createspace™
an Amazon.com company

Digital Proofer

How to Build Unique ...

Authored by Michael B Musgrove

6.0" x 9.0" (15.24 x 22.86 cm)
Color on White paper
172 pages

ISBN-13: 9781482367263
ISBN-10: 1482367262

Please carefully review your Digital Proof download for formatting, grammar, and design issues that may need to be corrected.

We recommend that you review your book three times, with each time focusing on a different aspect.

1. Check the format, including headers, footers, page numbers, spacing, table of contents, and index.
2. Review any images or graphics and captions if applicable.
3. Read the book for grammatical errors and typos.

Once you are satisfied with your review, you can approve your proof and move forward to the next step in the publishing process.

To print this proof we recommend that you scale the PDF to fit the size of your printer paper.

HOW TO BUILD UNIQUE TWITTER BOOTSTRAP THEMES QUICKLY AND EASILY

BY

MICHAEL B. MUSGROVE

LEARN ABOUT UPCOMING TITLES

BY MICHAEL B. MUSGROVE AT

FreeBootstrap.com

TABLE OF CONTENTS

INTRODUCTION

What is Twitter Bootstrap?

Let's quickly run through some basics first. Since you're reading this book, I'm assuming you have heard of Bootstrap. But what is Bootstrap, exactly?

Bootstrap is a free web development tool originating from Twitter that, with a little HTML, CSS and JavaScript experience, makes building websites quick, intuitive, and fun. It's a customizable and lightweight framework that allows you to create and deploy feature-rich websites by easily accessing built-in goodies in the form of CSS3, JavaScript, HTML5 and UI elements. It's built using what's commonly referred to as a "grid system," and specifically with Twitter Bootstrap, "scaffolding" and makes building attractive, responsive websites a breeze.

There are quite a few good reasons to use Bootstrap:

1. Time

The Bootstrap libraries offer existing snippets of code that immediately inject life into a website. Web designers and developers don't have to spend time working and writing out code, and can spend more time creating and building. Bootstrap allows you to grab the tool you need for your workspace quickly and easily. In addition, much of the styling and design aspects are already handled, since the CSS is built with LESS. Redundancy is minimized.

2. Totally customizable

No one wants to use a framework that, after much planning, designing and work, still gives you the same-looking website as a zillion others out there. You can make each Bootstrap site your own. Bootstrap also has a web app that allows you to only use the components you want, decreasing page load time and file size even more. You can override the core CSS easily and really have something unique in no time.

3. Design considerations

The Grid:

If you aren't familiar with working with grid layouts, fear not. They make your job a lot easier. Bootstrap's default is a platform with a 12 column grid which is 940px wide. With the responsive CSS file added, the grid adapts to be 724px and 1170px wide, depending on your viewport. Below 767px viewports, the columns become fluid and stack vertically. Here are the different column and gutter widths, depending:

Label	Layout width	Column width	Gutter width
Large display	1200px and up	70px	30px
Default	980px and up	60px	20px
Portrait tablets	768px and above	42px	20px
Phones to tablets	767px and below	Fluid columns, no fixed widths	
Phones	480px and below	Fluid columns, no fixed widths	

With a structure in place, all you could do is drop content into the HTML and go. Also, if you need to work through columns and nesting, then you're set with Bootstrap.

CSS & LESS:

LESS is becoming more and more popular with developers, and for good reason. With its additions to the Bootstrap platform, you can use LESS mixins and CSS manipulations to customize the built-in grid. Changes are automatically updated after each variable shift which is a nice touch.

JavaScript:

Bootstrap comes equipped with JavaScript libraries that are far beyond just basic structure and styling. JavaScript often needs to be incorporated heavily in some designs, and developers have to pull elements in from all over the place. With Bootstrap, you can easily manipulate modal windows alerts, tooltips, Scrollspy, Popover, Button, Typeahead, etc…, and again, only pull the JavaScript tools from the shelf that you want with Bootstrap's customization app. The best part, however, is that Bootstrap enables you to skip writing the <script> altogether.

4. Responsiveness:

What initially lures designers and developers to Bootstrap is that it's responsive. Responsive layouts aren't going anywhere and, in fact are becoming almost expected. More and more devices and appliances are entering the market, all with different sized viewports and browsers. That's not an issue with Bootstrap. If you shift from a desktop to an iPad and from an iPad to an iPhone, you

won't have to worry. Bootstrap adapts to the change in platforms seamlessly.

5. Consistency:

The main reason Bootstrap was developed was that developers at Twitter were seeing major inconsistencies among their internal developers that were working on various projects. This caused some issues both on the development side and the end-user side. Since Bootstrap works with a standardized core code, Bootstrap's results are uniform across platforms. You'll see the same thing on Internet Explorer, Opera, Safari, Chrome and Firefox.

6. Integration:

If you're working on a site which is live already, but need to iron out a few creases, then Bootstrap can help. For instance, if you use table styling, all you need to do is take the styles you need and copy them to the CSS file you're working with. Bootstrap will immediately kick in with its own style, and then all you do is link up the file you are working with or hotlink to Twitter (more or less). Integration is simple, fast and easy to accomplish.

7. Future Compatibility:

Bootstrap is forward-thinking and is equipped with many elements that are the future of design and development. For instance both HTML5 and CSS3 are the way the wind is blowing and Bootstrap relies on both. Since the framework takes into account the future of design and development, it's likely to be around for some time.

8. Competitiveness:

Bootstrap isn't the only framework out there. It has been pitted against JQuery UI and the HTML5 Boilerplate, but a more realistic competitor is Zurb Foundation. Bootstrap 2 has a toolset that Foundation will have difficulty beating. Third party plugins, themes, a plethora of features, codes, etc. are already available for Bootstrap, which aren't entirely available for Foundation just yet. Some might feel that Bootstrap has a comparatively bloated feel (which Foundation doesn't) but that can be easily done away with through the custom Bootstrap web app, which is simple to use, and only adds exactly what you need and nothing more. A main concern of Bootstrap's development is not only to keep it light, but continue to make it ultra-light. This is a focus of every release.

9. Documentation:

Bootstrap has quite a bit of high-quality documentation available, although it's relatively new. A catch-22 loop many new open-source platforms face is that they don't have adequate documentation. And without documentation, people won't use it. And if people don't use it, it doesn't (typically) get the attention it needs to compete as well as proper documentation. Even some theme shops that have been around for years, like some for WordPress, for example, can't offer the same level of available support. This book is a big part of that. And that's because in large part, Bootstrap has an active and thriving community that has sprung up around it.

10. The Bootstrap Community:

There's a large, and growing, fan base of Twitter Bootstrap that knows a lot about everything that goes into making Bootstrap better, from the UI to the UX to making admin panels and themes to testing beta versions. The more talented people that are getting on board, the better. And the better Bootstrap becomes, the more people want to be involved, and the more talented people that want to get in means Bootstrap becomes even better. See the circle?

