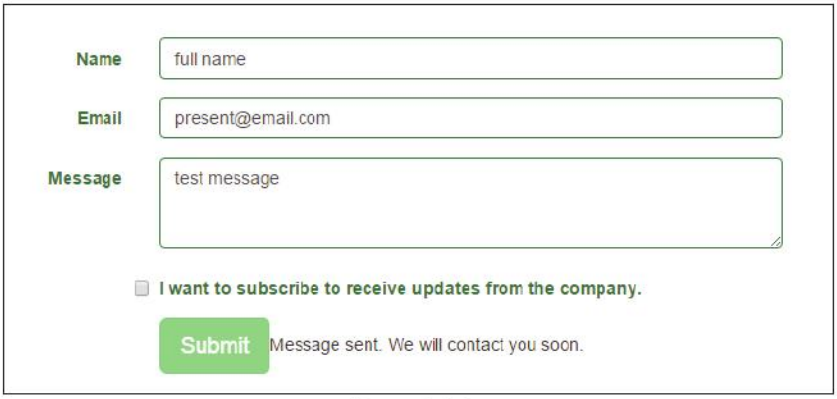


Рис. 4.11.

Форма регистрации

А теперь, после знакомства с несколькими стилями формы обратной связи, создадим еще одну форму – форму регистрации.

Вернемся в HTML-файл целевой страницы и добавим в элемент `.btn`, отвечающий за регистрацию, который находится в представительном заголовке, идентификатор `#sign-btn`:

```
<a id="sign-btn" class="btn btn-default btn-lg"
                 href="#" role="button">Sign in</a>
```

Сразу за элементом `<ul>`, обертывающим кнопки регистрации и входа, поместим код формы регистрации:

```
<form id="signin" class="form-inline text-center hidden-element">
  <div class="form-group">
    <div class="input-group">
      <div class="input-group-addon">@</div>
      <input type="text" class="form-control" id="signin-email"
placeholder="Email">
    </div>
  </div>
  <div class="form-group">
    <div class="input-group">
      <div class="input-group-addon">*</div>
      <input type="password" class="form-control" id="signin-
password" placeholder="Password">
    </div>
  </div>
  <button type="submit" class="btn btn-default">Sign in</button>
</form>
```

Результат показан на рис. 4.12, где форма регистрации находится под кнопками:

Перед завершением компоновки, дадим пояснения к `.input-group`. Bootstrap с помощью класса `.input-group-addon` предоставляет возможность добавления вставок до или после полей ввода. В данном случае перед полями ввода добавлены символы `@` и `*`. Можно было добавить их после, поместив тег с классом `.input-group-addon` после полей ввода.

В таблицу стилей CSS мы добавили простое правило `.hidden-element`. Здесь нельзя использовать Bootstrap-класс `.hidden`, поскольку он использует модификатор `!important`, не позволяющий сделать элемент видимым повторно, без удаления этого класса:

```
.hidden-element {
  display: none;
}
```

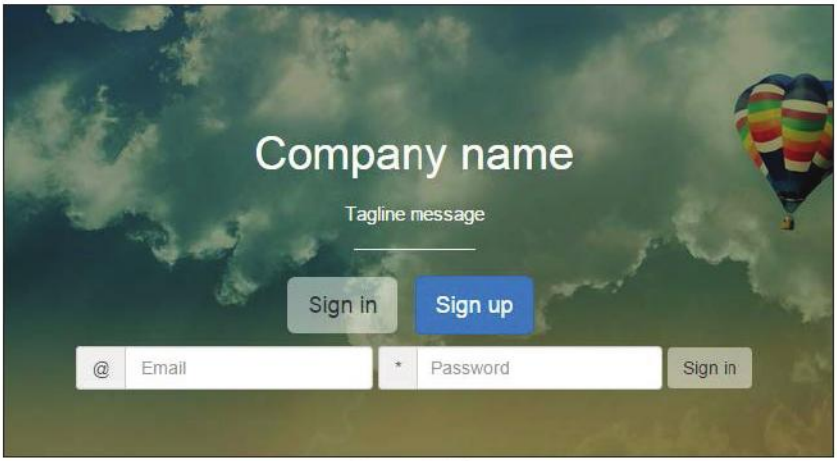


Рис. 4.12.

А теперь добавим немного анимации! Откройте JavaScript-файл и добавьте обработку щелчка на кнопке регистрации:

```
$(document).ready(function() {
    ... // прочий JavaScript-код
    $('#sign-btn').on('click', function(e) {
        $(e.currentTarget).closest('ul').hide();
        $('form#signin').fadeIn('fast');
    });
});
```

При этом будет скрыт элемент `<ul>`, содержащий кнопки регистрации и входа, и показана форма регистрации. Вся изюминка заключается в происходящем с формой! Обновите веб-страницу в браузере, щелкните на кнопке **Sign in** и вы увидите, как появляется новая форма. Далее, мы добавим несколько изображений и посмотрим, чем может помочь Bootstrap.

Изображения

Для работы с изображениями Bootstrap предлагает несколько классов, которые значительно облегчают жизнь разработчиков. Мы уже обсудили использование класса `.img-responsive`, выполняющего масштабирование изображения при наличии атрибутов `max-width: 100%` и `height: auto`.

Фреймворк также предлагает три удобных класса для стилизации изображений. Чтобы задействовать их, поместите следующий код после таблицы цен на целевой странице:

```
<section id="team">
  <div class="container">
    <div class="row">
      <div class="col-sm-12">
        <ul class="list-inline list-unstyled text-center">
          <li>
            <img src="imgs/jon.png" class="img-rounded">
            <h5>Jonny Doo</h5>
            <p>CEO</p>
          </li>
          <li>
            <img src="imgs/jon.png" class="img-circle">
            <h5>Jonny Doo</h5>
            <p>CTO</p>
          </li>
          <li>
            <img src="imgs/jon.png" class="img-thumbnail">
            <h5>Jonny Doo</h5>
            <p>CIO</p>
          </li>
        </ul>
      </div>
    </div>
</section>
```

Здесь просто создается еще один контейнер и строка с одним столбцом `.col-sm-12`. В столбец добавлен список с выстроенными в линию элементами, содержащими изображения с разными классами. Класс `.img-rounded` закругляет углы изображения, `.img-circle` придает изображению форму круга, а `.img-thumbnail` добавляет изображению красивую рамку с закругленными углами, как показано на рис. 4.13.

Рис. 4.13 демонстрирует внешний вид этого раздела. Нужно также добавить CSS-код для увеличения полей и отступов, и настройки шрифта:

```
section#team ul {
  margin: 5rem 0;
}
```

```
section#team li {
  margin: 0 5rem;
}
section#team h5 {
  font-size: 1.5rem;
  font-weight: bold;
}
```

Рис. 4.13.

Bootstrap неплохо справляется и с работой с изображениями, облегчая и ускоряя процесс разработки. Кстати, в Bootstrap имеется масса других вспомогательных классов, преследующих ту же цель. С некоторыми из них мы уже знакомы, а теперь рассмотрим другие.

Вспомогательные классы

Вспомогательные классы Bootstrap помогают выполнять определенные настройки. Они задумывались как средство, позволяющее уменьшить количество повторяющихся CSS-правил. Их цель все та же: ускорение разработки.

Плавающие и центрируемые блоки

Выше уже рассматривались классы `.pull-left` и `.pull-right`, обеспечивающие автоматическое смещение HTML-элементов влево или вправо. Для центрирования блока используется класс `.center-block`.

Чтобы задействовать его, перейдем к столбцу, обертывающему таблицу цен, и заменим классы `.col-sm-10.col-sm-offset-1` на `.center-block`, а в таблицу CSS добавим следующее правило:

```
section#pricing .center-block {
  width: 90%
}
```

Обновив веб-страницу, вы обнаружите, что строки таблицы остались центрированными, но теперь это сделано другим способом.

Контекстные цвета

Имеется возможность применить те же цвета, что использовались для кнопок и таблицы цен, ко всем элементам на странице. Для этого используются следующие классы: `.text-primary`, `.text-success`, `.text-warning`, `.text-info`, `.text-danger` и `.text-muted`.

В только что созданном разделе добавим класс `.text-info` в элементы `<h5>` и класс `.text-muted` в элементы `<p>`:

```
<section id="team">
  <div class="container">
    <div class="row">
      <ul class="list-inline list-unstyled text-center">
        <li>
          <img src="imgs/jon.png" class="img-rounded">
          <h5 class="text-info">Jonny Doo</h5>
          <p class="text-muted">CEO</p>
        </li>
        <li>
          <img src="imgs/jon.png" class="img-circle">
          <h5 class="text-info">Jonny Doo</h5>
          <p class="text-muted">CTO</p>
        </li>
        <li>
          <img src="imgs/jon.png" class="img-thumbnail">
          <h5 class="text-info">Jonny Doo</h5>
          <p class="text-muted">CIO</p>
        </li>
      </ul>
    </div>
</section>
```

Обновите веб-страницу и текст заголовков окрасится в светло-синий цвет, а абзацев – в серый.

Чтобы окрасить фон в контекстные цвета, используйте класс `.bg-*`, указав один из вариантов цвета (`primary`, `info`, `warning` или `danger`).

Интервалы

В Bootstrap 4 появились новые классы полей и отступов. С помощью Sass можно присвоить переменной `$spacer` величину интервала

и для всех элементов с этими классами будет использовано это значение ширины поля, которое по умолчанию равно `1rem`.

В табл. 4.1 перечислены классы для определения полей. В общем случае имена классов следуют шаблону `.m-*-*`, где первый параметр определяет положение, например, верх, низ и так далее, а второй – размер поля. Чтобы разобраться, просмотрите табл. 4.1.

Таблица 4.1.

	Без поля	Поле по умолчанию	Среднее поле (увеличено в 1.5 раза)	Большое поле (увеличено в 3 раза)
Все	`.m-a-0`	`.m-a`	`.m-a-md`	`.m-a-lg`
Верхнее	`.m-t-0`	`.m-t`	`.m-t-md`	`.m-t-lg`
Правое	`.m-r-0`	`.m-r`	`.m-r-md`	`.m-r-lg`
Нижнее	`.m-b-0`	`.m-b`	`.m-b-md`	`.m-b-lg`
Левое	`.m-l-0`	`.m-l`	`.m-l-md`	`.m-l-lg`
Горизонтальные	`.m-x-0`	`.m-x`	`.m-x-md`	`.m-x-lg`
Вертикальные	`.m-y-0`	`.m-y`	`.m-y-md`	`.m-y-lg`

Как показано в табл. 4.2, имена классов отступов следуют тому же шаблону, но с другим префиксом `.p-*-*`. Поскольку интервал по умолчанию равен `1rem`, средний равен `1.5rem`, а большой `3rem`:

Таблица 4.2.

	Без отступа	Отступ по умолчанию	Средний отступ (увеличен в 1.5 раза)	Большой отступ (увеличен в 3 раза)
Все	`.p-a-0`	`.p-a`	`.p-a-md`	`.p-a-lg`
Верхний	`.p-t-0`	`.p-t`	`.p-t-md`	`.p-t-lg`
Правый	`.p-r-0`	`.p-r`	`.p-r-md`	`.p-r-lg`
Нижний	`.p-b-0`	`.p-b`	`.p-b-md`	`.p-b-lg`
Левый	`.p-l-0`	`.p-l`	`.p-l-md`	`.p-l-lg`
Горизонтальные	`.p-x-0`	`.p-x`	`.p-x-md`	`.p-x-lg`
Вертикальные	`.p-y-0`	`.p-y`	`.p-y-md`	`.p-y-lg`

Адаптивные внедряемые элементы

Новая версия Bootstrap 4 позволяет делать адаптивными даже внедряемые элементы. В ней имеются классы для элементов `<iframe>`, `<embed>`, `<video>` и `<object>`. Для достижения нужного результата в соответствующий элемент добавляется класс `.embed-responsive`:

```
<div class="embed-responsive embed-responsive-16by9">
  <iframe class="embed-responsive-item" src="//www.youtube.com/
embed/dQw4w9WgXcQ" allowfullscreen></iframe>
</div>
```

При добавлении класса `.embed-responsive-16by9`, окно видеопроигрывателя будет адаптироваться с сохранением соотношения сторон *16:9*. С помощью классов `.embed-responsive-21by9` и `.embed-responsive-4by3` можно задать соотношение сторон *21:9* и *4:3*, соответственно.

Итоги

В этой главе мы переделали целевую страницу и применили к ней тему Bootstrap. В результате страница стала гораздо привлекательнее. На данный момент у нас имеется приведенная в порядок веб-страница, созданная с помощью методологии «сначала для мобильных».

Кроме того, вы познакомились с созданием форм на трех примерах, одной из которых является дополнительная страница с формой обратной связи. Помимо форм мы начали использовать и JavaScript! В частности, мы реализовали проверку и простую анимацию формы.

Наконец, были описаны способы вывода изображений и ряд вспомогательных классов Bootstrap. В этой главе были перечислены далеко не все вспомогательные классы, но не волнуйтесь, мы еще вернемся к ним в следующей главе.

Если вы думаете, что целевая страница уже прекрасно выглядит, то вас ждут доказательства, что ее можно сделать еще лучше! Об этом пойдет речь в следующей главе, где будут добавлены значки, другие формы, кнопки, навигационные панели.

Примите поздравления! Вы покорили промежуточную высоту. А теперь готовьтесь подняться на следующий уровень. Далее мы рассмотрим еще более сложные элементы и компоненты Bootstrap.

ГЛАВА 5. Что делает его фантастическим

Наконец, пришло время завершить пример целевой страницы. После знакомства с основами Bootstrap, сеточной системой, методологией разработки «сначала для мобильных» и использованием HTML-элементов Bootstrap, мы подошли к концу работы над примером целевой страницы. Сейчас мы займемся более глубоким и подробным изучением замечательного фреймворка Bootstrap.

Эта глава посвящена добавлению компонентов в целевую страницу. Также будет рассмотрен параметр `flexbox`, появившейся в версии 4. По завершении наша целевая страница будет готова к эксплуатации. Основные темы этой главы:

- значки Glyphicon;
- навигационные панели;
- компонент раскрывающегося списка;
- группировка элементов ввода;
- использование Flexbox в Bootstrap.

Использование значков в Bootstrap

Bootstrap – замечательный фрейворк! Он предлагает более 250 готовых к использованию значков (или пиктограмм), причем полноценно поддерживающих масштабирование. Значки созданы из набора Glyphicon Halflings (http://glyphicons.com/). Они реализованы в виде шрифтов, что позволяет изменять их размер и цвет. Для демонстрации их использования обратимся к разделу свойств на целевой странице. Пока этот раздел выглядит достаточно скромно. Добавим несколько шрифтов, чтобы приукрасить его:

```
<section id="features">
  <div class="container">
    <!-- строка 5 -->
    <div class="row">
      <div class="col-sm-12">
        <h3 class="text-center">Features</h3>
        <p class="text-center">Features headline message</p>
      </div>
    </div>

    <!-- строка 6 -->
    <div class="row">
      <div class="col-sm-2 col-md-4">
        <div class="feature">
          <span class="glyphicon glyphicon-screenshot"
aria-hidden="true"></span>
          <span class="feature-tag">Product focus</span>
        </div>
      </div>
      <div class="col-sm-2 col-md-4">
        <div class="feature">
          <span class="glyphicon glyphicon-education"
aria-hidden="true"></span>
          <span class="feature-tag">Teaching as a passion</span>
        </div>
      </div>
      <div class="col-sm-2 col-md-4">
        <div class="feature">
          <span class="glyphicon glyphicon-send" aria-
hidden="true"></span>
          <span class="feature-tag">Spreading knowledge </span>
        </div>
      </div>
      <div class="col-sm-2 col-md-4">
        <div class="feature">
          <span class="glyphicon glyphicon-hourglass"
aria-hidden="true"></span>
          <span class="feature-tag">Save your day time</span>
        </div>
      </div>
      <div class="col-sm-2 col-md-4">
        <div class="feature">
```

```
            <span class="glyphicon glyphicon-sunglasses"
aria-hidden="true"></span>
            <span class="feature-tag">Make it fancy</span>
          </div>
        </div>

        <div class="col-sm-2 col-md-4">
          <div class="feature">
            <span class="glyphicon glyphicon-heart" aria-
hidden="true"></span>
            <span class="feature-tag">Made with love</span>
          </div>
        </div>
        </div>
      </div>
    </div>
</section>
```

Жирным шрифтом выделен код, добавляющий значки. Добавить значок достаточно просто. Выберите нужный вариант на странице http://getbootstrap.com/components/#glyphicons, скопируйте код класса и используйте его в элементе. Обратите внимание, что оба класса – `.glyphicon` и `.glyphicon-*` – должны использоваться в паре.

Свойство aria-hidden

Вы, наверное, заметили, что во все значки добавлено свойство `aria-hidden="true"`. Причина в том, что элементы этих шрифтов интерпретируются как символы Unicode, то есть они могут представлять слова. Поэтому их следует сделать недоступными для экранных дикторов, которые воспроизводят вслух прочитанные символы. Эту недоступность и обеспечивает атрибут `aria-hidden`.

Кроме того, внесем изменения в CSS-файл, добавив несколько правил для текущего раздела. Добавим следующий стиль в файл `base.css`, находящийся в папке `css`:

```
section#features .feature {
  padding-top: 2rem;
  padding-bottom: 4rem;
  text-align: center;
```

```
}
section#features .glyphicon {
  font-size: 2rem;
}
section#features .glyphicon-heart {
  color: #E04C4C;
}
section#features .feature-tag {
  max-width: 10.7em;
  display: inline-block;
  text-align: left;
  margin-left: 1.5em;
  font-size: 1.7rem;
}
```

Здесь демонстрируются интересные возможности значков. Первая из них – возможность изменения размеров значка путем изменения размера шрифта. В данном случае это было сделано с помощью `font-size: 2rem`. Вторая возможность – изменение цвета значка простым добавлением CSS-правила `color`. Здесь это правило применяется к значку с изображением сердца, чтобы сделать его красным: `color: #E04C4C`.

Рис. 5.1.

Использование других наборов значков

Существует множество других наборов значков, которые можно использовать в Bootstrap подобно значкам Glyphicons. Из них особо стоит отметить Font Awesome (https://fortawesome.github.io/Font-Awesome/). Он выделяется среди других тем, что это был первый набор значков, использующий шрифт, а также широким разнообразием значков.

На рис. 5.1 представлен полученный в результате раздел **Features**. Как видите, использовать значки в Bootstrap довольно просто. Кроме того, возможности, предлагаемые фреймворком, отлично подходят для ежедневной смены цветов и размеров значков.

Займемся навигацией

Bootstrap предлагает отличную навигационную панель, которую можно разместить вверху или в любых других местах веб-сайта. Изменим раздел заголовка, превратив его в навигационную панель. Она будет расположена в верхней части веб-страницы и действовать как меню навигации.

Прежде всего, используем элемент `<nav>`, добавим в него классы `.navbar` и `.navbar-default`, необходимые компоненту, и класс `.navbar-fixed-top` для закрепления элемента вверху страницы. Заменим HTML-код раздела `<header>` на следующий:

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="navbar-header">
    <a class="navbar-brand" href="landing_page.html">Company
name</a>
  </div>
  <div class="navbar-right">
      <ul class="nav navbar-nav">
        <li><a href="#about">About</a></li>
        <li><a href="#features">Features</a></li>
        <li><a href="#pricing">Pricing</a></li>
        <li><a href="contact.html">Contact</a></li>
      </ul>
  </div>
</nav>
```

Как уже упоминалось, классы `.navbar` и `.navbar-default` необходимы компоненту навигации. В ссылку **Company name** добавим класс `.navbar-brand`, который выделит имя компании размером шрифта и отступами.

Затем, создадим тег `<div>` с классом `.navbar-right`, чтобы задействовать правила CSS, определяющие отступы, и сместить список вправо, где он находился раньше. В таблицу CSS добавим следующее правило, чтобы создать отступ над блоком `<body>` страницы:

```
body {
  padding-top: 3.6em;
```

```
}

#nav-menu {
  margin-right: 1rem;
}
```

В результате навигационная панель должна выглядеть, как показано на рис. 5.2.

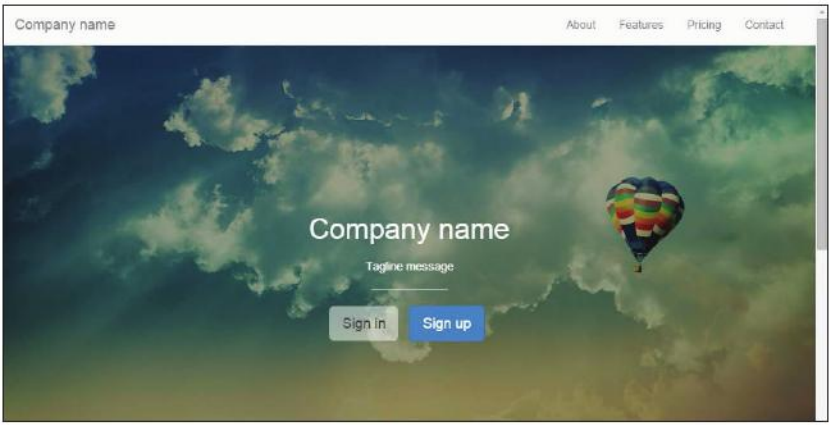


Рис. 5.2.

Свертывание панели навигации

Попробуйте изменить размер веб-страницы, и вы увидите, что для малых областей просмотра, горизонтальный список навигации превращается в вертикальный, как показано на рис. 5.3. К счастью, Bootstrap предоставляет возможность свертывания списка навигационной панели. Необходимая для этого процедура довольно проста и сейчас будет продемонстрирована.

Итак, сделаем панель `.nav-header` сворачиваемой и создадим кнопку переключения `toggle`, показывающую и скрывающую пункты меню:

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="navbar-header">
    <a class="navbar-brand" href="landing_page.html">
Company name</a>
    <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#nav-menu" aria-expanded=
"false">
```

```
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
        </div>

        <div id="nav-menu" class="collapse navbar-collapse navbar-
right">
            <ul class="nav navbar-nav">
              <li><a href="#about">About</a></li>
              <li><a href="#features">Features</a></li>
              <li><a href="#pricing">Pricing</a></li>
              <li><a href="contact.html">Contact</a></li>
            </ul>
        </div>
    </nav>
```

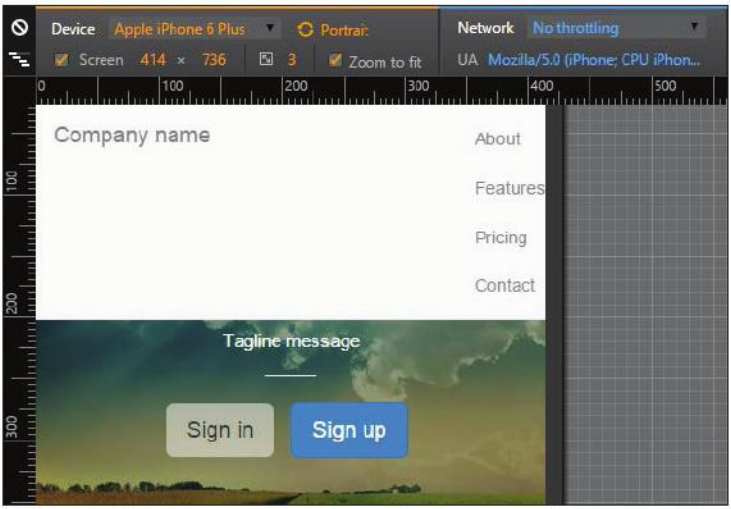


Рис. 5.3.

Новые строки выделены жирным. Здесь появился новый элемент `<button>`, создающий *кнопку со значком бутерброда* (с тремя черточками). В атрибуте `data-target` нужно указать идентификатор сворачиваемого элемента, в данном случае это `#nav-menu`.

Далее, нужно указать, что элемент будет сворачиваться только при попытке отображения в областях просмотра небольшого размера,

для этого к классу `.navbar-right` добавлены классы `.collapse` и `.navbar-collapse`. Теперь навигационная панель должна выглядеть, как показано на рис. 5.4. Ура! И снова Bootstrap сохранил нам массу времени!

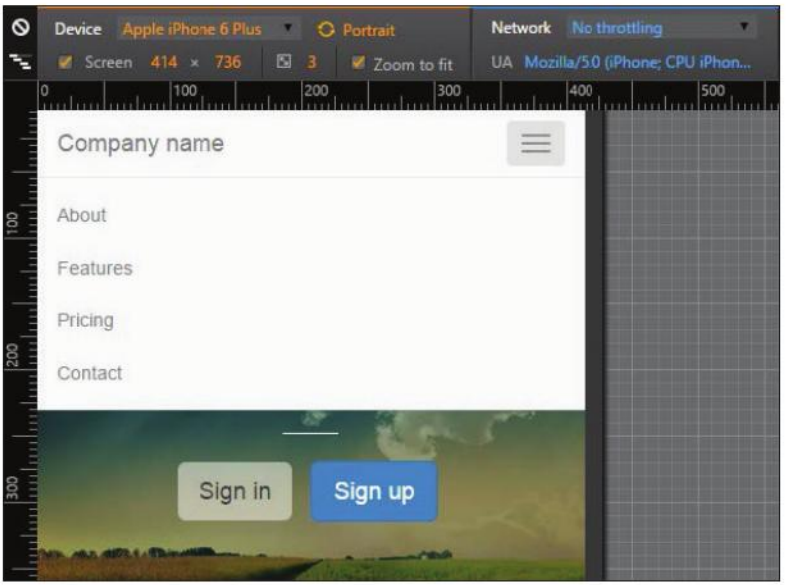


Рис. 5.4.

Используя класс `.navbar-collapse`, не забудьте загрузить JavaScript-библиотеку Bootstrap.

Иные способы размещения

В этом примере навигационная панель была помещена в верхнюю часть веб-страницы, однако ее можно разместить в других местах, например, вдоль нижней границы, использовав класс `.navbar-fixed-bottom`.

Если вы решите разместить панель снизу, не забудьте изменить отступ `<body>` в CSS-коде с `top` на `bottom`.

Панель также может быть статической. Для этого используется класс `.navbar-static-*`, где звездочкой обозначен вариант `top` или `bottom`. При использовании статических навигационных панелей необходимо добавить вложенный контейнер (статический или плавающий):

```
<nav class="navbar navbar-default navbar-static-top">
  <div class="container">
  ...
  </div>
</nav>
```

Цвет навигационной панели

Цвет навигационной панели можно изменить. Версия Bootstrap 3 предоставляет набор инвертированных цветов. Для его использования необходимо добавить класс `.navbar-inverse` в элемент `<nav>`:

```
<nav class="navbar navbar-inverse">
...
</nav>
```

В 4-й версии были добавлены другие варианты цветов. Так, если фон навигационной панели имеет темный цвет, добавление класса `.navbar-dark` сделает текст и прочие элементы белыми. Если фон светлый, применение класса `.navbar-light` приведет к противоположному результату.

Для управления цветом фона предназначен класс `.bg-*`, где звездочка обозначает один из набора цветов Bootstrap. Здесь можно использовать `default`, `primary`, `info`, `success`, `warning` или `danger`:

```
<nav class="navbar navbar-dark bg-danger">
...
</nav>
```

Раскрывающееся меню

Вернемся вновь к кнопкам. Теперь рассмотрим использование кнопок с раскрывающимся меню. Кнопка с раскрывающимся меню прекрасно подходит для присоединения набора пунктов к единственной кнопке. Ее можно использовать в различных ситуациях.

Имейте в виду, что при использовании кнопок с раскрывающимся меню необходимо подключать JavaScript-библиотеку Bootstrap.

Для их использования требуется немного разметки и несколько классов. Добавим кнопку с раскрывающимся меню в новую панель навигации. Ниже приводится законченный HTML-код тега <nav>:

```
<nav class="navbar navbar-default navbar-fixed-top">
   <div class="navbar-header">
      <a class="navbar-brand" href="landing_page.html">
Company name</a>
      <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#nav-menu" aria-expanded
="false">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <!-- <a class="btn btn-primary navbar-btn pull-right"
href="#" role="button">Sign up</a> -->
   </div>

   <div class="btn-group pull-right">
      <button type="button" class="btn btn-primary dropdown-
toggle" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
        Customer area <span class="caret"></span>
      </button>
      <ul class="dropdown-menu">
        <li><a href="#">Action</a></li>
        <li><a href="#">Another action</a></li>
        <li><a href="#">Something else here</a></li>
        <li role="separator" class="divider"></li>
        <li><a href="#">Separated link</a></li>
      </ul>
   </div>

   <div id="nav-menu" class="collapse navbar-collapse navbar-
right">
        <ul class="nav navbar-nav">
          <li><a href="#about">About</a></li>
          <li><a href="#features">Features</a></li>
          <li><a href="#pricing">Pricing</a></li>
          <li><a href="contact.html">Contact</a></li>
        </ul>
```

```
    </div>
  </nav>
```

Новый код выделен жирным. Мы добавили тег `<button>`, за которым следует список `<ul>`, и все это обернуто элементом `div.btn-group`. Компоненты раскрывающихся меню можно создавать только так, и никак иначе.

В таблицу CSS нужно также добавить небольшой интервал, между кнопкой и списком, как показано ниже:

```
nav .btn-group {
  margin: 0.8rem 2rem 0 0;
}
```

Результат представлен на рис. 5.5.

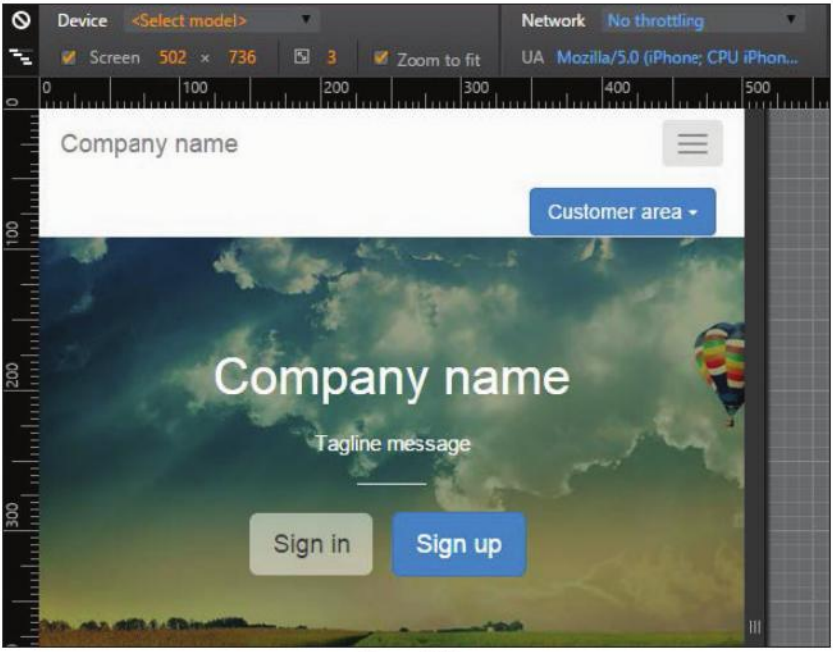


Рис. 5.5.

Однако не все так гладко: если на больших устройствах с новой кнопкой все в порядке, то на малых она съезжает вниз. Исправим этот недостаток с помощью медиа-запроса!

```
@media(max-width: 48em){
  nav .btn-group {
```

```
        position: absolute;
        top: 0;
        right: 4em;
    }
}
```

После внесения исправлений результат должен выглядеть, как показано на рис. 5.6.

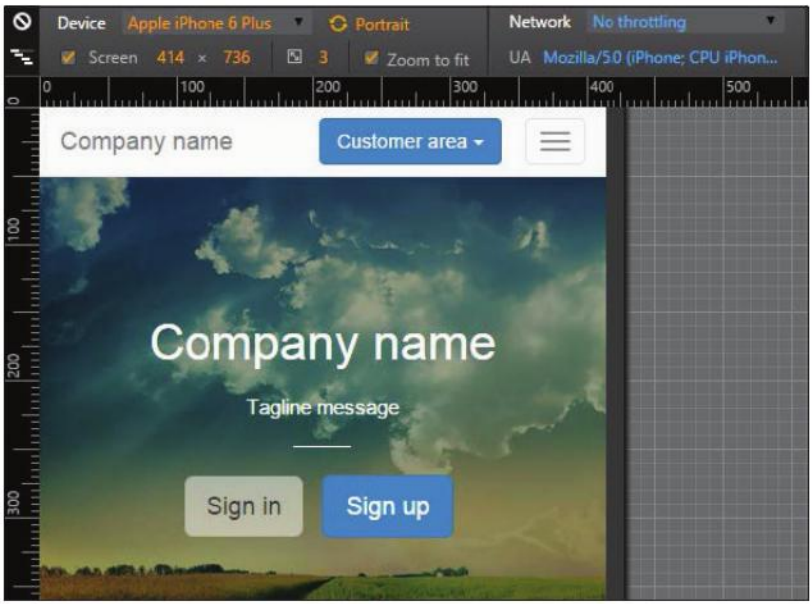


Рис. 5.6.

Настройка кнопок с раскрывающимся списком

Кнопки с раскрывающимся списком имеют несколько параметров настройки. Начнем с возможности разделения. Для этого немного изменим HTML-код:

```
<div class="btn-group pull-right">
  <button type="button" class="btn btn-primary">Customer
area</button>
  <button type="button" class="btn btn-primary dropdown-
toggle" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
```

```
      <span class="caret"></span>
      <span class="sr-only">Toggle Dropdown</span>
    </button>
    <ul class="dropdown-menu">
      <li><a href="#">Action</a></li>
      <li><a href="#">Another action</a></li>
      <li><a href="#">Something else here</a></li>
      <li role="separator" class="divider"></li>
      <li><a href="#">Separated link</a></li>
    </ul>
</div>
```

Основное изменение выделено жирным. В результате создается еще одна кнопка, как показано на рис. 5.7.

Рис. 5.7.

Имеется также возможность реализовать разворачивание меню вверх. Для этого достаточно добавить класс `div.btn-group`:

```
<div class="btn-group dropup»>
  ...
</div>
```

Группировка элементов ввода

Как рассказывалось в предыдущей главе, элементы ввода можно объединять с помощью группирующих компонентов, как это было сделано в форме регистрации. Однако это еще не все. Рассмотрим еще несколько полезных вариантов группировки.

Начнем с группировки элементов ввода и кнопок. Основная идея практически та же: создать `div.input-group` и включить в него элемент ввода и кнопку, как показано ниже:

```
<div class="input-group">
  <input type="text" class="form-control" placeholder="Type
the page title...">
  <span class="input-group-btn">
    <button class="btn btn-success" type="button">Search</
button>
```

```
    </span>
</div>
```

Результат показан на рис. 5.8.

Рис. 5.8.

Хитрость заключается в обертывании кнопки элементом . Если поменять местами элемент ввода и кнопку:

```
<div class="input-group">
  <span class="input-group-btn">
    <button class="btn btn-success"type="button">Search
</button>
  </span>
  <inputtype="text"class="form-control"placeholder="Type
the page title...">
</div>
```

кнопка окажется перед элементом ввода, как показано на рис. 5.9.

Рис. 5.9.

Аналогичным способом можно добавлять в группы кнопки других видов. Для иллюстрации сгруппируем элемент ввода и кнопку с раскрывающимся списком. Для этого заменим в предыдущем примере элемент <button> на кнопку с раскрывающимся списком:

```
<div class="input-group">
  <span class="input-group-btn">
    <div class="btn-group pull-right">
      <button type="button" class="btn btn-primary dropdown-
toggle" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
        Customer area <span class="caret"></span>
      </button>
      <ul class="dropdown-menu">
        <li><a href="#">Action</a></li>
```

```
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">Separated link</a></li>
          </ul>
        </div>
      </span>
      <input type="text" class="form-control" placeholder="Type
the page title...">
</div>
```

Вы легко сможете добавить кнопку практически любого вида слева или справа от элемента ввода. Результат предыдущего примера показан на рис. 5.10:

Рис. 5.10.

А можно добавить две кнопки?

Это станет вашим небольшим заданием. Сможете самостоятельно добавить две кнопки к элементу ввода? Попробуйте добавить несколько кнопок в `.input-group` и посмотрите, что при этом произойдет!

Готовимся к использованию разметки Flexbox!

В версии Bootstrap 4 наконец-то появилась поддержка Flexbox! Однако вам решать – будете ли вы использовать ее. Для начала давайте познакомимся с разметкой Flexbox, если вы этого еще не сделали, а затем перейдем к ее использованию.

Мы не будем использовать целевую страницу для демонстрации разметки Flexbox, так как ее поддержка появилась только в версии Bootstrap 4, а просто постараемся разобраться в этой новой возможности.

Основные понятия Flexbox

Определение Flexbox дано в спецификации CSS3. Главная цель состоит в улучшении организации элементов на веб-странице, делая эту организацию более предсказуемой. Flexbox можно рассматривать как аналог `float`, который предлагает массу вариантов, таких как переупорядочивание элементов, и позволяет избегать известных проблем `float`, таких как необходимость применения `clearfix`.

Для иерархической организации, в первую очередь, необходимо обернуть все flex-элементы (например, заключить столбцы в строки `.row`). Кроме того, можно менять направление и оси обернутых элементов.

Создадим для демонстрации пример разметки HTML. Создайте отдельный файл `flexbox.html`, использовав базовый шаблон, и поместите следующий код в тег `<body>`:

```
<body>
  <div class="wrapping-flex">
    <div class="item1">Item 1</div>
    <div class="item2">Item 2</div>
    <div class="item3">Item 3</div>
  </div>
</body>
```

В данном случае в качестве охватывающего flex-элемента выбран `div.wrapping-flex`. При применении следующего CSS-кода все дочерние элементы будут вытянуты в линию:

```
.wrapping-flex {
  display: -webkit-flex;
  display: flex;
  background-color: #CCC;
}
.wrapping-flex > div {
  background-color: #ECA45A;
  margin: 1rem;
  padding: 1.5rem;
}
```

Внешний вид страницы примера показан на рис. 5.11.

Существует масса вариантов использования Flexbox. Я настоятельно рекомендую, ознакомится с руководством Flexbox по адресу: https://css-tricks.com/snippets/css/a-guide-to-flexbox/, поскольку Flexbox не является основной темой книги.

Рис. 5.11.

Тем не менее, рассмотрим очень важный пример использования Flexbox. Вы сталкивались с проблемами, когда требовалось выровнять один блок `div` внутри другого по вертикали? Я надеюсь, что нет, потому что это весьма болезненный процесс, тем более, если нужна поддержка старых браузеров.

При использовании Flexbox для этого достаточно следующего CSS-кода:

```
.wrapping-flex {
  display: -webkit-flex;
  display: flex;
  background-color: #CCC;
  height: 12rem;
  width: 50%;
  margin-left: 20%;
}
.wrapping-flex > div {
  background-color: #ECA45A;
  margin: 1rem;
  padding: 1.5rem;
}
.wrapping-flex .item2 {
  align-self: center;
  height: 5rem;
}
```

Мы добавили правило `height: 12rem` для обертывающего элемента и определили выравнивание `align-self: center` с высотой `height: 5rem` для элемента `.item2`. Благодаря этому второй вложенный flex-блок `<div>` был выровнен по центру, хотя два других попрежнему растянуты на всю высоту, как показано на рис. 5.12.

Эксперименты с Bootstrap и Flexbox

Версия Bootstrap 4 поддерживает два варианта использования Flexbox. Первый состоит в применении Sass и требует присвоить переменной `$enable-flex` значение `true`.

Рис. 5.12.

Второй заключается в загрузке соответствующей скомпилированной версии CSS – ее можно найти в репозитории Bootstrap (https://github.com/twbs/bootstrap/releases).