11. Updates:

Bootstrap has taken on a life of its own, and as a living "project" it's updated frequently. The developers are committed to Bootstrap, and there's a high degree of quality assurance and tendency to always improve. As soon as web developers find a problem, the Bootstrap team starts looking to fix it.

WHAT WILL YOU LEARN?

In this book you'll learn how to download the Bootstrap framework and integrate CSS and JavaScript into your website. We'll cover Bootstrap's responsive 12-column grid system, which not only offers tons of flexibility -- no pun intended -- but also helps assemble highly workable web page layouts quickly.

We'll also look at the included CSS styling that comes with Bootstrap, including the correct (and easy) way to override and fine-tune those styles to your, or your client's, preferences.

We'll explore navigation systems that are built into Bootstrap, including tabs and pills, as well as pagination.

Topics you'll learn include:

- Understanding the difference between default and fluid grids
- Nesting using fluid grids
- Creating a thumbnail gallery
- Adding block quotes and lists of text
- Incorporating images and icons
- Adding breadcrumb navigation and pagination
- Custom CSS
- Building a thumbnail gallery
- Using tabs and pills navigation
- Making the navigation bar responsive with JavaScript
- Adding dropdown menus to the nav bar, tabs, and pill boxes
- Creating modal windows
- Building a photo carousel

And much more!

A few important notes:

If you're completely new to programming, check out some resources to learn how CSS and HTML work together and review the terminologies. You should feel comfortable poking around HTML, CSS and to a lesser degree, JavaScript. Although Bootstrap certainly doesn't require you to be a coding machine and able to code in your sleep whatsoever, a casual familiarity with them will make your experience with Bootstrap, and this book, much more fun and productive and take you further, faster. There are many good resources online to polish up your coding knowledge and skills, and I've included some I've found to be more helpful in the "Resources" section. If you've just

toyed with coding, going through this book will help you pick up basic coding quickly.

Web designers the world-over seem to wish for a template or application that removes all coding from the development process, however, it's my humble opinion that to build a world-class website, you'll need to know what's going on behind the scenes, and how to tweak things as necessary, which is going to require some coding knowledge. Also remember that Bootstrap is a *framework*, and meant to be built upon using additional resources for a truly customized website, whether those are code snippets you find from around the web, or your own code that you write yourself. Bootstrap's customization app helps with this even more, but you may eventually need to include additional styling and utilities that are beyond the customization app's capacities. Fortunately, it isn't hard and we'll learn how to do that here.

Hopefully you downloaded the sample files from FreeBootstrap.com when you purchased this book. They should be good jumping off points and are what I reference all along in this book. You certainly don't have to use them, but I included them in case you do, and they may make following along easier, as well as give you some ideas for your own projects. Some of the screenshots to illustrate what we're going over in this book must be shrunk to a point that they're difficult to read, so having the code on your screen will be beneficial. I've included those types of images purely for reference.

Speaking of code, you'll need to have a text editor installed on your computer, and know your way around it. This means a robust IDE (Integrated Development Environment) rather than something like the default Notepad that comes on Windows machines. There are quite a few text editors available online that are free (and paid), and

do a great job. Textmate 2, Notepad++, Sublime Text, Eclipse, Brackets and Aptana Studio 3 are several of the most popular. I've used all of those and also have Adobe Dreamweaver CS6, and I personally prefer Aptana Studio 3, at least for the moment, which is based on Eclipse. I find it to be full-featured but not bloated, customizable, extensible, fairly intuitive and Aptana works cross-platform. But any text editor you choose is fine. As the cliché goes: the best tool is the one you'll actually use.

If you don't have a favorite text editor yet, below is a list of quality text editors. Most are free, so try a few and see which you like best. A good text editor will improve your workflow and quickly become intuitive; a bad one can wreck your day.

In alphabetical order:

- BBEdit (Classic Mac OS, Mac OS X, $)
- Boxer Text Editor (Windows)
- Brackets by Adobe (Mac, Windows)
- Coda (Mac OS X, Shareware)
- Crimson Editor (Windows, Freeware)
- EditPad (cross-platform)
- EditPlus (Windows)
- Editra (cross-platform, Open Source, Free)
- emacs (Unices, Windows, Mac OS X, Open Source, Free)
- Fraise (Mac OS X, Open Source, Free, based on Smultron)
- gedit (Unices)
- JEdit (cross-platform)
- Kate (Unices)
- Komodo Edit (cross-platform, Open Source, Free)
- Kwrite (Unices)

- Notepad++ (Windows, Open Source, Free)
- Notepad2(Windows, Freeware)
- pico (Unices)
- PSPad (Windows, Free)
- Smultron (Mac OS X, Open Source, Free/$)
- Smultron 4 (req OS-X Lion) (v.cheap in MacApp store)
- SubEthaEdit (Mac OS X, $)
- Sublime Text 2 (Windows, Linux, Mac OS X)
- TextEdit (comes with Mac OS X)
- TextMate (Mac OS X, $)
- TextPad (Windows)
- TextWrangler (Mac OS X, Free)
- UltraEdit-32 vim
- (Unices, Windows, Mac OS X, Open Source, Free)
- WebTide Editor (Windows, Linux, Mac OS X, Freeware, Java)

Many of these are free, so you may want to experiment with several until you find one that best suits you and your workflow.

I would recommend taking your time and following along with this book a section at a time, to ensure you have a firm grasp on each concept we're reviewing. Quite a few of the features build upon one another as we progress through the book, so by the time you finish, you should have a good understanding of how Bootstrap operates.

You can find the most current version of Twitter Bootstrap, (version 2.3.0 as of this writing), at getbootstrap.com.

So, with all that out of the way, let's get to it!

CHAPTER ONE: SETTING UP YOUR SITE USING BOOTSTRAP

DOWNLOAD IT

First things first. Direct your browser to http://getbootstrap.com and download a copy of Bootstrap. You may notice that you also have the option in the top navigation to customize your downloaded files with some common options. This has several benefits: it not only saves you even more time setting up your site, but reduces your file size, which ultimately improves your page load speed, and increases the user experience (UX). This also reduces the likelihood of coding errors since you don't need to touch the code as much. For now, go ahead and just click the normal "Download Bootstrap" button and download it to a place that's convenient for you to work on, such as your desktop. Bootstrap comes as a .zip file, so you'll need to unzip it.

http://GetBootstrap.com

A few things to notice on the home page before we dive in, beginning with the fact that Bootstrap (as of this writing, v. 2.3.0) is hosted on GitHub, which is a software and project version control application that is used heavily by collaborators on open-source projects.

Since Bootstrap's introduction, it has been one of the most watched and forked projects on GitHub. This reflects that as more and more users discover how easy it is to build a site using Bootstrap, it has successfully begun to take on a life of its own. That's in large part to plenty of hard work by dedicated and knowledgeable developers, and a strong Bootstrap community. It is still managed by its original developers at Twitter, Mark Otto and Jacob Thornton. Also, Bootstrap is licensed under Apache License v 2.0. This may be about to change, and could soon be managed under an M.I.T. license, which is a little less restrictive. And last but not least, please note the very appealing facts that

Bootstrap offers a responsive layout, and also uses LESS, which is a CSS preprocessor that makes using CSS faster and easier.

LESS(LEss cSS) is a JavaScript library so it's processed client-side. This is information you'll want to know as you advance your customization skills with Bootstrap.

OK, enough of the Yappity-Yap! Let's get on with it!

Now that we've downloaded and unzipped our fresh copy of Bootstrap, let's get started building our first HTML web page inside of our text editor.

So crank up your text editor, and go to the folder with your unzipped copy of the Bootstrap files in it. The very first thing I'd recommend doing when building a new site is to rename your Bootstrap folder to something less generic, because you'll need to be able to tell your sites apart at high levels. You can name it whatever you'd like but make it descriptive of the site, if not simply the title of the website itself.

Next, in the folder that you unzipped your downloaded Bootstrap files to, add a new file to the folder, usually by right clicking within the open folder and choosing "New" and then "Text Document, "or alternately from the edit menu at the top of your editor. Inside of this folder should also be three folders that we just downloaded from getbootstrap.com.

The next thing is to create a new HTML web page. It's going to be an index with an HTML page type, as usual, with the file extension of .html or .htm (your choice, but be consistent in your programming). If your text editor asks for the doc type, make sure you choose "HTML 5," because that's what we'll be working with

for Bootstrap, which is specifically designed to work with HTML 5. You should now have a document very similar to this:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <!-- Always force latest IE rendering
engine (even in intranet) & Chrome Frame
        Remove this if you use the .htaccess-
><meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1" />
        <title>Twitter Bootstrap</title>
        <meta name="description" content="" />
        <meta name="author" content="Michael
Musgrove"/>
        <meta name="viewport"
content="width=device-width; initial-scale=1.0"
/>
   <body>
        <div>
            <header>
                <h1>about</h1>
            </header>
            <nav>
                <p>
                    <a href="/">Home</a>
                </p>
                <p>
                   <a href="/contact">Contact</a>
                </p>

   </nav>
   <div>
         </div>
   <footer>
    <p>&copy; Copyright by Michael B. Musgrove
    </p>
   </footer>
```



```
        </div>
    </body>
</html>
```

Of course, you'll want to modify the lines where necessary, such as the copyright, author and title.

Let's Make it Responsive!

Depending on your text editor and whether you used the default settings for HTML5 files, you may or may not have it initially optimized. We'll need to make sure we have a few items in our blank HTML file before we begin. If you don't see a line that begins with <meta name="viewport" as in the above example, then let's add it. That's simply a meta tag for our responsive design. This is a meta tag indicating that the viewport starts with an initial scale of 1.0, which is important when utilizing responsive design. If you don't have it, go ahead and copy the following line of code, just after "meta character set of UTF-8:"

```
<meta name="viewport"
content="width=devicewidth, initial-
scale=1.0">
```

For each of your pages, you'll want to change your title from "Untitled Document" to your site name or the title of that specific page, whatever that may be. This is very important for search engines, so be sure not to overlook that keyword opportunity. For our examples, however I'm just going to use something easy and relevant like "How to Use Bootstrap."

LINK THE STYLESHEETS

The next thing we need to do is attach the stylesheets that we'll be using in this particular document. Open your main Bootstrap CSS folder and inside of that you'll find that there are actually four stylesheets that are available to you, and you may be wondering what the differences are between these.

First of all, you'll see that there are actually two sets of two: "bootstrap" and "bootstrap.min," and then "bootstrap-responsive" and "bootstrapresponsive.min." The differences here are the .min files are what are called "minified" and they've simply been processed so that all the white space that's normally present in a stylesheet, between all the characters in the code, has been taken out. That will help the files download faster, which is great when you're actually using a production site, and you want to make sure your file sizes are small as possible so that mobile phone users have very small files to download, and desktop and laptop users can access your site in a flash.

If you're doing this in a development environment, you'll want to go ahead and link to the full version of these stylesheet files in case we ever want to have a look at them in the future, which is highly probable.

Of course, we want the more human readable format to work in, so link directly to just the regular Bootstrap file here first, after your title:

```
<link href="css/bootstrap.css" rel="stylesheet"
type="text/css">
<link href="css/bootstrap-responsive.css"
rel="stylesheet" type="text/css">
```

Right after that, we're going to attach the second stylesheet, and we want the bootstrap responsive stylesheet as above. *! important→ Make sure the bootstrap-responsive stylesheet comes **after** the bootstrap stylesheet, because in some cases, this bootstrap responsive stylesheet will override within the bootstrap stylesheet, so the order does indeed matter.

Now you should have two lines of code here, one calling for regular bootstrap.CSS, and one calling for the bootstrap responsive. Go ahead and save this file. Go to File > Save, and save it right into the root of your website here.

If you haven't renamed this file yet, go ahead and call this index.html, always a good named to call an HTML document, and we should have everything correctly linking to those two stylesheets.

ADDING AN HTML5 SHIM

Before we leave the <head> of this document, there's one last thing I'd like to add, and this will come just after the CSS for the bootstrap-responsive. Put a little extra space here inside of your <head> and add the following code:

```
<!-- HTML5 shim for IE backwards compatibility-->
    <!--[if lt IE 9]>
      <script
src="http://html5shim.googlecode.com/svn/trunk/
html5.js"></script>
     <![endif]-->
```

You'll see that we have a little snippet of code here, which is an HTML 5 shim. This is a particular piece of JavaScript that's available at Googlecode.com. It's designed to help provide backwards compatibility in older Internet Explorer browsers which, if you've been developing for very long, will know the ubiquitous Microsoft IE requires special attention. This shim itself is not a part of Bootstrap, but it will help make your websites more compatible with old Internet Explorer browsers, and you can just drop this right into your Bootstrap document right before your closing </head>.

ADDING JAVASCRIPT

Alright, so next you'll see that we have a starting <body> tag and an ending </body> tag. Put a few spaces in between those and just before the closing </body> tag, we're going to add some JavaScript declarations.

The first one that we're going to include is the <script> tag, and the source for that <script> tag will be http://code.jquery.com/jquery-latest.js and then a closing </script> tag. This will load the latest version of jQuery, which is a popular JavaScript framework from the jQuery website.

In order for this line of code to work, you need to make sure that you have an Internet connection, so that it's able to download it. If you're not going to be online, you may want to go to jquery.com and download the latest version of jQuery, and create that as a file, and then link that to your website's js file. But if you use this specific line of code, even if jQuery continually changes its version, which will happen, the latest version, whatever it may be, will always be located here, and that's a super easy way to keep your Bootstrap site up to date.

The next thing that we're going to do is add a line of code for the specific Bootstrap JavaScript, and if you take a look back in your local Bootstrap files, you'll see we also have two versions of Bootstrap JavaScript. One is bootstrap.js, and the other is bootstrap.min.js. As with our CSS, this is the minified version of the code along with the full-sized version. You can see the giant difference between the file sizes: 55K for the bootstrap.js and 25K for the minified version. By using the .min version, you'll definitely save some download time for mobile phone and dial-up users, in particular.

Since we're not going to be looking at JavaScript in great detail in this book, we'll just go ahead and link directly to the .min version of the JavaScript file. I have provided some JavaScript resources at the end of this book

Once again, include <script src="">, and insert the path to the .min version of the JavaScript. Be sure to close the script tag: </script>. You should have an index.html document resembling this:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My Bootstrap Site</title>
<link href="css/bootstrap.css" rel="stylesheet" type="text/css">
<link href="css/bootstrap-responsive.css" rel="stylesheet" type="text/css">
<!-- HTML5 shim for IE backwards compatibility -->
    <!--[if lt IE 9]>
      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
    <div class="container-fluid">
        <div class="row-fluid">
            <div class="span3">
                <p>This is our text on the left</p>
            </div>
            <div class="span9">
                <p>This is our text on the right</p>
            </div>
        </div>
    </div>
<script src="http://code.jquery.com/jquery-latest.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

So this basic document that you've put together is a great default setup for starting any Bootstrap web page. It has everything that you need here; it's got the links to the stylesheets in the right order; it has an HTML5 shim for backwards compatibility with older Internet Explorer versions; it has a meta tag, which is important for

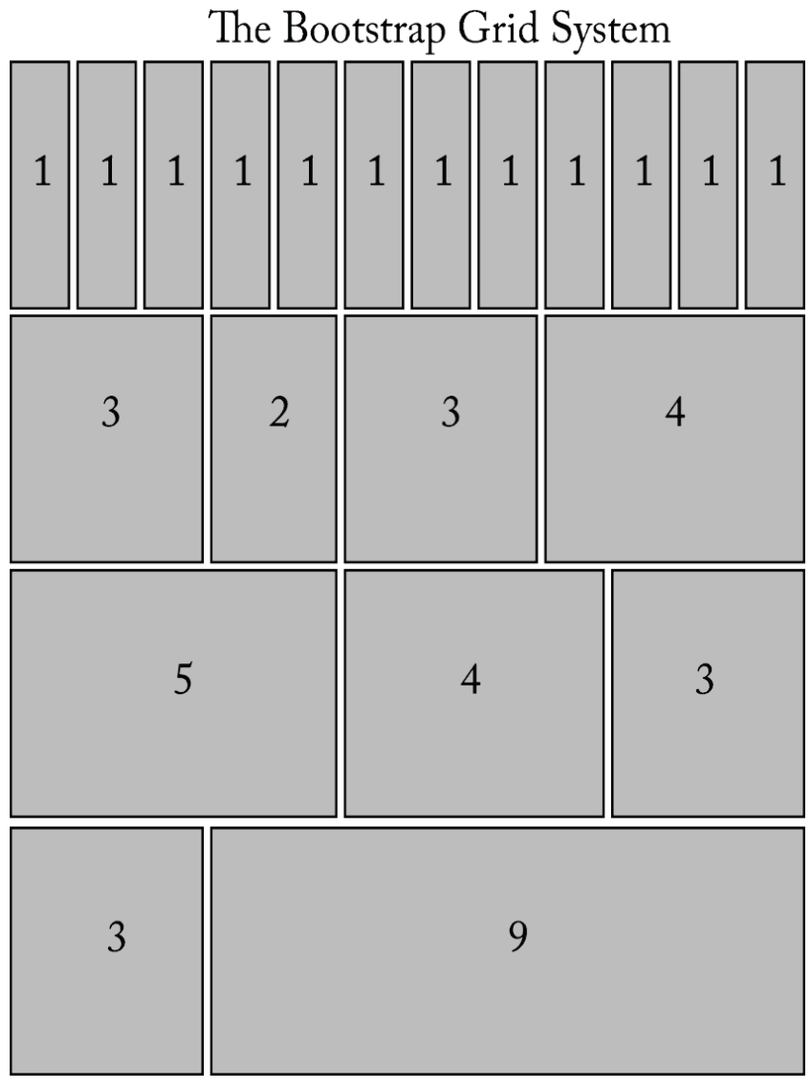
responsive design; and down towards the bottom of the page, we've linked to both of the JavaScript pieces that are required to make Bootstrap work correctly.

So once you have this in place, we're ready to start building our first page with Bootstrap!

CHAPTER 2: THE GRID SYSTEM IN BOOTSTRAP

THE DIFFERENCES BETWEEN THE DEFAULT AND FLUID LAYOUTS

Bootstrap ships with a 12-column grid system that presently comes in two varieties. If you take a look at the Bootstrap website at getbootstrap.com, and you click on the "Scaffolding" tab at the top of the page, and then you click on the link on the left side of the page for the grid system, you'll see Bootstrap's default grid system which, as you may have concluded, is what the grid system is referred to in Bootstrap. This is the default 940px wide grid which, as you can see, contains 12 columns per row, and will always contain 12 columns in a row, no matter how you slice it. Rows are going horizontally across the page and the columns are vertical, almost like cell-based spreadsheet programs. The essence of Bootstrap lies within the columns, because this "Scaffolding" is what we utilize to structure each web page. The following illustration depicts a "look under the hood" of how the columns of a Bootstrap page could be arranged:



It's designed to adapt somewhat via **adaptive layout** to the different screen widths our users may be using. In the example above, the numbers indicate, as examples, how many columns wide each box is. In the first row, there is only one column per box; the next row is grouped into four variously-sized columns, as

is the third and fourth. You can divide the columns up into any number of "boxes" you wish from one to twelve, as long as you come out spanning 12 across.

You can even divide these boxes up into subsets, with 12 columns inside those, and subdivide those, and on and on just like *Inception,* which you'll have the power to do soon.

If you visit GetBootstrap.com and hover your cursor over each box in the Scaffolding documentation, a tooltip will pop up to tell you how wide each box is. (That tooltip only appears in the documentation; it's not going to appear on your live site.)

Now you may have noticed that although I just told you this is a 12-column grid, the live example on the Bootstrap website only totals 9 columns across. Why is that? Well, the entire GetBootstrap.com site is built using Bootstrap, which shouldn't be a surprise. The left column on that specific website is three columns wide, which leaves nine columns available for the main content area over on the right.

In the live example on Bootstrap's website, the HTML grid works so that each row should comprise as many columns as its parent. So in this particular case, since the parent is nine columns wide, the grid example is also nine columns wide.

Now click on the link for the **Fluid** grid system--we'll jump down the page to that particular example. Notice the Fluid grid system is a truly responsive grid. Also note that it's 12 columns wide just like our adaptive layout example, even though this grid is embedded *inside* of the grid used to lay out the web page. This responsive system is somewhat different.

Since it's built with columns using a percentage width, instead of a fixed-pixel width, you need to keep all of the grids to 12

columns, or 100%, no matter where they occur on your page. So this is an embedded grid, 12 columns wide, even though it's inside of a larger area that's 9 columns wide.

Now I just used two terms to describe these grids: an "**adaptive layout**," which is for the default grid system, and "responsive design," which I'm applying to the **Fluid** grid system.

"Responsive design" was originally defined by Ethan Marcotte in a 2010 article published on alistapart.com. Marcotte wrote that responsive design must contain three specific aspects: 1) fluid grids, 2) flexible images, and 3) media queries. Let's review these individually.

1) A **Fluid grid** is one that's built using a relative measurement unit like ems or a percentage, instead of pixels.
2) **Flexible image** means that the images resize as part of the design. They generally display larger on a large-screen, and they scale down as the size of the viewport decreases.