Разметка Flexbox имеет ограниченное применение, поскольку не все браузеры готовы ее предоставить. Возможность использования Flexbox имеет смысл рассматривать, если требуется поддержка только новых браузеров, таких как Internet Explorer версии 10 и выше.

Ознакомиться с текущей поддержкой браузерами разметки Flexbox можно по адресу: http://caniuse.com/#feat=flexbox.

Итоги

Эта глава стала большим шагом в освоении более сложных элементов и теоретических понятий. Примите поздравления в связи с завершением первого примера, приведенного в книге!

В этой главе мы познакомились со значками в Bootstrap! Это очень удобный инструмент, позволяющий поместить на страницу совершенный значок совершенным способом, позволяющим указать размер и цвет значка. Из версии Bootstrap 4 исключена встроенная поддержка значков Glyphicons, но их все еще можно использовать как стороннюю библиотеку.

Затем было рассмотрено использование навигационной панели Bootstrap и предложена масса вариантов ее настройки для конкретных случаев. Было описано несколько интересных приемов, позволяющих свернуть меню в панели навигации и добавить к нему несколько компонентов, например, кнопку с раскрывающимся списком.

После этого вновь была затронута тема группировки элементов ввода и приведено еще несколько примеров ее использования, например, группировка полей ввода и кнопок.

И, наконец, были рассмотрены теоретические основы разметки Flexbox и описана возможность ее использования вместе с новой версией Bootstrap 4.

В следующих главах мы рассмотрим еще один пример – настоящее веб-приложение! При его создании будут использованы новые элементы и компоненты Bootstrap. Завершив изучение примера, вы будете обладать знаниями, достаточными для создания собственного веб-приложения!

ГЛАВА 6. Можно ли создать веб-приложение?

Из всех видов веб-страниц наиболее быстрыми темпами растет число веб-приложений. Поэтому здесь будет рассмотрен пример разработки продвинутого веб-приложения. На самом деле, Bootstrap изначально предназначен для приложений такого типа, поскольку в первую очередь создавался для веб-приложения Twitter.

В этой и следующих главах мы пройдем обратный путь. Вместо разработки Bootstrap для Twitter разработаем приложение, подобное Twitter, с помощью Bootstrap. При этом будут использованы другие компоненты и элементы Bootstrap, и рассмотрены следующие темы:

- определение веб-приложения;
- навигационная панель произвольного вида;
- карточки;
- миниатюры;
- разбивка на страницы;
- навигационные цепочки.

Эта глава несколько сложнее предыдущих, но я полагаю, что вы к этому готовы. Итак, можно ли создать веб-приложение?

Что такое веб-приложение

Очевидно, что веб-приложение – это объединение приложения и браузера! То есть, веб-приложение выполняется в веб-браузере. Соответственно, значительная доля вычислений производится на клиентской машине, а сервер занят только обработкой данных.

Интересно отметить , что клиент всегда получает самую последнюю версию приложения, независимо от его желания обновить программное обеспечение. Это ведет к быстрым изменениям и непрерывности процесса разработки приложения.

Создание структуры кода

Итак, начнем новый пример с использования того же HTML-шаблона, что и обычно, сохранив ту же структуру папок и всего прочего:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <title>Web App</title>

    <link rel="stylesheet" href="css/bootstrap.css">
    <link rel="stylesheet" href="css/base.css">

    <!--[if lt IE 9]>
      <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.
min.js"></script>
      <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.
js"></script>
    <![endif]-->
  </head>
  <body>
    <script src="js/jquery-1.11.3.js"></script>
    <script src="js/bootstrap.js"></script>
    <script src="js/main.js"></script>
  </body>
</html>
```

Добавление навигации

Прежде всего добавим навигационную панель, подобную той, что была реализована в предыдущей главе, поместив ее в начало тега <body>:

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
```

```
        <a class="navbar-brand" href="webapp.html">
          <img src="imgs/logo.png" class="img-responsive">
        </a>
        <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#nav-menu"
aria-expanded="false">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <!-- <a class="btn btn-primary navbar-btn pull-left"
href="#" role="button">Sign up</a> -->
      </div>

      <div id="nav-menu" class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
          </ul>
      </div>
    </div>
</nav>
```

Это простая навигационная панель с возможностью свертывания, подобная реализованной в предыдущей главе. Она отличается только наличием логотипа: `image <img src="imgs/logo.png" class="img-responsive">`. CSS-код для настройки логотипа приведен ниже:

```
.navbar-brand img {
  height: 100%
}
```

Далее, внутри тега списка `ul.nav.navbar-nav` нужно создать элементы:

```
<ul class="nav navbar-nav">
  <li>
    <a href="#">
      Home
    </a>
  </li>
  <li>
    <a href="#">
      Notifications
    </a>
```

```
    </li>
    <li>
      <a href="#">
        Messages
      </a>
    </li>
  </ul>
```

Добавим значки для каждого пункта меню. Помните, как это делается? Используем для этого Bootstrap Glyphicons. Добавление значков выделено жирным:

```
<ul class="nav navbar-nav">
  <li>
    <a href="#">
      <span class="glyphicon glyphicon-home" aria-hidden=
"true"></span>
      Home
    </a>
  </li>
  <li>
    <a href="#">
      <span class="glyphicon glyphicon-bell" aria-hidden=
"true"></span>
      Notifications
    </a>
  </li>
  <li>
    <a href="#">
      <span class="glyphicon glyphicon-envelope" aria-hidden=
"true"></span>
      Messages
    </a>
  </li>
</ul>
```

Получившийся результат должен выглядеть, как показано на рис. 6.1.

Рис. 6.1.

Добавление поля ввода для поиска

Добавим в навигационную панель поле ввода для поиска, применив два интересных приема. Во-первых, поле ввода должно быть реализовано как группа элементов, чтобы присоединить значок лупы справа. Во вторых, поле ввода должно быть прижато к правому, а не к левому краю панели `<nav>`. Создадим для этого форму, поместив ее после тега `ul.nav.navbar-nav`:

```
<div id="nav-menu" class="collapse navbar-collapse">
    <ul class="nav navbar-nav">
      ...
    </ul>

    <form id="search" role="search">
      <div class="input-group">
        <input type="text" class="form-control"
               laceholder="Search...">
        <span class="glyphicon glyphicon-search"
              aria-hidden="true"></span>
      </div>
    </form>
</div>
```

Добавим правила CSS, выравнивающие форму по правому краю с небольшим отступом:

```
nav form#search {
  float: right;
  padding: 0.5em;
}
nav form#search .glyphicon-search {
  z-index: 99;
  position: absolute;
  right: 0.7em;
  top: 50%;
  margin-top: -0.44em;
}

nav form#search .input-group .form-control {
  border-radius: 0.25em;
}
```

Теперь обновите веб-страницу. Получившееся поле ввода должно выглядеть, как показано на рис. 6.2.

Рис. 6.2.

Пришло время создания пунктов меню!

Навигационная панель уже похожа на навигационную панель веб-приложения, но это еще не совсем то, что нужно! Настала очередь пунктов меню.

Миниатюра с меню

А теперь создадим нечто экстремальное: добавим миниатюру, объединенную с кнопкой раскрывающегося меню непосредственно перед `form#search`:

```
<div id="nav-options" class="btn-group pull-right">
  <button type="button"
        class="btn btn-default dropdown-toggle thumbnail"
        data-toggle="dropdown" aria-haspopup="true"
        aria-expanded="false">
    <img src="imgs/jon.png">
  </button>
  <ul class="dropdown-menu">
    <li><a href="#">Profile</a></li>
    <li><a href="#">Setting</a></li>
    <li role="separator" class="divider"></li>
    <li><a href="#">Logout</a></li>
  </ul>
</div>
```

Здесь использован шаблон кнопки с раскрывающимся списком (из предыдущей главы), но без компонента `.caret`. Вместо текста использовано изображение из учетной записи пользователя, а в тег `.btn-group` добавлен вспомогательный класс `.pull-right`. Так как кнопка определяется перед формой, на странице она появится после формы.

А теперь займемся правилами CSS: нужно изменить размер изображения и настроить поля и отступы:

```
#nav-options {
  margin: 0.5em;
}

#nav-options button.thumbnail {
  margin: 0;
```

```
  padding: 0;
}

#nav-options img {
  max-height: 2.3em;
  border-radius: 0.3em;
}
```

Результат с добавленной кнопкой показан на рис. 6.3.

Рис. 6.3.

Добавление кнопки Tweet

Последним элементом навигационной панели станет кнопка **Tweet**. Чтобы добавить ее, вставим следующий HTML-код непосредственно перед только что добавленной кнопкой:

```
<button id="tweet" class="btn btn-default pull-right">
  <span class="glyphicon glyphicon-pencil" aria-hidden=
"true"></span> Tweet</button>
```

и добавим небольшие поля в CSS:

```
#tweet {
  margin: 0.5em;
}
```

Наконец навигационная панель содержит все элементы и компоненты, и должна выглядеть, как показано на рис.6.4.

Рис. 6.4.

Настройка навигационной панели

Теперь, когда навигационная панель практически закончена, нужно настроить тему Bootstrap и исправить проблемы, связанные с разными областями просмотра.

Настройка темы

Чтобы панель хоть немного отличалась от стандартной, окрасим ее фон в синий цвет. Для этого добавим несколько простых правил CSS:

```
.navbar-default {
  background-color: #2F92CA;
}

.navbar-default .navbar-nav > li > a {
  color: #FFF;
}
```

После этого сделаем активным один из пунктов навигационного списка, добавив класс `.active` в первый элемент списка `nav` (пункт **Home**). Соответствующая строка выделена жирным:

```
<ul class="nav navbar-nav">
  <li class="active">
    <a href="#">
    <span class="glyphicon glyphicon-home" aria-hidden=
"true"></span>
      Home
    </a>
  </li>
  ... <!-- другие теги li и остальной код -->
</ul>
```

И затем определим еще несколько правил CSS:

```
.navbar-default {
  background-color: #2F92CA;
}

.navbar-default .navbar-nav > li > a {
  color: #FFF;
}

.navbar-default .navbar-nav > .active > a {
  background-color: transparent;
  color: #FFF;
  padding-bottom: 10px;
  border-bottom: 5px solid #FFF;
}
```

Результат показан на рис. 6.5. Пункт **Home** находится в активном состоянии. Чтобы обозначить это, в его изображение добавлена нижняя граница:

Рис. 6.5.

Исправление списка навигационной панели с помощью псевдо-классов

При наведении указателя мыши на любой элемент в списке навигации, он окрашивается неправильным цветом. Чтобы исправить это, воспользуемся переходами CSS3! Ниже приводится полный набор необходимых правил CSS:

```
.navbar-default {
  background-color: #2F92CA;
}

.navbar-default .navbar-nav > li > a,
.navbar-default .navbar-nav > li > a:hover {
  color: #FFF;
  -webkit-transition: all 150ms ease-in-out;
  -moz-transition: all 150ms ease-in-out;
  -ms-transition: all 150ms ease-in-out;
  -o-transition: all 150ms ease-in-out;
  transition: all 150ms ease-in-out;
}

.navbar-default .navbar-nav > .active > a {
  background-color: transparent;
  color: #FFF;
  padding-bottom: 0.62em;
  border-bottom: 0.45em solid #FFF;}

.navbar-default .navbar-nav > .active > a:hover,
.navbar-default .navbar-nav > li > a:hover {
  background-color: transparent;
  color: #F3F3F3;
  padding-bottom: 0.62em;
  border-bottom: 0.45em solid #F3F3F3;
}
```

Переходы CSS3

Добавленные в CSS3 переходы позволяют плавно изменять значения свойств. Чтобы определить переход, нужно указать свойство (в данном случае `all`), протяженность во времени и функцию (здесь используется `ease-in-out`).

С помощью этих правил мы изменили цвет по умолчанию навигационного списка, а определив переходы, создали красивый анимационный эффект – плавное появление нижней границы под элементом списка в момент наведения указателя мыши.

Вам нужен бейдж!

Перед завершением работы над навигационной панелью неплохо было бы добавить бейдж для отображения количества новых оповещений, подобно тому, как это сделано в Twitter. Для этого, нужно освоить использование бейджей Bootstrap.

Добавьте в реализацию пункта оповещений HTML-код, выделенный жирным:

```
<ul class="nav navbar-nav">
  <li class="active">
    <a href="#">
      <span class="glyphicon glyphicon-home" aria-hidden=
"true"></span>
      Home
    </a>
  </li>
  <li>
    <a href="#">
      <span class="badge">5</span>
      <span class="glyphicon glyphicon-bell" aria-hidden=
"true"></span>
      Notifications
    </a>
  </li>
  <li>
    <a href="#">
      <span class="glyphicon glyphicon-envelope" aria-hidden=
"true"></span>
      Messages
    </a>
  </li>
</ul>
```

Определим в CSS-коде параметры позиционирования, отступы и границы:

```
.navbar-nav .badge {
  color: #2F92CA;
```

```
    background-color: #FFF; font-size: 0.7em;
    padding: 0.27rem 0.55rem 0.2rem 0.4rem;
    position: absolute;
    left: 0.37rem; top: 0.7rem;
    z-index: 99;
    border: 0.2rem solid #2F92CA;
}
```

Отлично! Обновите страницу и вы увидите красивый бейдж, как показано на рис. 6.6.

Рис. 6.6.

Решение проблем навигационной панели

Сейчас навигационная панель имеет три недостатка. Догадываетесь, какие?

Это положение кнопки **Tweet** при отображении страницы в малых областях просмотра, сворачивание навигационного меню и цвет кнопки с пиктограммой *бутерброда*.

Начнем с самого легкого – исправим кнопку **Tweet**! Для этого создадим еще один элемент, поместив его слева от кнопки сворачивания, и будем выводить его только при малом разрешении. Прежде всего, добавим класс `.hidden-xs` в кнопку `Tweet`:

```
<button id="tweet" class="btn btn-default pull-right hidden-xs">
  <span class="glyphicon glyphicon-pencil" aria-hidden=
"true"></span>
  Tweet
</button>
```

Затем, в `.navbar-header`, после `button.navbar-toggle`, добавим еще одну кнопку (выделена жирным):

```
<div class="navbar-header">
  <a class="navbar-brand" href="webapp.html">
    <img src="imgs/logo.png" class="img-responsive">
  </a>
  <button type="button" class="navbar-toggle collapsed"
          data-toggle="collapse" data-target="#nav-menu"
          aria-expanded="false">
    <span class="sr-only">Toggle navigation</span>
```

```
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>

    <button id="tweet" class="btn btn-default pull-right
visible-xs-block">
    <span class="glyphicon glyphicon-pencil" aria-hidden=
"true"></span>
      Tweet
    </button>
</div>
```

Итак, мы скрыли кнопку **Tweet** при отображении на сверхмалых устройствах и показали новую, в другом элементе. Выберите область просмотра для мобильных устройств и убедитесь, что положение кнопки исправлено, как показано на рис. 6.7.

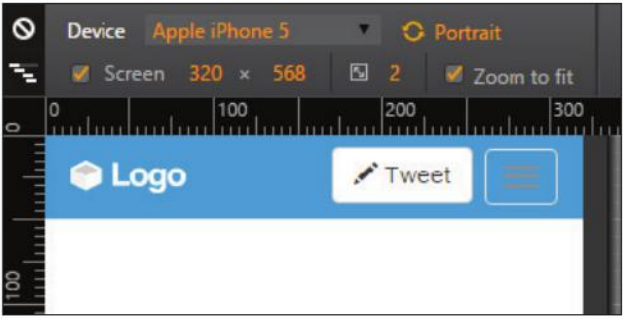


Рис. 6.7.

Далее, исправим цвет кнопки с пиктограммой *бутерброда*, добавив следующие правила:

```
.navbar-header .navbar-toggle,
.navbar-default .navbar-toggle:focus {
  background-color: #57A5D2;
}

.navbar-default .navbar-toggle:hover {
  background-color: #3986B3;
}

.navbar-default .navbar-toggle .icon-bar {
  background-color: #FFF;
}
```

Наконец, настроим сворачивание навигационной панели с помощью вспомогательных классов Bootstrap. Добавим класс `.hidden-xs` в элемент `.nav-options` и класс `.hidden-sm` в элемент `form#search`, сделав их невидимыми на сверхмалых и малых устройствах, соответственно, как это было сделано выше с кнопкой **Tweet**:

```
<div id="nav-options" class="btn-group pull-right hidden-xs">
  ...
</div>

<form id="search" role="search" class="hidden-sm">
  ...
</form>
```

Затем, в навигационном списке `ul.nav.navbar-nav` создадим два пункта, которые заменят невидимые в текущей области просмотра:

```
<ul class="nav navbar-nav">
  ...
  <!-- прочие скрытые элементы списка -->
  <li class="visible-xs-inline">
    <a href="#">
      <span class="glyphicon glyphicon-user" aria-hidden=
"true"></span>
      Profile
    </a>
  </li>
  <li class="visible-xs-inline">
    <a href="#">
      <span class="glyphicon glyphicon-off" aria-hidden=
"true"></span>
      Logout
    </a>
  </li>
</ul>
```

Здесь с помощью класса `.visible.xs-inline` создаются элементы, выстроенные в линию, которые видны при сверхмалых разрешениях.

В заключение уберем нижнюю границу активного пункта списка, так как она плохо смотрится при такой компоновке, и добавим левую, с помощью следующего медиа-запроса:

```
@media(max-width:34em){
  .navbar-default .navbar-nav > .active > a {
```

```
        border-bottom: none;
        border-left: 0.45em solid #FFF;
        padding-left: 0.5em;
    }
}
```

Мы сделали это! Обновите веб-страницу и вы увидите законченную навигационную панель, показанную на рис. 6.8. Она выглядит превосходно!

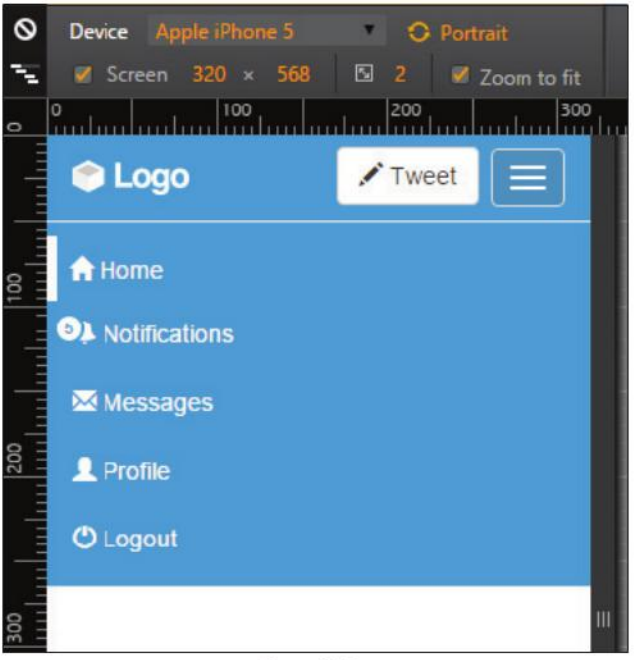


Рис. 6.8.

И снова сетка

Мы закончили навигационную панель. Теперь, пришло время заняться основным разделом. Для этого создадим сетку страницы. Поскольку Twitter использует компоновку, основанную на трех столбцах, создадим такую же. HTML-код основы необходимо поместить после элемента `<nav>`:

```
<div class="container">
  <div class="row">
```

```
    <div id="profile" class="col-md-3 hidden-sm hidden-
xs"> </div>
    <div id="main" class="col-sm-12 col-md-6"> </div>
    <div id="right-content" class="col-md-3 hidden-sm
hidden-xs"> </div>
  </div>
</div>
```

Здесь просто создается контейнер `.container` с единственной строкой `.row`. Элемент `.row` содержит три столбца, первый и последний из которых видимы только на средних и больших устройствах. Это обеспечивают классы `.hidden-sm` и `.hidden-xs`. Когда два крайних столбца невидимы, средний столбец полностью занимает всю строку. На это указывает класс `.col-sm-12`.

В заключение добавим отступ сверху `padding-top` в `<body>`, чтобы скорректировать положение страницы, относительно навигационной панели:

```
body {
  padding-top: 4em;
  background-color: #F5F8FA;
}
```

Эксперименты с карточками

Далее, создадим новый компонент карточки *Card* с информацией о веб-приложении. Отвлечемся от разработки страницы, чтобы подробнее обсудить этот момент.

Карточки – это гибкие расширения для контейнеров, которые включают внутренние элементы, такие как верхний и нижний колонтитулы, а также прочие отображаемые элементы. В Bootstrap 4 имеется готовый компонент `Card`, но так как в книге речь идет о версиях 3 и 4, мы рассмотрим реализацию карточек в обеих версиях.

Карточки в Bootstrap 4

Как уже было упомянуто, Bootstrap 4 содержит готовый компонент `Card`. Чтобы воспользоваться им, нужно создать элемент `div.card` и добавить в него такие элементы, как `.card-block` и `.card-img-top`:

```
<div class="card">
  <img class="card-img-top img-responsive" src="imgs/
landscape.jpg">
```

```
    <div class="card-block">
      <h4 class="card-title">Name</h4>
      <p class="card-text">About text</p>
      <a href="#" class="btn btn-primary">Can add buttons</a>
    </div>
  </div>
```

На экране эта разметка отображается, как показано на рис. 6.9. Как видите, компонент `Card` достаточно прост и понятен. Он поддерживает несколько вариантов использования, которые мы будем рассматривать по мере возникновения такой необходимости.

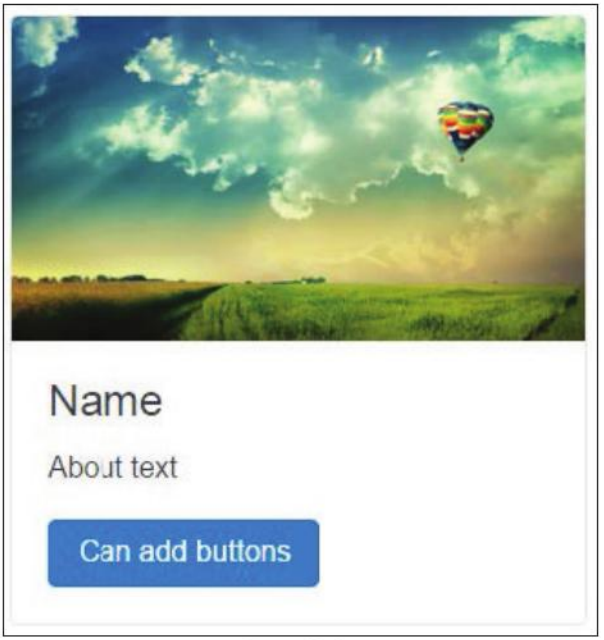


Рис. 6.9.

Создание собственных карточек

В Bootstrap 3 нет компонента `Card`, зато имеются все необходимые инструменты для создания собственного компонента карточек! Поэтому будем следовать поговорке: *свои беды превращай в победы*, и создадим свои карточки!

Для этого используем те же классы и структуры, что применяются в Bootstrap 4, и сделаем все необходимое только с помощью CSS-кода.

То есть, если вы пользуетесь версией Bootstrap 3, предыдущая разметка HTML будет выглядеть точно так же, как показано на рис. 6.10:

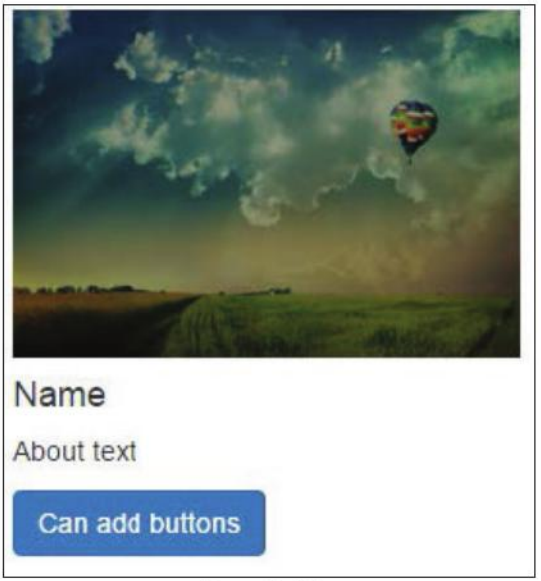


Рис. 6.10.

Сначала определим CSS-класс `.card`:

```
.card {
  position: relative;
  border: 0.1rem solid #e5e5e5;
  border-radius: 0.4rem;
  position: relative;
  background-color: #FFF;
}
```

Затем, добавим два CSS-правила для `img.card-img-top` и `.card-block`:

```
.card-img-top {
  border-radius: 0.4rem 0.4rem 0 0;
}
.card-block { padding: 1.25rem;
}
```

Готово! Мы создали собственный компонент карточки, готовый к использованию с версией Bootstrap 3. На рис. 6.11 представлен конечный результат. Конечно, имеются некоторые отличия в оформлении

и цветах кнопки, но это вызвано различием версий, сам компонент воссоздан идеально.

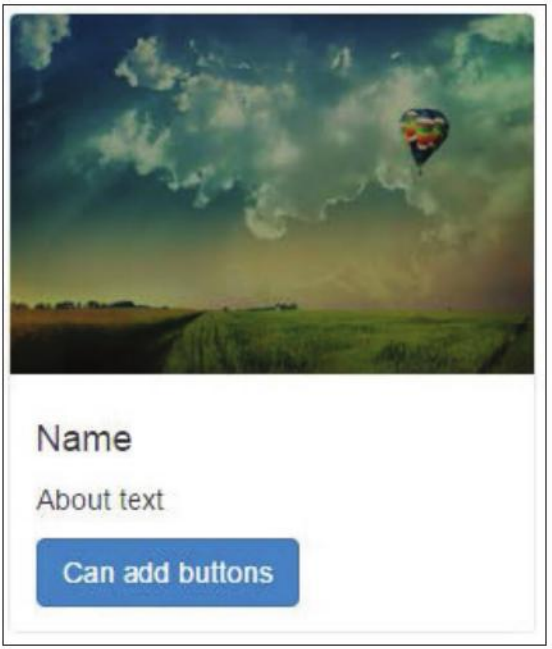


Рис. 6.11.

А вы сможете самостоятельно закончить компонент карточки?

Компонент для версии Bootstrap 3 поддерживает только часть настроек. Сможете доделать поддержку всех остальных? Попробуйте создать CSS-правила для прочих классов, таких как `.card-img-bottom`, `.card-header` и `.card-footer`.

Добавление карточки в веб-приложение

Вернемся разработке к веб-приложения и добавим компонент карточки в блок `div#profile` главного контейнера. Ниже приводится HTML-код для этого раздела:

```
<div id="profile-resume" class="card">
  <img class="card-img-top img-responsive" src="imgs/
```

```
landscape.jpg">
  <div class="card-block">
    <img src="imgs/jon.png" class="card-img">
    <h4 class="card-title">Jonny Doo <small>@jonnydoo</
small></h4>
    <p class="card-text">Dog goes woofy. Did you said
squitly?</p>
    <ul class="list-inline list-unstyled">
      <li id="card-tweets">
        <a href="#">
          <span class="profile-stats">Tweets</span>
          <span class="profile-value">99k</span>
        </a>
      </li>
      <li class="card-following">
        <a href="#">
          <span class="profile-stats">Following</span>
          <span class="profile-value">7</span>
        </a>
      </li>
      <li class="card-followers">
        <a href="#">
          <span class="profile-stats">Followers</span>
          <span class="profile-value">132k</span>
        </a>
      </li>
    </ul>
  </div>
</div>
```

Здесь в блок `.card-block` было добавлено несколько компонентов. Во-первых, это элемент `.card-img` с фотографией из профиля пользователя. Претерпел изменения элемент `.card-title` – в него были добавлены теги `<small>` в `<h4>`. И, наконец, был добавлен список `<ul>` со статистикой учетной записи.

Это фрагмент HTML-кода не содержит никаких секретов, мы просто добавили несколько элементов самым обычным способом. А теперь перейдем к правилам CSS. Прежде всего, изменим параметры позиционирования и размеры элемента `img.card-img`:

```
.card-block img.card-img {
  top: 50%;
  margin-top: -36px; width: 72px;
  border: 3px solid #FFF;
```

```
  border-radius: 0.4rem;
  float: left;
  position: relative;
  z-index: 99;
}
```

Так как название смещено вправо, скорректируем выравнивание `.card-title` и добавим отступ для `.card-text`:

```
.card-block .card-title {
  float: left;
  margin: 0;
  margin-left: 0.5em;
}

.card-block .card-title small {
  display: block;
}
.card-block .card-text {
  clear: both;
  padding-top: 0.25em;
   margin-bottom: 1.5em;
}
```

Сможете ли вы изменить блок карточки так, чтобы его можно было использовать с Flexbox?

Здесь возникает другая проблема. Так как вы уже знакомы с Flexbox, попробуйте заменить плавающие блоки в предыдущем коде соответствующими CSS-правилами Flexbox. Имейте в виду, что разметку Flexbox рекомендуется применять с версией Bootstrap 4, но поддерживается она только новыми браузерами.

Карточка получилась похожей на карточку в Twitter слева, осталось только изменить стиль списка. Добавим еще несколько правил CSS:

```
.card-block ul a:hover { text-decoration: none;
}

.card-block ul .profile-stats {
  color: #777;
  display: block;
  text-transform: uppercase;
  font-size: 0.63em;
```

```
}

.card-block ul .profile-value {
   color: #000;
   font-size: 1.2em;
   font-weight: bold;
   color: #2F92CA;
}
```

Отлично сработано! Выглядит красивее, чем компонент Twitter. На рис. 6.12 представлен полученный результат:

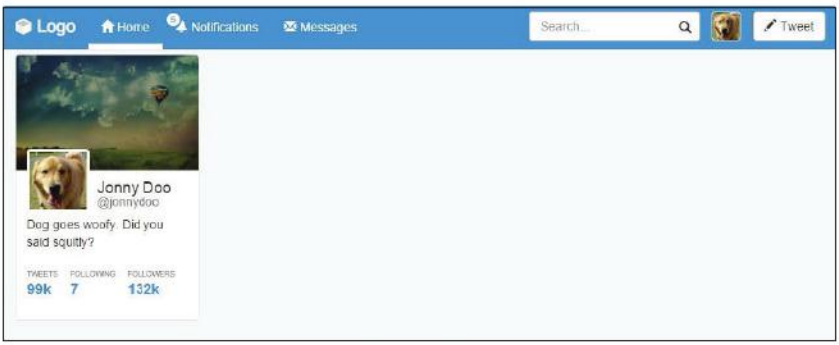


Рис. 6.12.

Карточка с миниатюрами

Вслед за карточкой `#profile-resume` создадим еще одну карточку `#profile-photo`, содержащую фотографии пользователя. Используем уже известную методику, чтобы поместить ее вслед за `#profile-resume`:

```
<div id="profile-photo" class="card">
   <div class="card-header">Photos</div>
   <div class="card-block">
      <ul class="list-inline list-unstyled">
         <li>
            <a href="#" class="thumbnail"><img class="img-
responsive" src="imgs/landscape-02.jpg"></a>
         </li>
         <li>
            <a href="#" class="thumbnail"><img class="img-
responsive" src="imgs/landscape-03.jpg"></a>
         </li>
         <li>
```

```
          <a href="#" class="thumbnail"><img class="img-
responsive" src="imgs/landscape-04.jpg"></a>
        </li>
        <li>
          <a href="#" class="thumbnail"><img class="img-
responsive" src="imgs/landscape-05.jpg"></a>
        </li>
      </ul>
    </div>
</div>
```

Для этой карточки был создан новый элемент `.card-header`. В Bootstrap 4 можно воспользоваться существующим классом, но в Bootstrap 3 придется добавить следующее CSS-правило:

```
.card .card-header {
  border-radius: 0.4rem 0.4rem 0 0;
  padding: .75rem 1.25rem;
  background-color: #f5f5f5;
  border-bottom: 0.1em solid #e5e5e5;
  color: #4e5665;
  font-weight: bold;
}
```

Остальной CSS-код для этой карточки довольно прост. Он задает ширину изображения и определяет поля и отступы:

```
#profile-photo {
  margin-top: 2rem;
}
#profile-photo ul {
  margin: 0;
}
#profile-photo li {
  width: 48%;
  padding: 0;
}
```

Отметим также, что в теге `<a>`, обертывающем изображение, используется класс `.thumbnail` для красивого оформления миниатюр изображений. Его также можно применять к тексту, сопровождающему изображение.

Карточка с фотографиями должна выглядеть, как показано на рис. 6.13. В веб-приложении будут использованы еще несколько карточек, но мы рассмотрим их позднее, когда возникнет такая необходимость.

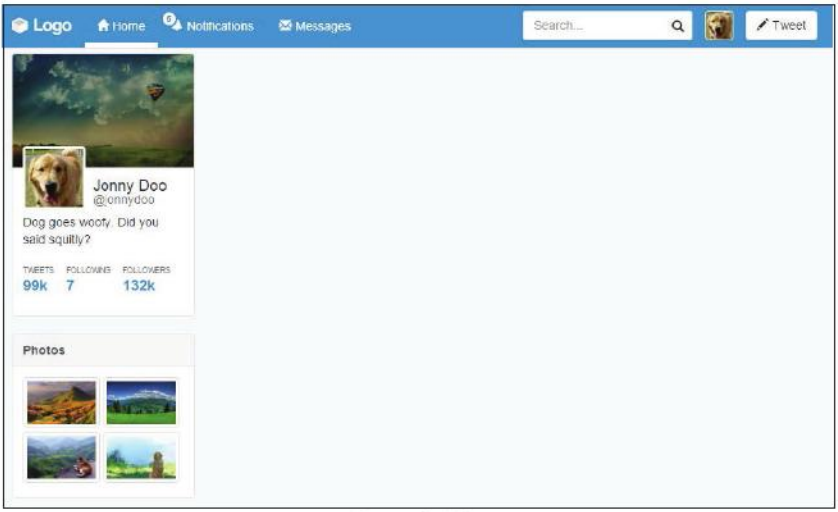


Рис. 6.13.

Наполнение содержимым основного раздела

Двинемся дальше и реализуем вывод основного контента в середине страницы, содержащего ленты и форму ввода новых сообщений.

Создадим элемент ввода для отправки нового сообщения, добавив его в элемент `div#main`:

```
<div id="main" class="col-sm-12 col-md-6">
  <div id="main-card" class="card">
    <form id="new-message">
      <div class="input-group">
        <input type="text" class="form-control"
placeholder= "What is happening?">
        <span class="input-group-addon">
          <span class="glyphicon glyphicon-camera" aria-
hidden="true"></span>
        </span>
      </div>
    </form>
  </div>
</div>
```

Здесь была создана форма и вновь использованы группы элементов ввода, значки и карточки. Почувствовали легкость разработки с Bootstrap? Мы просто разместили элементы с нужными классами и получили превосходный результат. Цвета и отступы для формы определяют следующие правила CSS:

```
form#new-message {
  border-radius: 0.4rem 0.4rem 0 0;
  padding: 1em;
  border-bottom: 0.1em solid #CEE4F5;
  background-color: #EBF4FB;
}

form#new-message .input-group-addon {
  background-color: #FFF;
}
```

На данный момент, результат должен выглядеть, как показано на рис. 6.14.

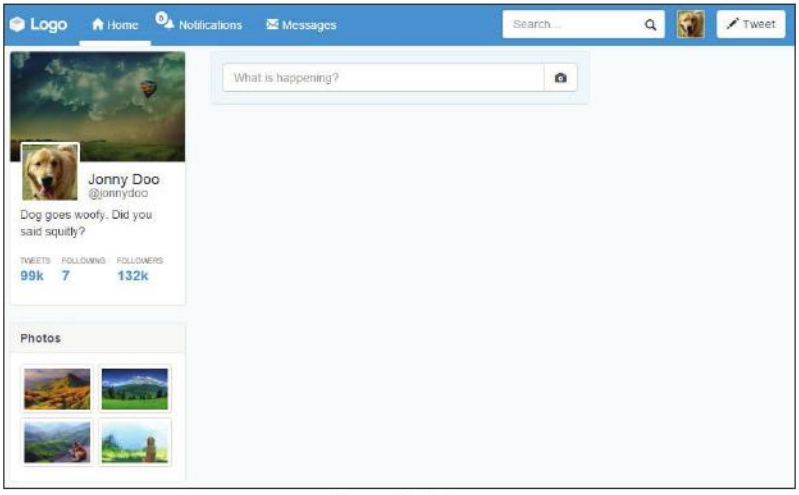


Рис. 6.14.

Теперь создадим следующие элементы.

Создание ленты

Мы уже добавили несколько элементов, но основой страницы является лента сообщений. Поэтому создадим красивую и удобную ленту для веб-приложения.

Как обычно, начнем с HTML-кода. Лента будет отображаться как вертикальный список. Учитывая это, создадим первый элемент списка:

```
<div id="main" class="col-sm-12 col-md-6">
  <div id="main-card" class="card">
    <form id="new-message">
      <div class="input-group">
        <input type="text" class="form-control"
placeholder="What is happening?">
        <span class="input-group-addon">
          <span class="glyphicon glyphicon-camera"
aria-hidden="true"></span>
        </span>
      </div>
    </form>
    <ul id="feed" class="list-unstyled">
      <li>
        <img src="imgs/doge.jpg" class="feed-avatar img-
circle">
        <div class="feed-post">
          <h5>Name <small>@namex - 3h</small></h5>
          <p> You can't hold a dog down without staying
down with him!</p>
        </div>
        <div class="action-list">
          <a href="#">
            <span class="glyphicon glyphicon-share-alt"
aria-hidden="true"></span>
          </a>
          <a href="#">
            <span class="glyphicon glyphicon-refresh"
aria-hidden="true"></span>
            <span class="retweet-count">6</span>
          </a>
          <a href="#">
            <span class="glyphicon glyphicon-star"
aria-hidden="true"></span>
          </a>
        </div>
      </li>
    </ul>
  </div>
</div>
```

Разметка, формирующая список, выделена жирным. Для наглядности в список добавлен типовой элемент сообщения, включающий, изображение, имя, текст и параметры. Изображение в списке стилизовано с помощью класса `.img-circle`.

Добавим несколько правил CSS, чтобы привести стиль списка и изображения в соответствие со стилем страницы:

```
ul#feed {
  margin: 0;
}
ul#feed li {
  padding: 1em 1em;
}
ul#feed .feed-avatar {
  width: 13%;
  display: inline-block; v
  ertical-align: top;
}
```

Эти правила корректируют размеры изображения, а также поля и отступы. Для оформления раздела с текстом сообщения определим следующие правила CSS:

```
ul#feed .feed-post {
  width: 80%;
  display: inline-block;
  margin-left: 2%;
}
ul#feed .feed-post h5 {
  font-weight: bold;
  margin-bottom: 0.5rem;
}

ul#feed .feed-post h5 > small {
  font-size: 1.2rem;
}
```

И, в заключение, добавим стили для элемента `.action-list`:

```
ul#feed .action-list {
  margin-left: 13%;
  padding-left: 1em;
}
ul#feed .action-list a {
  width: 15%;
  display: inline-block;
```

```
}
ul#feed .action-list a:hover {
  text-decoration: none;
}

ul#feed .action-list .retweet-count {
  padding-left: 0.2em;
  font-weight: bold;
}
```

Обновив страницу, вы должны получить результат, показанный на рис. 6.15.