3) **Media queries** are also included in the CSS3 standard, and they allow you to define different styles for different screen widths.

"Adaptive layout" is something of a precursor to responsive design and was also described on the alistapart.com website back in 2006 by Marc den Dobbelsteen. An adaptive layout uses a fixed-width grid instead. In the particular implementation we're going to be looking at, the layout of the screen resizes once specific pixel widths are reached, sometimes referred to as "break points." Be aware this design may not fill the entire screen at all times, in the way that a fully responsive layout does.

Many people, including Jeffrey Zeldman, dubbed "The King of Web Standards" by BusinessWeek, are calling for a broader definition of responsive design that will encompass adaptive design and adaptive

layouts as well. These two approaches are still viewed as separate, at least as of this writing. Bootstrap's default grid is the adaptive layout; while Bootstrap's fluid grid lets you make a *truly* responsive website.

So which grid is best?

Well, as with so many things in life, it depends. Consider what your goals are for your website and your audience.

In general, most designers, including your humble author, are leaning in the truly responsive direction, using percentage-based layouts for our sites, as well as images that resize. I believe this helps future-proof our websites as more and more web-enabled devices come on the market, of all shapes and sizes.

However, if you're not comfortable with that approach, you can certainly use the default grid and its adaptive layout. For the purpose of this book, we'll be working with the Fluid grid, as I feel that a major advantage of working with Bootstrap is the ability to make responsive websites quickly and easily.

So let's take a look at the Fluid grid system in a bit more detail.

EXPLORING THE FLUID GRID AND CREATING NEW ROWS

Now that we're clear on the difference between Bootstrap's two styles of grids, I'd like to go into more detail concerning the fluid responsive grid to show you how to mark up a page with some content.

Let's go ahead and get started building out a grid right inside of the <body> tag, so you can see how this works exactly. Let's go to our index.html file we just created.

In general, when you start an HTML web page to use inside of Bootstrap, you're going to start it with a , which is pretty standard when constructing any website.

You'll have some kind of wrapper that goes around the entire web page, and inside of a Bootstrap site that we want to have a fluid layout, we'll simply use a with a class of "container-fluid." Let's start by going ahead and putting that in place along with its mate, a closing at the bottom. So that's going to encompass your entire web page; that's the outside wrapper that holds the columns on your page together. Then the next thing we want to add to this is the start of our first row.

If you have a with a class of "row-fluid," that will indicate where this particular row is going to start and stop. This is the very first part of the grid, indicating where each row of that grid happens to be. Inside of that, you may have another couple of s. Let's say you have a with a class of "span3." In other words: span3 columns so we can insert our HTML inside of that.

For example, we could say here's a paragraph<p>, here's a left column on this web page spanning 3 columns, and then we end that particular . Then we can have a second inside of that row. In this case, we have nine columns left over. So we'll just put in a class of "span9," in order to always add up to a total of 12 across. Just always remember 12 columns. Incidentally, Bootstrap was originally 16-columns but was recently changed. The problem with the 16-column grid system was that it wasn't flexible enough to have a 3column layout without adding extra CSS. So with version 2 of Bootstrap came a 12-column system.

As I briefly mentioned earlier, “span1” through “span12” are the classes that are available to you, and each one of those classes indicates how many columns it should “span” over. Easy? Yes!

So in your html file, create a starting and an ending , and then inside of that you’ve got some text wrapped in <p> tags. Here’s what we should be looking at now:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My Bootstrap Site</title>
<link href="css/bootstrap.css" rel="stylesheet" type="text/css">
<link href="css/bootstrap-responsive.css" rel="stylesheet" type="text/css">
<!-- HTML5 shim for IE backwards compatibility -->
    <!--[if lt IE 9]>
      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
    <div class="container-fluid">
        <div class="row-fluid">
            <div class="span3">
                <p>This is our text on the left</p>
            </div>
            <div class="span9">
                <p>This is our text on the right</p>
            </div>
        </div>
    </div>
<script src="http://code.jquery.com/jquery-latest.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

I put just a simple line of text inside of each one of these s here, but of course, you could have quite an extensive set of markup inside of each one of these particular divs, the span of 3, and the span of 9. You could have images, you could have headings, you could have any kind of markup that you want inside of those s. I’m just making this very simple for our first example.

So now, go ahead and save this web page, (File > Save), and view this inside of your browser. You may very well (and should) have a way to easily do that in your text editor by typically clicking on a globe icon at the top or via a pull-down menu, and clicking “Preview in Firefox,” or Chrome, IE, Opera, Safari, or whichever is the browser you’re interested in and have configured. Ideally, you’ll view your site in all the major browsers when you’re done to ensure compatibility.

So you will see now that we have the very start of a grid on the website; we have our left column, and we have a main content area on the right. The main content area doesn’t look that large because I only put in one sentence, but if you put in a more impressive block of text, you would see that it stretches nine columns across the web page.

Remember at this point that Bootstrap was built to use HTML5, so we can actually use real HTML5 markup tags with Bootstrap. You certainly don’t have to, but as with responsive design, I believe HTML5 is the way the wind is currently blowing and, if possible, you should use it. Bootstrap encourages it. The only reason we’ve begun this with s to start with, is because most developers/designers understand s, and there are a number of you who are just learning HTML5 at this point in time.

So let’s take a look at a way that we could do that.

One of the things that we might want to have in our particular web page is to include a header at the top. Pretty standard for a website, right? So inside of our existing with a class of “container-fluid,” let’s go ahead and put in a header for our particular web page here, and in HTML5, the tag that you use for that is the tag of <header>, which you can use here as well. Also, give this a class of “**row-fluid**.” Bootstrap makes it very easy to switch back and forth

between fluid and standard layouts, with the simple addition of adding the term "fluid" to our classes. Gotta love it.

So, no extra here in this particular markup; we just have the start of the header, and we have the finish of the <header>, and inside of that you're going to put in the <title> of your web page. You might do that with an <h1>, and then set it to have a class of "span12" because we want it to be huge and go all the way across the web page.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My Bootstrap Site</title>
<link href="css/bootstrap.css" rel="stylesheet" type="text/css">
<link href="css/bootstrap-responsive.css" rel="stylesheet" type="text/css">
<!-- HTML5 shim for IE backwards compatibility -->
    <!--[if lt IE 9]>
      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
    <div class="container-fluid">
        <header class="row-fluid">
            <h1 class="span12">My Very First Bootstrap Page!</h1>
        </header>
        <section class="row-fluid">
            <aside class="span3">
                <p>This is our text on the left</p>