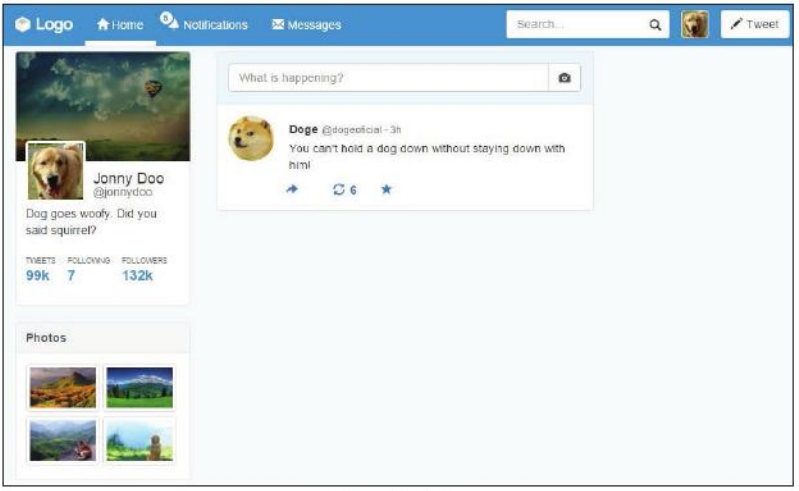


Рис. 6.15.

Потрясающе! Обратите внимание, что все интервалы для элемента сообщения определены исключительно с использованием процентных значений. Это отличный выбор, так как гарантирует правильное изменение размеров страницы с изменением разрешения устройства пользователя.

Осталось решить только одну проблему. Если теперь добавить еще одно сообщение, вы увидите, что они ничем не разделены. Для наглядности добавим в HTML-разметку второе сообщение:

```
<ul id="feed" class="list-unstyled">
  <li>
    <img src="imgs/doge.jpg" class="feed-avatar img-circle">
     <div class="feed-post">
       <h5>Doge <small>@dogeoficial - 3h</small></h5>
```

```
      <p>You can't hold a dog down without staying down
with him!</p>
    </div>
    <div class="action-list">
      <a href="#">
        <span class="glyphicon glyphicon-share-alt"
aria-hidden="true"></span>
      </a>
      <a href="#">
        <span class="glyphicon glyphicon-refresh"
aria-hidden="true"></span>
        <span class="retweet-count">6</span>
      </a>
      <a href="#">
        <span class="glyphicon glyphicon-star" aria-
hidden="true"></span>
      </a>
    </div>
  </li>
  <li>
    <img src="imgs/laika.jpg" class="feed-avatar img-circle">
    <div class="feed-post">
      <h5>Laika <small>@spacesog - 4h</small></h5>
      <p>That's one small step for a dog, one giant leap
for giant</p>
    </div>
    <div class="action-list">
      <a href="#">
        <span class="glyphicon glyphicon-share-alt"
aria-hidden="true"></span>
      </a>
      <a href="#">
        <span class="glyphicon glyphicon-refresh"
aria-hidden="true"></span>
        <span class="retweet-count">6</span>
      </a>
      <a href="#">
        <span class="glyphicon glyphicon-star" aria-
hidden="true"></span>
      </a>
    </div>
  </li>
</ul>
```

Следующие правила CSS вносят небольшие коррективы в отступы и границы между элементами:

```
ul#feed li {
  padding: 1em 1em;
  border-bottom: 0.1rem solid #e5e5e5;
}
ul#feed li:last-child {
  border-bottom: none;
}
```

Результат показан на рис. 6.16.

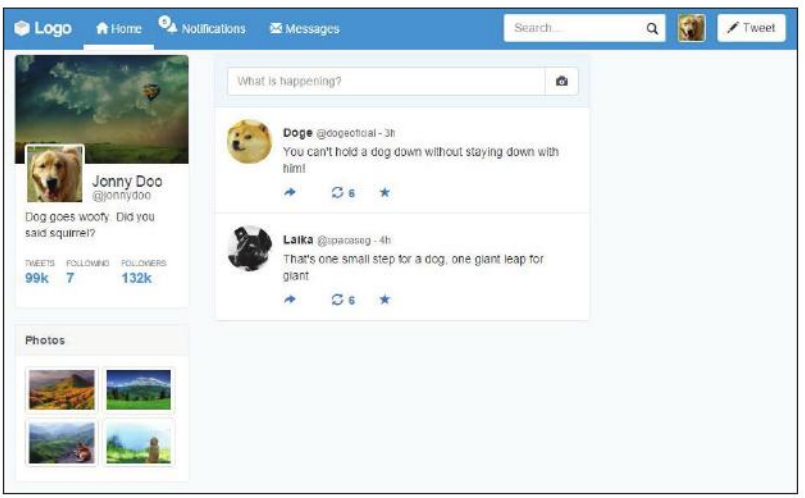


Рис. 6.16.

Разделение списка сообщений на страницы

Веб-приложения, подобные Twitter, обычно используют бесконечную ленту, а не разбивку на страницы, но ведь мы хотим научиться ею пользоваться! Bootstrap предлагает потрясающие варианты разбивки на страницы, поэтому давайте познакомимся с некоторыми из них прямо сейчас.

Прежде всего, создадим компонент и вставим его после блока `div.main-card`:

```
<nav class="text-center">
  <ul class="pagination pagination-lg">
```

```
    <liclass="disabled"><ahref="#"aria-label="Previous">
<span aria-hidden="true">&laquo;</span></a></li>
    <li class="active"><a href="#">1 <span class="sr-only">
(current)</span></a></li>
     <li><a href="#">2</a></li>
     <li class="disabled"><a href="#">...</a></li>
     <li><a href="#">3</a></li>
     <li><a href="#">4</a></li>
    <li> <a href="#" aria-label="Next"><span aria-hidden=
"true">&raquo;</span></a></li>
  </ul>
</nav>
```

Здесь необходимо сделать несколько пояснений. Во-первых, для центрирования блока ссылок на страницы используется вспомогательный класс `.text-center`. Это нужно потому, что `ul.pagination` не использует стиль `display: inline-block`.

Во-вторых, вновь созданный список `<ul>` с классом `.pagination` управляется фреймворком. Кроме того, добавление класса `.pagination-lg` увеличивает размеры элементов со ссылками на страницы.

И, наконец, класс `.disabled` скрывает ссылку на предыдущую страницу элемент и элемент с многоточием .... Кроме того, страница 1 в списке выделяется как активная изменением цвета фона.

Результат добавления постраничного просмотра показан на рис. 6.17.

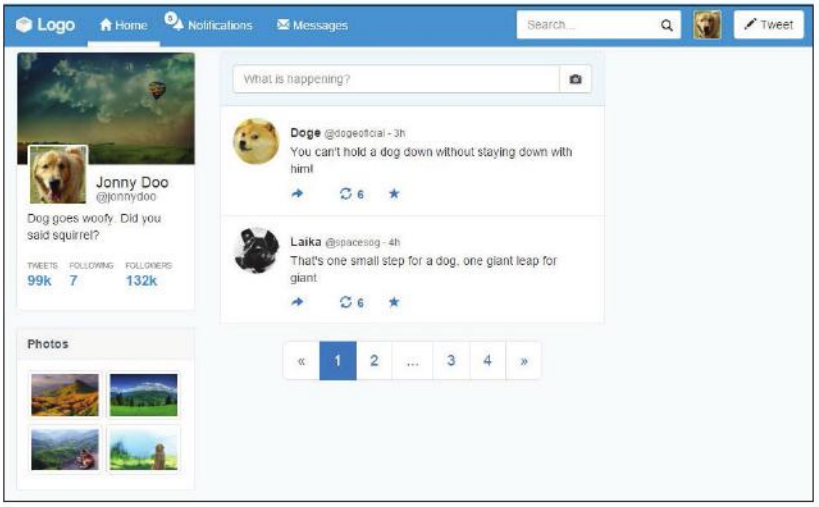


Рис. 6.17.

Создание навигационной цепочки

Для демонстрации навигационных цепочек Bootstrap добавим одну из них в наше веб-приложение. Обратите внимание, что этот шаг выполняется исключительно в учебных целях, чтобы при необходимости вы могли создавать навигационные цепочки сами.

Так же как для разделения на страницы, фреймворк Bootstrap предоставляет свой компонент навигационной цепочки. Для его использования создадим упорядоченный список непосредственно после открывающего тега `div#main`:

```
<div id="main" class="col-sm-12 col-md-6">

  <ol class="breadcrumb card">
    <li><a href="#">Home</a></li>
    <li><a href="#">Profile</a></li>
    <li class="active">Feed</li>
  </ol>
  ...
</div>
```

Одной из замечательных особенностей навигационных цепочек в Bootstrap является автоматическое добавление разделителей посредством параметров `:before` и `content`, поэтому нам не придется беспокоиться о них.

Обратите внимание, что для сохранения единства стиля веб-приложения в навигационную цепочку добавлен класс `.card`. Законченная навигационная цепочка показана на рис. 6.18:

Добавление контента в колонку справа

Итак, почти все готово. Пришло время наполнить колонку справа. Правая колонка будет содержать такие сведения, как «*Who to follow*» (Кто следует за мной) и информацию о странице. Давайте наполним ее!

Вернемся к разметке HTML и добавим еще одну карточку в блок `div.right-content`:

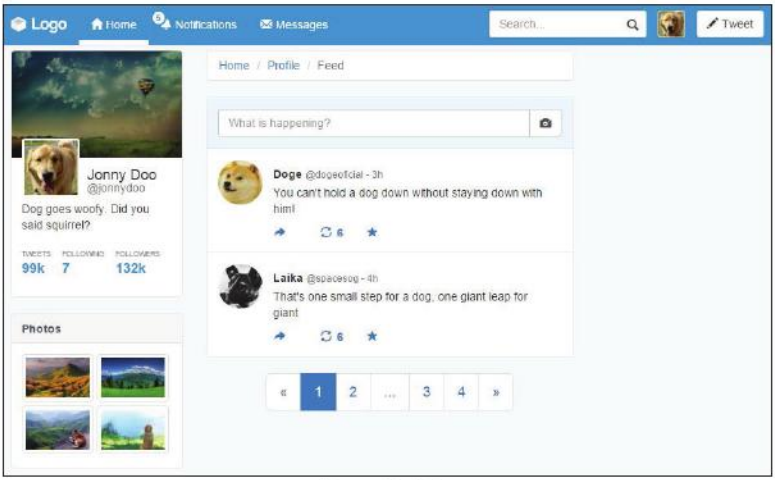


Рис. 6.18.

```
<div id="right-content" class="col-md-3 hidden-sm hidden-xs">
  <div id="who-follow" class="card">
    <div class="card-header">
      Who to follow
    </div>
    <div class="card-block">

    </div>
  </div>
</div>
```

Внутри `.card-block` создадим вертикальный список:

```
<div id="right-content" class="col-md-3 hidden-sm hidden-xs">
  <div id="who-follow" class="card">
    <div class="card-header">
      Who to follow
    </div>
    <div class="card-block">
      <ul class="list-unstyled">
        <li>
          <img src="imgs/cat.jpg" class="img-rounded">
          <div class="info">
            <strong>Crazy cats</strong>
            <button class="btn btn-default">
              <span class="glyphicon glyphicon-plus" aria-
hidden="true"></span> Follow
```

```
                </button>
              </div>
            </li>
            <li>
              <img src="imgs/ration.jpg" class="img-rounded">
              <div class="info">
                <strong>Free ration alert</strong>
                <button class="btn btn-default">
                  <span class="glyphicon glyphicon-plus" aria-
hidden="true"></span> Follow
                </button>
              </div>
            </li>
          </ul>
        </div>
      </div>
    </div>
```

Итак, результат без применения правил CSS выглядит не очень хорошо, как показано на рис. 6.19. Необходимо исправить это.

Рис. 6.19.

Начнем с добавления полей вокруг элементов списка:

```
div#who-follow li {
  margin-bottom: 2em;
}
div#who-follow li:last-child {
  margin-bottom: 0;
}
```

Затем, ограничим размер изображения и следующего за ним текста:

```
div#who-follow li img {
  width: 26%;
  display: inline-block;
  vertical-align: top;
  margin-right: 2%;
}
div#who-follow li .info {
  width: 68%;
  display: inline-block;
}
```

И в завершение приведем в порядок содержимое элемента `.info`:

```
div#who-follow li .info strong {
  display: block;
  overflow:hidden;
  text-overflow: ellipsis;
}

div#who-follow li .info .glyphicon {
  color: #2F92CA;
}
```

Результат показан на рис. 6.20.

Чтобы закончить с основным содержимым веб-страницы, создадим еще одну карточку со ссылками на справочный раздел, соглашение и политику конфиденциальности, и так далее. Сразу за карточкой `div#who-follow` создадим еще одну:

```
<div id="app-info" class="card">
  <div class="card-block">
    © 2015 SampleApp
    <ul class="list-unstyled list-inline">
      <li><a href="#">About</a></li>
      <li><a href="#">Terms and Privacy</a></li>
```

```
        <li><a href="#">Help</a></li>
        <li><a href="#">Status</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </div>
    <div class="card-footer">
      <a href="#">Connect other address book</a>
    </div>
</div>
```

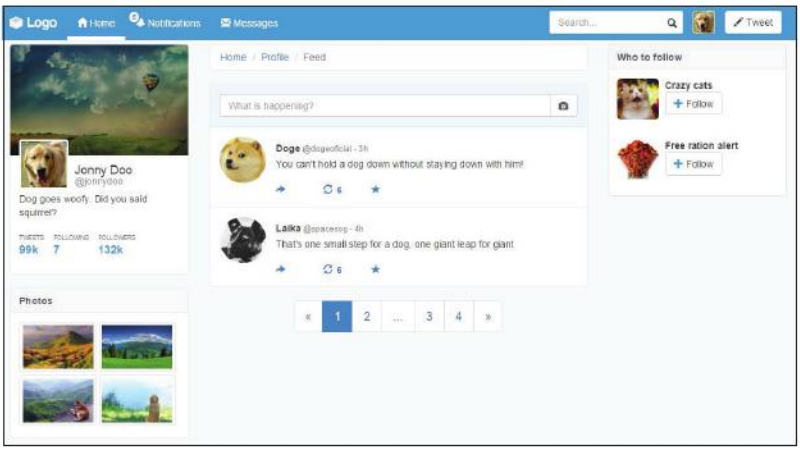


Рис. 6.20.

Прежде всего обратите внимание на элемент карточки `.card-footer`. Если вы используете Bootstrap 3, добавьте следующий CSS-код:

```
.card .card-footer {
  border-radius: 0 0 0.4rem 0.4rem;
  padding: .75rem 1.25rem;
  background-color: #f5f5f5;
  border-top: 0.1em solid #e5e5e5;
  color: #4e5665;
}
```

Для этой карточки необходимо также добавить поля, чтобы отделить ее от карточки выше:

```
div#app-info {
  margin-top: 2rem;
}
```

Выглядит отлично! Работа над большей частью компонентов веб-приложения завершена! На рис. 6.21. представлен законченный результат наших усилий в этой главе. Отличная работа!

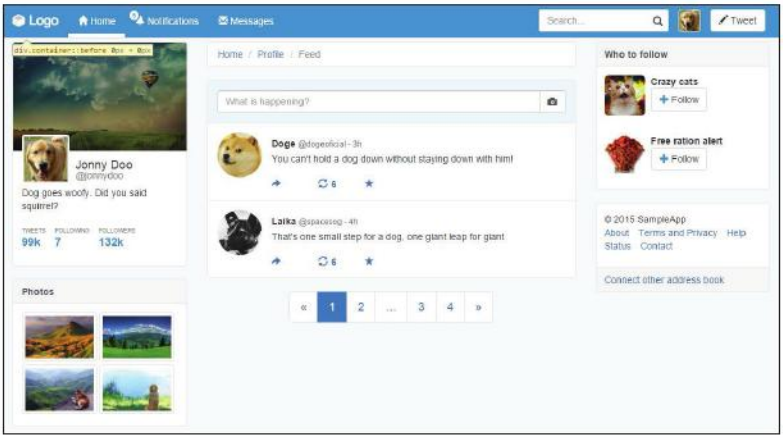


Рис. 6.21.

Итоги

В этой главе начата работа над следующим примером – красивым веб-приложением, похожим на Twitter. Разработка началась с создания всех необходимых компонентов Bootstrap и их настройки. К концу главы была добавлена большая часть компонентов.

Во-первых, была создана и полностью настроена навигационная панель, поддерживающая любые устройства. Особое внимание было уделено отображению компонентов на различных мобильных и настольных устройствах.

Были подробно рассмотрены карточки. Это новый компонент, появившейся в Bootstrap 4, но мы создали собственную версию для поддержки Bootstrap 3. В разных столбцах карточки наполнены разным содержимым, и элементы в них размещены по-разному.

Так же было рассмотрено использование других компонентов Bootstrap, таких как навигационные цепочки, постраничный вывод и миниатюры.

Надеюсь, что теперь вы чувствуете себя достаточно уверенно, так как в следующей главе мы продолжим разработку этого веб-приложения, использовав другие компоненты Bootstrap и еще больше настроек.

ГЛАВА 7. Конечно можно создать веб-приложение!

В этой главе мы завершим разработку веб-приложения, добавив в него новые элементы и компоненты Bootstrap. К концу этой главы вы познакомитесь с большинством элементов, имеющихся в Bootstrap, что сделает вас экспертом по ним, а также получите ответ на вопрос из предыдущей главы: можно ли создать веб-приложение? Конечно можно!

Здесь будут рассмотрены более сложные компоненты и элементы Bootstrap.

Основные темы этой главы:

- использование сообщений Bootstrap;
- настройка сообщений;
- индикаторы хода выполнения операций;
- ключевые кадры CSS;
- компоненты навигации;
- вкладки;
- этикетки и бейджи;
- списки групп.

Несмотря на большое количество тем, они достаточно просты и легки в освоении. Поэтому, я уверен, что вы справитесь с ними.

Сообщения в веб-приложении

В предыдущей главе мы познакомились практически со всеми типичными компонентами страниц. А теперь создадим компоненты, отвечающие за взаимодействие с пользователями. Для начала познакомимся с предупреждающими сообщениями, часто используемыми в любых веб-приложениях.

Для изучения создадим несколько предупреждающих сообщений. Делается это очень просто, но не забудьте добавить загрузку JavaScript-кода Bootstrap, как это делалось во всех примерах в книге.

Обычно предупреждающие сообщения создаются с помощью класса `.alert`. Вместе с этим классом могут использоваться классы, определяющие вид сообщения, например, `.alert-success` для сообщения об успешном завершении действия, и другие подобные классы, такие как, `.alert-info` и `.alert-danger`. Просто замените суффикс на нужный.

А теперь создадим первое предупреждающее сообщение! В веб-приложении из предыдущей главы, сразу после `div#main`, находится список `ol.breadcrumb`. Замена `ol.breadcrumb` на `.alert` приведет к результату, показанному на рис. 7.1.

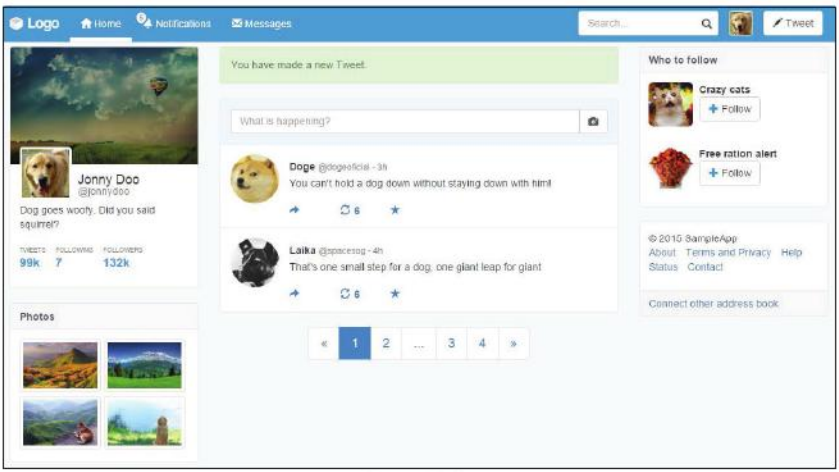


Рис. 7.1.

Разметка HTML для создания этого сообщения выглядит очень просто:

```
<div class="alert alert-success" role="alert">
  You have made a new Tweet.
</div>
```

Как упоминалось выше, просто добавим элемент с классом `.alert` вместе с классом, определяющим вид сообщения, в данном случае с классом `.alert-success`.

Зачем нужен атрибут роли?

В предыдущем примере компонент `.alert` имеет атрибут `role="alert"`. Атрибут роли включен в HTML5 согласно спецификации ARIA 1.0. Сделано это с целью сохранение семантики различных элементов, например, в данном случае, когда используется обычный `<div>` для объявления еще одного семантического элемента, являющегося предупреждающим сообщением.

Отключаемые сообщения

Все таки Bootstrap поразителен! Знаете почему? Потому что для создания предупреждающего сообщения требуется лишь три строки кода!

Еще одной причиной так считать является возможность создавать отключаемые сообщения. Чтобы получить нужный результат, просто добавьте в компонент сообщения строки, выделенные жирным:

```
<div class="alert alert-success" role="alert">
  <button type="button" class="close" data-dismiss="alert"
aria-label="Close"><span aria-hidden="true">&times;</span>
</button>
  You have made a new Tweet.
</div>
```

Они создают кнопку, отключающую компонент с помощью атрибута `data-dismiss="alert"`. Обновите страницу и вы увидите, что сообщения приобрело другой вид, как показано на рис. 7.2.

Рис. 7.2.

Настройка сообщений

А теперь создадим сообщение по собственному рецепту. Поставим перед собой две задачи: добавим в элемент `.alert` заголовок и ссылки.

Сначала создадим элемента заголовка внутри сообщения:

```
<div class="alert alert-success" role="alert">
  <button type="button" class="close" data-dismiss="alert"
```

```
aria-label="Close"><span aria-hidden="true">&times;</span>
</button>
  <h3>Tweet alert</h3>
  You have made a new Tweet.
</div>
```

А затем определим его оформление:

```
.alert h3 {
  margin: 0 0 1rem;
  font-size: 1.4em;
}
```

Конечный результат показан на рис. 7.3.

Рис. 7.3.

Вторая задача, вставка ссылок в компонент, так же легко решается с помощью Bootstrap, с применением его класса `.alert-link` для создания ссылок. Он придает ссылке нужный цвет, соответствующий виду сообщения.

Как результат, разметка HTML получается очень простой:

```
<div class="alert alert-success" role="alert">
  <button type="button" class="close" data-dismiss="alert"
aria-label="Close"><span aria-hidden="true">&times;</span>
</button>
  <h3>Tweet alert</h3>
  You have made a new Tweet.
  <a href="#" class="alert-link">Click here to review your
tweets.</a>
</div>
```

Чтобы закончить с первым сообщением, добавьте последний штрих в виде следующего CSS-кода, затем обновите страницу в браузере и просмотрите получившийся результат, изображенный на рис. 7.4.

```
.alert {
  border-left-width: 0.5rem;
}
```

Рис. 7.4.

Индикатор выполнения

Индикаторы выполнения часто применяются в веб-приложениях, когда, например, необходимо подождать окончания действия, связанного с отправкой данных на сервер, и дать пользователю обратную связь, отразив ход выполнения задания.

Для демонстрации создадим индикатор выполнения, сообщающий о выполнении операции публикации сообщения. Индикатор выполнения может пригодиться в самых разных ситуациях, например, при выполнении операции выгрузки файла на сервер или при загрузке информации веб-клиентом.

В качестве примера, создадим еще одно сообщение, содержащее индикатор выполнения, как бы оповещающий: «Эй, подождите, я еще не закончил!»

Заменим недавно созданный элемент `.alert` следующим:

```
<div class="alert alert-info" role="alert">
  <button type="button" class="close" data-dismiss="alert"
aria-label="Close"><span aria-hidden="true">&times;</span>
</button>
  <h3>Posting new Tweet</h3>
</div>
```

Он отображается как предупреждающее сообщение `.alert-info`, окрашенное в синий цвет. Добавим в него индикатор выполнения:

```
<div class="alert alert-info" role="alert">
  <button type="button" class="close" data-dismiss="alert"
aria-label="Close"><span aria-hidden="true">&times;</span>
</button>
  <h3>Posting new Tweet</h3>
  <div class="progress">
    <div class="progress-bar progress-bar-info"
role="progressbar" aria-valuenow="25" aria-valuemin="0"
aria-valuemax="100" style="width: 25%">
```

```
        </div>
      </div>
    </div>
```

В результате получится индикатор выполнения, показанный на рис. 7.5. Разберемся с каждой частью нового компонента.

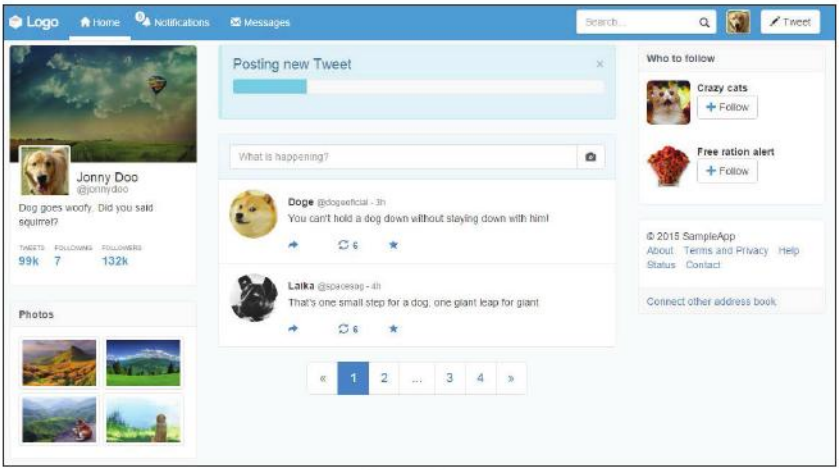


Рис. 7.5.

Внутри компонента предупреждающего сообщения создан элемент `div.progress`, отображающийся как серый прямоугольник, заполняемый в процессе выполнения. Внутрь него помещен еще один тег `div.progress-bar`, служащий для создания внутреннего наполнения, с классом `.progress-bar-info`, окрашивающим индикатор в синий цвет в соответствии с контекстным цветом класса `.info`.

Индикатор выполнения также имеет атрибуты `aria-*`, а его размер устанавливается с помощью стиля `width: 25%`. Чтобы изменить его размер, нужно изменить значение стиля `width`.

Параметры индикатора выполнения

Подобно предупреждающим сообщениям, индикаторы выполнения поддерживают те же контекстные цвета Bootstrap, но с префиксом `.progress-bar-*`. Кроме того, с помощью класса `.progress-bar-striped` индикатор выполнения можно сделать полосатым:

```
<div class="alert alert-info" role="alert">
  <button type="button" class="close" data-dismiss="alert"
aria-label="Close"><span aria-hidden="true">&times;</span>
```

```
</button>
  <h3>Posting new Tweet</h3>
  <div class="progress">
    <div class="progress-bar progress-bar-info progress-
bar-striped active" role="progressbar" aria-valuenow="25"
aria-valuemin="0" aria-valuemax="100" style="width: 25%">
    </div>
  </div>
</div>
```

Наконец, добавив класс `.active` в сочетании с классом `.progress-bar-striped`, полосы можно анимировать. Результат добавления этих классов показан на рис. 7.6.

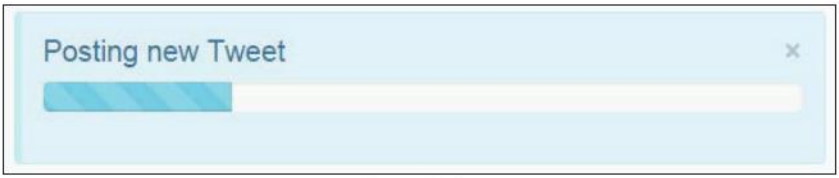


Рис. 7.6.

Анимация индикатора выполнения

Сейчас у нас есть хорошая возможность использовать CSS-анимацию `@keyframes`.

Если заглянуть в определения CSS-классов `.progress-bar-striped` и `.active`, можно увидеть, что Bootstrap выполняет следующую анимацию:

```
.progress-bar.active {
    animation: progress-bar-stripes 2s linear infinite;
}
```

Здесь используется CSS-селектор `@keyframes` с именем `progress-bar-stripes`:

```
@keyframes progress-bar-stripes {
  from {
    background-position: 40px 0;
  }
  to {
    background-position: 0 0;
  }
}
```

Он описывает плавное смещение полосатого фона от отметки 40 пикселей до отметки 0 пикселей в течение двух секунд.

Ладно, приятно было познакомиться с `progress-bar-stripes`, но ведь имеется возможность реализовать и другую анимацию! Создадим свой селектор `@keyframes`:

```
@keyframes w70 {
  from { width: 0%; }
  to { width: 70%; }
}
```

Этот селектор ключевых кадров изменяет ширину индикатора выполнения от 0% до 70% максимального значения при загрузке страницы. Теперь применим новую анимацию к `.progress-bar.active`:

```
.progress-bar.active {
  animation: w70 1s ease forwards,
             progress-bar-stripes 2s linear infinite;
}
```

Этот анимационный эффект выполняется в течение 1 секунды и только один раз, так как указан атрибут `forwards`. Обратите внимание, что эта анимация переопределяет предыдущую, поэтому после новой анимации `w70` следует добавить текущую `progress-bar-stripes`. Обновите страницу и оцените великолепный эффект.

Создание страницы с настройками

Продолжим работу над примером веб-приложения. На этот раз создадим страницу с настройками приложения. У нас уже имеется ссылка на нее в группе кнопок. Помните?

Итак, в папке с HTML-файлом веб-приложения создадим еще один файл, `settings.html`, и обновим ссылку в навигационной панели:

```
<div id="nav-options" class="btn-group pull-right hidden-xs">
  <button type="button" class="btn btn-default dropdown-
togglethumbnail"data-toggle="dropdown"aria-haspopup="true"
aria-expanded="false">
    <img src="imgs/jon.png">
  </button>
  <ul class="dropdown-menu">
    <li><a href="#">Profile</a></li>
```

```
        <li><a href="settings.html">Setting</a></li>
        <li role="separator" class="divider"></li>
        <li><a href="#">Logout</a></li>
      </ul>
    </div>
```

За основу возьмем тот же шаблон, что был использован для веб-приложения, скопировав навигационную панель, компоновку сетки и левый столбец `#profile`. В результате страница с настройками должна выглядеть, как показано на рис. 7.7.

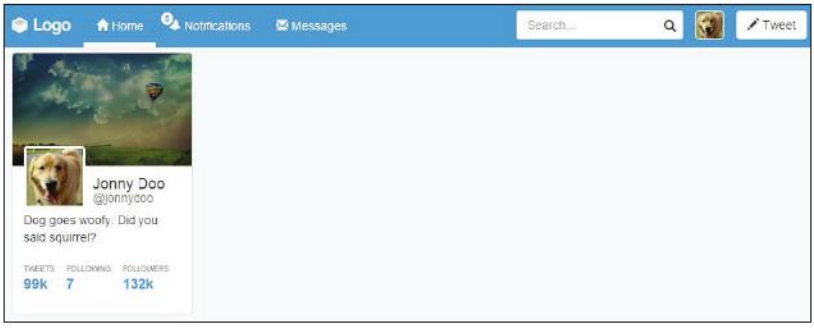


Рис. 7.7.

А теперь наполним эту страницу. Главная наша цель – использовать другие компоненты навигации Bootstrap, более удобные в этом случае.

Вертикальное навигационное меню

Здесь мы впервые воспользуемся навигационным меню Pills с вертикальным расположением пунктов. Pills – это компонент Bootstrap для создания горизонтальных и вертикальных меню. Многие веб-приложения используют его для реализации боковых меню, которое чуть позже будет создано и нами.

В основном для определения навигационных компонентов используется класс `.nav`, за которым следует класс варианта навигации. В данном случае будет использован класс `.nav-pills` для создания нужного эффекта, в сочетании с классом `.nav-stacked`, обеспечивающим вертикальное расположение пунктов. Создадим элемент `.card`, поместив его непосредственно после `#profile`, и добавим навигационное меню:

```
<div id="profile-settings" class="card">
  <ul class="nav nav-pills nav-stacked">
```

```
      <li role="presentation" class="active">
        <a href="#">
          Account
          <span class="glyphicon glyphicon-chevron-right
pull-right" aria-hidden="true"></span>
        </a>
      </li>
      <li role="presentation">
        <a href="#">
          Security
          <span class="glyphicon glyphicon-chevron-right
pull-right" aria-hidden="true"></span>
        </a>
      </li>
      <li role="presentation">
        <a href="#">
          Notifications
          <span class="glyphicon glyphicon-chevron-right
pull-right" aria-hidden="true"></span>
        </a>
      </li>
      <li role="presentation">
        <a href="#">
          Design
          <span class="glyphicon glyphicon-chevron-right
pull-right" aria-hidden="true"></span>
        </a>
      </li>
    </ul>
  </div>
```

а так же следующее правило в CSS-файл:

```
.row .card + .card {
  margin-top: 2rem;
}
```

Это правило автоматически добавит верхнее поле ко всем элементам `.card`, следующим за другим элементом `.card`, как определяет селектор +. Откройте страницу настройки, и вы увидите результат, изображенный на рис. 7.8.

Мы создали список с использованием классов `.nav`, `.nav-pills` и `.nav-stacked`. Список состоит из четырех элементов, каждый из которых содержит ссылку и значок со стрелкой вправо. Для разме-

щения стрелки с правой стороны был использован вспомогательный класс `.pull-right` со значком `.glyphicon-chevron-right`. В первый элемент мы добавили класс `.active`, изменяющий его цвет фона на синий и меняющий цвет его текста и значка.

Рис. 7.8.

Выглядит почти идеально! Необходимо внести небольшие корректировки в CSS, чтобы он выглядел безупречно. Используем селектор `:nth-child` для стилизации элементов по их расположению, например, первого (`:first-child`) и последнего элемента (`:last-child`) меню:

```
#profile-settings .nav-stacked li {
  border-bottom: 1px solid #e5e5e5;
  margin: 0;
}

#profile-settings .nav-stacked a {
  border-radius: 0;
}

#profile-settings .nav-stacked li:first-child a {
```

```
    border-radius: 0.4rem 0.4rem 0 0;
}

#profile-settings .nav-stacked li:last-child {
    border-bottom: 0;
}

#profile-settings .nav-stacked li:last-child a {
    border-radius: 0 0 0.4rem 0.4rem;
}
```

Мы просто удалили ненужные поля и закругления, и добавили границу снизу ко всем элементам списка, кроме последнего. Это обеспечил селектор `#profile-settings .nav-stacked li:last-child`, позволяющий определить отдельное правило для последнего элемента списка.

Также мы изменили свойство `border-radius` для первого и последнего элементов, чтобы создать закругления для всего списка. Получившееся меню `.nav-pill` показано на рис. 7.9.

Рис. 7.9.

Вкладки в середине

В столбец `#main` нужно поместить вкладки с фактическими настройками. В числе прочих навигационных компонентов Bootstrap предлагает компонент с вкладками. Сначала, поработаем над разметкой, а затем займемся стилями CSS и задействуем JavaScript.

Итак, внутрь тега `#main` поместим следующую разметку, определяющую вкладки:

```
<ul id="account-tabs" class="nav nav-tabs nav-justified">
  <li role="presentation" class="active">
    <a href="#account-user">User info</a>
  </li>
  <li role="presentation">
    <a href="#account-language">Language</a>
  </li>
  <li role="presentation">
    <a href="#account-mobile">Mobile</a>
  </li>
</ul>
```

По аналогии с меню, вкладки создаются путем определения списка с классами `.nav` и `.nav-tabs`. Класс `.nav-justified` обеспечивает равномерное распределение вкладок в компоненте.

Кроме того, в список добавлено несколько элементов. Каждая ссылка содержит идентификатор для связывания вкладки с соответствующим контентом. Запомните эту информацию.

Добавим несколько CSS-правил, определяющих цвета границы:

```
#account-tabs > li {
  border-bottom: 0.1rem solid #e5e5e5;
}
#account-tabs a {
  border-bottom: 0;
}
#account-tabs li.active {
  border-bottom: 0;
}
```

Обновив веб-страницу, вы должны увидеть результат, изображенный на рис. 7.10.

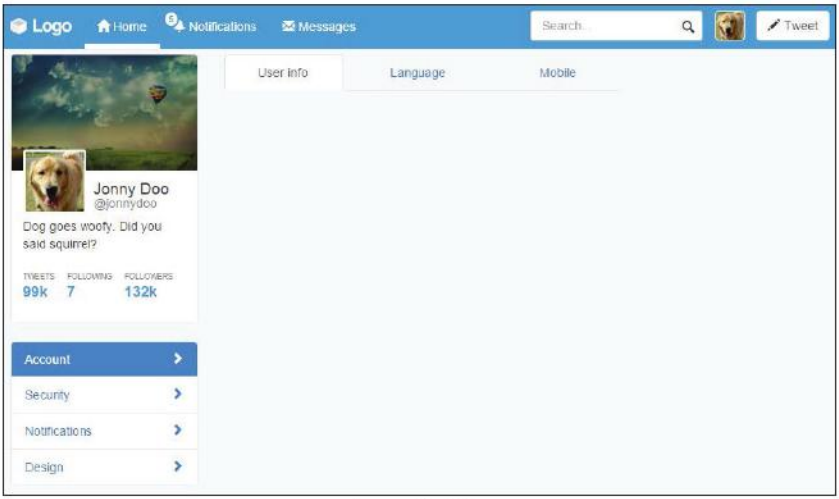


Рис. 7.10.

Добавление содержимого вкладок

Для наполнения вкладок контентом, используем именованные *панели вкладок*. Все панели должны находиться в элементе с классом `.tab-pane` и иметь идентификаторы, соответствующие ссылкам во *вкладках*. Вставим разметку HTML панелей сразу после списка вкладок:

```
<ul id="account-tabs" class="nav nav-tabs nav-justified">
  <li role="presentation" class="active">
    <a href="#account-user">User info</a>
  </li>
  <li role="presentation">
    <a href="#account-language">Language</a>
  </li>
  <li role="presentation">
    <a href="#account-mobile">Mobile</a>
  </li>
</ul>

<div class="tab-content">
  <div role="tabpanel" class="tab-pane active" id="account-
user">
    User info tab pane
```

```
    </div>
    <div role="tabpanel" class="tab-pane" id="account-language">
      Language tab pane
    </div>
    <div role="tabpanel" class="tab-pane" id="account-mobile">
      Mobile tab pane
    </div>
  </div>
```

Обновив веб-страницу, вы обнаружите, что отображается только содержимое вкладки `#account-user`, а щелчки на других вкладках не приводят к переключению между ними. Это произошло из-за того, что компонент не был инициализирован посредством JavaScript.

Использование плагина вкладок Bootstrap

Вкладки можно инициализировать с помощью простого кода на JavaScript:

```
$('#account-tabs a').click(function (e) {
  e.preventDefault()
  $(this).tab('show')
})
```

Однако мы поступим разумнее и задействуем разметку данных. Для этого добавим атрибут `data-toggle="tab"` в элементы ссылок в списке вкладок:

```
<ul id="account-tabs" class="nav nav-tabs nav-justified">
  <li role="presentation" class="active">
    <a href="#account-user" data-toggle="tab">User info</a>
  </li>
  <li role="presentation">
    <a href="#account-language" data-toggle="tab">Language</a>
  </li>
  <li role="presentation">
    <a href="#account-mobile" data-toggle="tab">Mobile</a>
  </li>
</ul>
```

Если вы используете компонент Pills, добавьте атрибут `data-toggle="pill"` в элементы меню, как это было сделано для вкладок.

Обновите страницу в веб-браузере и попробуйте переключать вкладки. Все отлично работает! Для большей оригинальности добавьте класс `.fade` ко всем `.tab-pane`, создающий эффект растворения/проявления при смене вкладок.

Наполнение вкладки с информацией о пользователе

Наполним панель `.tab-pane` с идентификатором `#account-user`. Поскольку это конфигурационное меню, создадим форму с пользовательскими настройками. И снова используем для этого формы Bootstrap. Вы еще помните, как ими пользоваться?

Форма будет содержать три поля ввода (имя пользователя, логин и адрес электронной почты) и кнопку для сохранения изменений. Чтобы расположить поля ввода со своими метками рядом друг с другом, используем класс `.form-horizontal`. Итак, добавим следующий код внутрь элемента `#account-user`:

```
<div id="account-user" role="tabpanel" class="tab-pane active">
  <form class="form-horizontal">
    <div class="form-group">
      <label class="col-sm-3 control-label">Name</label>
      <div class="col-sm-9">
        <input type="text" class="form-control" value=
"Jonny Doo">
      </div>

    </div>
    <div class="form-group">
      <label class="col-sm-3 control-label">Username</
label>
      <div class="col-sm-9">
        <div class="input-group">
          <div class="input-group-addon">@</div>
            <input type="text" class="form-control"
value="jonnydoo">
        </div>
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-3 control-label">Email</label>
      <div class="col-sm-9">
```

```
            <input type="email" class="form-control" value=
"jonnydoo@dogmail.com">
          </div>
        </div>
        <div class="form-group">
          <div class="col-sm-offset-3 col-sm-9">
            <button type="submit" class="btn btn-primary">
              Save changes
            </button>
          </div>
        </div>
      </form>
    </div>
```

Согласно этой разметке, метки занимают четверть ширины строки, а поля ввода – оставшееся пространство. Снова была использована группировка элементов ввода для добавления символа @ перед полем ввода логина.

Каждой метке присвоен класс `.col-sm-3`, а каждому полю ввода класс `.col-sm-9`. Как уже упоминалось, формы поддерживают сеточную систему и поэтому в них можно применять те же классы. Также был добавлен класс `.col-sm-offset-3` для придания кнопке отправки нужного смещения.

На данный момент страница выглядит, как показано на рис. 7.11. Разве она не хороша?!

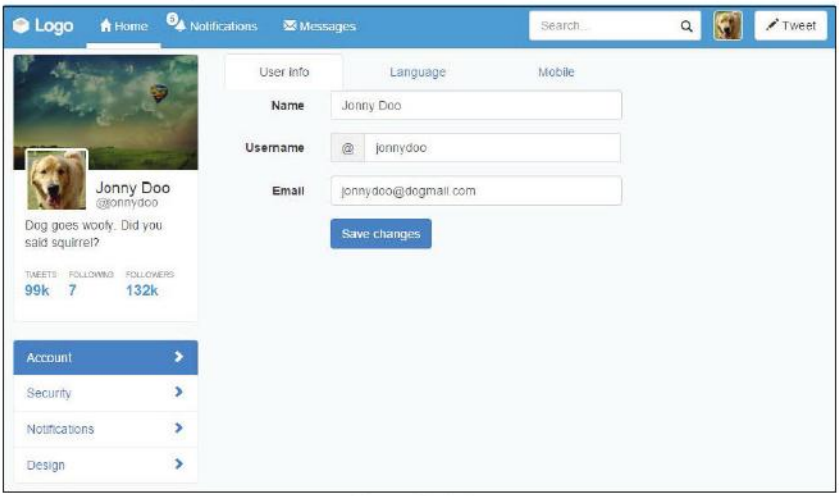


Рис. 7.11.

Перейдем к правилам CSS! Внешний вид страницы можно улучшить, определив отступы. Однако, прежде добавим класс `.card` в `.tab-content`, а затем исправим отступы и поля в `.tab-content`, как это показано ниже:

```
#main .tab-content {
  padding: 2em;
  margin-top: -0.1rem;
}
```

Удалим нижнюю границу списка вкладок и изменим `z-index` вкладок:

```
#account-tabs > li {
  border-bottom: 0;
}
#account-tabs {
  position: relative;
  z-index: 99;
}
```

И, наконец, удалим нижнее поле у кнопки сохранения изменений с помощью еще одного селектора `nth`:

```
#main .form-horizontal .form-group:last-child {
  margin-bottom: 0;
}
```

Отлично! Обновите веб-приложение и вы увидите результат, изображенный на рис. 7.12.

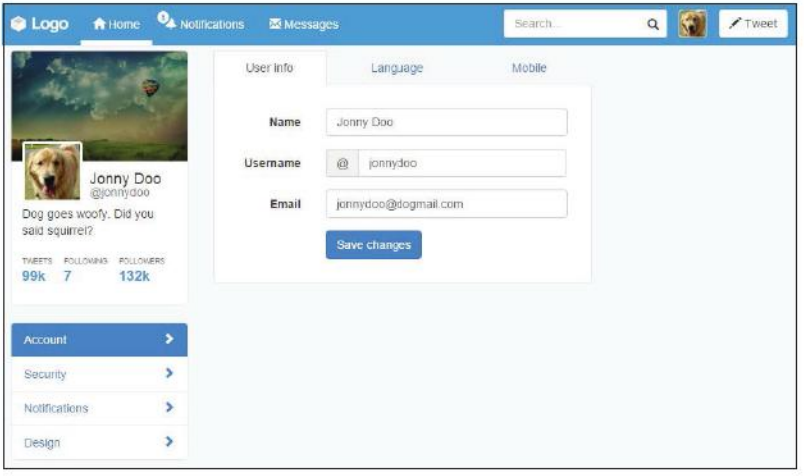


Рис. 7.12.

С этими изменениями форма в `.tab-content` стала выглядеть как карточка. Мы немного скорректировали размеры полей и отступов для улучшения внешнего вида формы. А удаление верхнего поля и установка Z-индекса обеспечили правильное соединение выбранной вкладки с контентом.

Столбец статистики

Чтобы завершить страницу с настройками, нужно заполнить правый столбец. Создадим для этого меню, содержащее общие статистические данные пользователя.

Используем компонент списка групп Bootstrap. Список групп – очень мощный компонент. Он чем-то похож на карточки, но в большей степени ориентирован на создание меню и поддерживает такие особенности, как заголовки, недоступные элементы и многое другое.

Итак, приступим! Создадим HTML-разметку в элементе `#right-content`:

```
<div id="right-content" class="col-md-3 hidden-sm hidden-xs">
  <ul class="list-group">
    <li class="list-group-item list-group-item-info">
      Dog stats
    </li>
    <li class="list-group-item">
      Number of day rides
    </li>
    <li class="list-group-item">
      Captured mice
    </li>
    <li class="list-group-item">
      Postmen frightened
    </li>
    <li class="list-group-item">
      Always alert badge
    </li>
  </ul>
</div>
```

Это простой список с классом `.list-group`, в каждый элемент которого добавлен класс `.list-group-item`. В первый элемент списка добавлен класс контекстного цвета `.list-group-item-info`, чтобы окрасить его в синий цвет, соответствующий указанному контекстному цвету Bootstrap.

Перед тем как продолжить, изменим цвет границ списка групп на соответствующий цвет границы:

```
.list-group-item {
  border-color: #e5e5e5;
}
```

Результат представлен на рис. 7.13. Список групп – очень удобный компонент, который позволяет получить практически тот же эффект, что и компонент навигационного меню, но требует меньше CSS-правил.

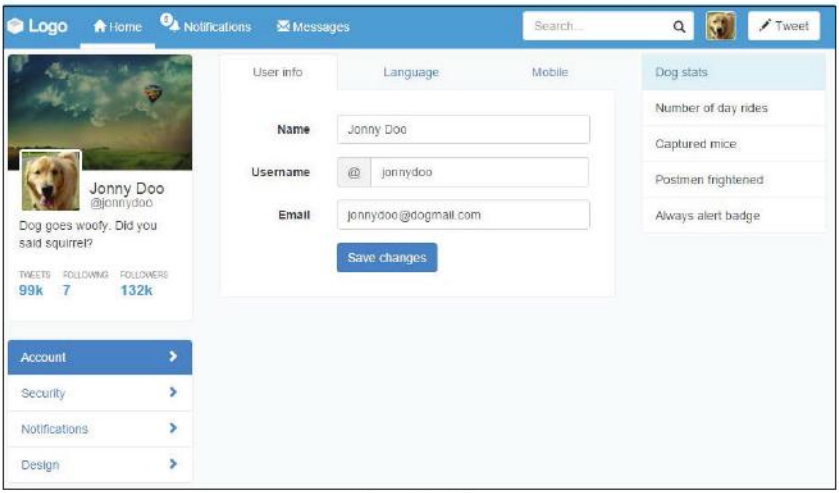


Рис. 7.13.

Этикетки и бейджи

Этикетки и бейджи – компоненты Bootstrap – служат для выделения текста и прочей информации. Они могут быть использованы где угодно, например, в элементе оповещений навигационной панели.

Подобно большинству компонентов Bootstrap, этикетки поддерживают контекстные цвета, а бейджи – нет. Чтобы познакомиться с особенностями использования, добавим их меню статистики справа, потому что, в конце концов, всех нас интересуют наши статистические данные!

Это будет достаточно просто, так как мы уже делали это раньше. Для первых трех пунктов используем этикетки, а для последнего бейдж со значком. Добавьте выделенный код в элемент `.list-group`:

```
<div id="right-content" class="col-md-3 hidden-sm hidden-xs">
  <ul class="list-group">
    <li class="list-group-item list-group-item-info">
      Dog stats
    </li>
    <li class="list-group-item">
      Number of day rides
      <span class="label label-success">3</span>
    </li>
    <li class="list-group-item">
      Captured mice
      <span class="label label-danger">87</span>
    </li>
    <li class="list-group-item">
      Postmen frightened
      <span class="label label-default">2</span>
    </li>
    <li class="list-group-item">
      Always alert badge
      <span class="badge glyphicon glyphicon-star" aria-
hidden="true">
      </span>
    </li>
  </ul>
</div>
```

Здесь следует отметить некоторые моменты. Первый: класс `.label` используется вместе с классом контекстного цвета `.label-*`, где `*` обозначает имя класса контекста.

Второй момент связан с выравниванием. Если сейчас обновить веб-страницу, вы увидите, что Bootstrap содержит CSS-правила для выравнивания бейджей по правому краю, но они не применяются к этикеткам. Поэтому добавим следующее CSS-правило:

```
#right-content .list-group .label {
  float: right;
}
```

Третий момент касается одновременного использования бейджей и значков. Такое возможно, но необходимо настроить CSS-код отображения:

```
#right-content .list-group .label {
  padding: 0.3rem 0.6rem;
```

```
}

#right-content .list-group .badge.glyphicon-star {
  background-color: #f0ad4e;
  padding: 0.4rem;
  padding-left: 0.5rem;
}
```

С помощью этих правил мы скорректировали отступы для этикеток и бейджей, и определили желтый фон для бейджа с изображением звезды. Результат должен выглядеть, как показано на рис. 7.14.

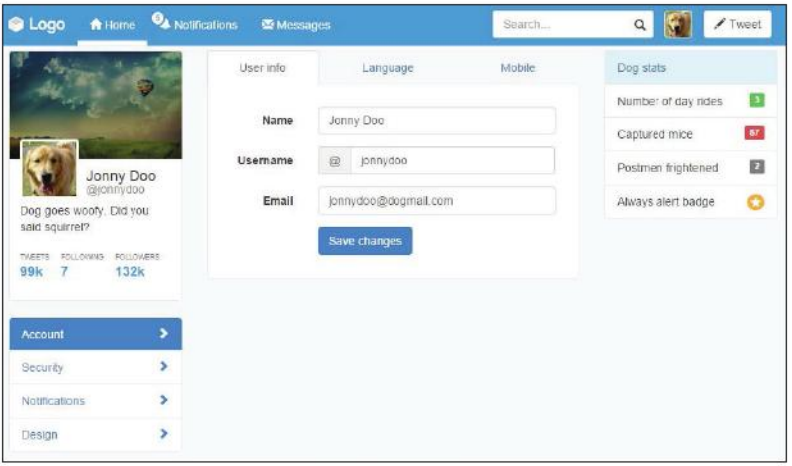


Рис. 7.14.

Круто! Мы создали еще одну страницу для веб-приложения. Меню статистики выглядит вполне привлекательно. Мы не будем рассматривать другие меню страницы настроек, так как они имеют практически такую же структуру, как в рассмотренном примере. Попробуйте создать их самостоятельно.

Итоги

В этой главе был рассмотрен ряд компонентов Bootstrap. Теперь вы можете утверждать, что знакомы со всеми наиболее важными компонентами Bootstrap и даже с новыми, появившимися в версии 4. Примите поздравления, поскольку глава была сложной, и вам необходимо было познакомиться с множеством подробностей о различных компонентах!

Сначала мы исследовали предупреждающие сообщения. Здесь был использован только сам компонент с атрибутами, но в следующих главах он будет использоваться как плагин JavaScript, с множеством настроек и анимацией. Не волнуйтесь! Это лишь анонс будущих глав.

Затем, мы познакомились с индикаторами выполнения и их настройкой, а также с применением к ним анимационных эффектов с помощью CSS-правил. Мы получили нужный результат, и теперь вы сможете использовать индикаторы выполнения в своих страницах.

После этого, мы переключились на страницу с настройками. В ней была использована та же базовая структура, что и для веб-приложения, но изменена компоновка. Мы создали навигационное меню и компонент с вкладками в основной области.

При создании вкладок использовался плагин Bootstrap на JavaScript. Не волнуйтесь, если что-то осталось для вас непонятным – этой теме будет посвящена вся следующая глава.

И, в завершении главы, была описана работа с другим компонентом, списком групп. Этот компонент предоставляет еще один способ создания вертикальных меню.

В пункты меню были помещены этикетки и бейджи – удобные компоненты, предлагаемые Bootstrap.

Следующая глава будет посвящена работе с плагинами Bootstrap на JavaScript. В ней мы продолжим работу над главной страницей веб-приложения и добавим временную шкалу для сообщений. Это будет увлекательно. Встретимся там!

ГЛАВА 8. Работа с JavaScript

Интернет без JavaScript не стал бы тем, чем он является. То же самое можно сказать о Bootstrap. Плагины Bootstrap на JavaScript внесли свой вклад в успех Bootstrap. Они позволяют управлять модальностью, выводить предупреждения и подсказки, и реализуют массу других возможностей, предоставляемых Bootstrap.

Поэтому основное внимание в этой главе будет уделено использованию в веб-приложениях основных плагинов Bootstrap на JavaScript. В предыдущих главах мы уже получили небольшой опыт использования плагинов, а в этой главе рассмотрим их во всех подробностях.

Основные темы этой главы:

- общие сведения об использовании плагинов на JavaScript в Bootstrap;
- атрибуты данных;
- модальность;
- всплывающие подсказки;
- всплывающие панели;
- аффиксы.

Введение в плагины на JavaScript

Как уже упоминалось, Bootstrap включает ряд плагинов на JavaScript. Они загружаются вместе с фреймворком и могут использоваться, при условии подключения к HTML-файлу сценария `bootstrap.js`. Кроме того любой плагин можно загрузить отдельно с веб-сайта Bootstrap.

Минификация JavaScript

На стадии эксплуатации желательно использовать минифицированные версии JavaScript-файлов. Здесь минификация не используется, так как она помешает изучению плагинов, но в реальных приложениях рекомендуется использовать только минифицированные версии.

Зависимости

При настройке среды разработки уже упоминалась необходимость импорта библиотеки JQuery. Фактически, на настоящий момент jQuery является единственной внешней зависимостью Bootstrap.

Загляните в файл `bower.json`, находящийся в репозитории Bootstrap, который содержит подробную информацию о зависимостях.

Bower

Bower – это система управления пакетами клиентских компонентов. Для использования Bower необходимо установить Node и npm. Система Bower разработана программистами Twitter.

Атрибуты данных

В HTML5 появилась возможность добавлять в теги документа нестандартные атрибуты для хранения пользовательской информации. Таким образом, атрибуты с префиксом `data-*` можно использовать для передачи информации в JavaScript, без применения специализированных плагинов. Подавляющее большинство веб-браузеров поддерживает пользовательские атрибуты данных.

На этой идеологии основаны все плагины Bootstrap – они используют только атрибуты данных. Это позволило фреймворку увеличить скорость разработки и моделирования. И именно поэтому вы можете использовать плагины, не написав ни строчки кода на JavaScript.

Для поддержки этой технологии в Bootstrap реализован прикладной программный интерфейс API, открывающий доступ к плагинам исключительно через атрибуты данных. Но иногда требуется отключить доступ через API. Для этого достаточно вставить следующую команду в начало вашего сценария на JavaScript:

```
$(document).off('.data-api');
```

Чтобы отключить API только определенного плагина, нужно добавить пространство имен перед именем плагина. Например, следующая инструкция отключит API предупреждений:

```
$(document).off('.alert.data-api');
```

JavaScript-события в Bootstrap

Существует ряд событий, о которых Bootstrap оповещает все плагины. Их обработчики вызываются, как правило, до и после события. Для демонстрации, допустим, что имеется модальный элемент Bootstrap (не волнуйтесь, управление модальностью будет рассмотрено чуть ниже в этой же главе), и его нужно открыть, как показано ниже:

```
$('#some-modal').modal();
```

В этом случае Bootstrap вызовет события `show.bs.modal` и `shown.bs.modal`. Первое из них будет вызвано перед *открытием модального элемента*, а второе – после. Допустим нужно настроить модальный элемент до его отображения. Для этого следует использовать первое событие:

```
$('#some-modal').on('shown.bs.modal', function(e) {
  // настройка перед отображением
});
```

События можно использовать для всех плагинов. Просто измените пространство имен (в данном случае пространством имен является `.modal`) для получения нужного результата.

Потрясающая реализация модальности в Bootstrap

Пришло время познакомиться с модальностью! Модальность широко используется в современной веб-разработке, а плагины Bootstrap обеспечивают полноценное и простое управление ею. Для демонстрации вернемся к главной странице веб-приложения, содержащей ленты.

Для начала добавим вспомогательный класс `.hide` в блок `div.alert`, созданный в разделе `#main`. Сообщениями мы займемся позже, а сейчас найдите кнопку `#tweet` в навигационной панели. Попробуем реализовать вывод формы ввода сообщения в модальном режиме

при нажатии этой кнопки. Для этого добавим в элемент следующую разметку:

```
<!-- кнопка модального открытия -->
<button id="tweet" class="btn btn-default pull-right
visible-xs-block" data-toggle="modal" data-target="#tweet-
modal">
  <span class="glyphicon glyphicon-pencil" aria-hidden=
"true"></span> Tweet
</button>
```

Здесь реализован вызов элемента с идентификатором `#tweet-modal` и атрибутом данных `data-toggle="modal"` в модальном режиме. То же самое можно реализовать в JavaScript:

```
$('#tweet-modal').modal();
```

Создадим модальную панель, как показано ниже. Поместите ее в конец HTML-файла, за пределами элементов Bootstrap, непосредственно перед загрузкой библиотек JavaScript:

```
<div class="modal fade" id="tweet-modal" tabindex="-1" role=
"dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss=
"modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title">Modal title</h4>
      </div>
      <div class="modal-body">
        Modal content
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default"
data-dismiss="modal">
          Close
        </button>
        <button type="button" class="btn btn-primary">
          Save changes
        </button>
      </div>
    </div>
```

```
        </div>
    </div>
```

Обновите веб-приложение, щелкните на кнопке **Tweet** и произойдет чудо! Этот код достаточно сложен, поэтому разберем его по частям. На рис. 8.1 показан черновой вариант модальной панели. А теперь разберемся, как она реализована.

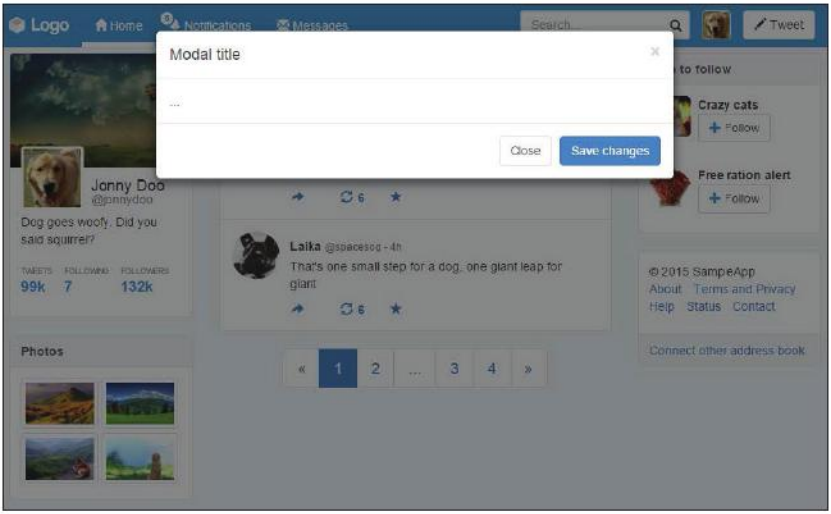


Рис. 8.1.

Общие сведения о модальности и модальный контент

Основной тег модальной панели – тег `<div>` с классом `.modal`. Обратите внимание, что в него также добавлен класс `.fade` для создания эффекта проявления и растворения при открытии и закрытии модальной панели.

Внутрь элемента `.modal` добавлены два вложенных тега. Первый – элемент `.modal-dialog` – обертывает весь модальный диалог. Внутри него создан элемент `.modal-content` с содержимым модальной панели.

Заголовок модальной панели

Внутрь элемента `.modal-content` помещен элемент `.modal-header`. Он является заголовком модальной панели. В этом примере

также добавлена кнопка закрытия, которая скрывает модальную панель с помощью атрибута данных `data-dismiss="modal"`.

Тело модальной панели

Тело модальной панели служит для размещения основного ее содержимого. Что особенно важно, внутри модальной панели можно использовать сетки.

Для этого не требуется создавать контейнер. Просто создайте элемент `.row` внутри элемента `.modal-body` и добавьте столбцы, выделенные жирным в следующем примере:

```
<div class="modal fade" id="tweet-modal" tabindex="-1" role=
"dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss=
"modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title">Modal title</h4>
      </div>
      <div class="modal-body">
        <div class="row">
          <div class="col-sm-2">Use</div>
          <div class="col-sm-4">the</div>
          <div class="col-sm-6">grid system</div>
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default"
data-dismiss="modal">
          Close
        </button>
        <button type="button" class="btn btn-primary">
          Save changes
        </button>
      </div>
    </div>
  </div>
</div>
```

Нижний колонтитул модальной панели

Наконец, в модальную панель можно добавить элемент `.modal-footer` для размещения нескольких компонентов, таких как кнопки, как в предыдущем примере.

Создание собственной модальной панели

Теперь, после знакомства с поддержкой модальности в Bootstrap, настроим созданную панель для конкретного примера. Прежде всего, добавим контент внутрь элемента `.modal-body` и несколько изменим `.modal-header` и `.modal-footer`:

```
<div class="modal fade" id="tweet-modal" tabindex="-1"
role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss=
"modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title">Dog a new tweet</h4>
      </div>
      <div class="modal-body">
        <textarea class="form-control" rows="4" placeholder=
"What you want to bark?" maxlength="140"></textarea>
      </div>
      <div class="modal-footer">
        <span class="char-count pull-left" data-max=
"140">140</span>
        <button type="button" class="btn btn-default"
data-dismiss="modal">
          Close
        </button>
        <button type="button" class="btn btn-primary">
          Tweet
        </button>
      </div>
    </div>
```

```
    </div>
  </div>
```

Здесь в элемент `.modal-header` добавлен заголовок, в элемент `.modal-body` – поле для ввода текста, а в нижний колонтитул – элемент `<span>` с классом `.char-count`.

Наша цель – дать возможность ввести текст сообщения в элемент `textarea` и показать счетчик символов в нижнем колонтитуле, чтобы пользователь мог видеть, сколько символов он еще сможет ввести.

Перейдем в CSS-файл и добавим правило для элемента `.char-count`:

```
#tweet-modal .char-count {
  padding: 0.7rem 0;
}
```

Обновите веб-страницу и вы увидите модальное окно, изображенное на рис. 8.2. А теперь нужно добавить JavaScript-код для подсчета оставшихся символов сообщения.

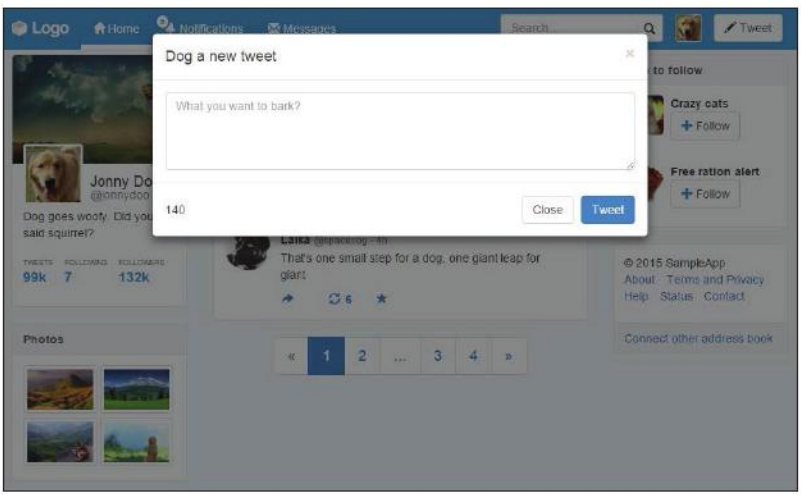


Рис. 8.2.

Для ввода JavaScript-кода откроем (или создадим, если он не был создан ранее) файл `main.js`. Чтобы убедиться, что документ готов к выполнению сценария, поместим в файл следующий код:

```
$(document).ready(function() {
  // место для добавления кода
});
```

Затем создадим функцию, которая будет обновлять количество оставшихся символов при вводе каждой буквы, то есть, обработчик события `keyup`.

Событие `keyup` возбуждает библиотека jQuery, содержащая массу обработчиков событий, возникающих при различных действиях. Имеются и другие события, такие как `click`, `hover` и так далее. Событие `keyup`, используемое в данном случае, возникает, когда пользователь отпускает нажатую клавишу.

Как правило, для установки обработчика события используется функция `.on`, которой в первом аргументе передается тип события (в данном случае `keyup`), а во втором – обработчик (в данном случае, функция). Ниже приводится JavaScript-код, в котором обработчик события выделен жирным:

```
$(document).ready(function() {
  var $charCount, maxCharCount;

  $charCount = $('#tweet-modal .char-count')
  maxCharCount = parseInt($charCount.data('max'), 10);

  $('#tweet-modal textarea').on('keyup', function(e) {
    var tweetLength = $(e.currentTarget).val().length;

    $charCount.html(maxCharCount - tweetLength);

  });
});
```

Здесь обрабатывается событие `keyup`, возникающее после отпускания клавиши и ввода символа. То есть, когда новая длина текста в `textarea` уже будет вычислена.

Инструмент для вывода подсказок

Всплывающие подсказки – очень полезный компонент для подробного описания элемента или веб-страницы. Например, когда имеется изображение и требуется его описание, для этого часто применяется всплывающая подсказка. При наведении указателя мыши на изображение, на экране отображается дополнительная информация.

В данном случае всплывающие подсказки будут использованы для описания кнопок `Reply`, `Retweet` и `Start`, используемых при вводе сообщений. Плагин всплывающих подсказок прост в использовании и полезен во многих ситуациях. Для его использования добавим в сообщения (элементы `<li>` в элементе `ul#feed`) разметку, выделенную жирным:

```
<ul id="feed" class="list-unstyled">
  <li>
    <img src="imgs/doge.jpg" class="feed-avatar img-circle">
    <div class="feed-post">
      <h5>Doge <small>@dogeoficial - 3h</small></h5>
      <p>You can't hold a dog down without staying down
with him!</p>
    </div>
    <div class="action-list">
      <a href="#" data-toggle="tooltip" data-placement=
"bottom" title="Reply">
        <span class="glyphicon glyphicon-share-alt" aria-
hidden="true"></span>
      </a>
      <a href="#" data-toggle="tooltip" data-placement=
"bottom" title="Retweet">
        <span class="glyphicon glyphicon-refresh" aria-
hidden="true"></span>
        <span class="retweet-count">6</span>
      </a>
      <a href="#" data-toggle="tooltip" data-placement=
"bottom" title="Start">
        <span class="glyphicon glyphicon-star" aria-hidden=
"true"></span>
      </a>
    </div>
  </li>

  <li>
    <img src="imgs/laika.jpg" class="feed-avatar img-circle">
    <div class="feed-post">
      <h5>Laika <small>@spacesog - 4h</small></h5>
      <p>That's one small step for a dog, one giant leap
for giant</p>
    </div>
    <div class="action-list">
      <a href="#" data-toggle="tooltip" data-placement=
```

```
"bottom" title="Reply">
          <span class="glyphicon glyphicon-share-alt" aria-
hidden="true"></span>
        </a>
        <a href="#" data-toggle="tooltip" data-placement=
"bottom" title="Retweet">
          <span class="glyphicon glyphicon-refresh" aria-
hidden="true"></span>
          <span class="retweet-count">6</span>
        </a>
        <a href="#" data-toggle="tooltip" data-placement=
"bottom" title="Star">
          <span class="glyphicon glyphicon-star" aria-hidden=
"true"></span>
        </a>
      </div>
    </li>
</ul>
```

Как видите, чтобы создать всплывающую подсказку нужно добавить три атрибута данных. Первый – `data-toggle` – определяет вид (в данном случае, `tooltip`). Атрибут `data-placement` определяет местоположение подсказки (не требует пояснений). В данном случае он указывает, что подсказка должна находиться внизу, но ему можно также присвоить значение `left`, `top`, `bottom`, `right` или `auto`. И, наконец, атрибут `title`, который не является атрибутом данных, поскольку в HTML уже имеется атрибут с таким названием и мы можем пользоваться им.

Обновите веб-приложение в браузере, наведите указатель мыши на значок и вы увидите, что ... ничего не происходит! В отличие от других плагинов, для активации всплывающих подсказок и всплывающих панелей недостаточно простого определения атрибутов данных. Они имеют определенные проблемы, и их необходимо инициализировать с помощью команд JavaScript. Поэтому добавим следующую строку в файл `main.js`:

```
$(document).ready(function() {
    ... // остальной код
    $('[data-toggle="tooltip"]').tooltip();
});
```

Селектор `[data-toggle="tooltip"]` извлекает все элементы всплывающих подсказок и инициализирует их. При вызове функции

инициализации `.tooltip()` можно передать параметры. В табл. 8.1 перечислены основные параметры (полный список можно найти в официальной документации Bootstrap), которые можно передать как напрямую, из JavaScript, так и через атрибуты данных.

Таблица 8.1.

Параметр	Тип	По умолчанию	Описание
`animation`	Boolean	`true`	Добавляет к подсказке анимацию проявления и растворения.
`placement`	String или function	`top`	Определяет местоположение подсказки. В этом параметре можно передавать те же значения, что и в атрибутах данных (`top`, `bottom`, `left`, `right` и `auto`). Значение `auto` можно комбинировать с другими значениями, например, `auto left` (подсказка будет выведена слева, если это возможно, иначе – справа).
`selector`	String	`false`	При передаче селектора подсказка будет делегирована указанной цели.
`trigger`	String	`hover focus`	Событие активации подсказки. Поддерживаются значения `click`, `hover`, `focus` и `manual`. Можно передать несколько вариантов. Для варианта `manual` необходимо написать функцию активации плагина.

Плагин всплывающих подсказок содержит несколько вспомогательных методов, например, метод вывода сразу всех всплывающих подсказок. Их можно вызывать с помощью метода `.tooltip()`. Чтобы вывести все всплывающие подсказки, выполните вызов `$('.tooltip-selector').tooltip('show')`. Кроме `show` имеются также методы `hide`, `toggle` и `destroy`.

Парящие над всеми

Иногда объем сведений может не умещаться в обычном компоненте всплывающей подсказки. Для таких ситуаций Bootstrap предлагает всплывающие панели, перекрывающие основное содержимое страницы для отображения подробной информации.

Плагин всплывающих панелей является расширением плагина всплывающих подсказок, следовательно, при выборочном использовании плагинов необходимо загрузить их оба. Кроме того, подобно всплывающим подсказкам, всплывающие панели нельзя активировать только с помощью атрибутов данных. Для этого потребуется написать JavaScript-код.

Используем всплывающие панели в примере веб-приложения для правого столбца с идентификатором `div#who-follow`. Добавим всплывающую панель к кнопке **Follow**, для чего необходимо, во-первых, заменить элемент `<button>` элементом `<a>` и, во-вторых, добавить в него разметку всплывающей панели.

Зачем менять кнопку на ссылку при использовании всплывающей панели?

На самом деле, можно было не менять разметку для кнопки, это сделано из соображений совместимости с браузерами. Существуют браузеры, которые не в полной мере поддерживают функциональность, например щелчок на пункте Dismiss всплывающей панели.

Начнем с элемента `<buttons>` в элементе `div#who-follow`. Заменим его элементом `<a>`, и добавим атрибуты `role="button"` и `tabindex="-1"` для лучшей совместимости с браузерами:

```
<div id="who-follow" class="card">
  <div class="card-header">
    Who to follow
  </div>
  <div class="card-block">
    <ul class="list-unstyled">
      <li>
        <img src="imgs/cat.jpg" class="img-rounded">
        <div class="info">
          <strong>Crazy cats</strong>
          <a href="#" role="button" tabindex="-1"
class="btn btn-default">
            <span class="glyphicon glyphicon-plus"
aria-hidden="true"></span> Follow
          </a>
        </div>
      </li>
```

```
        <li>
          <img src="imgs/ration.jpg" class="img-rounded">
          <div class="info">
            <strong>Free ration alert</strong>
            <a href="#" role="button" tabindex="-1"
class="btn btn-default">
              <span class="glyphicon glyphicon-plus"
aria-hidden="true"></span> Follow
            </a>
          </div>
        </li>
      </ul>
    </div>
```

Код, выделенный жирным, связан с заменой кнопки на ссылку. А теперь, добавим разметку для всплывающей панели. Она достаточно проста и использует большинство атрибутов данных, что использовались с плагином всплывающих подсказок:

```
<div id="who-follow" class="card">
  <div class="card-header">
    Who to follow
  </div>
  <div class="card-block">
    <ul class="list-unstyled">
      <li>
        <img src="imgs/cat.jpg" class="img-rounded">
        <div class="info">
          <strong>Crazy cats</strong>
          <a href="#" role="button" tabindex="-1" class=
"btn btn-default" data-toggle="popover" data-trigger= "focus"
title="You may want to follow">
          <span class="glyphicon glyphicon-plus" aria-
hidden="true"></span> Follow
          </a>
        </div>
      </li>
      <li>
        <img src="imgs/ration.jpg" class="img-rounded">
        <div class="info">
          <strong>Free ration alert</strong>
          <a href="#" role="button" tabindex="-1" class=
"btn btn-default" data-toggle="popover" data-trigger=
```

```
"focus" title="You may want to follow">
              <span class="glyphicon glyphicon-plus"
aria-hidden="true"></span> Follow
            </a>
          </div>
        </li>
      </ul>
    </div>
```

По аналогии со всплывающими подсказками добавим активацию всплывающих панелей:

```
$(document).ready(function() {
    ... // прочий код на JavaScript
    $('[data-toggle="popover"]').popover();
});
```

Обновите веб-страницу, щелкните на кнопке **Follow** и вы увидите всплывающую панель справа от кнопки. А теперь изменим кое-какие настройки с помощью параметров. Прежде всего, добавим текст с описанием и изменим местоположение всплывающей панели через JavaScript:

```
$(document).ready(function() {
  ... // прочий код на JavaScript
  var popoverContentTemplate = '' +
    '<img src="imgs/breed.jpg" class="img-rounded">' +
    '<div class="info">' +
      '<strong>Dog Breeds</strong>' +
      '<a href="#" class="btn btn-default">' +
        '<span class="glyphicon glyphicon-plus" aria-
hidden="true"></span>' +
        'Follow' +
      '</a>' +
    '</div>';
  $('[data-toggle="popover"]').popover({
    placement: 'bottom',
    html: true,
    content: function() {
      return popoverContentTemplate;
    }
  });
});
```

Здесь мы изменили местоположение всплывающей панели на `bottom` и, с помощью параметра `html: true` указали, что содержимое всплывающей панели определено в виде HTML-разметки. Само содержимое определяется функцией, которая просто возвращает переменную `popoverContentTemplate`.

Текст описания можно определить разными и более оптимальными способами, но здесь я хотел показать способ добавления разметки HTML во всплывающую панель через JavaScript с помощью функции. В функции можно было бы использовать параметры нажатой кнопки, обращаясь к ней с помощью ссылки `this`.

События всплывающих панелей

Всплывающие панели и подсказки имеют несколько удобных событий. Как уже упоминалось выше, Bootstrap возбуждает несколько событий при отображении, скрытии и вставке элементов. Используем для демонстрации событие `show.bs.popover`, возникающее непосредственно перед отображением всплывающей панели. Например, изменим надпись на кнопке **Follow** на **Following** и значок с изображением плюса на значок с изображением галочки перед отображением всплывающей панели. Эти изменения можно выполнить в обработчике события `show.bs.popover`. Добавьте следующий код в JavaScript-файл:

```
$(document).ready(function() {
   ... // прочий JavaScript-код
   $('[data-toggle="popover"]').on('show.bs.popover',
function()
{
      var $icon = $(this).find('span.glyphicon');

      $icon.removeClass('glyphicon-plus').addClass
('glyphicon-ok');
      $(this).append('ing');
   });
});
```

Область видимости события привязана к элементу `data-toggle`, который является кнопкой **Follow**. Обработчик отыскивает значок кнопки и заменяет класс `glyphicon-plus` классом `glyphicon-ok`. После этого он добавляет суффикс `ing` к надписи **Follow**, это значит, что пользователь уже последовал за **Crazy cats** или **Free ration alert**.

И качестве последнего штриха, изменим цвет значка с синего на зеленый после появления значка галочки:

```
div#who-follow li .info .glyphicon-ok {
  color: #5cb85c;
}
```

Обновите веб-страницу и щелкните на кнопке **Follow**. Вы должны увидеть результат, изображенный на рис. 8.3.

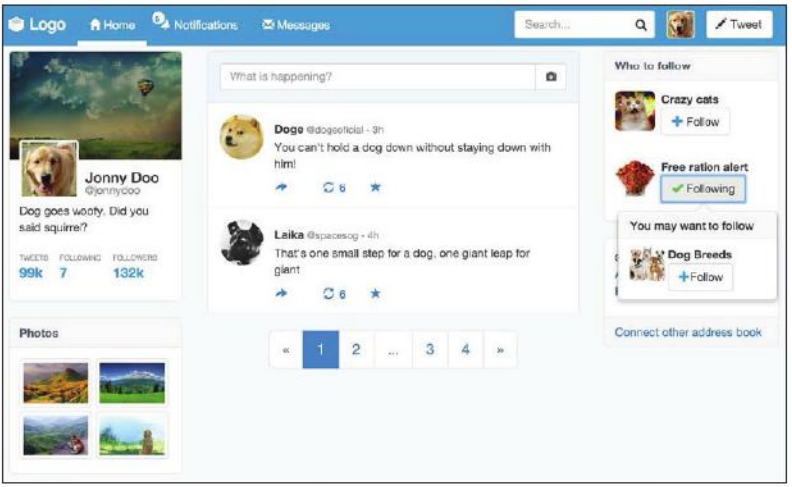


Рис. 8.3.

События Bootstrap можно использовать в самых разных случаях. Пример выше наглядно демонстрирует, как изменять элементы, с которыми взаимодействует пользователь. Имейте в виду, что изменения должны выполняться при каждом взаимодействии.

Создание аффикса меню

Плагин аффикса присутствует только в Bootstrap 3 (он был удален в версии 4), и служит для переключения способа позиционирования элемента между фиксированным и относительным, имитируя эффект прилипания `position: sticky`, который поддерживают не все браузеры.

Мы применим эффект прилипания к элементу `#profile`, но сейчас в веб-странице недостаточно элементов, чтобы стала возможной ее прокрутка. Поэтому, просто скопируем элемент `<li>` из списка `ul#feed`, чтобы увеличить количество элементов в списке. Повторите

копирование не менее трех раз, чтобы в веб-браузере появилась полоса вертикальной прокрутки.