            </aside>
            <article class="span9">
                <p>This is our text on the right</p>
            </article>
        </section>
    </div>
<script src="http://code.jquery.com/jquery-latest.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

Now we have better markup there for our header. Let's go ahead and fix what we did before with all of those s in

what's now the second row, where we have the with a class of "**row-fluid**."

That might be a "section" of your web page, so we can change the tag with a class of "**row-fluid"** to a <section> with a class of "row-fluid," and then inside of that we'll have a left column, and we have some main content. There are HTML5 tags that describe those as well.

Depending on what exactly is in the left area, it may very well be an <**aside>**, so replace the with an <**aside>** and keep a class of "span3." And it's quite likely that whatever goes on the right side of this page is an <**article>**. So we'll use the <article> tag, with a class of "span9."

So, now that you have two rows in your particular web page (you have a header, and you have a section with an aside) go ahead and save this web page and see what it looks like in your browser. Hopefully yours looks similar to this:

So there's your heading. You have your left column, and you have your right column. Of course, just because you've gone ahead and established this web page as having a left column and a right side with an article, doesn't mean that your following rows have to be laid out exactly the same way, whatsoever.

Go back to your text editor, and I'll show you exactly how that might work. Perhaps next you want to flip it around and add a row that has a very large left column and a very small right column. You'll add a <section>, once again, with the class of "row-fluid," and inside of that, perhaps this time have the <article> on the right side, this time an <article> with a class of "span9."

Of course you could divide those columns into whatever numbers you wanted to use. We can add another <article>, </article> and maybe put the <aside> over on the right, so add an <aside> with a class of "span3," then maybe we have some more information about that <article> over on the <aside>, and close that, and you can close out the </section>.

So once again (we're going to be doing this a lot), just go to File > Save, and if you take a look at your web browser, you'll see that now you have a second row of information. We have our <article> over on the left side of the screen, and over on the right side, we have more information about the <article>; that's that <aside> that you put in. In this case it's displaying in a right column. Please

note that I'm using a larger screen in these examples, so the proportions may look slightly different than what you have on your screen.

Okay, so the very last thing that we ought to do is add a footer to your web page, so hop back into your text editor. HTML5 gives us a great and logical <footer> tag for doing this. So in this case, <footer> will have a class of "row-fluid", and also like our <header>, just give this a span of 12 so that it goes all the way across the page, and, just for example, you may want to include some text in here wrapped in an <h4> tag, such as your copyright, and some bottom navigational elements.

You should make sure that your closing is down there at the bottom. That closes the container- fluid from all the way at the top of the page. Go ahead and save this one more time, File > Save, preview this in a browser, and you'll see that we now have something resembling a footer that's in place at the bottom of the web page, just like we wanted.

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My Bootstrap Site</title>
<link href="css/bootstrap.css" rel="stylesheet" type="text/css">
<link href="css/bootstrap-responsive.css" rel="stylesheet" type="text/css">
<!-- HTML5 shim for IE backwards compatibility -->
  <!--[if lt IE 9]>
    <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
</head>
<body>
    <div class="container-fluid">
        <header class="row-fluid">
            <h1 class="span12">My Very First Bootstrap Page!</h1>
        </header>
        <section class="row-fluid">
            <aside class="span3">
                <p>This is our text on the left</p>
            </aside>
            <article class="span9">
                <p>This is our text on the right</p>
            </article>
        <section class="row-fluid">
            <article class="span9">
                <p>Yet more Text on the left as an Article</p>
            </article>
            <aside class="span3">
                <p>And here's our text over on the right in the Aside</p>
            </aside>
        </section>
        </section>
        <footer class="row-fluid">
            <h4 class="span12">Copyright 2013 Michael B. Musgrove "-" Navigation Here</h4>
        </footer>
    </div>
<script src="http://code.jquery.com/jquery-latest.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

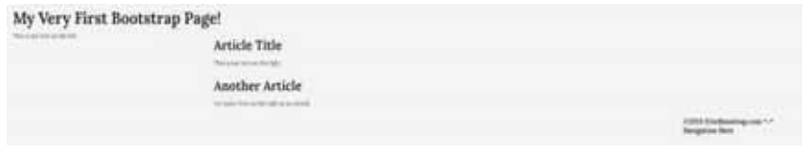
Will give you:

Finally, you may be wondering about the HTML5 markup that you just put into place, and that's currently being used in the web page that you're building. Bootstrap, as a system and a framework, indicates that it's mostly backwards-compatible to IE7. However, sometimes you're going to find backwards compatibility issues with older browsers if you're unfortunate enough to have to support them, particularly if you're working with the newer HTML5 semantic tags that we just used, such as <header>, <footer>, <aside>, <article>, and there are others. So what do we do?

This is as good of a time as any to introduce you to **Modernizr**, which is a JavaScript solution that can help make your Bootstrap website HTML5 backwards-compatible with older browsers. It's not perfect, and it definitely isn't the only solution to make this work, but it is a solution that's widely used, and one that you may want to try if you find your Bootstrap projects are producing bugs in older browsers that you may end up having to support, unfortunately. You can find this by going to modernizr.com, and that website will explain everything that you need and want to know about Modernizr. Modernizr itself is outside the scope of this book, but you can find it covered fairly comprehensively online. Modernizr is, of course, compatible with Bootstrap, so no compatibility worries there.

UNDERSTANDING FLUID OFFSETTING

So here's our web page that we've built so far:



Let's say what you really wanted to have was a left column on this page, and you wanted to have a few articles listed on the right side of the page. Although this third row looks like I have an article on the left side of the page and more information about it over on the right, it would obviously look a lot better if we had the articles all lined up one after another.

But for the second article, I don't really want anything to appear over in the left-hand column. What can we do? In Bootstrap, this

is called "**offsetting**," and it's very easy to set up, and it does exactly what the term implies.

We can set this up so that you can have the two articles appear underneath each another and not have a left column in the second row. There must be 12 columns, no matter what you do. So offsetting is a way of allowing you to have those 12 columns represented, even if they're not all containing content.

So, go back to your text editor and let's make a couple of quick changes to your page. Just to make your articles stand out a little bit more in the second row, give them <h2> tags, and call this "**Article title**," and then down in the second <section>, delete the <aside> with class of "span3." That's what we'll offset. Also, clean up the markup in this particular article a little bit as well, so it has a heading of <h2>, another article, and of course, underneath that would be our text we inserted earlier.

Now what we need to do is inside of this third row that you see, we have a "span9," but we deleted the other three columns. We can account for them with the "offset" that was just mentioned. The way to utilize it is to include it with the span9 class as "**offset3**," all one word.

So what your code should look like now is this:


```
<title>My Bootstrap Site</title>
<link href="css/bootstrap.css" rel="stylesheet" type="text/css">
<link href="css/bootstrap-responsive.css" rel="stylesheet" type="text/css">
<!-- HTML5 shim for IE backwards compatibility -->
    <!--[if lt IE 9]>
      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
    <div class="container-fluid">
        <header class="row-fluid">
            <h1 class="span12">My Very First Bootstrap Page!</h1>
        </header>

            <section class="row-fluid">
                <aside class="span3">
                    <p>This is our text on the left</p>
                </aside>
                <article class="span9">
                <h2>Article Title</h2>
                    <p>This is our text on the right</p>
                </article>
            </section>

            <section class="row-fluid">
                <article class="span9 offset3">
                    <h2>Another Article</h2>
                    <p>Yet more Text on the right as an Article</p>
                </article>
            </section>

        <footer class="row-fluid">
            <h4 class="span2 offset10">02013 FreeBoostrap.com *-* Navigation Here</h4>
        </footer>
    </div>
<script src="http://code.jquery.com/jquery-latest.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

These offset classes, just like our content columns, range from 1 to 12, and what they'll do is they'll add blank columns on the left side of your particular row. So, with an offset of 3, it's going to add 3 blank columns on the left side of the screen. They'll be placeholders, followed by the <article>, which will occupy the 9 columns on the right side.
Go ahead and save this page, File > Save, and put it into your browser. Now you can see that you have a left column on this web page, and you have the two articles that are appearing underneath. All those line up much better on the page.

We may also want to do something with that "Copyright" while we're here. Right now the copyright is located down at the bottom of this web page. It's got a span of 12 on it so it goes all the way across the bottom of the page. But perhaps what you really want to do is shorten that up and have it appear just on the right side at the bottom the page. You can do that also with offsets. So back to your text editor.

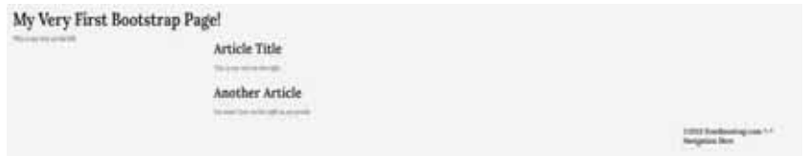
On the bottom, rather than a class of "span12," try a class of "Span2," and an offset of 10 and then down on the bottom, put in a copyright symbol under Insert > HTML > Special > Characters > Copyright (or hold alt and type 0169), and then you could put in some actual text such as: 2013 FreeBootstrap.com. Go ahead and save that with Ctrl+S or Command+S, or just File > Save.

And now when you go back to your browser, and refresh your web page, you'll see that you have the copyright statement now appearing over towards the right side of the screen, exactly the way that you wanted it to.

So in this particular section, we've taken a look at the offset property inside of the grid. This allows you to have a row with all 12 columns accounted for even if not all 12 of those columns contain some type of content.

NESTING WITH FLUID GRIDS

Occasionally, you may want to segment a portion of the grid that you're working with into smaller pieces. For example, here's the web page that we've built so far in this particular chapter, and inside of the content area, maybe we'd like to have two information boxes appear side by side, underneath the two articles that are present:

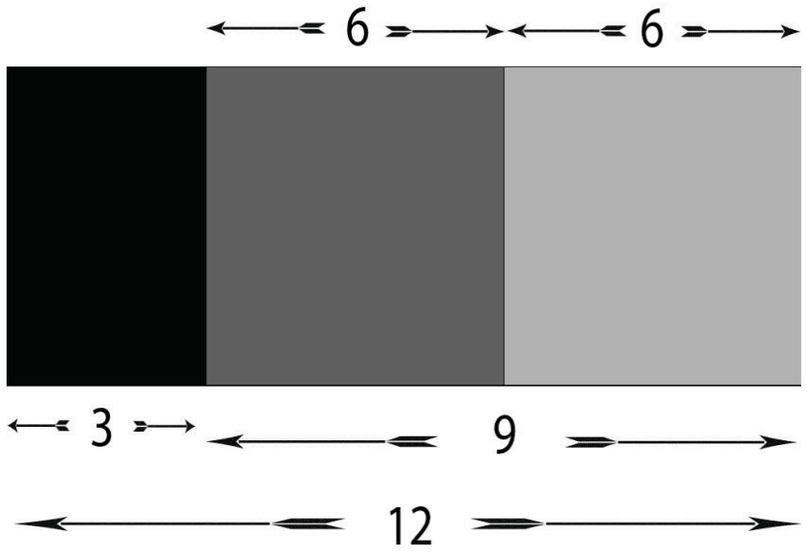


Unfortunately, since that particular part of the web page has a span of nine, in other words nine columns, we can't make two evenly-sized boxes of four and a half, so the closest we could get is that one would be four columns wide and one would be five columns wide.

That's no good.

However, there's another way that we can attack this.

We can **nest** another row inside of that row so that we would have 12 columns to work with, and then divide those 12 columns into 2 "sub-rows" of 6 columns each. It's easier than it sounds. These are called **nested rows** inside of Bootstrap and it's how to subdivide your columns. Here's an illustration of what we're doing:



If you go back into your text editor and open up your HTML web page, and scroll on down past the second <article>, we can add some more code. Add this as a **<section>** with a class of "**row-fluid**," and put it in a with a class of "span9 offset3," and then you may want to go ahead and close those so you don't lose track of those two tags later.

Now, if you went ahead and just put your information boxes in here, at this point we could have one that was four columns wide and one that was five columns. But let's say for example what you want are two evenly-sized boxes. To do that, just go with an inside with a class of "span9 offset3." Then go ahead and add another row.

Inside of this , add another with a class of "row-fluid," exactly the way we've done before, and then inside of that particular you'll be able to insert your equally-spaced information boxes.