Добавим в `div#profile` разметку, необходимую для закрепления меню:

```
<div id="profile" class="col-md-3 hidden-sm hidden-xs"
data-spy="affix" data-offset-top="0">
    ...
    // остальная часть профильного HTML-кода
</div>
```

Обновите страницу. Вы увидите, что закрепление (аффикс) не работает. Поскольку плагин аффикса фиксирует положение левого столбца, весь столбец удаляется из сетки, ломая порядок столбцов.

Нам нужен обходной путь. Напишем на JavaScript обработчики событий, возбуждаемых плагином.

Воспользуемся событием `affix.bs.affix`, возникающим перед фиксацией элемента:

```
$(document).ready(function() {
    ... // прочий JavaScript-код

    $('#profile').on('affix.bs.affix', function() {
        $(this).width($(this).width() - 1);
        $('#main').addClass('col-md-offset-3');
    }).on('affix-top.bs.affix', function() {
        $(this).css('width', '');
        $('#main').removeClass('col-md-offset-3');
    });
});
```

Здесь мы использовали несколько интересных трюков.

Первый обработчик события `.on('affix.bs.affix', handler)` сохраняет ширину левого столбца, когда для элемента устанавливается режим позиционирования `position: fixed`. Это необходимо потому, что класс `.col-md-3` не имеет фиксированной ширины, его ширина задается в процентах.

Кроме того, применив класс `.col-md-offset-3`, он добавляет смещение для среднего столбца, соответствующее ширине отделенного левого столбца.

Событие `affix-top.bs.affix` возникающее при возврате элемента в начальную позицию, производит обратное действие, удаляет фиксированную ширину и класс смещения для среднего столбца.

Удаление фиксированной ширины и возврат к ширине в процентах выполняет строка `$(this).css('width', '')`. Кроме того, из элемента `#main` удаляется класс `.col-md-offset-3`.

Обновите веб-страницу и прокрутите ее вниз, результат показан на рис. 8.4. Обратите внимание, что колонка с профилем слева остается на месте, а остальная часть страницы прокручивается:

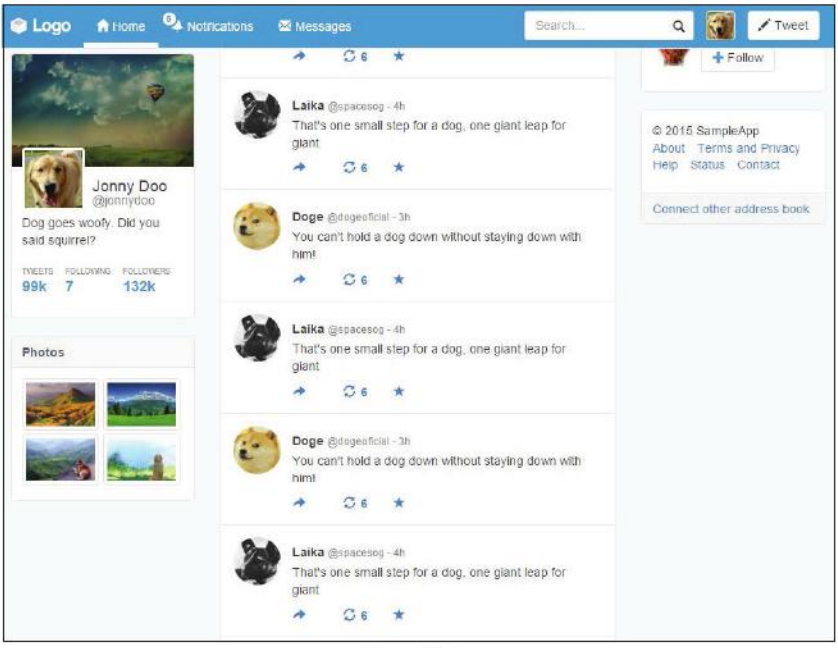


Рис. 8.4.

Завершение веб-приложения

Чтобы закончить пример веб-приложения, осталось только создать еще одну модальную панель, которая должна открываться щелчком на ссылке **Messages** в навигационной панели.

Создадим ее по аналогии с модальной панелью для кнопки **Tweet**. Итак, добавим атрибут данных в ссылку **Messages**, находящуюся в элементе `.nav navbar-nav`:

```
<ul class="nav navbar-nav">
  <li class="active">
    <a href="#">
```

```
        <span class="glyphicon glyphicon-home" aria-hidden=
"true"></span>
        Home
      </a>
    </li>
    <li>
      <a href="#">
        <span class="badge">5</span>
        <span class="glyphicon glyphicon-bell" aria-
hidden="true"></span>
        Notifications
      </a>
    </li>
    <li>
      <a href="#" role="button" data-toggle="modal" data-
target="#messages-modal">
        <span class="glyphicon glyphicon-envelope" aria-
hidden="true"></span>
        Messages
      </a>
    </li>
    <li class="visible-xs-inline">
      <a href="#">
        <span class="glyphicon glyphicon-user" aria-hidden=
"true"></span>
        Profile
      </a>
    </li>
    <li class="visible-xs-inline">
      <a href="#">
        <span class="glyphicon glyphicon-off" aria-hidden=
"true"></span>
        Logout
      </a>
    </li>
</ul>
```

Выделенный код присваивает ссылке роль кнопки, открывающей панель с идентификатором `#messages-modal` в модальном режиме. Создадим основу модальной панели в конце HTML-файла, сразу после `#tweet-modal`:

```
<div id="messages-modal" class="modal fade" tabindex="-1"
role="dialog">
```

```
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss=
"modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
          <h4 class="modal-title">Dog messages</h4>
          <button type="button" class="btn btn-primary btn-
message">New message</button>
        </div>
        <div class="modal-body">
        </div>
      </div>
    </div>
</div>
```

Вновь созданная панель имеет следующие отличия от `#tweet-modal`: во-первых, из модальной панели удален элемент `.modal-footer`, поскольку в нем нет необходимости (Bootstrap, как и многие другие фреймворки, позволяет включать и исключать элементы по мере необходимости), и во-вторых, в заголовке создана новая кнопка **New message** с классом `.btn-message`. Для правильного отображения кнопки определим следующий CSS-стиль:

```
#messages-modal .btn-message {
  position: absolute;
  right: 3em; top: 0.75em;
}
```

А сейчас наполним модальную панель, добавив в нее список сообщений:

```
<div class="modal fade" id="messages-modal" tabindex="-1"
role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss=
"modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title">Dog messages</h4>
        <button type="button" class="btn btn-primary
```

```
btn-message">New message</button>
        </div>
        <div class="modal-body">
          <ul class="list-unstyled">
            <li>
              <a href="#">
                <img src="imgs/laika.jpg" class="img-circle">
                <div class="msg-content">
                  <h5>Laika <small>@spacesog</small></h5>
                  <p>Hey Jonny, how is down there?</p>
                </div>
              </a>
            </li>
            <li>
              <a href="#">
                <img src="imgs/doge.jpg" class="img-circle">
                <div class="msg-content">
                  <h5>Doge <small>@dogeoficial </small></h5>
                  <p>Wow! How did I turned in to a meme?</p>
                 </div>
              </a>
            </li>
            <li>
              <a href="#">
                <img src="imgs/cat.jpg" class="img-circle">
                <div class="msg-content">
                  <h5>Cat <small>@crazycat</small></h5>
                  <p>You will never catch me!</p>
                </div>
              </a>
            </li>
            <li>
              <a href="#">
                <img src="imgs/laika.jpg" class="img-circle">
                <div class="msg-content">
                  <h5>Laika <small>@spacesog</small></h5>
                  <p>I think I saw you in Jupiter! Have
you been there recently?</p>
                 </div>
              </a>
            </li>
```

```
            </ul>
          </div>
        </div>
      </div>
    </div>
```

В заключение определим стиль в CSS-файле для правильного отображения списка:

```
#messages-modal .modal-body {
  max-height: 32rem;
  overflow: auto;
}
#messages-modal li {
  padding: 0.75rem;
  border-bottom: 0.1rem solid #E6E6E6;
}
#messages-modal li:hover {
  background-color: #E6E6E6;
}
#messages-modal li a:hover {
  text-decoration: none;
}
#messages-modal li img {
  max-width: 15%;
}
#messages-modal .msg-content {
  display: inline-block;
  color: #000;
}
#messages-modal .msg-content h5 {
  font-size: 1em;
  font-weight: bold;
}
```

Здесь мы просто установили максимальную высоту тела модальной панели, при превышении которой появляется вертикальная полоса прокрутки. Для списка и ссылки изменен стиль, который применяется при наведении указателя мыши, и определены вид, размер и цвет шрифта.

Обновите веб-страницу, щелкните на ссылке **Messages** в навигационной панели и перед вами появится красивая модальная панель, как на рис. 8.5.

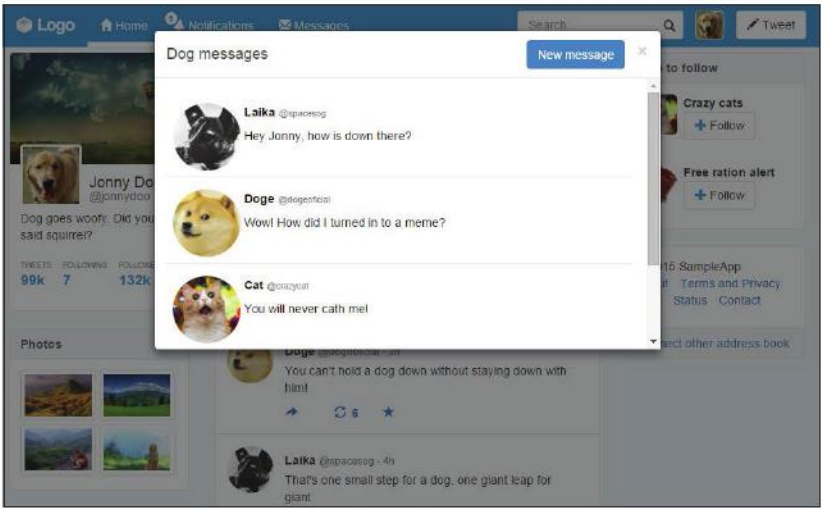


Рис. 8.5.

Итоги

В этой главе закончена работа над примером веб-приложения. Основной целью главы было знакомство с плагинами Bootstrap, которые не рассматривались ранее.

Сначала мы познакомились с атрибутами данных и особенностями их использования в Bootstrap. Затем, рассмотрели различные способы вызова плагинов: с помощью JavaScript или только посредством атрибутов данных.

Начали и закончили исследование универсального и гибкого плагина модальных панелей, одного из основных плагинов Bootstrap. Этот плагин можно использовать во множестве ситуаций, когда необходимо реализовать взаимодействие с пользователем без перехода на другую страницу.

В середине главы речь шла о двух плагинах, тесно связанных между собой: всплывающих подсказках и панелях. Оба компонента используют один и тот же базовый плагин, но в разных контекстах. Всплывающие подсказки применяются для отображения дополнительной информации, а всплывающие панели являются чем-то средним между модальными панелями и всплывающими подсказками, поскольку могут вмещать больше информации, чем всплывающие подсказки, но не настолько навязчивы, как модальные панели.

Навыки создания Twitter-подобного веб-приложения весьма важны, поскольку им можно найти много различных применений. Веб-приложения произвели революцию в Веб и Bootstrap сыграл в этом важную роль, позволяя быстро создавать красивые веб-страницы.

Следующая глава будет посвящена описанию решения более сложной задачи – созданию с нуля панели мониторинга! Такие панели весьма популярны в Интернете и умение создавать их поставит вас вровень с лучшими веб-разработчиками. Готовы к переходу на продвинутый уровень?

ГЛАВА 9. Введение в продвинутый режим

Ладно, нет больше времени на совершенствование навыков. Настало время проверить себя на настоящей большой задаче, разработке панели управления для администратора. Bootstrap нам поможет но, для этого нужно уметь профессионально обращаться с фреймворками.

Нам нужен план разработки панели мониторинга, с самого начала и до полного завершения. Поэтому используем обычный порядок разработки и пройдем путь от рисунка до готовой веб-страницы. Двигаясь по этому пути, мы рассмотрим следующие вопросы:

- плавающий контейнер;
- Flexbox-макет;
- настраиваемая вертикальная панель навигации Bootstrap;
- плагин сворачивания;
- Bootstrap и продвинутые правила CSS;
- интеграция с внешними плагинами;
- загрузка одностраничного приложения.

Это пример – последний в книге. Закончив его, вы будет знать о Bootstrap почти все. Я знаю, что вы способны победить в финале!

Общий план

Как уже упоминалось, перед нами стоит профессиональная задача, и она требует профессионального подхода. Будем строго следовать руководству по разработке. На рис. 9.1 показана панель мониторинга, которую надо получить:

Как видите, панель мониторинга состоит из навигационного заголовка с некоторой информацией, панели поиска и вывода оповещений. Слева расположено меню с разделами веб-приложения. В цент-

ре присутствует ряд диаграмм, отражающих состояние страницы. Панель хорошо выглядит на снимке экрана, но еще лучше будет смотреться в браузере!

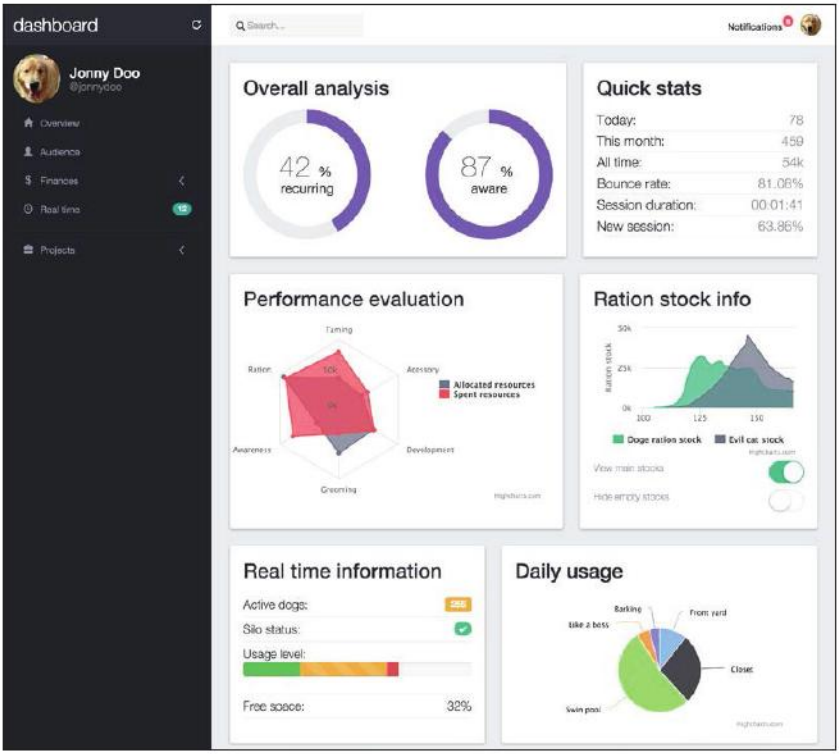


Рис. 9.1.

Основу страницы составляют:

- навигационный компонент Bootstrap в заголовке, прикрепленный к верхней границе страницы;
- плавающий контейнер с двумя столбцами;
- левый столбец содержит меню и зафиксирован;
- правый столбец включает основной контент с набором карточек, отображающих статистические сведения.

Прежде чем добавить первый элемент, создадим новый файл с той же структурой, что использовалась во всех предыдущих примерах (за более подробной информацией обращайтесь к разделу «Теги, необходимые Bootstrap» в главе 1). Сохраним вновь созданный файл с име-

нем dashboard.html и добавим в него начальную разметку HTML. Теперь можно двигаться дальше!

Последняя навигационная панель с Flexbox

Вам, наверное, уже надоели навигационные панели, но благодаря приобретенному ранее опыту, мы быстро реализуем ее, воспользовавшись кодом из предыдущих примеров.

Создадим элемент <nav> и добавим в него элементы .container-fluid и .row:

```
<nav class="navbar navbar-fixed-top">
  <div class="container-fluid">
    <div class="row">
    </div>
  </div>
</nav>
```

Этот элемент .row содержит два столбца, как и основной контейнер, о чем уже упоминалось выше. В первом столбце создадим заголовок панели мониторинга и кнопку обновления:

```
<nav class="navbar navbar-fixed-top">
  <div class="container-fluid">
    <div class="row">
      <div class="col-sm-3 top-left-menu">
        <div class="navbar-header">
          <a class="navbar-brand" href="webapp.html">
            <h1>dashboard</h1>
          </a>
        </div>
        <a href="#" data-toggle="tooltip" data-placement=
"bottom" data-delay="500" title="Refresh data" class= "header-
refresh pull-right">
          <span class="glyphicon glyphicon-repeat" aria-
hidden="true"></span>
        </a>
      </div>
    </div>
  </div>
</nav>
```

Отметьте, что в кнопку обновления добавлен значок `.glyphicon` и всплывающая подсказка. Не забудьте активировать всплывающие подсказки в файле `main.js`:

```
$(document).ready(function() {
    $('[data-toggle="tooltip"]').tooltip();
});
```

Для всплывающей подсказки предусмотрена задержка перед выводом, на что указывает атрибут `data-delay="500"`. Этот параметр упоминался выше, но мы его еще не использовали. То есть, всплывающая подсказка будет появляться с задержкой в 500 миллисекунд после наведения указателя мыши на кнопку.

Внутрь элемента `.nav-header` добавим элемент `.navbar-toggle`, который отображается на малых экранах и сворачивает меню:

```
<nav class="navbar navbar-fixed-top">
  <div class="container-fluid">
    <div class="row">
       <div class="col-sm-3 top-left-menu">
         <div class="navbar-header">
           <a class="navbar-brand" href="webapp.html">
             <h1>dashboard</h1>
           </a>
           <button type="button" class="navbar-toggle
collapsed" data-toggle="collapse" data-target="#nav-menu"
aria-expanded="false">
             <span class="sr-only">Toggle navigation</span>
             <span class="icon-bar"></span>
             <span class="icon-bar"></span>
             <span class="icon-bar"></span>
           </button>
         </div>
         <a href="#" data-toggle="tooltip" data-placement=
"bottom" data-delay="500" title="Refresh data" class="header-
refresh pull- right">
           <span class="glyphicon glyphicon-repeat" aria-
hidden="true"></span>
         </a>
       </div>
    </div>
  </div>
</nav>
```

Пока в разметке нет ничего загадочного. Все эти компоненты мы уже использовали раньше. Следуя плану, создадим несколько CSS-правил для оформления страницы, но начнем с создания некого общего CSS-стиля. В начало файла `base.css` добавим стиль:

```
.transition,
.transition:hover,
.transition:focus {

  -webkit-transition: all 150ms ease-in-out;
  -moz-transition: all 150ms ease-in-out;
  -ms-transition: all 150ms ease-in-out;
  -o-transition: all 150ms ease-in-out;
  transition: all 150ms ease-in-out;
}

html, body {
  position: relative;
  height: 100%;
  background-color: #e5e9ec;
}
```

Здесь определяется общий класс `.transition`, который будет использован в нескольких местах (в этой главе). Переходы впервые появились в спецификации CSS3 и помогают создавать анимационные эффекты. В данном случае ко всем элементам с этим классом применяется эффект `ease-in-out`.

Кроме того, для элементов `html` и `body` изменены фон, режим позиционирования и высота во весь экран.

Далее, добавим CSS-стиль для заголовка навигационной панели:

```
nav.navbar-fixed-top {
  background-color: #FFF;
  border: none;
}
nav .top-left-menu {
  background-color: #252830;
  display: -webkit-flex; display: flex;
  align-items: center;
}
.navbar-brand {
  height: auto;
}
.navbar-brand h1 {
```

```
  margin: 0;
  font-size: 1.5em;
  font-weight: 300;
  color: #FFF;
}
nav .header-refresh {
  margin-left: auto;
  color: #FFF;
}
```

Здесь был изменен цвет элементов. Но более важно здесь использование Flexbox-правил (помните обсуждение Flexbox в разделе «*Основные понятия Flexbox*» *главы 5*, «*Что делает его фантастическим*»?). Bootstrap 4 будет поддерживать разметку Flexbox, поэтому продолжим использовать ее, учитывая еще и тот факт, что в ближайшем будущем она должна стать стандартом для каждого браузера.

Результат проделанной работы показан на рис. 9.2.

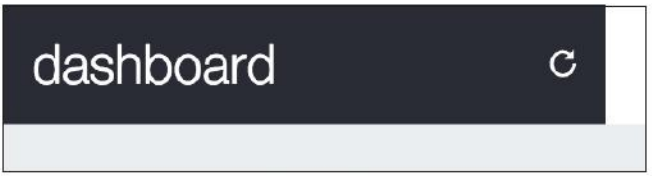


Рис. 9.2.

Поиск

Следуя проекту, необходимо создать форму поиска. Сразу после элемента `.top-left-menu` добавим код формы (выделен жирным в следующем листинге):

```
<nav class="navbar navbar-fixed-top">
  <div class="container-fluid">
    <div class="row">
      <div class="col-sm-3 top-left-menu">
        ...
      </div>

      <form id="search" role="search" class="hidden-xs col-
sm-3">
        <div class="input-group">
          <span class="glyphicon glyphicon-search" aria-
hidden="true"></span>
```

```
            <input type="text" class="form-control transition"
placeholder="Search...">
          </div>
        </form>

      </div>
    </div>
</nav>
```

И, как обычно, правила CSS:

```
nav form#search {
  padding: 0.9em;
}

nav form#search .input-group {
  display: -webkit-flex;
  display: flex;
  align-items: center;
}

nav form#search .input-group .form-control {
  border-radius: 0.25em;
  border: none; width: 70%;
  padding-left: 1.9em;
  background-color: #F3F3F3;
  box-shadow: none;
}

nav form#search .input-group .form-control:focus {
  width: 100%;
  box-shadow: none;
}

nav form#search .glyphicon-search {
  z-index: 99;
  left: 1.7em;
}
```

Здесь снова используется свойство `display: flex`. Кроме того, создано правило для псевдо-класса `.form-control`. Псевдо-класс `:focus` активируется при получении фокуса полем ввода, другими словами, когда оно принимает вводимый текст. Правило `:focus` изменяет ширину поля ввода в момент щелчка мышью на нем.

Обновите веб-страницу и щелкните на поле ввода в форме поиска. В этот элемент добавлен класс `.transition`, поэтому при получении

фокуса он изменяет ширину не сразу, а постепенно, в соответствии с функцией перехода. Результат приведен на рис. 9.3.

Рис. 9.3.

Меню навигации

Чтобы закончить навигационную панель, нужно создать элемент навигационного меню, размещенный справа, который мы назовем `#nav-menu`. Меню будет содержать список оповещений в виде кнопки с раскрывающимся списком.

Добавьте следующий код HTML сразу после элемента `<form>`:

```
<div id="nav-menu" class="collapse navbar-collapse pull-
right">
  <ul class="nav navbar-nav">
  </ul>
</div>
```

Поместим оповещения внутрь тега `<ul>`, и сразу добавим в этот список несколько элементов:

```
<div id="nav-menu" class="collapse navbar-collapse pull-
right">
  <ul class="nav navbar-nav">
    <li>
      <div id="btn-notifications" class="btn-group">
        <span class="badge">3</span>
        <button type="button" class="btn btn-link
dropdown-toggle" data-toggle="dropdown" aria-haspopup=
"true" aria-expanded="false">
          Notifications
        </button>
      </div>
    </li>
  </ul>
</div>
```

Этот элемент реализован как кнопка, состоящая из обертывающего элемента `#btn-notifications`, внутри которого находится элемент `.badge`, отображающий количество новых оповещений, и сама

кнопка `button.btn`, которая выглядит как ссылка, благодаря классу `.btn-link`. Кроме того, кнопка содержит все, необходимое для создания кнопки с раскрывающимся списком: класс `.dropdown-toggle` и атрибут данных `data-toggle="dropdown"`.

Каждой кнопке `button.dropdown-toggle` необходим список `ul.dropdown-menu`, поэтому добавим его сразу после элемента `<button>`:

```
<div id="nav-menu" class="collapse navbar-collapse pull-
right">
  <ul class="nav navbar-nav">
    <li>
      <div id="btn-notifications" class="btn-group">
        <span class="badge">3</span>
        <button type="button" class="btn btn-link dropdown-
toggle" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
          Notifications
        </button>
        <ul id="notification-list" class="dropdown-menu
pull-right">
          <li>
            <a href="#">
              <span class="badge"></span>
              <img src="imgs/laika.jpg" class="img-circle">
              <div class="notification-message">
                <strong>Laika</strong>
                <p>Hey! How are you?</p>
                <em class="since">2h ago</em>
              </div>
            </a>
          </li>
          <li>
            <a href="#">
              <span class="badge"></span>
              <img src="imgs/cat.jpg" class="img-circle">
              <div class="notification-message">
                <strong>Devil cat</strong>
                <p>I will never forgive you...</p>
                <em class="since">6h ago</em>
              </div>
            </a>
```

```
            </li>
            <li>
              <a href="#">
                <span class="badge"></span>
                <img src="imgs/doge.jpg" class="img-circle">
                <div class="notification-message">
                  <strong>Doge</strong>
                  <p>What are you doing? So scare. It's
alright now.</p>
                  <em class="since">yesterday</em>
                </div>
              </a>
            </li>
          </ul>
        </div>
      </li>
    </ul>
</div>
```

Список выделен в этом листинге жирным. Несмотря на то, что список получился большим, он включает всего лишь три одинаковых элемента оповещений с различным содержимым.

Обновите страницу, откройте раскрывающийся список, и вы почувствуете непреодолимое желание добавить несколько правил CSS, чтобы этот список выглядел не так безобразно:

```
/*навигационное меню*/
nav #nav-menu {
  padding: 0.4em;
  padding-right: 1em;
}

/*навигационное меню и оповещения*/
#nav-menu #btn-notifications > .badge {
  color: #FFF;
  background-color: #f35958;
  font-size: 0.7em;
  padding: 0.3rem 0.55rem 0.3rem 0.5rem;
  position: absolute;
  right: -0.4rem; top: 1rem;
  z-index: 99;
}

#btn-notifications .btn-link {
```

```
  padding-top: 1.5rem;
  color: #252830;
  font-weight: 500;
}

#btn-notifications .btn-link:hover {
  text-decoration: none;
}
```

Отлично! Кнопка и бейдж приобрели более привлекательный вид. А теперь, займемся списком `#notification-list`:

```
#notification-list {
  max-height: 20em;
  overflow: auto;
}

#notification-list a {
  display: -webkit-flex;
  display: flex;
  opacity: 0.7;
  margin: 1.5rem;
  border-radius: 0.5rem;
  padding: 0.5rem 1.3rem;
  background-color: #EFEFEF;
  position: relative;
}

#notification-list a:hover {
  color: #262626;
  text-decoration: none;
  opacity: 1;
}

#notification-list img {
  display: inline-block;
  height: 35px;
  width: 35px;
  margin-right: 1em;
  margin-top: 1em;
}

#notification-list .notification-message {
  display: inline-block;
  white-space: normal;
  min-width: 25rem;
```

```
}

#notification-list .badge:empty {
  display: inline-block;
  position: absolute;
  right: 0.5rem;
  top: 0.5rem;
  background-color: #f35958;
  height: 1.4rem;
}

#notification-list em.since {
  font-size: 0.7em;
  color: #646C82;
}
```

Для оповещений добавлены обычные правила, касающиеся интервалов, цвета и так далее. Единственной отличительной чертой здесь является использование `flexbox` для выравнивания контента. На рис. 9.4 представлена законченная навигационная панель:

Рис. 9.4.

Заметили, что изображение имеет круглую форму? А знаете почему? Причиной этому является класс `.img-circle`, добавленный в элемент `<img>`.

Просмотр учетной записи

Последний компонент в панели навигации – изображение, щелчок на котором должен открывать меню пользователя, как в примере веб-приложения. Поэтому, не мешкая, добавим следующий HTML-код непосредственно после списка `#nav-menu`:

```
<div id="nav-menu" class="collapse navbar-collapse pull-
right">
  <ul class="nav navbar-nav">
    ...
  </ul>

  <div id="nav-profile" class="btn-group pull-right">
```

```
      <button type="button" class="btn btn-link dropdown-
toggle thumbnail" data-toggle="dropdown" aria-haspopup=
"true" aria-expanded="false">
        <img src="imgs/jon.png" class="img-circle">
      </button>
      <ul class="dropdown-menu">
        <li><a href="#">Profile</a></li>
        <li><a href="settings.html">Setting</a></li>
        <li role="separator" class="divider"></li>
        <li><a href="#">Logout</a></li>
      </ul>
    </div>
</div>
```

Это еще одна кнопка с раскрывающимся списком. Определим для нее правила CSS:

```
#nav-profile {
  margin: 0.5em;
  margin-left: 1em;
}

#nav-profile button.thumbnail {
  margin: 0;
  padding: 0;
  border: 0;
}

#nav-profile img {
  max-height: 2.3em;
}
```

Готово! Обновите страницу в браузере – результат должен быть похожим на рис. 9.5.

Рис. 9.5.

Наполнение основного плавающего раздела

Завершив работу над панелью навигации, приступим к заполнению основного раздела. Для этого создадим элемент-контейнер `.container-fluid`. Внутрь контейнера добавим единственный элемент `.row` с двумя столбцами, шириной три и девять, соответственно:

```
<div class="container-fluid">
  <div class="row">
    <div id="side-menu" class="col-md-3 hidden-xs">
    </div>

    <div id="main" class="col-md-9">
    </div>
  </div>
</div>
```

Это обычная сетка с одной строкой. Первый столбец `#side-menu` отображается на устройствах, от малых до больших, при этом столбец `#main` занимает 9 из 12 ячеек сетки при среднем разрешении.

Не стоит забывать, что компонент `#side-menu` должен быть закреплен. Поэтому добавим свойства данных, чтобы прикрепить его к верхней части страницы, как это было сделано в примере веб-приложения:

```
<div class="container-fluid">
  <div class="row">
    <div id="side-menu" class="col-md-3 hidden-xs" data-
spy="affix" data-offset-top="0">
    </div>

    <div id="main" class="col-sm-offset-3 col-md-9">
    </div>

  </div>
</div>
```

Обратите внимание, что из-за добавлении аффикса необходимо сместить блок `#main`, добавив в него класс `.col-sm-offset-3`.

Боковое вертикальное меню

Теперь заполним элемент `#side-menu`. Прежде всего, создадим блок учетной записи со сведениями о пользователе. Поместим следующую разметку HTML внутрь соответствующего элемента:

```
<div id="side-menu" class="col-md-3 hidden-xs" data-spy=
"affix" data-offset-top="0">
  <div class="profile-block">
    <img src="imgs/jon.png" class="img-circle">
    <h4 class="profile-title">Jonny Doo
<small>@jonnydoo</small></h4>
  </div>
</div>
```

Обновите страницу в браузере, и вы увидите, что она выглядит неважно. Попробуем улучшить оформление:

```
#side-menu {
  background-color: #1b1e24;
  padding-top: 7.2rem;
  height: 100%;
  position: fixed;
}

#side-menu .profile-block > * {
  display: inline-block;
}

#side-menu .profile-block img {
  width: 70px;
}

#side-menu .profile-title {
  color: #FFF;
  margin-left: 1rem;
  font-size: 1.5em;
  vertical-align: middle;
}

#side-menu .profile-title small {
  display: block;
}
```

Боковое меню `#side-menu` слева должно быть растянуто на всю доступную высоту, но если уменьшить ширину окна браузера, можно за-

метить, что размеры элемента `#nav-header` не изменяются синхронно с содержимым основного раздела. Это проблема. Вы понимаете, что происходит?

Вот такая беда! Чем тут можно помочь? Элемент `#side-menu` включает класс `.col-md-3`, предназначенный для средних разрешений. Чтобы улучшить адаптивность для малых разрешений и обеспечить синхронность изменения размеров с прочими элементами, включающими класс `.col-sm-*`, добавим в него класс поддержки малых устройств. В данном случае достаточно изменить классы в элементах `#side-menu` и `#main`:

```
<div class="container-fluid">
  <div class="row">
    <div id="side-menu" class="col-sm-3 hidden-xs" data-
spy="affix" data-offset-top="0">
    </div>

    <div id="main" class="col-sm-offset-3 col-sm-9">
    </div>
  </div>
</div>
```

Получившееся боковое меню показано на рис. 9.6:

Рис. 9.6.

Левое меню отлично смотрится!

Веб-приложение не было бы веб-приложением, если бы у него не было меню. Закончив с выводом данных о пользователе в боковое меню `#side-menu`, займемся вертикальным меню.

Что вам приходит в голову, когда вы слышите определение «вертикальное», применительно к меню? Конечно же, это меню с классом `.nav-stacked`! Поэтому создадим компонент меню с классом `.nav-stacked`, добавив его после элемента `#profile-block`:

```
<ul class="nav nav-pills nav-stacked">
  <li>
    <a href="#" class="transition">
      <span class="glyphicon glyphicon-home" aria-hidden=
"true"></span>
      Overview
    </a>
  </li>
  <li>
    <a href="#" class="transition">
      <span class="glyphicon glyphicon-user" aria-hidden=
"true"></span>
      Audience
    </a>
  </li>
  <li>
    <a href="#" class="transition">
      <span class="glyphicon glyphicon-usd" aria-hidden=
"true"></span>
      Finances
      <span class="glyphicon glyphicon-menu-left pull-right
transition" aria-hidden="true"></span>
    </a>
  </li>
  <li>
    <a href="#" class="transition">
      <span class="glyphicon glyphicon-time" aria-hidden=
"true"></span>
      Real time
      <span class="badge pull-right">12</span>
    </a>
  </li>
  <li class="nav-divider"></li>
  <li>
    <a href="#" class="transition">
      <span class="glyphicon glyphicon-briefcase" aria-
hidden="true"></span>
      Projects
      <span class="glyphicon glyphicon-menu-left pull-right
transition" aria-hidden="true"></span>
    </a>
  </li>
</ul>
```

Здесь нет никаких секретов! Был создан обычный вертикальных список с классами `.nav`, `.nav-pills` и `.nav-stacked`. Все остальное обеспечила магия Bootstrap. А теперь рассмотрим трюк, связанный с применением плагина сворачивания.

Плагин сворачивания

Плагин сворачивания, входящий в состав Bootstrap, позволяет управлять видимостью элемента. С его помощью элемент можно свернуть или развернуть.

Плагин сворачивания зависит от плагина, реализующего функции переходов.

Чтобы добавить в элемент возможность сворачивания, нужно добавить в него атрибут данных `data-toggle="collapse"`. Если элемент является ссылкой `<a>`, ее нужно связать с идентификатором сворачиваемого элемента в виде: `href="#my-collapsed-element"`. Если элемент является кнопкой `<button>`, в нее нужно добавить атрибут данных, например: `data-target="#my-collapsed-element"`. Различие использования `href` для ссылки и `data-target` для `button` определяется разной семантикой элементов. Действительно, любая ссылка должна иметь атрибут `href`, но это требование неприменимо к кнопке, поэтому связь реализуется посредством атрибута данных `data-target`.

Добавим вложенные списки в пункты **Finances** и **Projects** с помощью плагина сворачивания. После каждой из ссылок на эти элементы, создадим вложенные списки, выделенные жирным в следующем листинге. Кроме того, поскольку мы решили использовать теги `<a>`, добавим атрибут `href` с идентификатором сворачиваемого элемента и атрибут `data-toggle` для сворачивания:

```
<ul class="nav nav-pills nav-stacked">
  <li>
    <a href="#" class="transition">
      <span class="glyphicon glyphicon-home" aria-hidden=
"true"></span>
      Overview
    </a>
  </li>
  <li>
```

```
      <a href="#" class="transition">
        <span class="glyphicon glyphicon-user" aria-hidden=
"true"></span>
        Audience
      </a>
    </li>
    <li>
      <a href="#finances-opts" class="transition" role=
"button" data-toggle="collapse" aria-expanded="false"
aria-controls="finances-opts">
        <span class="glyphicon glyphicon-usd" aria-hidden=
"true"></span>
        Finances
        <span class="glyphicon glyphicon-menu-left pull-right
        transition" aria-hidden="true"></span>
      </a>
      <ul class="collapse list-unstyled" id="finances-opts">
        <li>
            <a href="#" class="transition">
            Incomes
          </a>
        </li>
        <li>
            <a href="#" class="transition">
            Outcomes
          </a>
        </li>
      </ul>
    </li>
    <li>
      <a href="#" class="transition">
        <span class="glyphicon glyphicon-time" aria-hidden=
"true"></span>
        Real time
        <span class="badge pull-right">12</span>
      </a>
    </li>
    <li class="nav-divider"></li>
    <li>
      <a href="#projects-opts" class="transition" role=
"button" data-toggle="collapse" aria-expanded="false"
aria-controls="projects-opts">
        <span class="glyphicon glyphicon-briefcase" aria-
```

```
hidden="true"></span>
       Projects
      <span class="glyphicon glyphicon-menu-left pull-right
transition" aria-hidden="true"></span>
    </a>
    <ul class="collapse list-unstyled" id="projects-opts">
      <li>
          <a href="#" class="transition">
          Free ration nation
        </a>
      </li>
      <li>
          <a href="#" class="transition">
          Cats going crazy
        </a>
      </li>
    </ul>
  </li>
</ul>
```

Разберем в качестве примера первый сворачиваемый пункт меню **Finances**. За ссылкой **Finances** следует сворачиваемый список `#finances-opt`. В него добавлен класс `.collapse`, который присваивается сворачиваемым элементам.

Вернемся к ссылке **Finances**. Она имеет атрибут `href` с идентификатором сворачиваемого списка `#finances-opt`, атрибут `data-toggle="collapse"`, необходимый плагину сворачивания, а также атрибуты области `aria-expanded` и `aria-controls`, определяющие семантику сворачивания.

Обновите страницу и убедитесь, что фреймворк Bootstrap сделал практически все необходимое сам.

Теперь добавим несколько простых стилей, определяющих цвет и интервалы:

```
#side-menu ul.nav {
  margin-top: 1rem;
}

#side-menu ul.nav a {
  color: #8b91a0;
}

#side-menu ul.nav a:hover,
#side-menu ul.nav a:focus {
```

```
    color: #FFF;
    background-color: inherit;
}

#side-menu ul.nav a .glyphicon {
    margin-right: 0.7rem;
}

#side-menu ul.nav a .glyphicon.pull-right {
    margin-top: 0.2rem;
}
#side-menu ul.nav a .badge {
    background-color: #1ca095;
}

#side-menu ul.nav .nav-divider {
    background-color: #252830;
}

#side-menu ul.nav ul {
    margin-left: 10%;
}

#side-menu ul.nav ul a {
    display: block;
    background-color: #2b303b;
    padding: 1rem;
    margin-bottom: 0.3rem;
    border-radius: 0.25em;
}

#side-menu ul.nav ul a:hover {
    text-decoration: none;
    background-color: #434857;
}
```

Вернемся в браузер, обновим страницу и проверим результат. Он показан на рис. 9.7.

Использование продвинутых CSS-стилей

Давайте внесем некую изюминку, применив другие CSS-свойства. Что вы думаете о повороте значка стрелки в сворачиваемых элементах меню на 90 градусов против часовой стрелки для создания эффекта разворачивания? Это будет отлично выглядеть, тем более что для реализации достаточно лишь определить стили CSS.

Рис. 9.7.

Добавим следующее CSS-правило, использующее расширенные возможности свойства `transform`:

```
#side-menu ul.nav a:focus .glyphicon.pull-right {
       -moz-transform: rotate(-90deg);
       -webkit-transform: rotate(-90deg);
       -o-transform: rotate(-90deg);
       -ms-transform: rotate(-90deg);
       transform: rotate(-90deg);
    }
```

Такое свойство `transform` будет делать именно то, что требуется, то есть, при получении ссылкой фокуса (в результате щелчка), значок со стрелкой повернется на 90 градусов против часовой стрелки (то есть, на 90 градусов со знаком «минус»).

Чтобы это выглядело более профессионально, используем недавно появившееся свойство `will-change`. Добавим стиль для следующего селектора:

```
#side-menu ul.nav a .glyphicon.pull-right {
  margin-top: 0.2rem;
  will-change: transform;
}
```

Щелкните на элементе меню, чтобы открыть его и проверьте, как действует анимация поворота стрелки. На рис. 9.8 показано открытое меню:

Свойство will-change

Свойство `will-change` оптимизирует анимацию, указывая браузеру, какие элементы будут изменяться и требуют повышенного внимания. В настоящее время это свойство не поддерживается браузерами, но в скором времени такая поддержка появится. О его доступности можно узнать на http://caniuse.com/#feat=will-change.

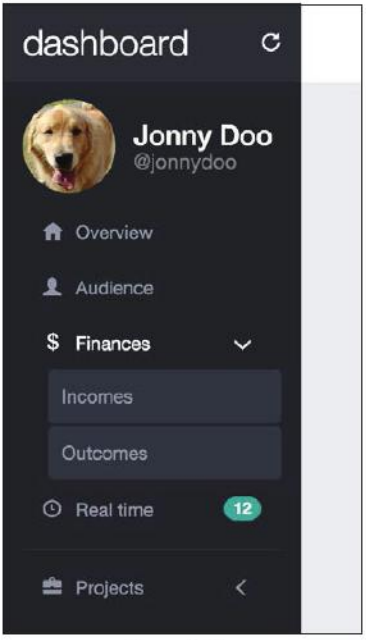


Рис. 9.8.

Добавление основного контента

Чтобы завершить первый этап работы над панелью мониторинга, займемся основным контентом в столбце `#main`. В этом разделе будет создан набор карточек, практически аналогичных карточкам в демонстрационном примере веб-приложения, и использованы несколько внешних плагинов для вывода диаграмм.

Но прежде определим общие CSS-правила для основного контента. Добавим следующие стили в файл `base.css`:

```
#main {
  padding-top: 7.2rem;
  display: -webkit-flex;
  display: flex;
  align-items: stretch;
  flex-flow: row wrap;
}

.card {
  position: relative;
  border-radius: 0.25em;
  box-shadow: 0 1px 4px 0 rgba(0, 0, 0, 0.37);
  background-color: #FFF;
  margin: 1.25rem;
  flex-grow: 5;
}

.card * {
  color: #252830;
}

.card-block {
  padding: 2rem;
}

.card-block h2 {
  margin: 0;
  margin-bottom: 1.5rem;
  color: #252830;
}
```

Как уже упоминалось, элемент будет заполнен карточками, поэтому для них нужно определить классы, практически те же, что использовались в карточках в примере веб-приложения. В данном случае, даже при том, что мы используем Bootstrap 4, эти классы обязательно следует добавить, чтобы обеспечить правильное оформление компонентов карточек.

Поместим в элемент `#main` первую карточку **Overall analysis**:

```
<div id="main" class="col-sm-offset-3 col-sm-9">
  <div class="card" id="pie-charts">
    <div class="card-block">
      <h2>Overall analysis</h2>

    </div>
  </div>
</div>
```

Круговые диаграммы

Первый плагин, который мы задействуем, называется *Easy Pie Chart* (https://rendro.github. io/easy-pie-chart/). Этот легковесный, узкоспециализированный плагин выводит круговые диаграммы.

Чтобы использовать этот плагин, его нужно установить с помощью диспетчера пакетов bower или npm, или просто загрузить ZIP-файл из репозитория, а затем подключить к HTML-файлу.

Воспользуемся jQuery-версией плагина, поместив файл JavaScript в папку `js` и добавив его загрузку в конец HTML-файла:

```
<script src="js/jquery-1.11.3.js"></script>
<script src="js/bootstrap.js"></script>
<script src="js/jquery.easypiechart.min.js"></script>
<script src="js/main.js"></script>
```

Внутрь только что созданной карточки `#pie-charts` поместим HTML-код, предназначенный для подключаемого модуля:

```
<div class="card" id="pie-charts">
  <div class="card-block">
    <h2>Overall analysis</h2>
    <div class="round-chart" data-percent="42">
      <span>
        42
          <small>
            % <br>
            recurring
          </small>
      </span>
    </div>
    <div class="round-chart" data-percent="87">
      <span>
        87
          <small>
            % <br>
            aware
          </small>
      </span>
    </div>
  </div>
</div>
```

Чтобы запустить плагин *Easy Pie Chart*, нужно применить его к элементу и с помощью атрибутов данных передать ему аргументы.

Например, в данном случае, был указан атрибут `data-percent`, определяющий процент наполнения диаграммы.

Перейдем в JavaScript-файл (файл `main.js`) и внутрь функции `ready` (точно так же, как в разделе «*Создание собственной модальной панели*» главы 8, «*Работа с JavaScript*»), добавим следующий код инициализации подключаемого модуля:

```
$(document).ready(function() {
  $('.round-chart').easyPieChart({
    'scaleColor': false,
    'lineWidth': 20,
    'lineCap': 'butt',
    'barColor': '#6d5cae',
    'trackColor': '#e5e9ec',
    'size': 190
  });
});
```

Здесь определяется стиль диаграммы. Но этого недостаточно! Добавим следующие CSS-правила в файл `base.css`:

```
.round-chart {
  display: inline-block;
  position: relative;
}

.round-chart + .round-chart {
  float: right;
}

.round-chart span {
  font-size: 3em;
  font-weight: 100;
  line-height: 1.7rem;
  width: 12rem;
  height: 4.4rem;
  text-align: center;
  position: absolute;
  margin: auto;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
}

.round-chart span > small {
```

```
    font-size: 2rem;
    font-weight: 400;
}
```

Здесь, помимо изменения некоторых интервалов, добавлено выравнивание текста по центру. Обновите страницу, и вы должны увидеть что-то похожее на рис. 9.9.

Рис. 9.9.

Как видите, карточка занимает всю строку. Это обусловлено макетом `flexbox` в элементе `#main`, оформление которого приводится ниже еще раз:

```
#main {
    padding-top: 7.2rem;
    display: -webkit-flex;
    display: flex;
    align-items: stretch;
    flex-flow: row wrap;
}
```

При наличии атрибутов `display: flex` и `align-items: stretch`, содержимое элемента `#main` будет растягиваться по горизонтали на полную его ширину.

Стиль `flex-flow` является сокращенной формой записи свойств `flex-direction` и `flex-wrap`. Можно одновременно использовать оба свойства, определив направление размещения элементов, в данном случае как `row` (в строку) и установить возможность переноса строки (`wrap`).

Кроме того, для каждой карточки определено свойство `flex-grow: 5`, указывающее, что она может принимать пять разных размеров контейнера `#main`.

Создание карточки оперативной статистики

Следующая карточка содержит статистическую информацию и для ее создания используются только компоненты Bootstrap. Итак, добавим после карточки `#pie-charts` следующий HTML-код:

```
<div class="card" id="quick-info">
  <div class="card-block">
    <h2>Quick stats</h2>
    <div class="quick-stats">
      <strong>Today:</strong>
      <span>78</span>
    </div>
    <div class="quick-stats">
      <strong>This month:</strong>
      <span>459</span>
    </div>
    <div class="quick-stats">
      <strong>All time:</strong>
      <span>54k</span>
    </div>
    <div class="quick-stats">
      <strong>Bounce rate:</strong>
      <span>81.08%</span>
    </div>
    <div class="quick-stats">
      <strong>Session duration:</strong>
      <span>00:01:41</span>
    </div>
    <div class="quick-stats">
      <strong>New session:</strong>
      <span>63.86%</span>
    </div>
  </div>
</div>
```

Карточка `#quick-info` содержит только обычные элементы, которые выводятся в отдельных строках внутри блока `.card`. Добавим несколько CSS-стилей для корректного отображения карточки:

```
#quick-info .card-block {
  display: flex;
  flex-direction: column;
}

#quick-info .quick-stats {
  font-size: 2rem;
  line-height: 3rem;
  border-bottom: 0.1rem solid #e5e9ec;
}

#quick-info .quick-stats strong {
  font-weight: 300;
}

#quick-info .quick-stats span {
  font-weight: 300;
  float: right;
  color: #8b91a0;
}
```

В браузере должен получиться результат, изображенный на рис. 9.10.

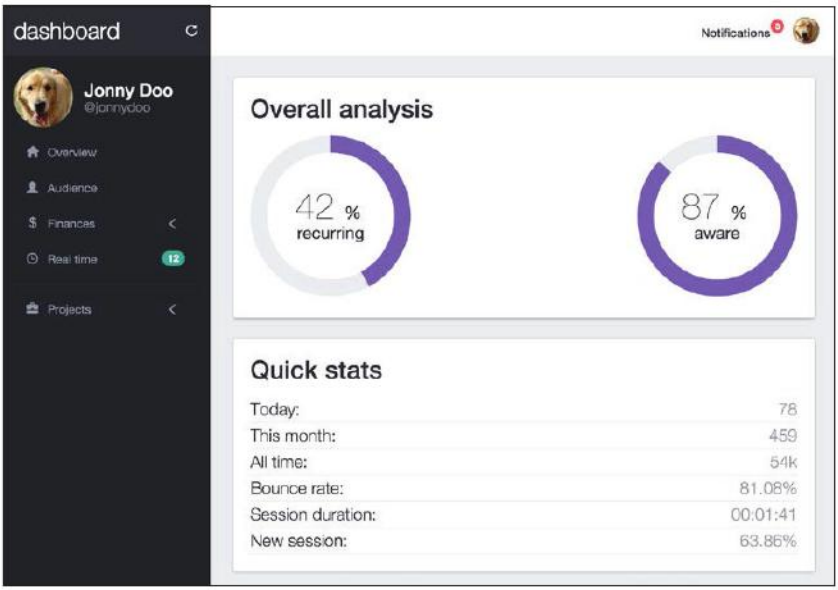


Рис. 9.10.

Но подождите! Если посмотреть на начальный макет, эти две карточки должны находиться в одной строке! Что же здесь произошло?

Это еще одно преимущество использования Flexbox! При использовании разметки Flexbox каждый элемент внутри контейнера адаптируется к доступной области. Скриншот на рис. 9.10 получен при среднем разрешении. На экране с большим разрешением элементы окажутся рядом, как показано на рис. 9.11.

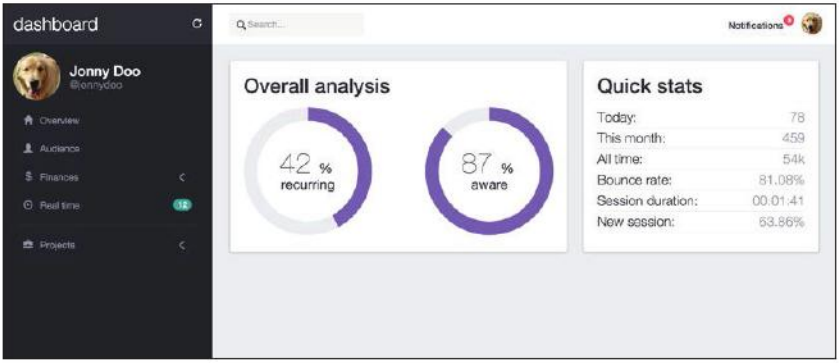


Рис. 9.11.

Радиальная диаграмма

Следующая диаграмма называется радиальной. Плагин Highcharts (http://www. highcharts.com/) является одним из самых популярных плагинов для создания диаграмм. Он поддерживает широкий спектр диаграмм, реализованных в отдельных модулях, благодаря чему вы можете загружать только те из них, которые будете использовать.

Подобно Easy Pie Chart, плагин Highcharts можно получить из различных источников. Загрузим первый нужный модуль Highcharts, добавив следующие строки в конец HTML-файла (в данном случае плагин загружается с помощью CDN):

```
<script src="https://code.highcharts.com/highcharts.js"></script>
<script src="https://code.highcharts.com/highcharts-more.js"></script>
```

Создадим еще один элемент `.card` ниже элемента `#quick-info` и присвоим ему идентификатор `#performance-eval`:

```
<div class="card" id="performance-eval">
  <div class="card-block">
    <h2>Performance evaluation</h2>
    <div class="spider-chart"></div>
```

```
    </div>
  </div>
```

Плагину Highcharts требуется совсем немного HTML-кода, поскольку он настраивается с помощью JavaScript. Для инициализации и настройки плагина добавим следующий код в функцию `ready`:

```
$('#performance-eval.spider-chart').highcharts({

  chart: {
    polar: true,
    type: 'area'
  },

  title: {
    text: ''
  },

  xAxis: {
    categories: ['Taming', 'Acessory', 'Development',
'Grooming', 'Awareness', 'Ration'],
    tickmarkPlacement: 'on',
    lineWidth: 0
  },

  yAxis: {
    gridLineInterpolation: 'polygon',
    lineWidth: 0,
    min: 0
  },

  tooltip: {
    shared: true,
    pointFormat: '<span style="color:{series.color}">
{series.name}: <b>${point.y:,.0f}</b><br/>'
  },

  legend: {
    align: 'right',
    verticalAlign: 'top',
    y: 70,
    layout: 'vertical'
  },

  series: [{
```

```
        name: 'Allocated resources',
        data: [45000, 39000, 58000, 63000, 38000, 93000],
        pointPlacement: 'on',
        color: '#676F84'
      },
      {
        name: 'Spent resources',
        data: [83000, 49000, 60000, 35000, 77000, 90000],
        pointPlacement: 'on',
        color: '#f35958'
      }]

});
```

В JavaScript-коде вызывается функция `highcharts` для селектора диаграммы `#performance-eval.spider-chart`. Описание всех использованных свойств можно найти в официальной документации. Отметим только, что выбор полярной проекции определяется ключом `polar: true` свойства `chart`.

Внешний вид панели мониторинга показан на рис. 9.12.

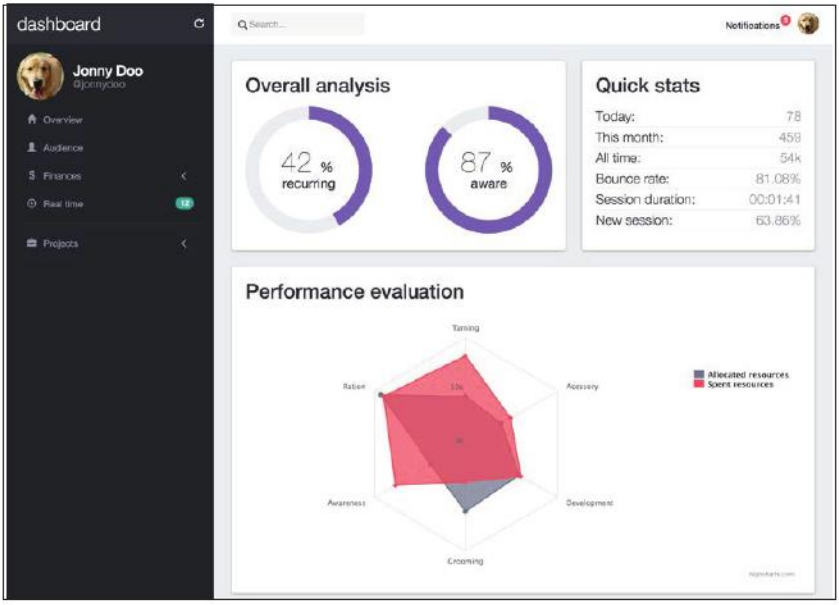


Рис. 9.12.

Параллельная загрузка

Еще одной интересной особенностью этих плагинов является поддержка анимации диаграмм, что делает отображаемый результат более наглядным.

Поместив загрузку JavaScript-кода в конец HTML-файла, мы увеличили скорость отображения страницы. Как результат, элементы, создаваемые библиотеками JavaScript, будут выводиться после отображения страницы, из-за чего на экране могут возникать временные искажения.

Для решения этой проблемы, многие страницы используют индикатор загрузки, который скрывается, как только документ будет готов к отображению.

Для этого, сразу после открывающего тега `<body>`, создадим блок `<div>` для вывода индикатора загрузки:

```
<body>
  <div class="loading">
  </div>
  ... <!-- прочая разметка HTML -->
</body>
```

Добавим файл с анимированным изображением индикатора загрузки с расширением `.svg` в папку изображений и определим CSS-правила:

```
.loading {
  position: fixed;
  z-index: 999;
  width: 100%;
  height: 100%;
  background-image: url('../imgs/loading.svg');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: center;
  background-color: #e5e9ec;
}
```

В результате будет создан элемент, отображаемый поверх всех прочих элементов, кроме панели навигации. Этот элемент заполнит всю ширину и высоту страницы, и в его центре будет находиться анимированное изображение индикатора загрузки.

Обновите страницу, и вы увидите индикатор, закрывающий страницу веб-приложения, как показано на рис. 9.13.

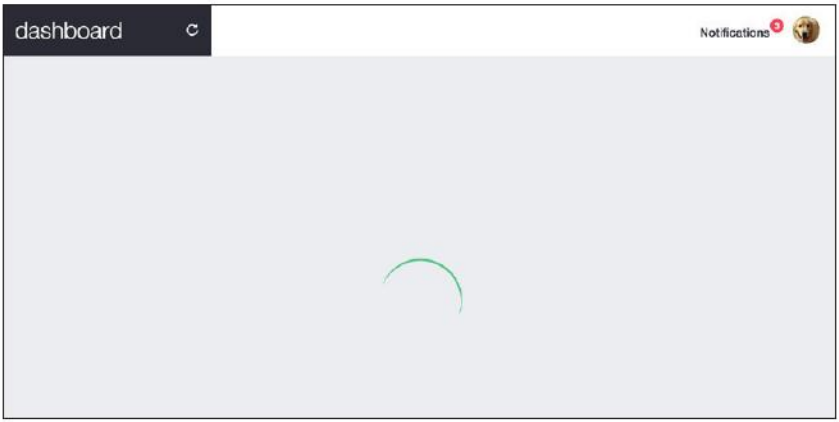


Рис. 9.13.

Как только страница будет готова к отображению, индикатор нужно удалить. Сначала в файле JavaScript, перед первой строкой в функции `$(document).ready`, вставим код удаления индикатора:

```
$(document).ready(function() {
   // удалить индикатор после загрузки страницы
   $('.loading').remove();
   // ниже следует прочий код JavaScript
});
```

Готово! Обновите веб-браузер, и то, что вы увидите на экране, будет зависеть от производительности вашего компьютера и скорости передачи по сети.

Элемент индикатора сейчас может показаться излишним, но только из-за того, что панель мониторинга пока содержит не все карточки, но они будут добавляться, поэтому уже сейчас стоит заняться решением этой проблемы.

Исправление вывода кнопки переключения на мобильных устройствах

При создании страницы был применен подход «в первую очередь для мобильных устройств», но некоторые компоненты все же выводятся неправильно или вообще не отображаются, и это нужно исправить.

Начнем с кнопки переключения `.navbar-toggle` в навигационной панели. Она выводится, но ее цвет оставляет желать лучшего. Исправим это с помощью CSS-правила:

```
.navbar-toggle {
  border-color: #252830;
  background-color: #e5e9ec; margin-top: 13px
}
.navbar-toggle .icon-bar {
  background-color: #252830;
}
```

Кнопка переключения теперь окрашена в серые тона и выглядит, как показано на рис. 9.14.

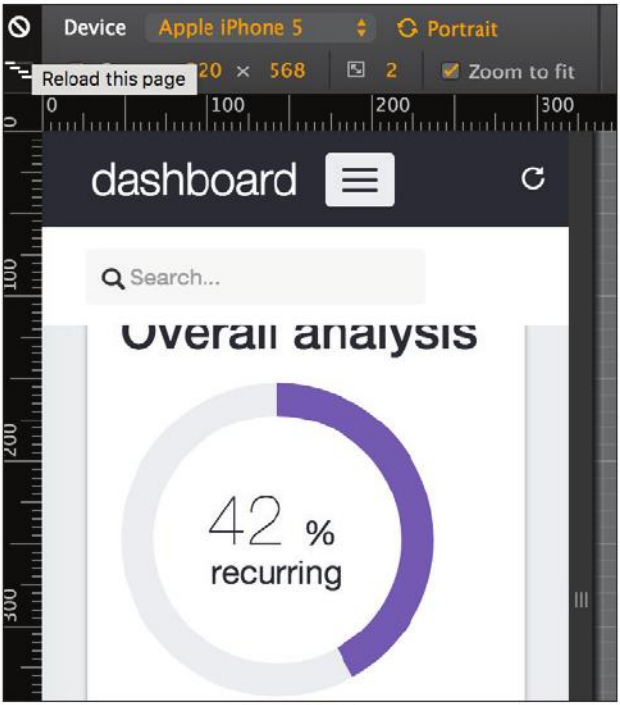


Рис. 9.14.

Как видите, здесь есть еще, чем заняться, чтобы улучшить отображение на мобильных устройствах и устройствах с малыми разрешениями. Этим мы займемся в следующих главах и добавим еще несколько очень интересных эффектов. Проявите терпение и увидите все сами!

Итоги

В этой главе начата работа над новым примером – веб-приложением панели мониторинга.

На первый взгляд пример кажется сложным, но мы рассмотрели каждую его строку в отдельности и применили несколько новых методик разработки внешнего интерфейса.

В этот раз мы нарисовали начальный макет, который нужно воссоздать. И это, здорово, потому что он служит путеводителем на пути к конечной цели. Как правило, работе над любым проектом начинается с создания подобного макета.

Сначала, была создана еще одна, более сложная навигационная панель, основанная на плавающем контейнере. Однако все остальное было реализовано практически по той же методике, что рассматривалась при знакомстве с этим компонентом Bootstrap.

Для меню слева был использован многоуровневый компонент навигации Bootstrap и плагин сворачивания.

Работу над основным разделом мы начали с импортирования внешних плагинов, позволивших создать красивые диаграммы для панели мониторинга. Кроме того, для лучшей адаптивности была использована flex-компоновка совместно с CSS-правилами.

Под конец был создан элемент индикатора загрузки и исправлена первая проблема, связанная с отображением при разных разрешениях. Дальнейшее исправление этих проблем будет продолжено в следующих главах.

Примите поздравления! Вы преодолели первую главу, посвященную последнему примеру в книге! Я уверен, что вы уже разобрались в идеях разработки с применением Bootstrap, позволяющих значительно увеличить ее производительность.

В следующей главе мы продолжим разработку панели мониторинга, точнее – ее основного раздела, а также добавим другие внешние плагины и карточки с помощью компонентов Bootstrap. Кроме того, мы исправим проблемы, связанные с отображением при разных разрешениях и рассмотрим другие плагины Bootstrap.

ГЛАВА 10. Оживление компонентов

Предыдущая глава была непростой! Панель мониторинга еще не полностью соответствует макету, нужно реализовать еще три карточки в основном разделе и исправить проблемы, связанные с выводом на устройствах с разным разрешением. После этого, предстоит создать еще несколько компонентов для панели мониторинга. Вперед, навстречу новым задачам!

В этой главе будут рассмотрены следующие темы:

- нестандартный флажок;
- интеграция с внешними плагинами;
- продвинутые медиа-запросы Bootstrap;
- расширенная настройка для разных разрешений экрана;
- плагин Carousel;
- плагин Scrollspy.

Создание карточек в основном разделе

Сверившись с макетом, нетрудно заметить, что нужно создать еще три карточки. Первая из них самая сложная, поэтому начнем с нее!

Следующий элемент `.card` является соединением комбинированной двумерной диаграммы с двумя сериями и флажков в стиле iOS. Скриншот на рис. 10.1 послужит напоминанием, как должна выглядеть эта карточка:

Для создания комбинированной диаграммы будет использована все та же библиотека Highcharts, а флажки будут созданы с помощью плагина switchery (https://github.com/abpetkov/switchery). После ознакомления с его документацией, создадим следующую HTML-разметку:

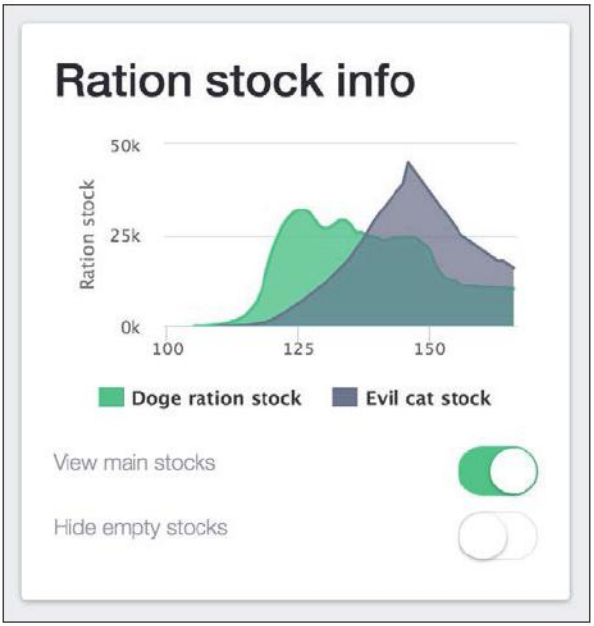


Рис. 10.1.

```
<div class="card" id="ration-stock">
  <div class="card-block">
    <h2>Ration stock info</h2>
    <div class="stacked-area"></div>
    <div class="switch">
      View main stocks
      <input type="checkbox" class="swithcery" checked />
      <div class="clearfix"></div>
    </div>
    <div class="switch">
      Hide empty stocks
      <input type="checkbox" class="swithcery" />
      <div class="clearfix"></div>
    </div>
  </div>
</div>
```

Разберем фрагмент выше. В нем определен элемент `div.stacked-area`, где будет создана диаграмма, и два элемента ввода с атрибутом `type="checkbox"` и классом `.switchery` для флажков.

Подключим CSS-файл плагина switchery в <head>, после CSS-файла Bootstrap:

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

и импортируем библиотеку switchery в конце разметки HTML:

```
<script src="js/switchery.min.js"></script>
```

Здесь потребуется минимум CSS-правил, поскольку все настройки будут реализованы в JavaScript-коде. Поэтому добавим лишь правила, определяющие высоту диаграммы и стиль текста флажков:

```
#ration-stock .stacked-area {
  height: 200px;
}

#ration-stock .switch {
  font-weight: 300;
  color: #8b91a0;
  padding: 0.5rem 0;
}

#ration-stock .switchery {
  float: right;
}
```

Основные настройки этой карточки выполняются в JavaScript. Начнем с инициализации плагина switchery и добавим следующие три строки в функцию .ready файла main.js:

```
var elems, switcheryOpts;

elems =
Array.prototype.slice.call(document.querySelectorAll
('.switchery'));

switcheryOpts = {
  color: '#1bc98e'
};

elems.forEach(function(el) {
  var switchery = new Switchery(el, switcheryOpts);
});
```

В переменной elems сохраняются элементы с классом .switchery. Этот плагин не использует библиотеку jQuery, поэтому поиск осу-

ществляется стандартными средствами JavaScript. Поиск выполняется, как предлагается в документации к плагину, а поскольку плагин switchery не является основной темой книги, за более подробной информацией я рекомендую обращаться к его документации.

Поиск выполняется с помощью команды `document.querySelectorAll('.switchery')`. Объект `Array` является глобальным объектом JavaScript и в большинстве современных браузеров используется для создания высокоуровневых списков объектов.

`prototype` – это объект, присутствующий в любом объекте JavaScript. Он содержит набор свойств и методов, присущих данному объекту.

Функция `slice` служит для извлечения поверхностной копии фрагмента массива и передачи ее в другой массив. То есть, в результате всех этих действий получается массив элементов с классом `.switchery`.

Далее устанавливаются параметры для плагина, в данном случае определяется только цвет фона с помощью свойства `color` переменной `switcheryOpts`. И, наконец, цикл `forEach` инициализирует объекты `Switchery`.

Обновите веб-страницу, вновь созданная карточка должна выглядеть, как показано на рис. 10.2.

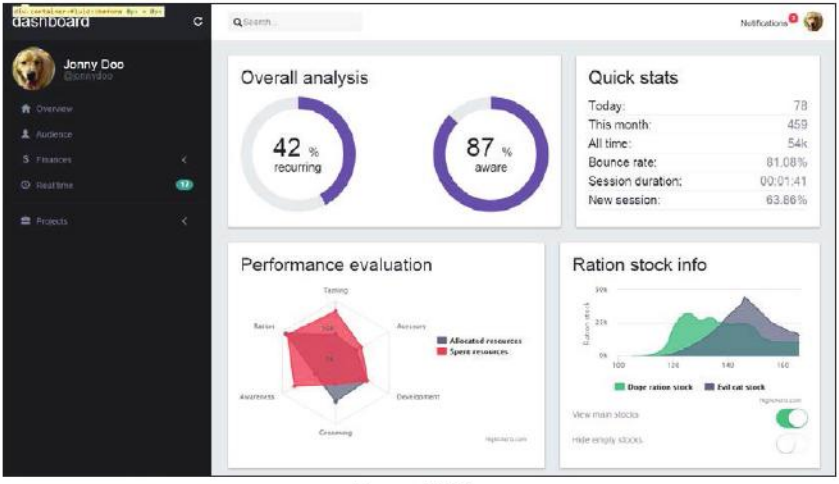


Рис. 10.2.

Создание карточки из компонентов Bootstrap

Для создания следующей карточки будут использованы индикатор выполнения, метки и бейджи. Эта карточка отображает определенную информацию в режиме реального времени с помощью индикатора выполнения, анимация которого реализована на JavaScript.

Прежде всего, добавим новую карточку с идентификатором `#real-time` после предыдуей карты `#ration-stock`:

```
<div class="card" id="real-time">
  <div class="card-block">
  <h2>Real time information</h2>
  </div>
</div>
```

Вслед за тегом `<h2>` добавим список со всеми необходимыми элементами – меткой, бейджем, индикатором выполнения и фрагментом текста – как показано ниже:

```
<div class="card" id="real-time">
  <div class="card-block">
  <h2>Real time information</h2>
    <ul class="list-unstyled">
      <li>
        Active dogs:
        <span class="label label-warning pull-right">
255</span>
      </li>
      <li>
        Silo status:
        <span class="badge ok pull-right">
          <span class="glyphicon glyphicon-ok" aria-hidden=
"true"></span>
        </span>
      </li>
      <li>
        Usage level:
        <div class="progress">
          <div class="progress-bar progress-bar-success"
role="progressbar" aria-valuenow="25" aria-valuemin="0"
aria-valuemax="100" style="width: 25%">
            <span class="sr-only">25%</span>
```

```
            </div>
            <div class="progress-bar progress-bar-warning
progress-bar-striped active" role="progressbar" aria-valuenow=
"38" aria-valuemin="0" aria-valuemax="100" style="width: 38%">
              <span class="sr-only">38% alocated</span>
            </div>
            <div class="progress-bar progress-bar-danger"
role="progressbar" aria-valuenow="5" aria-valuemin="0"
aria-valuemax="100" style="width: 5%">
              <span class="sr-only">5% reserved</span>
            </div>
          </div>
        </li>
        <li>
          Free space:
            <span id="free-space" class="pull-right"> 32%
          </span>
        </li>
      </ul>
    </div>
</div>
```

Поскольку здесь используются только элементы и компоненты Bootstrap, нам не понадобится слишком много CSS-правил:

```
#real-time li {
  font-size: 1.8rem;
  font-weight: 300;
  border-bottom: 0.1rem solid #e5e9ec;
  padding: 0.5rem 0;
}

#real-time .badge.ok {
  background-color: #1bc98e;
}

#real-time .badge span,
#real-time .label {
  color: #FFF;
}

#real-time .badge,
#real-time .label {
  margin-top: 0.25rem;
}
```

Эти правила определяют размер шрифта текста карточки и границы между элементами списка, а также цвет и размеры полей для бейджей и меток.

После обновления страница должна выглядеть, как показано на рис. 10.3.

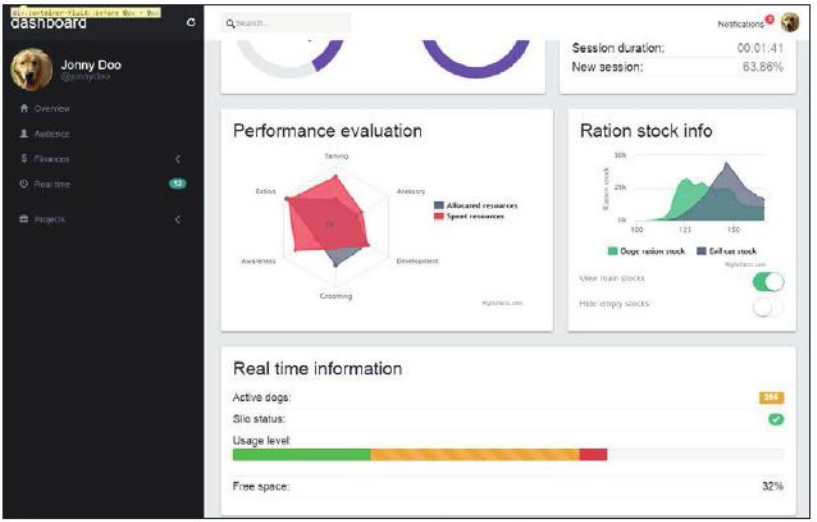


Рис. 10.3.

Новая карточка выглядит отлично! Добавим несколько CSS-правил для ее анимации, и периодически будем изменять процент свободного пространства. Для этого создадим следующую JavaScript-функцию:

```
changeMultiplier = 0.2;
window.setInterval(function() {
  var freeSpacePercentage;

  freeSpacePercentage = $('#free-space').text();
  freeSpacePercentage = parseFloat(freeSpacePercentage);

  delta = changeMultiplier * (Math.random() < 0.5 ? -1.0 : 1.0);

  freeSpacePercentage = freeSpacePercentage +
freeSpacePercentage * delta;
  freeSpacePercentage = parseInt(freeSpacePercentage);

  $('#free-space').text(freeSpacePercentage + '%');
}, 2000);
```

Данная функция будет вызываться каждые 2 секунды. За это отвечает метод `setInterval`, активируемый каждые 2000 мс (или 2 секунды).

Сначала функция выполняет парсинг текста в элементе `#free-space`. Затем с помощью параметра `changeMultiplier` рассчитывает случайное смещение в пределах 20 процентов вверх или вниз.

И, наконец, смещение умножается на текущее значение и добавляется к нему. Обновления значения элемента производит функция `.text()` из библиотеки jQuery. Она сохраняет переданный ей текст в качестве контента элемента, в данном случае, таким текстом является процент `freeSpacePercentage`, сгенерированный случайным образом.

Обновите страницу и убедитесь, что значение обновляются каждые 2 секунды.

Создание последней карточки

Последняя карточка содержит еще один график, на этот раз круговую диаграмму. И снова воспользуемся библиотекой Highcharts для ее реализации. Сначала создадим HTML-разметку карточки, поместив ее за карточкой `#real-time`:

```
<div class="card" id="daily-usage">
  <div class="card-block">
    <h2>Daily usage</h2>
    <div class="area-chart"></div>
  </div>
</div>
```

Затем определим высоту карточки в CSS-правилах:

```
#daily-usage .area-chart {
  height: 200px;
}
```

И, наконец, самое важное – вызовем функции в JavaScript-коде:

```
$('#daily-usage .area-chart').highcharts({
  title: {
    text: '',
  },
  tooltip: {
    pointFormat: '{series.name}:
<b>{point.percentage:.1f}%</b>'
```

```
    },
    plotOptions: {
      pie: {
        dataLabels: {
          enabled: true,
          style: {
            fontWeight: '300'
          }
        }
      }
    },
    series: [{
      type: 'pie',
      name: 'Time share',
      data: [
            ['Front yard', 10.38],
            ['Closet', 26.33],
            ['Swim pool', 51.03],
            ['Like a boss', 4.77],
            ['Barking', 3.93]
      ]
    }]
  });
```

Здесь выбирается вид графика `pie` и с помощью массива `data` определяются доли всех сегментов.

На рис. 10.4 показано, как должна выглядеть последняя карточка:

Мы сделали это! Разработка главной страницы панели мониторинга завершена. Теперь можно переходить к следующим страницам.

Исправление неправильного отображения на мобильных устройствах

Уменьшив размер окна браузера до размеров экранов мобильных устройств (которые в Bootstrap интерпретируются как сверхмалая область просмотра), можно заметить проблемы, возникающие при отображении некоторых элементов. Обратите внимание на рис. 10.5, что панель поиска и карточка с круговой диаграммой не выровнены друг относительно друга.

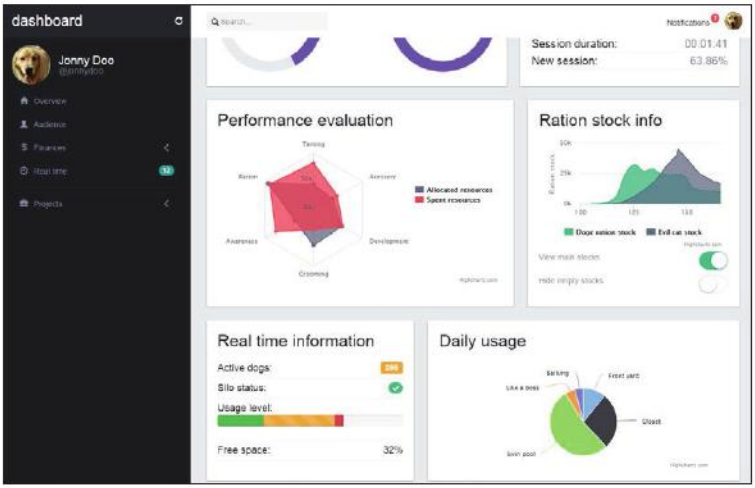


Рис. 10.4.

Этот скриншот получен в браузере Chrome после выбора области просмотра iPhone 6 с портретной ориентацией:

Рис. 10.5.

Что касается панели поиска, ее лучше выводить только при необходимости, например, после щелчка на кнопке. Для этого рядом с кнопкой обновления создадим еще один значок, щелчок на котором будет выводить панели поиска:

```
<div class="col-sm-3 top-left-menu">
  <div class="navbar-header">
    <a class="navbar-brand" href="dashboard.html">
      <h1>dashboard</h1>
    </a>

    <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#nav-menu"
aria-expanded="false">
      <span class="sr-only">Toggle navigation</span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
  </div>
  <a href="#" id="search-icon" data-toggle="tooltip"
data-placement="bottom" data-delay="500" title="Display search
bar" class="header-buttons pull-right visible-xs">
    <span class="glyphicon glyphicon-search" aria-hidden="true">
 </span>
  </a>
  <a href="#" data-toggle="tooltip" data-placement="bottom"
data-delay="500" title="Refresh data" class="header-buttons
pull-right">
    <span class="glyphicon glyphicon-repeat" aria-hidden="true">
</span>
  </a>
</div>
```

Обсудим этот код. Во-первых, мы изменили имя класса. Ссылка в определении значка обновления прежде имела класс `.header-refresh`. Теперь, поскольку кнопок стало несколько, выбрано более общее имя `.header-button`.

Кроме того, во вновь созданную кнопку добавлена всплывающая подсказка Bootstrap с текстом: `"Display search bar"` (Показать панель поиска), по аналогии со значком обновления.

В соответствии с этими изменениям, сменим имя класса в CSS-правиле:

```
nav .header-buttons {
  margin-left: auto;
  color: #FFF;
}
```

Теперь заголовок будет выглядеть, как показано на рис. 10.6.

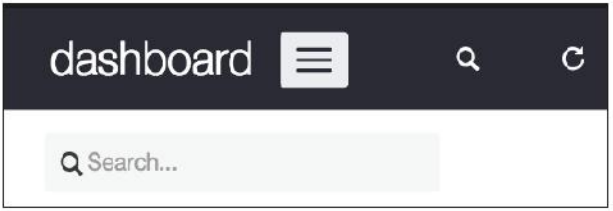


Рис. 10.6.

Проблема отображения панели поиска решена. Давайте теперь изменим классы элемента `form#search`. Чтобы улучшения его отображение, заменим классы `.hidden-sm` и `.col-md-3` на `.col-sm-3`.

Помните основные принципы сеточной системы? После добавления этого класса в элемент `form`, он будет занимать 3 из 12 столбцов шаблона на экранах с низким разрешением, и выводиться как линейный блок на экранах со сверхнизким разрешением.

Теперь скроем форму с помощью медиа-запроса CSS при отображении в сверхмалых областях просмотра:

```
@media(max-width:48em){
  form#search {
    display: none;
  }
}
```

Для отображения и скрытия панели поиска добавим JavaScript-события. Первое предназначено для отображения панели поиска по щелчку на значке лупы с идентификатором `#search-icon`. Перейдем в файл `main.js` и добавим следующую функцию:

```
$('#search-icon').on('click', function(e) {
  e.preventDefault();
  $('form#search').slideDown('fast');
  $('form#search input:first').focus();
});
```

Первым делом отключим реакцию по умолчанию на щелчок вызовом `e.preventDefault()`. Затем используем функцию `.slideDown` из библиотеки jQuery, чтобы выдвинуть элемент вниз. В данном случае – форму `form#search`.

После появления формы, передадим фокус полю ввода, при этом на экране мобильного телефона откроется клавиатура.

Было бы неплохо скрыть панель поиска после потери фокуса полем ввода. Для этого добавим следующий обработчик события:

```
$('form#search input').on('blur', function(e) {
  if($('#search-icon').is(':visible')) {
    $('form#search').slideUp('fast');
  }
});
```

Здесь использовано событие `blur`, возникающее при потере элементом фокуса. Обработчик события отыскивает элемент `#search-icon` и проверяет, является ли он видимым, что означает отображение в сверхмалой области просмотра, и тогда скрывает панель поиска вызовом функции `slideUp`, производящей действие, обратное действию функции `slideDown`.

Исправление навигационного меню

Щелкните на кнопке свертывания навигации (кнопка с пиктограммой бутерброда) и вы увидите, что меню `#nav-menu` выглядит неаккуратно, как показано на рис. 10.7. Это нужно исправить так же, как это было сделано в предыдущем примере веб-приложения.

Для этого необходимо удалить класс `.pull-right` из элемента `#nav-menu`. Классы `.pull-*` смещают элемент влево или вправо, при этом применяется модификатор `!important`, который невозможно переопределить. В данном случае, чтобы переопределить это правило, нужно удалить класс `.pull-right` и добавить стилевое правило, обеспечивающее смещение текущего элемента:

```
#nav-menu {
  float: right;
}
```

Создадим медиа-запрос для отображения элемента `#nav-menu` на сверхмалых устройствах и удалим в нем атрибут `float: right`:

```
@media(max-width:48em){
  #nav-menu {
    float: none;
```

```
        }
    }
```

После этого скроем `#nav-profile` и поместим его кнопку в список `#nav-menu`. Но прежде добавим класс `.hidden-xs` в элемент для вывода сведений о пользователе:

```
<div id="nav-profile" class="btn-group pull-right hidden-xs">
...
</div>
```

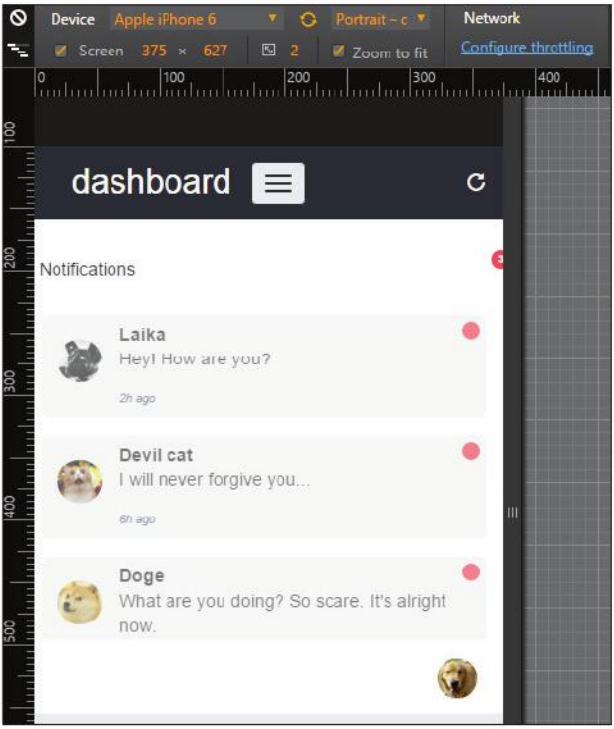


Рис. 10.7.

Это предотвратит вывод элемента на сверхмалых устройствах. Затем, в списке `#nav-menu > ul` добавим пункты в раскрывающийся список кнопки `#nav-profile`:

```
<div id="nav-menu" class="collapse navbar-collapse">
  <ul class="nav navbar-nav">
    <li>...</li>
```

```
        <li class="visible-xs">
          <a href="#">Profile</a>
        </li>
        <li class="visible-xs">
          <a href="settings.html">Setting</a>
        </li>
        <li class="visible-xs">
          <a href="#">Logout</a>
        </li>
      </ul>
    </div>
```

Обратите внимание, что новые элементы списка видимы только на сверхмалых устройствах, благодаря классу `.visible-xs`.

Новый список должен выглядеть так же, как уже имеющийся список оповещений. Поэтому добавим селектор нового списка элементов в CSS-стиль `#btn-notification`:

```
#btn-notifications .btn-link,
#nav-menu li a {
  padding-top: 1.5rem;
  color: #252830;
  font-weight: 500;
}
```

В развернутом виде список должен выглядеть, как показано на рис. 10.8.

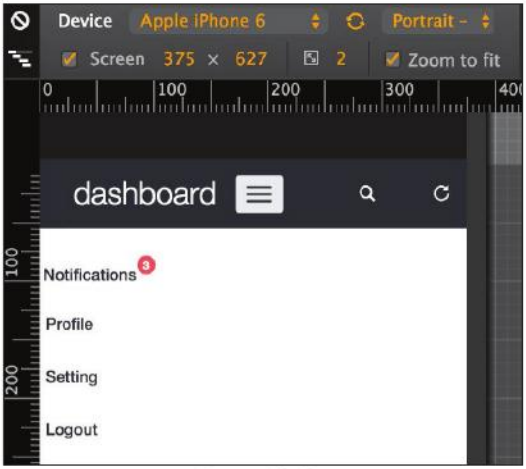


Рис. 10.8.

А теперь изменим область просмотра и проверим правильность отображения элементов в заголовке. Элемент `#nav-profile` будет выводиться только в областях просмотра от малых до больших и свертываться в `#nav-menu ul` при отображении в сверхмалых областях просмотра.

Оформление списка оповещений

Если щелкнуть на списке оповещений, чтобы открыть его, станут видны три проблемы: во-первых, бейдж, отображающий количество новых уведомлений сдвинут вправо; кнопка уведомлений не растянута на всю доступную ширину; и, наконец, внешний вид открытого списка оповещений оставляет желать лучшего.

Чтобы исправить смещение бейджа на кнопке оповещений, просто добавьте следующий CSS-код:

```
@media(max-width:48em){
  #nav-menu #btn-notifications > .badge {
    right: inherit;
    left: 10rem;
  }
}
```

Обратите внимание, что для изменения положения бейджа на сверхмалых устройствах использован медиа-запрос.

Чтобы изменить ширину кнопки оповещений, также нужно создать медиа-запрос:

```
@media(max-width:48em){
  #btn-notifications,
  #btn-notifications > button {
    width: 100%;
    text-align: left;
  }
}
```

Этот стиль изменит ширину раскрывающегося списка и самой кнопки.

Наконец, нужно изменить стиль списка оповещений в файле `main.css`:

```
@media(max-width:48em){
  #notification-list {
    margin: 1.25rem;
    margin-left: 2rem;
```

```
    background-color: #e5e9ec;
  }
  #notification-list a {
    background-color: #FFF;
    opacity: 1;
  }
}
```

Потрясающе! Обновите веб-страницу, список `#notification-list` должен выглядеть, как показано на рис. 10.9.

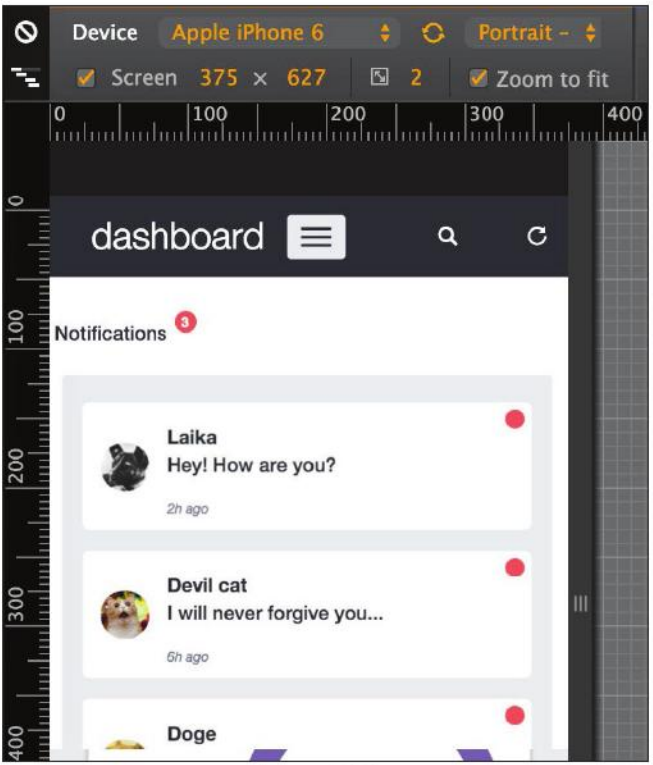


Рис. 10.9.

Добавление потерянного левого меню

Куда подевались элементы левого меню? Если внимательно изучить HTML-разметку меню `#side-menu`, можно заметить в ней класс `.hidden-xs`. То есть, при отображении на сверхмалых устройствах навигационные пункты нужно перенести в другое место.

Добавим ссылки в `#nav-menu ul`, как это было сделано для `#nav-profile`:

```
<div id="nav-menu" class="collapse navbar-collapse">
  <ul class="nav navbar-nav">
    <li>...</li>
    <li class="visible-xs">
      <a href="#">Audience</a>
    </li>
    <li class="visible-xs">
      <a href="#">Finances</a>
    </li>
    <li class="visible-xs">
      <a href="#">Realtime</a>
    </li>
    <li class="visible-xs">
      <a href="#">Projects</a>
    </li>

    <li role="separator" class="divider visible-xs"></li>
    ...
  </ul>
</div>
```

Изменим максимальную высоту `#nav-menu` при свертывании с помощью стиля:

```
#nav-menu.navbar-collapse {
  max-height: 39rem;
}
```

Для элемента `.divider` в списке создадим следующее CSS-правило:

```
#nav-menu .divider {
  height: 0.1rem;
  margin: 0.9rem 0;
  overflow: hidden;
  background-color: #e5e5e5;
}
```

Обратите внимание, что в списке `#nav-menu ul` кнопка оповещений будет отображаться выше вновь добавленных элементов, а пункты из `#nav-profile` – ниже. На рис. 10.10 показано законченное меню `#nav-menu`:

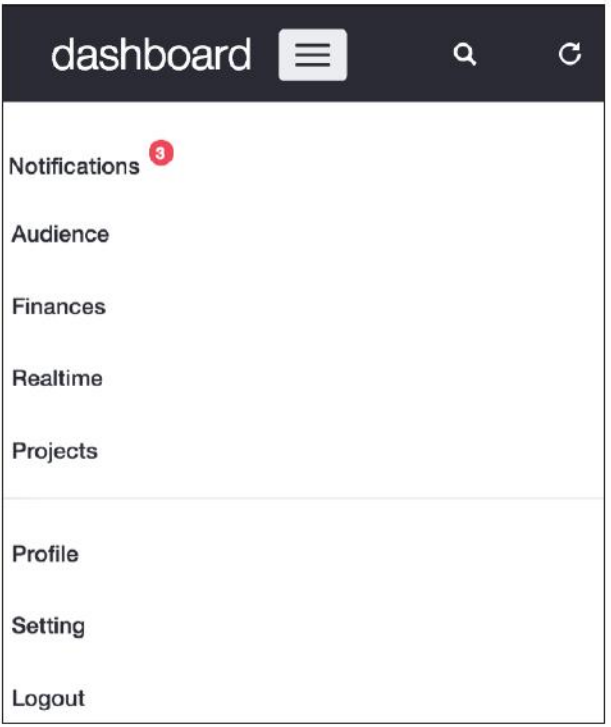


Рис. 10.10.

Выравнивание круговых диаграмм

Элементы `.round-chart` неправильно выравниваются внутри элемента `#pie-charts`. Этот недостаток можно быстро исправить с помощью двух CSS-правил, использующих медиа-запросы:

```
@media(max-width:48em){
  .round-chart,
  .round-chart canvas {
    display: block;
    margin: auto;
  }
  .round-chart + .round-chart {
    margin-top: 2rem;
    float: none;
  }
}
```

Обновив веб-страницу, вы должны увидеть результат, изображенный на рис. 10.11.

Рис. 10.11.

Отлично! Теперь панель мониторинга поддерживает все области просмотра и все устройства! Это было сложное, но очень полезное задание, в ходе выполнения которого была создана законченная панель мониторинга. А теперь, перейдем к другим страницам примера.

Знакомство с другими продвинутыми плагинами

Теперь, закончив главную страницу примера панели мониторинга и освоив практически все элементы, плагины и компоненты Bootstrap, займемся использованием еще не использовавшихся ранее плагинов JavaScript.

Создадим еще один файл – `audience.html` – в той же папке, где находится `dashboard.html`. Скопируем в него весь код из файла `dashboard.html`, кроме разметки внутри элемента `div#main`, поскольку в него будут внесены изменения.

Использование каруселей Bootstrap

В составе Bootstrap имеется плагин для создания слайд-шоу, который выполняет цикличную смену элементов. Однако при первом знакомстве он кажется излишне сложным и громоздким.

Прежде всего создадим следующий элемент внутри блока `div#main`:

```
<div id="main" class="col-sm-offset-3 col-sm-9">
  <div id="carousel-notification" class="carousel" data-
ride="carousel">
    ...
  </div>
</div>
```

Для взаимодействия с плагином карусели этому элементу следует дать идентификатор, в данном случае для внешнего блока `div` выбран идентификатор `#carousel-notification`.

Bootstrap организует карусель для элемента, содержащего атрибут данных `data-ride="carousel"`. Кроме того, этот элемент должен содержать класс `.carousel`, что обеспечивает применение к нему соответствующего CSS-стиля.

Внутри такого элемента нужно разместить слайды, которые будут сменять друг друга:

```
<div id="main" class="col-sm-offset-3 col-sm-9">
  <div id="carousel-notification" class="carousel" data-
ride="carousel">
    <div class="carousel-inner" role="listbox">
      <div class="item active">
        <img src="imgs/doge.jpg" width="512">
        <div class="carousel-caption">
          <p>What are you doing? So scare. It's alright
now.</p>
        </div>
      </div>
      <div class="item">
        <img src="imgs/cat.jpg" width="512">
        <div class="carousel-caption">
          <p>I will never forgive you...</p>
```

```
            </div>
          </div>
          <div class="item">
            <img src="imgs/laika.jpg" width="512">
            <div class="carousel-caption">
              <p>Hey! How are you?</p>
            </div>
          </div>
        </div>
      </div>
    </div>
```

Всего было создано три элемента. Все они помещены внутрь элемента `.carousel-inner`. Всем элементам внутри этого элемента присвоен класс `.item`.

Внутрь каждого элемента `.item` помещено изображение и элемент с классом `.carousel-caption`, содержащий текст подписи к слайду. Заметим, что первый слайд, кроме всего прочего, содержит класс `.active`, который обязательно должен быть добавлен к одному (и только к одному) из слайдов.

Если сейчас обновить страницу с каруселью Bootstrap в браузере, должно появиться изображение, показанное на рис. 10.12.

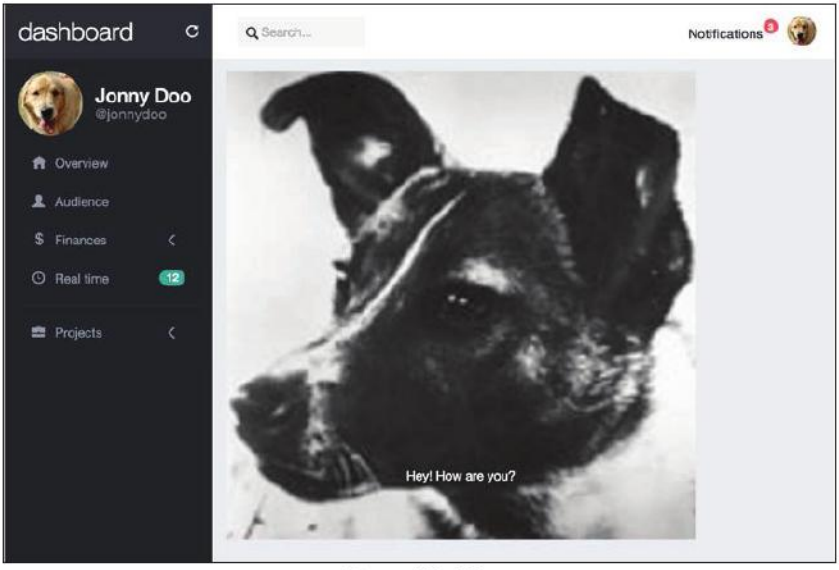


Рис. 10.12.

Если подождать 5 секунд, произойдет смена изображения и подписи. Определить другое значение интервала можно с помощью атрибута данных `data-interval` и или с помощью параметра в JavaScript, как это сделано в следующем примере:

```
$('.carousel').carousel({
  interval: 1000 // значение в миллисекундах
})
```

Также отметим, что при изменении слайдов не используется анимация. Для ее включения добавьте класс `.slide` в элемент `.carousel` и изображения станут выдвигаться слева.

Имейте в виду, что версии 8 и 9 браузера Internet Explorer не поддерживают CSS-анимацию. Поэтому плагин карусели будет работать только при наличии альтернативной реализации перехода, например, с помощью JQuery.

Настройка элементов карусели

Любой элемент карусели должен содержать обычный элемент `<p>`. Но так же можно добавлять другие элементы, как в следующем примере, где добавлен заголовок `h3`:

```
<div class="item">
  <img src="imgs/laika.jpg" width="512">
  <div class="carousel-caption">
    <h3>Laika said:</h3>
    <p>Hey! How are you?</p>
  </div>
</div>
```

Создание индикаторов слайдов

Карусели Bootstrap поддерживают создание индикаторов слайдов с маркерами. Чтобы воспользоваться этой возможностью, добавим следующий код после элемента `.carousel-inner`:

```
<div id="carousel-notification" class="carousel slide"
data-ride="carousel">
  <div class="carousel-inner" role="listbox">
    ...
```

```
  </div>

  <!-- Индикаторы -->
  <ol class="carousel-indicators">
    <li data-target="#carousel-notification" data-slide-to=
"0" class="active"></li>
     <li data-target="#carousel-notification" data-slide-to=
"1"></li>
     <li data-target="#carousel-notification" data-slide-to=
"2"></li>
  </ol>
</div>
```

Мы добавили упорядоченный список `<ol>`. Для каждого элемента указали идентификатор элемента карусели в атрибуте данных `data-target` (в данном случае `#carousel-notification`) и порядковый номер слайда с помощью атрибута `data-slide-to`. В результате создается список с количеством элементов, соответствующим количеству изображений.

Обновите страницу. Теперь должны появиться маркеры и включиться дополнительные функции карусели (переход между слайдами, вывод заголовков изображений и маркеров слайдов), как показано на рис. 10.13.

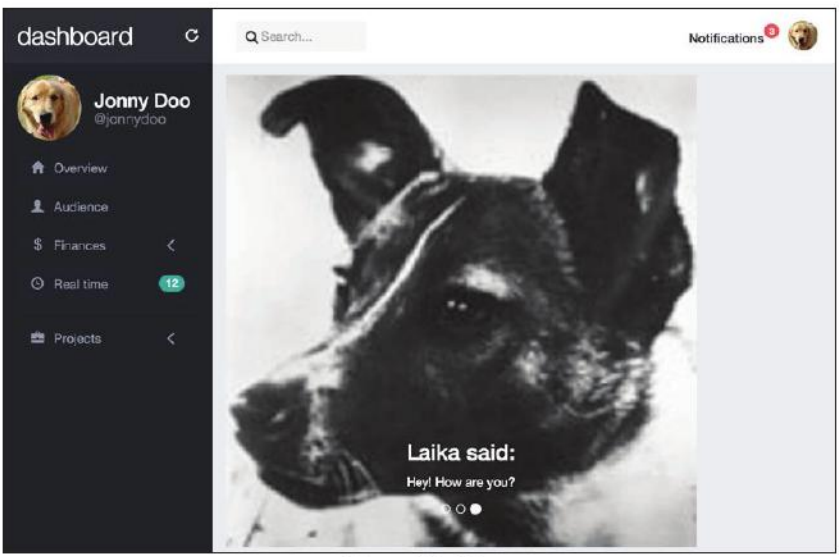


Рис. 10.13.

Добавление элементов управления навигацией

Еще один полезной возможностью каруселей Bootstrap является поддержка боковых навигационных элементов для смены слайдов.

Воспользуемся ею и добавим после индикаторов следующий HTML-код:

```
<div id="carousel-notification" class="carousel slide"
data-ride="carousel">
  <div class="carousel-inner" role="listbox">
    ...
  </div>

  <!-- Индикаторы -->
  <ol class="carousel-indicators">
  </ol>

  <!-- Элементы управления -->
  <a class="left carousel-control" href="#carousel-
notification" role="button" data-slide="prev">
    <span class="glyphicon glyphicon-chevron-left" aria-
hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="right carousel-control" href="#carousel-
notification" role="button" data-slide="next">
    <span class="glyphicon glyphicon-chevron-right" aria-
hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

Здесь созданы два элемента управления каруселью для смены слайда. Каждый из них заключен в тег `<a>` с классом `.carousel-control` и классом `.right` или `.left`, определяющим действие.

В атрибуте `href` указан идентификатор элемента карусели, по аналогии с атрибутом `data-target` в маркерах индикатора. Атрибут `data-slide` определяет действие, которое реализует элемент управления – это может быть `next`, для перехода к следующему слайду, или `prev`, для возврата к предыдущему.

На рис. 10.14 представлен законченный вариант карусели:

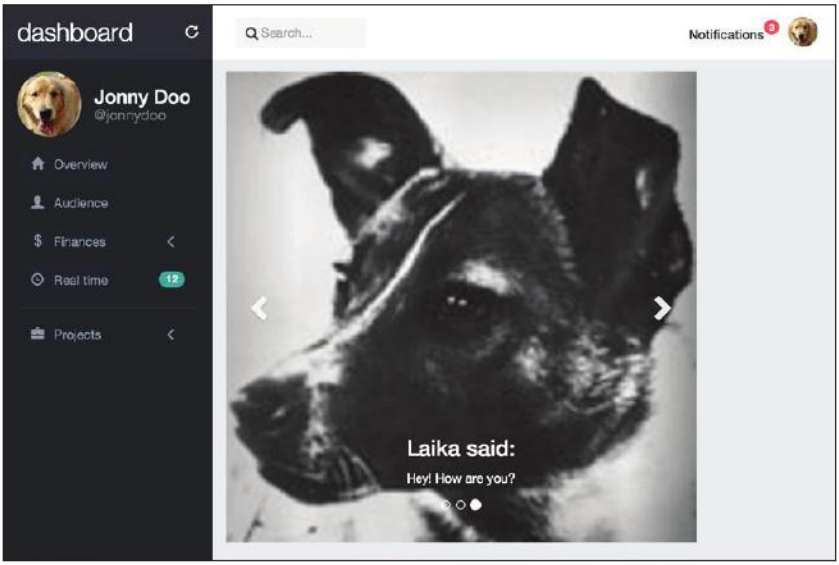


Рис. 10.14.

Использование нескольких каруселей Bootstrap на одной странице

Используя несколько каруселей Bootstrap на одной странице, не забывайте о присваивании уникальных идентификаторов родительским элементам (с классом `.carousel`). Кроме того, не забудьте привести в соответствие целевые элементы индикаторов и управления слайдами.

Прочие методы и параметры каруселей

Подобно другим плагинам Bootstrap, карусели имеют ряд параметров и методов. Их описание можно найти в официальной документации (http://getbootstrap.com/javascript/#carousel-options).

Однако некоторые параметры и методы хотелось бы упомянуть особо, например, `wrap`, определяющий необходимость циклического перехода из конца в начало. По умолчанию этот параметр имеет значение `true`.

Также с помощью JavaScript можно заставить карусель перейти к определенному слайду или просто принудить ее сменить слайд:

```
$('.carousel').carousel(2); // где 2 - порядковый номер слайда
                            // в атрибуте data-slide-to
```

Сменить слайд можно с помощью той же функции, передав ей в качестве аргумента строку `'prev'` или `'next'`:

```
$('.carousel').carousel('next') // или 'prev'
```

Подобно другим плагинам Bootstrap, карусель отлично справляется с задачей создания слайд-шоу. Карусель обладает широкими возможностями настройки для получения нужного результата. За дополнительной информацией обратиться к документации с описанием плагина.

Отслеживание прокрутки в Bootstrap

А теперь рассмотрим другой плагин Bootstrap, помогающий отслеживать прокрутку. Плагин отслеживания прокрутки часто используется для автоматического обновления элементов навигации Bootstrap в зависимости от позиции прокрутки. Многие сайты применяют его для управления боковой панелью навигации, в том числе и сайт электронной документации Bootstrap – в процессе прокрутки страницы происходит активизация разных элементов навигационной панели.

Для демонстрации этого плагина, создадим элемент `.card` на странице `audience.html`:

```
<div class="card">
  <div class="card-block">

  </div>
</div>
```

Внутри элемента `.card-block` создадим два столбца, левый для навигации и правый для размещения прокручиваемого содержимого. Никогда не забывайте добавлять классы `.col-*-*` в элементы `.row`:

```
<div class="card">
  <div class="card-block">
    <div class="row">
      <div class="col-sm-3" id="content-spy"></div>
      <div id="content" class="col-sm-9"></div>
    </div>
  </div>
</div>
```

Столбцу навигации присвоен идентификатор `#content-spy`, а столбцу с содержимым – идентификатор `#content`.

Начнем с создания левого навигационного столбца, использовав компонент `.nav-pills.nav-stacked`. Вы его помните? Освежим нашу память, задействовав его снова:

```
<div class="card">
  <div class="card-block">
    <div class="row">
      <div class="col-sm-3" id="content-spy">
        <ul class="nav nav-pills nav-stacked">
          <li role="presentation" class="active">
            <a href="#lorem">The Lorem</a>
          </li>
          <li role="presentation">
            <a href="#eros">The Eros</a>
          </li>
          <li role="presentation">
            <a href="#vestibulum">The Vestibulum</a>
          </li>
        </ul>
      </div>
      <div id="content" class="col-sm-9"></div>
    </div>
  </div>
</div>
```

Здесь мы создали список `<ul>` с классами `.nav`, `.nav-pills` и `.nav-stacked`, и три элемента списка со ссылками на идентификаторы (`#lorem`, `#eros` и `#vestibulum`). Эти идентификаторы мы используем позднее при отслеживании прокрутки.

А теперь создадим содержимое второго столбца. Оно включает три элемента `<div>`, помещенных в элемент `#content`, с идентификаторами, соответствующими атрибуту `href` в ссылках:

```
<div class="card">
  <div class="card-block">
    <div class="row">
      <div class="col-sm-3" id="content-spy">
        <ul class="nav nav-pills nav-stacked">
          <li role="presentation" class="active">
            <a href="#lorem">The Lorem</a>
          </li>
```

```
            <li role="presentation">
              <a href="#eros">The Eros</a>
            </li>
            <li role="presentation">
              <a href="#vestibulum">The Vestibulum</a>
            </li>
          </ul>
        </div>
        <div id="content" class="col-sm-9">
          <div id="lorem">
            <h2>The Lorem</h2>
            <p>
              Lorem ipsum dolor sit amet... <!-- Прочий текст -->
            </p>
          </div>
          <div id="eros">
            <h2>The Eros</h2>
            <p>
              Curabitur eget pharetra risus... <!-- Прочий текст -->
            </p>
          </div>
          <div id="vestibulum">
            <h2>The Vestibulum</h2>
            <p>
              Integer eleifend consectetur... <!-- Прочий текст -->
              <img src="imgs/jon.png" class="img-responsive">
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
```

Обратите внимание, что идентификаторы во всех элементах `<div>` соответствуют указанным в атрибутах `href` ссылок в элементах списка. Это необходимо синхронизации прокрутки с активным пунктом в элементе `nav`.

Не забудьте добавить класс `.img-responsive` в изображение в конце третьего элемента содержимого.

Активировать плагин можно двумя способами: с помощью атрибутов данных или в коде на JavaScript. Если вы выбрали JavaScript, добавьте в файл `main.js` следующей вызов:

```
$('#content').scrollspy({
  target: '#content-spy'
})
```

Обновите страницу и понаблюдайте за работой плагина. Если вы решите выбрать способ на основе атрибутов данных, добавьте атрибуты `data-spy` и `data-target` в элемент `#content`:

```
<div class="card">
  <div class="card-block">
    <div class="row">
      <div class="col-sm-3" id="content-spy">
        <ul class="nav nav-pills nav-stacked">
          <li role="presentation" class="active">
            <a href="#lorem">The Lorem</a>
          </li>
          <li role="presentation">
            <a href="#eros">The Eros</a>
          </li>
          <li role="presentation">
            <a href="#vestibulum">The Vestibulum</a>
          </li>
        </ul>
      </div>
      <div id="content" class="col-sm-9" data-spy="scroll"
data-target="#content-spy">
        <div id="lorem">
          <h2>The Lorem</h2>
          <p>
            Lorem ipsum dolor sit amet... <!-- Прочий
текст -->
          </p>
        </div>
        <div id="eros">
          <h2>The Eros</h2>
          <p>
            Curabitur eget pharetra risus... <!-- Прочий
текст -->
          </p>
```

```
            </div>
            <div id="vestibulum">
              <h2>The Vestibulum</h2>
              <p>
                Integer eleifend consectetur... <!-- Прочий
текст -->
                <img src="imgs/jon.png" class="img-responsive">
              </p>
            </div>
          </div>
        </div>
      </div>
    </div>
```

Атрибуту `data-spy` должно быть присвоено значение `scroll`, чтобы сообщить плагину, что слежение должно вестись за действием прокрутки. Атрибут `data-target` играет ту же роль, что и параметр `target` в JavaScript. Он определяет отслеживаемый элемент, в данном случае на `#content-spy`.

Для усиления эффекта при прокрутке, создадим следующее CSS-правило, ограничивающее высоту и регулирующую прокрутку контента:

```
#content {
  height: 30em;
  overflow: auto;
}
```

Обновите веб-страницу, теперь карточка должна выглядеть, как показано на рис. 10.15. Обратите внимание, что здесь выполнена прокрутка до второго пункта.

Отлично! Только что, вы освоили еще один плагин Bootstrap! Плагин отслеживания прокрутки особенно полезен на страницах с объемным контентом, разбитым на разделы. Пользуйтесь им.

Итоги

В этой главе был успешно завершен пример панели мониторинга. Мы создали еще несколько карточек и настроили вывод для разных устройств. В результате получилась отличная панель мониторинга, которую можно использовать в различных контекстах и веб-приложениях.

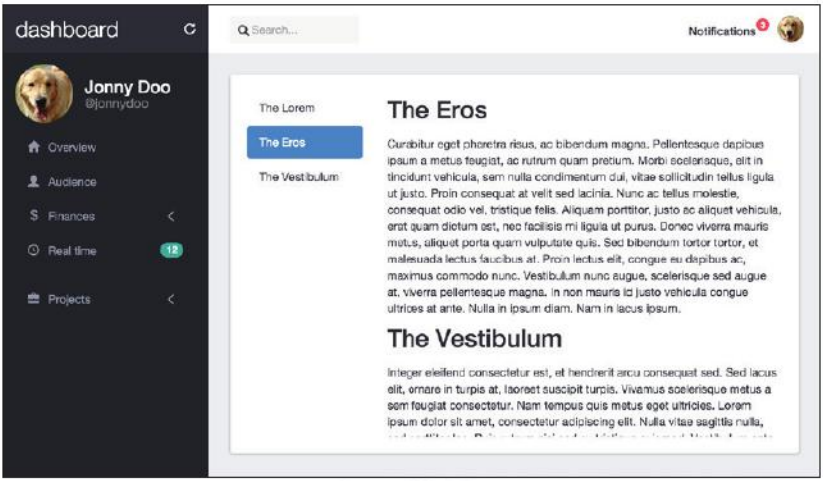


Рис. 10.15.

В конечном итоге был достигнут впечатляющий уровень выразительности и качества панели мониторинга, которая поддерживает любые устройства.

Затем, были рассмотрены еще два плагина Bootstrap. Мы познакомились с плагином отслеживания прокрутки, который отлично подходит для страниц с объемным контентом и необходимостью навигации между разделами. В данном примере плагин использовался с образцом заранее подготовленного текста, но его можно применять везде, где потребуется.

Кроме того, мы исследовали карусели Bootstrap. Каруселью называется замечательный плагин для создания слайд-шоу изображений с подписями. Я считаю, что единственный недостаток этого подключаемого модуля заключается в необходимости ввода слишком большого объема кода. Представьте, насколько было бы проще, если бы такую же карусель можно было создать с помощью нескольких строк кода. Я думаю, мы сможем добиться этого в последней главе!

Следующая глава посвящена настройке плагинов и, в качестве примера такого плагина, использована карусель. Мы создадим свою обертку, чтобы уменьшить объем кода, вводимого вручную при использовании карусели, и автоматизировать создание плагинов. Кроме того, подробно будет рассмотрена настройка плагинов Bootstrap.

Конечно, создание подключаемых модулей для Bootstrap – непростая задача, но я уверен, что вы с ней успешно справитесь.

ГЛАВА 11. Переделка на свой лад

На данный момент, вы уже можете называть себя мастером Bootstrap! Вы освоили фреймворк, что удается немногим, и должны гордиться этим!

А сейчас, вы столкнетесь уже не с учебной задачей. В этой главе описывается процесс создания и настройки своего плагина для Bootstrap. Это будет тяжело, но если вы уже дошли до этого места, стоит сделать еще шаг, чтобы стать настоящим мастером Bootstrap.

В главе будут рассмотрены следующие темы:

- настройка компонентов Bootstrap;
- настройка плагинов Bootstrap;
- создание плагинов Bootstrap.

Эта глава завершает книгу. Я надеюсь, что эта последняя глава поможет вам усовершенствовать навыки работы с фреймворком Bootstrap.

Чтобы следовать за обсуждением в этой главе, создайте файл `sandbox.html` и добавьте в него стандартный шаблон, использовавшийся во всех предыдущих примерах. Все фрагменты кода в этой главе мы будем помещать в этот файл.

Настройка компонентов Bootstrap

За все годы работы с Bootstrap, одним из основных вопросов, которые вставали передо мной, был вопрос изменения внешнего вида компонентов Bootstrap.

Обычно ответом было создание CSS-правил, переопределяющих их стили. Однако такой подход не всегда позволяет достичь желаемого.

Этот раздел посвящен настройке компонентов Bootstrap. Мы уже выполняли такую настройку в предыдущих главах, но здесь эта тема будет рассмотрена более глубоко. Начнем с настройки кнопки.

Настройка кнопки

Начать с кнопок следует по двум причинам. Во-первых, это очень простой компонент и, во-вторых, кнопки приходится настраивать чаще всего.

Предположим, что кнопка размещена на странице, которая загружает полную поддержку Bootstrap. Назовем эту страницу песочницей. HTML-разметка кнопки приведена ниже:

```
<button type="button" class="btn btn-primary" aria-pressed=
"false" autocomplete="off">
  This is a simple button
</button>
```

Такая кнопка не раз использовалась нами. Это простая кнопка с классами `.btn` и `.btn-default` синего цвета, как показано на рис. 11.1.

Рис. 11.1.

Чтобы окрасить кнопку в другой цвет нужно воспользоваться одним из контекстных классов Bootstrap (`.btn-success`, `.btn-info`, `.btn-warning`, `.btn-danger` и так далее), применив один из них совместно с базовым классом `.btn`.

Если потребуется добавить новый цвет, можно создать новый класс и определить псевдо-классы. Предположим, что понадобилось создать фиолетовую кнопку, определяемую классом `.btn-purple`. Для этого нужно определить основное CSS-правило:

```
.btn-purple {
  color: #fff;
  background-color: #803BDB;
  border-color: #822FBA;
}
```

и все псевдо-классы кнопки:

```
.btn-purple:hover,
.btn-purple:focus,
.btn-purple:active,
.btn-purple.active {
  color: #ffffff;
  background-color: #6B39AD;
  border-color: #822FBA;
}
```

Теперь, при любом взаимодействии с кнопкой (например, при наведении на нее указателя мыши), кнопка будет менять цвет фона на чуть более темный. Но все псевдо-классы не обязательно должны иметь один и тот же стиль, он может быть разным.

На рис. 11.2 представлена новая кнопка. В кнопке слева мы заменили класс `.btn-default` классом `.btn-purple`, а в кнопке справа – классом `.btn-purple:hover`:

Рис. 11.2.

Использование кнопок-переключателей

В Bootstrap имеется отличное средство для создания кнопок-переключателей, встроенное в фреймворк и достаточно удобное. Рассмотрим отдельную кнопку-переключатель. Для этого создадим на странице-песочнице обычную кнопку:

```
<buttontype="button"class="btnbtn-primary"autocomplete="off">
    Single toggle
</button>
```

Чтобы превратить эту кнопку в кнопку-переключатель нужно добавить в нее атрибуты данных `data-toggle="button"` и `aria-pressed="true"`. При этом кнопка будет преобразована в кнопку-переключатель. Теперь при щелчке на кнопке Bootstrap добавит в нее класс `.active`, придающий вид нажатой кнопки:

```
<button type="button" class="btn btn-default" data-toggle=
"button" aria-pressed="false" autocomplete="off">
      Single toggle
</button>
```

Кнопки-переключатели в виде флажков

Кнопки-переключатели можно превратить в флажки или радиокнопки. Прежде вспомним идею группы кнопок. Создадим обычный элемент `.btn-group`:

```
<div class="btn-group">
  <button class="btn btn-default">
    Laika
  </button>
  <button class="btn btn-default">
    Jonny
  </button>
  <button class="btn btn-default">
    Doge
  </button>
</div>
```

Идея группы кнопок подразумевает создание элемента `div` с классом `.btn-group` и размещения в нем набора элементов `button`. Однако нам нужен набор флажков, а не кнопок, поэтому заменим элементы `button` элементами `label` и `input` вида `checkbox`:

```
<div class="btn-group">
  <label class="btn btn-default">
    <input type="checkbox" autocomplete="off"> Laika
  </label>
  <label class="btn btn-default">
    <input type="checkbox" autocomplete="off"> Jonny
  </label>
  <label class="btn btn-default">
    <input type="checkbox" autocomplete="off"> Doge
  </label>
</div>
```

После обновления страницы список кнопок превратится в группу флажков с метками, как показано на рис. 11.3.

Рис. 11.3.

Чтобы превратить их в переключатели и скрыть флажки, нужно лишь добавить к атрибут данных `data-toggle="buttons"`.

Существует также возможность предварительно установить один из флажков, для этого нужно добавить класс `.active` в метку, и атрибут `checked="checked"` в элемент `input`:

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-default">
    <input type="checkbox" autocomplete="off"> Laika
  </label>
  <label class="btn btn-default active">
    <input type="checkbox" autocomplete="off" checked="checked">
Jonny
  </label>
  <label class="btn btn-default">
    <input type="checkbox" autocomplete="off"> Doge
  </label>
</div>
```

На рис. 11.4 показан окончательный вариант с выбранным при загрузке страницы вторым флажком:

Рис. 11.4.

Кнопка в качестве радиокнопки

Кнопки-переключатели можно также преобразовать в радиокнопки. Процедура очень похожа на проделанную выше с флажками. Нужно лишь изменить тип элемента ввода с `type="checkbox"` на `type="radio"`:

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-default">
    <input type="radio" autocomplete="off"> Laika
  </label>
  <label class="btn btn-default active">
    <input type="radio" autocomplete="off" checked="checked">
Jonny
  </label>
```

```
    <label class="btn btn-default">
      <input type="radio" autocomplete="off"> Doge
    </label>
  </div>
```

При этом будет создана группа кнопок `.btn-group`, допускающая возможность включения только одной кнопки.

Настройка с помощью JavaScript

Кнопки можно настраивать с помощью JavaScript. Например, любую кнопку-переключатель можно переключить вызовом метода:

```
$('button selector').button('toggle')
```

При этом состояние кнопки измениться с активного на неактивное.

В версиях Bootstrap до 3.3.6, из JavaScript можно было изменить надпись на кнопке, вызвав метод `button` со строковым аргументом. Но перед этим требовалось определить статический текст. Например, определим кнопку с атрибутом `data-statesample-text="What a sample"`:

```
<buttontype="button"class="btnbtn-primary"autocomplete="off"
data-statesample-text="What a sample">
  Single toggle
</button>
```

Из JavaScript-кода можно изменить надпись кнопки с помощью вызова:

```
$('button').button('statesample');
```

Вернуть оригинальную надпись можно следующим образом:

```
$('button').button('reset');
```

Однако эта функция является уже устаревшей для версий новее 3.3.6 и будет удалена в версии Bootstrap 4.

Настройка плагинов

Имеется возможность настраивать не только компоненты, но и плагины Bootstrap.

Продемонстрируем эту возможность на примере модальной панели. Этот плагин является одним из самых востребованных. Модальные панели могут создаваться в отдельном потоке, без изменения контента веб-страницы.

Создадим поле ввода и кнопку для открытия модальной панели. По имени пользователя получим данные из открытого прикладного программного интерфейса GitHub API и выведем часть сведений в модальную панель. Для этого, добавим в страницу-песочницу следующую разметку:

```
<!-- Кнопка открытия модального окна -->
<input id="github-username" type="text" class="form-control"
placeholder="Type your github username here">
<button type="button" class="btn btn-success btn-lg btn-block"
data-toggle="modal" data-target="#githubModal">
  Launch demo modal
</button>

<!-- Модальное окно -->
<div class="modal fade" id="githubModal" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal"
aria-label="Close"><span aria-hidden="true">&times;</span>
</button>
        <h4 class="modal-title"></h4>
      </div>
      <div class="modal-body">
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
        <button type="button" class="btn btn-success">Save
changes</button>
      </div>
    </div>
  </div>
</div>
```

Обновите страницу и вы увидите поле ввода с кнопкой. Если щелкнуть на кнопке, откроется пустая модальная панель, для взаимодействия с которой будет использован код на JavaScript.

В коде будет использоваться событие `show.bs.modal`, которое возникает при каждом отображении модальной панели (оно уже рассматривалось выше):

```
$('#githubModal').on('show.bs.modal', function (e) {
  var $element = $(this),
    url = 'https://api.github.com/users/{username}';
});
```

Внутри функции определяются две переменные. В переменную `$element` записывается ссылка на элемент, сгенерировавший событие, в данном случае – на модальную панель `#githubModal`. Переменной `url` присваивается адрес GitHub API. Заменим параметр `{username}` текстом из поля ввода:

```
$('#githubModal').on('show.bs.modal', function (e) {
  var $element = $(this),
    url = 'https://api.github.com/users/{username}';

  url = url.replace(/{username}/, $('#github-username').
val());
});
```

Затем реализуем отправку GET-запроса к API, чтобы получить информацию о пользователе в формате JSON.

JSON – это открытый, стандартный формат передачи данных в виде набора пар ключ/значение. Он широко используется для получения данных из веб-служб и прикладных программных интерфейсов API, таких как GitHub.

Чтобы выполнить запрос, воспользуемся функцией `$.get` из библиотеки jQuery. Ей следует передать адрес URL и функцию обратного вызова, принимающую JSON-объект `data`, возвращенный сервером:

```
$('#githubModal').on('show.bs.modal', function (e) {
  var $element = $(this),
    url = 'https://api.github.com/users/{username}';

  $.get(url, function(data) {
    console.log(data);
  });
});
```

Теперь обновите страницу, введите имя пользователя в поле ввода и щелкните на кнопке. Когда откроется модальная панель, загляните в консоль, она должна содержать данные запроса, как показано на рис. 11.5.

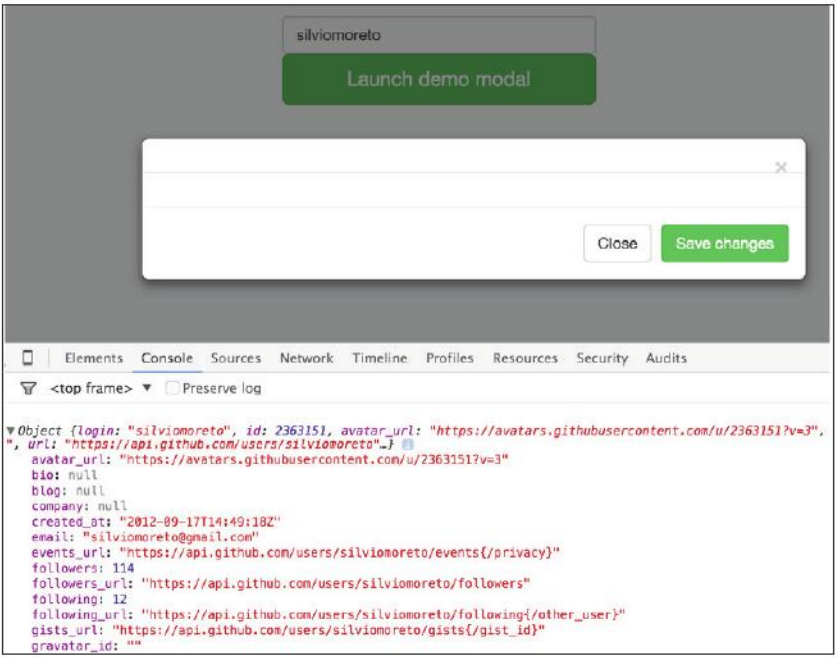


Рис. 11.5.

А теперь займемся парсингом объекта `data` и выводом части полученных сведений в модальную панель. Используем для этого тот же принцип замены переменной `url`. Добавим к уже существующим переменные, связанные с шаблоном.

Создадим шаблон с двумя столбцами: левый – для изображения аватара пользователя, полученного из объекта, а правый – для основных сведений о пользователе. Итак, добавим в код на JavaScript строки, выделенные жирным:

```
$('#githubModal').on('show.bs.modal', function (e) {
  var $element = $(this),
    url = 'https://api.github.com/users/{username}',
    title = 'Hi, my name is {name}',
    content = '' +
      '<div class="row">' +
        '<img src="{img}" class="col-sm-3">' +
        '<p class="col-sm-9" id="bio">{bio}</p>' +
      '</div>',

    bio = '' +
```

```
            'At moment I have {publicRepo} public repos ' +
            'and {followers} followers.\n' +
            'I joined Github on {dateJoin}';

        $.get(url, function(data) {
            console.log(data);
        });
    });
```

Здесь создаются три переменные шаблона, которые мы заменим данными из объекта `data`. Внутри функции `get` заменим переменные и создадим законченный шаблон.

Принцип тот же, что выше применялся в отношении `url`, то есть простая замена ключей, заключенных в фигурные скобки, значениями из объекта `data`:

```
$('#githubModal').on('show.bs.modal', function (e) {
    var $element = $(this),
        url = 'https://api.github.com/users/{username}',
        title = 'Hi, my name is {name}',
        content = '' +
            '<div class="row">' +
                '<img src="{img}" class="col-sm-3">' +
                '<p class="col-sm-9" id="bio">{bio}</p>' +
            '</div>',
        bio = '' +
            'At moment I have {publicRepo} public repos ' +
            'and {followers} followers.\n' +
            'I joined Github on {dateJoin}';

    url = url.replace(/{username}/, $('#github-username').val());

    $.get(url, function(data) {
        title = title.replace(/{name}/, data.name);

        bio = bio.replace(/{publicRepo}/, data.public_repos)
                 .replace(/{followers}/, data.followers)
                 .replace(/{dateJoin}/, data.created_at.split('T')[0]);

        content = content.replace(/{img}/, data.avatar_url)
                         .replace(/{bio}/, bio);

        $element.find('.modal-title').text(title);
        $element.find('.modal-body').html(content);
    });
});
```

После выполнения всех замен, переменные шаблона помещаются в модальную панель. Выполняется поиск элемента заголовка `.modal-title` и в него записывается простой текст, затем, в элемент `.modal-body` вставляется HTML-разметка.

Разметка HTML и простой текст передаются в jQuery по-разному. При передаче HTML-разметки следует проверить ее правильность, иначе у клиента могут возникнуть проблемы. Итак, обратите внимание, что при записи простого текста в `.modal-title` используется метод `text`, а при записи HTML-разметки в `.modal-body` используется метод `html`.

В браузере введите имя пользователя GitHub, щелкните на кнопке и вы должны увидеть модальную панель, изображенную на рис. 11.6.

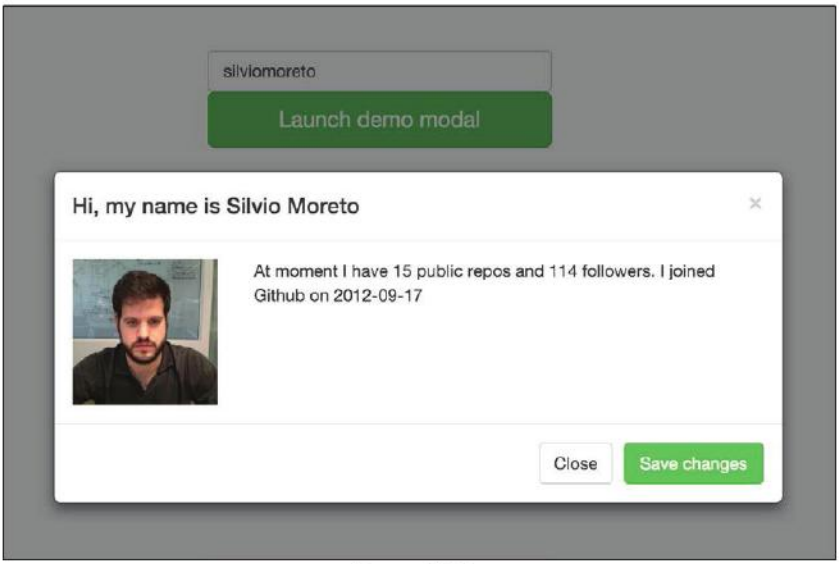


Рис. 11.6.

Итак, мы увидели пример активного взаимодействия с плагинами Bootstrap, заключающийся в их подстройке под собственные нужды.

Имейте в виду, что в Bootstrap реализованы события для всех плагинов. Их удобно использовать для взаимодействий с плагинами, что и было продемонстрировано на примере выполнения некоторых операций перед отображением модальной панели.

Дополнительные плагины Bootstrap

В Bootstrap имеются плагины практически на все случаи жизни. Однако порой нужные компоненты или плагины отсутствуют, например, предназначенные для выбора даты или цвета. Такие плагины не были включены в состав фреймворка Bootstrap, потому что они нужны далеко не во всех приложениях и при необходимости их можно подключать отдельно.

Список дополнительных ресурсов для Bootstrap можно найти по адресу: http://expo.getbootstrap.com/resources/.

Создание плагина Bootstrap

В предыдущей главе мы обсуждали плагин карусели. Помните необходимую ему HTML-разметку? Она была очень громоздкой, в чем можно убедиться, взглянув на следующий листинг:

```
<div id="carousel-notification" class="carousel slide"
data-ride="carousel">

  <!-- Обертка слайдов -->
  <div class="carousel-inner" role="listbox">
    <div class="item active">
      <img src="imgs/doge.jpg" width="512">
      <div class="carousel-caption">
        <h3>Doge said:</h3>
        <p>What are you doing? So scare. It's alright now.</p>
      </div>
    </div>
    <div class="item">
      <img src="imgs/cat.jpg" width="512">
      <div class="carousel-caption">
        <h3>Crazy cat said:</h3>
        <p>I will never forgive you...</p>
      </div>
    </div>
    <div class="item">
      <img src="imgs/laika.jpg" width="512">
      <div class="carousel-caption">
        <h3>Laika said:</h3>
```

```
            <p>Hey! How are you?</p>
          </div>
        </div>
      </div>

      <!-- Индикаторы -->
      <ol class="carousel-indicators">
        <li data-target="#carousel-notification" data-slide-
to="0" class="active"></li>
        <li data-target="#carousel-notification" data-slide-
to="1"></li>
        <li data-target="#carousel-notification" data-slide-
to="2"></li>
      </ol>

      <!-- Элементы управления -->
      <a class="left carousel-control" href="#carousel-
notification" role="button" data-slide="prev">
        <span class="glyphicon glyphicon-chevron-left" aria-
hidden="true"></span>
        <span class="sr-only">Previous</span>
      </a>
      <a class="right carousel-control" href="#carousel-
notification" role="button" data-slide="next">
      <span class="glyphicon glyphicon-chevron-right" aria-
hidden="true"></span>
        <span class="sr-only">Next</span>
      </a>
    </div>
```

Есть важная причина, объясняющая, зачем нужны все эти строки кода. С их помощью можно настроить плагин, приспособив его под собственные нужды. Однако хорошо было бы иметь простую карусель с меньшим объемом необходимой разметки. Можно ли реализовать такое?

Шаблон для нового плагина, который мы попытаемся создать, будет содержать только разметку, обеспечивающую все то же, что и приведенный выше код:

```
<div id="carousel-notification" class="bootstrap-carousel">
  <img src="imgs/doge.jpg" data-title="doge" data-content="Hey
there!">
  <img src="imgs/laika.jpg" data-title="laika" data-content="
```

```
Hey...!">
  <img src="imgs/cat.jpg" data-title="cat">
</div>
```

В плагине будет только один элемент `div`, обертывающий все остальное. Внутри него будет находиться последовательность элементов `img`, определяющих изображения, заголовки (через атрибут `data-title`) и описание слайда (через атрибут `data-content`).

Создание плагина с нуля является достаточно сложной задачей, но мы уже овладели основами Bootstrap и усовершенствуем свои навыки при работе над плагином.

Создание основы плагина

Прежде всего создадим необходимые папки и файлы. Для HTML-разметки создадим новый файл с тем же базовым шаблоном, который использовался в других примерах.

В папке `imgs` будут располагаться изображения питомцев из предыдущей главы. В этой главе не будут использоваться CSS-правила, поэтому о них можно не волноваться.

Создадим файл `bootstrap-carousel.js` в папке `js` и импортируем его в HTML-файле, поместив ниже загрузки файла `bootstrap.js` (в конце страницы):

```
<script src="js/jquery-1.11.3.js"></script>
<script src="js/bootstrap.js"></script>
<script src="js/bootstrap-carousel.js"></script>
```

Создадим фундамент плагина, добавив в файл `bootstrap-carousel.js` следующие строки:

```
+function ($) {
  'use strict';

  // ОПРЕДЕЛЕНИЕ КЛАССА КАРУСЕЛИ BOOTSTRAP
  // =====================
  var BootstrapCarousel = function (element, options) {
    this.$element = $(element);
    this.options = $.extend({}, BootstrapCarousel.DEFAULTS,
options);
  }

  BootstrapCarousel.VERSION = '1.0.0'
  BootstrapCarousel.DEFAULTS = {
```

```
    };

    BootstrapCarousel.prototype = {
    };

}(jQuery);
```

Это новая функция для jQuery. Она сначала определяет класс `BootstrapCarousel`, который и будет играть роль плагина. Функции передается элемент, к которому будет применена карусель, и параметры, которые могут быть получены либо через атрибуты данных, либо путем инициализации с помощью JavaScript.

Почему функции предшествует плюс?

Плюс (`+`) указывает, что функцию надо рассматривать как выражение. Благодаря этому любая функция, которой предшествует знак плюс, будет вызвана немедленно. Вместо этого символа, можно, с тем же эффектом, использовать любой другой унарный оператор (такой, как `!`, `~` или `()`).

Без начального символа оператор `function` будет воспринят как объявление функции, а не выражение, что, скорее всего, приведет к синтаксической ошибке.

Затем, переменная `options` дополняется параметрами `BootstrapCarousel.DEFAULT`. Таким образом, в отсутствие параметра с настройками будут использоваться настройки по умолчанию.

Далее, определяется версия (`VERSION`) плагина, значения настроек по умолчанию (`DEFAULT`) и прототип `prototype`, содержащий все свойства и методы класса. В `prototype` будут созданы методы и классы плагина с основной логикой его работы.

Прежде чем переходить к логике карусели, нужно сначала закончить с ее инициализацией. Добавим ее после создания прототипа `prototype`:

```
+function ($) {
  'use strict';

  // ОПРЕДЕЛЕНИЕ КЛАССА КАРУСЕЛИ BOOTSTRAP
  // =====================
  var BootstrapCarousel = function (element, options) {
    this.$element = $(element);
```

```
      this.options = $.extend({}, BootstrapCarousel.DEFAULTS,
options);
   }
   BootstrapCarousel.VERSION = '1.0.0'
   BootstrapCarousel.DEFAULTS= {
   };

   BootstrapCarousel.prototype = {
   };

   // ОПРЕДЕЛЕНИЕ ПОДКЛЮЧАЕМОГО МОДУЛЯ КАРУСЕЛИ BOOTSTRAP
   // ======================
   function Plugin(option) {

      var args = arguments;
      [].shift.apply(args);

      return this.each(function () { var $this = $(this),
        data = $this.data('bootstrap-carousel'),
        options = $.extend({}, BootstrapCarousel.DEFAULTS, $this.
data(), typeof option == 'object' && option),
        value;

        if (!data) {
          $this.data('bootstrap-carousel', (data = new
BootstrapCarousel(this, options)));
        }

        if (typeof option == 'string') {
          if (data[option] instanceof Function) {
            value = data[option].apply(data, args);
          } else {
            value = data.options[option];
          }
        }
      })
   }

}(jQuery);
```

Класс `Plugin` получает при вызове параметр `option` и `arguments` для элемента и вызывает его. Не стоит беспокоиться об этой части кода. Это распространенный способ инициализации плагинов, который применяется практически повсеместно.

Для завершения инициализации плагина добавьте код, выделенный жирным, после класса `Plugin`:

```
+function ($) {
  'use strict';

  // ОПРЕДЕЛЕНИЕ КЛАССА КАРУСЕЛИ BOOTSTRAP
  // =====================
    var BootstrapCarousel= function (element, options) {
      this.$element = $(element);
      this.options = $.extend({}, BootstrapCarousel.DEFAULTS,
options);
  }

  BootstrapCarousel.VERSION = '1.0.0'
  BootstrapCarousel.DEFAULTS = {
  };

  BootstrapCarousel.prototype = {
  };

  // ОПРЕДЕЛЕНИЕ ПОДКЛЮЧАЕМОГО МОДУЛЯ КАРУСЕЛИ BOOTSTRAP
  // =====================
  function Plugin(option) {
    ... // the plugin definition
  }

  var old = $.fn.bCarousel;
  $.fn.bCarousel = Plugin;
  $.fn.bCarousel.Constructor = BootstrapCarousel;

  // РАЗРЕШЕНИЕ КОНФЛИКТОВ КАРУСЕЛИ BOOTSTRAP
  // ================
  $.fn.bCarousel.noConflict = function () {
  $.fn.bCarousel = old; return this;
  }

  // КЛАСС ЗАГРУЗКИ КАРУСЕЛИ BOOTSTRAP
  // ==============
  $(window).on('load', function () {
    $('.bootstrap-carousel').each(function () {
      var $carousel = $(this);
      Plugin.call($carousel, $carousel.data());
    })
  })

}(jQuery);
```

Здесь сначала выполняется привязка плагина с помощью JQuery в строке `$.fn.bCarousel = Plugin;`. Затем определяется конструктор для инициализации класса: `$.fn.bCarousel.Constructor = BootstrapCarousel;`. Плагину присвоено имя `bCarousel`, поэтому его можно вызвать из JavaScript как:

```
$('some element selected').bCarousel();
```

Затем, плагин снова добавляется, на случай вероятного конфликта из-за присутствия еще одного плагина с таким же именем.

В последней части кода плагин инициализируется с помощью класса данных. Таким образом, для каждого элемента с классом `.bootstrap-carousel` плагин будет автоматически инициализироваться соответствующим набором атрибутов данных.

Определение методов плагина

Теперь, после объявления плагина, нужно создать его логику. Мы будем создавать методы в прототипе `prototype`. Далее будет показана только часть кода плагина, имеющая отношение к его работе.

Первый метод – метод `init()`. Он будет вызываться при запуске плагина и должен:

- выполнить начальную проверку;
- настроить элементы плагина и проверить предварительные условия;
- загрузить оригинальный шаблон Bootstrap;
- запустить плагин Bootstrap.

Инициализация и проверка подключаемого модуля

Фактически, оригинальный плагин Bootstrap имеет всего одно требование: главный элемент `div` должен иметь атрибут `id`. Создадим функцию `init` для проверки этого условия:

```
BootstrapCarousel.prototype = {
  init: function () {
    if(!this.$element.attr('id')){
      throw 'You must provide an id for the Bootstrap
Carousel element.';
    }

    this.$element.addClass('slide carousel');
```

```
    }
};
```

Здесь с помощью метода `this.$element.attr('id')` проверяется наличие у элемента атрибута `id`. Если его нет, в консоль выводится сообщение об ошибке и разработчик сможет исправить эту ошибку. Обратите внимание, что доступ к элементу плагина через `this.$element` возможен благодаря присваиванию плагина в начале.

В последней строке выполняется добавление необходимых классов, таких как `.slide` и `.carousel`, если они отсутствуют в `$element`.

Добавление шаблона Bootstrap

Для загрузки шаблона карусели Bootstrap создадим еще одну функцию `load`. Добавим ее вызов в метод `init`:

```
BootstrapCarousel.prototype = {
  init: function () {
    if(!this.$element.attr('id')){
      throw 'You must provide an id for the Bootstrap
Carousel element.';
    }

    this.$slides = this.$element.find('> img');
    this.$element.addClass('slide carousel');
    this.load();
  }

  load: function() {
  },
};
```

Прежде всего, необходимо удалить элементы карусели, которые находятся внутри элемента `$element`. К ним относятся элементы с классами `.carousel-inner`, `.carousel-indicators` и `.carousel-control`. Необходимо, также, загрузить изображения слайдов в переменную `this.$slides` и сделать их невидимыми:

```
load: function() {
  // удаление элементов карусели
  this.$element.find('.carousel-inner, .carousel-indicators,
.carousel-control').remove();

  // загрузка и скрытие изображений
  this.$slides = this.$element.find('> img');
```

```
    this.$slides.hide();
},
```

Далее, следует убедиться, что к данному элементу не подключены другие плагины карусели Bootstrap:

```
this.$element.carousel('pause');
this.$element.removeData('bs.carousel');
```

Сначала выполняется приостановка карусели, чтобы остановить любые взаимодействия с элементом, а затем вызывается функция `removeData` с аргументом `bs.carousel`, хранящим имя плагина карусели.

Далее, необходимо загрузить шаблон карусели Bootstrap. Для этого внутри прототипа класса необходимо создать переменную, которая будет хранить оригинальный шаблон. Переменная имеет следующий формат:

```
template: {
  slide: '...',
  carouselInner: '...',
  carouselItem: '...',
  carouselIndicator: '...',
  carouselIndicatorItem: '...',
  carouselControls: '...',
},
```

Здесь будет показан не весь код шаблонов, потому что он довольно обширный и желающие смогут найти его в прилагаемых к книге файлах. Тем более что ничего загадочного в шаблонах нет, это просто длинные строки с маркированными фрагментами, подлежащими замене. Маркированные фрагменты заключены в фигурные скобки, например: `{keyName}`. При создании шаблона необходимо заменить эти части строк вызовом метода `.replace(/{keyName}/, 'value')`.

Каждый ключ внутри шаблона соответствуют определенной части шаблона. Поясним каждый из них:

- `slide`: шаблон слайдов нового плагина, используется для добавления слайдов из JavaScript;
- `carouselInner`: внутренний элемент карусели, родительский элемент пунктов;
- `carouselItem`: элемент, содержащий изображение и заголовок слайда;
- `carouselIndicator`: набор маркеров в нижней части карусели;

- carouselIndicatorItem: представляет круглый маркер индикатора;
- carouselControls: элементы управления перемещением слайдов карусели вправо и влево.

В конец метода load добавим еще две строки:

```
load: function() {
  this.$element.find('.carousel-inner, .carousel-indicators,
.carousel-control').remove();
  this.$slides = this.$element.find('> img');
  this.$slides.hide();
  this.$element.carousel('pause');
  this.$element.removeData('bs.carousel');

  this.$element.append(this.createCarousel());
  this.initPlugin();
},
```

Эти строки добавляют в конец элемента this.$element шаблон, сгенерированный функцией createCarousel. После этого, выполняется инициализация оригинального плагина карусели Bootstrap.

Создание оригинального шаблона

Оригинальный шаблон создается в функции createCarousel в два этапа:

- создается колода слайдов для элемента .carousel-inner;
- создаются индикаторы и элементы управления, если необходимо.

То есть, метод createCarousel вызывает следующие три функции, добавляющие в переменную строки шаблона:

```
createCarousel: function() {
  var template = '';

  // создание слайдов
  template += this.createSlideDeck();

  // создание индикаторов
  if(this.options.indicators) {
    template += this.createIndicators();
  }

  // создание элементов управления
```

```
    if(this.options.controls) {
      template += this.createControls();
    }

    return template
  },
```

Обратите внимание, что индикаторы и элементы управления создаются после проверки их необходимости. Проверка выполняется с помощью переменной `this.options`, которая определяет, указал ли разработчик необходимость создания этих компонентов.

Итак, мы определили первые две переменные нашего плагина. Они могут передаваться через атрибуты данных `data-indicators` и `data-controls` элемента и определяют необходимость добавления соответствующих элементов.

Колода слайдов

Колода слайдов создается путем обхода всех элементов `this.$slide` и загрузки изображений, `data-title` и `data-content`. Кроме того, в первый элемент добавляется класс `.active`:

```
createSlideDeck: function() {
  var slideTemplate = '',
      slide;

  for (var i = 0; i < this.$slides.length; i++) {
    slide = this.$slides.get(i);

    slideTemplate += this.createSlide(
      i == 0 ? 'active' : '',
      slide.src,
      slide.dataset.title,
      slide.dataset.content
    );
  };

  returnthis.template.carouselInner.replace(/{innerContent}/,
slideTemplate);
},
```

В каждой итерации вызывается еще одна функция, `createSlide`, которой передается признак активности слайда, источник изображения, элемент заголовка и элемент контента. Затем эта функция изменяет шаблон в соответствии с аргументами:

```
createSlide: function(active, itemImg, itemTitle, itemContent) {
  return this.template.carouselItem
    .replace(/{activeClass}/, active)
    .replace(/{itemImg}/, itemImg)
    .replace(/{itemTitle}/, itemTitle ||
this.options.defaultTitle)
    .replace(/{itemContent}/, itemContent ||
this.options.defaultContent);
}
```

Функция поверяет наличие заголовка и контента. Если заголовок и/или контент отсутствуют, они замещаются значениями по умолчанию из `this.options`. Подобно индикаторам и элементам управления, эти параметры могут передаваться через атрибуты данных `data-default-title` и `data-default-content`.

Эти параметры могут также передаваться в момент инициализации плагина из JavaScript-кода, вызовом метода `.bCarousel({ defaultTitle: 'default title' })`.

Индикаторы карусели

Функция `createIndicators` используется для создания индикаторов. В ней применяется тот же метод, что и при создании колоды слайдов. Сначала она создает каждый маркер и заключает его в список `.carousel-indicators`:

```
createIndicators: function() {
  var indicatorTemplate = '',
      slide,
      elementId = this.$element.attr('id');

  for (var i = 0; i < this.$slides.length; i++) {
    slide = this.$slides.get(i);

    indicatorTemplate += this.template.carouselIndicatorItem
      .replace(/{elementId}/, elementId)
      .replace(/{slideNumber}/, i)
      .replace(/{activeClass}/, i == 0 ? 'class="active"' : '');
  }

  return this.template.carouselIndicator.replace(/
{indicators}/, indicatorTemplate);
},
```

Вся хитрость заключается в том, чтобы пронумеровать маркеры и добавить в них ссылки на идентификаторы `id` родительских элементов. Поэтому мы выполняем необходимые замены для каждого `this.$slides` и возвращаем шаблон индикатора.

Почему заменяемый ключ и окружающие его фигурные скобки заключены в прямые слеши?

Так в JavaScript выполняется поиск по регулярному выражению в предоставленном образце текста. Этот прием может пригодиться для реализации собственной замены и для поиска отдельных объектов.

Элементы управления карусели

Элементы управления создают стрелки для перемещения слайдов влево и вправо. Они конструируются тем же способом, что и другие шаблоны: извлекается шаблон и производится замена ключей:

```
createControls: function() {
  var elementId = this.$element.attr('id');

  return this.template.carouselControls
    .replace(/{elementId}/g, elementId)
    .replace(/{previousIcon}/, this.options.previousIcon)
    .replace(/{previousText}/, this.options.previousText)
    .replace(/{nextIcon}/, this.options.nextIcon)
    .replace(/{nextText}/, this.options.nextText);
},
```

Обратите внимание, что в первой операции замены `{elementId}` к регулярному выражению добавлен атрибут `g` – он требует заменить все совпадения с моделью. В отсутствие атрибута `g` JavaScript заменит только первое совпадение. В этом шаблоне имеются два ключа `{elementId}` и они оба должны быть заменены.

Также имеются параметры, передаваемые при инициализации подключаемого модуля для значков перехода к предыдущему и следующему слайду и соответствующих им текстов.

Инициализация оригинального плагина

После создания оригинального шаблона, нужно запустить оригинальный плагин карусели. Определим для этого функцию `initPlugin`:

```
initPlugin: function() {
  this.$element.carousel({
    interval: this.options.interval,
    pause: this.options.pause,
    wrap: this.options.wrap,
    keyboyard: this.options.keyboard
  });
},
```

Она просто запускает плагин, вызывая `this.$element.carousel` с параметрами карусели. Параметры загружаются в точности, как описывалось выше – в определении класса плагина, в следующей строке:

```
this.options = $.extend({}, BootstrapCarousel.DEFAULTS, options);
```

Если какой-то параметр был передан, он переопределят параметр по умолчанию в `BootstrapCarousel.DEFAULTS`. Параметры по умолчанию создаются, как показано ниже:

```
BootstrapCarousel.DEFAULTS = {
  indicators: true,
  controls: true,
  defaultTitle: '',
  defaultContent: '',
  nextIcon: 'glyphicon glyphicon-chevron-right',
  nextText: 'Next',
  previousIcon: 'glyphicon glyphicon-chevron-left',
  previousText: 'Previous',
  interval: 5000,
  pause: 'hover',
  wrap: true,
  keyboard: true,
};
```

Запуск плагина в работу

До загрузки плагина остается выполнить лишь одно действие. Создадим следующий HTML-код:

```
<div id="carousel-notification" class="bootstrap-carousel"
data-indicators="true" data-controls="true">
  <img src="imgs/doge.jpg" data-title="doge" data-content=
"Hey there!">
  <img src="imgs/laika.jpg" data-title="laika" data-content=
```

```
"Hey...!">
  <img src="imgs/cat.jpg" data-title="cat">
</div>
```

В JavaScript-коде нашего плагина нужно запустить прототип, вызвав функцию `init`:

```
var BootstrapCarousel = function (element, options) {
  this.$element = $(element);
  this.options = $.extend({}, BootstrapCarousel.DEFAULTS,
options);

  this.init();
}
```

Ура! Откройте HTML-файл в браузере и вы увидите подключаемый модуль в действии, как показано на рис. 11.7. Заглянув в модель DOM, вы сможете оценить, насколько хорошо сымитирован подключаемый модуль карусели Bootstrap, уменьшивший объем необходимого кода почти на 35 строк:

Рис. 11.7.

Добавление методов в плагин

Разработка плагина практически завершена. Теперь добавим в плагин несколько методов, чтобы их можно было вызывать подобно методу `.carousel('pause')` карусели Bootstrap, например.

Создавая основу плагина, мы создали класс `Plugin`, который определяет плагин. Эта часть кода часто используется в плагинах и она включена во все встроенные плагины Bootstrap:

```
function Plugin(option) {

  var args = arguments;
  [].shift.apply(args);

  return this.each(function () {
    var $this = $(this),
        data = $this.data('bootstrap-carousel'),
        options = $.extend({}, BootstrapCarousel.DEFAULTS,
$this.data(), typeof option == 'object' && option),
        value;

    if (!data) {
      $this.data('bootstrap-carousel', (data = new
BootstrapCarousel(this, options)));
    }

    if (typeof option == 'string') {
      if (data[option] instanceof Function) {
        value = data[option].apply(data, args);
      } else {
        value = data.options[option];
      }
    }
  })
}
```

В строках, выделенных жирным, проверяется тип переданной переменной `option`. Если она содержит строку, вызывается функция `apply`, которая вызовет функцию с именем в переменной `option`.

После этого, нужно экспортировать функцию определения класса `BootstrapCarousel`. Для этого добавим два поля, одно для хранения ссылки на функцию перезагрузки плагина, а другое – на функцию добавления слайда в карусель:

```
var BootstrapCarousel = function (element, options) {
  this.$element = $(element);
  this.options = $.extend({}, BootstrapCarousel.DEFAULTS,
options);

  // Экспортирование общедоступных методов
  this.addSlide = BootstrapCarousel.prototype.addSlide;
  this.reload = BootstrapCarousel.prototype.load;

  this.init();
}
```

Строки, выделенные жирным, осуществляют экспортирование методов. Теперь их нужно реализовать в прототипе.

Поскольку один из методов `BootstrapCarousel.prototype.load` уже реализован, при его экспортировании он переименовывается с `load` на `reload`. При вызове этого метода будут удалены все оригинальные плагины карусели Bootstrap, заново создан шаблон на основе изображений, переданных через наш плагин, и заново сгенерирован оригинальный плагин.

Реализуем метод добавления слайда `BootstrapCarousel.prototype.addSlide`. В прототипе `Bootstrap.prototype` создадим следующую функцию:

```
addSlide: function(itemImg, itemTitle, itemContent) {
  var newSlide = this.template.slide
     .replace(/{itemImg}/, itemImg)
     .replace(/{itemTitle}/, itemTitle)
     .replace(/{itemContent}/, itemContent);
   this.$element.append(newSlide); this.load();
},
```

В эту функцию нужно передать источник изображения `itemImg`, текст заголовка слайда `itemTitle` и абзац для подписи `itemContent`.

Чтобы создать новый слайд, прежде всего нужно использовать шаблон нового слайда, который можно найти в переменной `this.template.slide`:

```
template: {
  slide: '<img class="hide" src="{itemImg}" data-
title="{itemTitle}" data-content="{itemContent}">',
... // другие переменные шаблона
}
```

Так же как при создании колоды слайдов, индикаторов и элементов управления, в шаблоне определяются ключи в фигурных скобках, которые заменяются в функции.

После замены, новый слайд добавляется в `this.$element`, который уже содержит другие слайды. И, наконец, нужно вызвать функцию загрузки, которая выполнит самую тяжелую работу по назначению переменных, сокрытию элементов и запуску оригинального плагина.

Теперь, чтобы добавить слайд достаточно выполнить следующий вызов:

```
$('.bootstrap-carousel').bCarousel('addSlide', 'imgs/jon.png',
'New title image', 'This is awesome!');
```

С этой функцией мы завершили работу! Как видите, писать плагины не так уж сложно. Теперь, можно заняться расширением модуля, сделать его более удобным и добавить параметры для улучшения настройки.

Итоги

Мне кажется, последняя глава получилась внушительной! Она была посвящена настройке Bootstrap, как с точки зрения стилизации компонентов, так и со стороны организации взаимодействия с плагинами. Bootstrap – великий фреймворк, но великим его делает, прежде всего, возможность расширения. Он представляет собой идеальный мир, где предварительно подготовленные компоненты и средства их настройки существуют в симбиозе.

Достойным завершением книги стала разработка плагина для фреймворка Bootstrap, обертывающего плагин карусели Bootstrap. Наш плагин обеспечивает практически все возможности оригинального плагина Bootstrap и очень удобен при создании простой карусели с минимальным объемом кода.

Этот плагин можно найти по адресу: github.com/silviomoreto/bootstrap-carousel. Найдите его и загрузите! Его можно значительно улучшить и расширить. Например, можно добавить метод для удаления слайдов.

Кроме того, цель описания процесса создания плагина заключалась в том, чтобы научить вас создавать новые плагины для Bootstrap и освоить их настолько, чтобы уметь при необходимости их настраивать. Я полагаю, что теперь вы легко разберетесь в коде плагинов и сможете улучшить их.

Хотел бы поблагодарить вас за то, что дочитали эту книгу до конца. Освоение продвинутого фреймворка, подобного Bootstrap, далеко непростая задача, и она по плечу не всем разработчикам. Гордитесь своими достижениями.

Освоение плагина надо начать с базового оснащения и двигаться дальше к созданию его компонентов, элементов и так далее. Это было продемонстрировано на полезных примерах, которые вы сможете применить в собственных разработках.

Изюминкой книги стало знакомство с версией 4 фреймворка Bootstrap, которая появился совсем недавно. Это означает, что вы вошли в число немногих разработчиков, готовых к использованию новой версии фреймворка Bootstrap.

Я надеюсь, вам понравилось путешествие по миру Bootstrap и вы узнали много нового, прочтя эту книгу. Теперь ваша очередь! Вы должны справиться с любой задачей, с который вы столкнетесь, используя клиентский фреймворк Bootstrap. Я полагаю, что знания, почерпнутые из приведенных в книге примеров, позволят вам стать настоящим мастером Bootstrap.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

И

К

М

Н

О

П

Р

С

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «Планета Альянс» наложенным платежом, выслав открытку или письмо по почтовому адресу: **115487, г. Москва, 2-й Нагатинский пр-д, д. 6А**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в интернет-магазине: **www.alians-kniga.ru**.

Оптовые закупки: **тел. +7 (499) 782-38-89**.

Электронный адрес: **books@alians-kniga.ru**.

Сильвио Морето

Bootstrap в примерах

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Перевод с английского *Рагимов Р. Н.*
Научный редактор *Киселев А. Н.*
Корректор *Синяева Г. И.*
Верстка *Паранская Н. В.*
Дизайн обложки *Мовчан А. Г.*

Формат 60×90 1/16. Гарнитура «Петербург».
Печать офсетная. Усл. печ. л. 18,25.
Тираж 200 экз.

Веб-сайт издательства: www.дмк.рф