To do that you would put in a there with a class of "span6," and inside of that we'll add a title as an <h3>, and you can call that something like "Info Box 1."

Since we're going to be setting up a complete page, we'll need some images here as well. If you don't have any handy, please feel free to download these as placeholders. I have provided image files at http://FreeBootstrap.com/resources for your use. Just right click each image, choose "Save As" and save them to your desktop or another convenient place.

So go ahead and add those with an image source tag <img src="">, and you can browse for those files inside your Bootstrap images folder to put the images' path between the quotation marks. For example, if they're in the root folder, you can just insert "img/northernlights.jpg." You could also add some information about the images below them by wrapping your text in <p> tags. If you're getting these examples online, copy that one little snippet of code here (this with a class of "span6" and the image path), highlight and Ctrl+C to copy it, and right after that closing , paste that into place. You can add the other image here, just changing the path so it's correct, and giving it some descriptive text in a <p> tag, so what we have now looks like this:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My Bootstrap Site</title>
<link href="css/bootstrap.css" rel="stylesheet" type="text/css">
<link href="css/bootstrap-responsive.css" rel="stylesheet" type="text/css">
<!-- HTML5 shim for IE backwards compatibility -->
    <!--[if lt IE 9]>
      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
    <div class="container-fluid">
        <header class="row-fluid">
            <h1 class="span12">My Very First Bootstrap Page!</h1>
        </header>

            <section class="row-fluid">
                <aside class="span3">
                    <p>This is our text on the left</p>
                </aside>
                <article class="span9">
                <h2>Article Title</h2>
                    <p>This is our text on the right</p>
                </article>
            </section>

            <section class="row-fluid">
                <article class="span9 offset3">
                    <h2>Another Article</h2>
                    <p>Yet more Text on the right as an Article</p>
                </article>
            </section>

            <section class="row-fluid">
                <div class="span9 offset3">
                    <div class="row-fluid">
                        <div class="span6">
                            <h3>Information Box One</h3>
                             <img src="/Free Bootstrap/img/northern-lights.jpg">
                             <p>These are Northern Lights</p>
                        </div>
                        <div class="span6">
                            <h3>Information Box Two</h3>
                             <img src="/Free Bootstrap/img/portugal.jpg">
                             <p>This was taken in Portugal</p>
                        </div>
                    </div>
                </div>
            </section>

        <footer class="row-fluid">
            <h4 class="span2 offset10">©2013 FreeBoostrap.com *-* Navigation Here</h4>
        </footer>
    </div>
<script src="http://code.jquery.com/jquery-latest.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```


So now we also have Information Box number 2, and it'll be your second image instead of your first, of course, so that needs to be changed. And we've gone ahead and we've added an additional row to our grid and inside of that row we've nested another row, so we have to make sure all of those s are closed and everything looks good. Go ahead and save the file, and now when you take a look at this inside of your browser there are your two images with the text underneath, and we have our two articles that appear on the top.

This is a great opportunity to demonstrate the responsiveness of the Bootstrap grid as well, now that we have both images and text available. Right now your browser window is probably maximized so it's occupying the full width of your particular screen in a very typical desktop configuration. However, if you resize your browser window slowly--just grab the corner and drag it on over--you'll see that your images are in fact actively resizing. That's part of the definition of a responsive website. You'll notice that the grid is resizing as well, continuously.

You don't see it collapsing as you move the corner inward, and it stays the same until it suddenly jumps to the next position. That would be an adaptive layout, which would jump between different sizes. But a responsive design should be continuous and smooth, as you can see that this one is. And as you continue to shrink the screen you'll see that the grid continues to adapt, so what you see happening here is, at the very top of the screen, you see the left column where we had a "span3" and the <article> with a "span9." You should see that those are actually stacked on top of each other, then underneath we had an offset of 3 with a span of 9 for those articles and for these info boxes.

Notice that those images continue to resize as well, and if you scroll on down the page, you can see that everything is stacking

on top of one another, which is exactly the way that we would expect this grid to respond.

So now we've taken a look at the three critical aspects of this grid and how they work together, including the actual grid itself and how we can create any rows we want in it, and we've taken a look at fluid offsetting— which is simply shifting items over on the page--and finally, we've seen how we can nest a row inside of another row to achieve an even finer level of control on our grid.

BOOTSTRAP MEDIA QUERIES AND DEVICE-SPECIFIC STYLING

One of the last things you may want to do when working with these responsive design grids inside of Bootstrap is that you may wish to have some particular parts of the layout that are only visible on a desktop, or visible only on a tablet or a phone, or you may want to hide specific elements on one of those devices. Bootstrap includes some nice classes that will allow you to easily do exactly that as well.

Head back over to GetBootstrap.com, and under the "Scaffolding" tab under "Responsive design," if you'll just scroll on down a little on that page, you'll see a very useful chart under "Responsive utility classes:"

Responsive utility classes

For faster mobile-friendly development, use these utility classes for showing and hiding content by device. Below is a table of the available classes and their effect on a given media query layout (labeled by device). They can be found in `responsive.less`.

Class	Phones 767px and below	Tablets 979px to 768px	Desktops Default
`.visible-phone`	Visible	Hidden	Hidden
`.visible-tablet`	Hidden	Visible	Hidden
`.visible-desktop`	Hidden	Hidden	Visible
`.hidden-phone`	Hidden	Visible	Visible
`.hidden-tablet`	Visible	Hidden	Visible
`.hidden-desktop`	Visible	Visible	Hidden

This explains exactly how these particular classes work and which specific devices using those classes will produce content that's either visible or invisible. We're going to put those to work here in just a moment.

So, what we'd like to do is to set up a row on the web page that we've been working on so far, and set it up in a way that you're going to have some text that's going to show up only on certain devices. The important part to remember about this is the way that these particular CSS classes work.

First of all, the big thing to remember here is that these are CSS, so they are testing the width of the window of the *browser*. They aren't actually testing the device itself, which is something very different that actually comes over as part of the header in the request process for getting web pages. It's just testing the width of the screen, so it's calling devices such as phones, tablets, and desktops, but just by making your screen wider or making your screen narrower, you'll be able to simulate the screen sizes of phones, tablets, and desktops, and they'll work just fine.

Again, remember that phone, tablet, and desktop actually refer to specific *widths* of the screen, and those widths are detailed in the chart above. Phones are 767 pixels and below, tablets are 768 pixels to 979 pixels, and desktops are the default, which is anything wider than 979 pixels.

So, inside of your text editor, if you scroll down to just before the footer, let's go ahead and add another part to this web page. Let's add another <section> with a class of "row-fluid," just as we've been doing all along, and then add three s, so each will have a class of "span4."For the text in our first , type "This is visible on the desktop." Now add a second with the class of "span4" and the text "This is visible on a tablet," and then finally, the last with a class of "span4" and the text "This is visible on a phone."

Now what you'll need to do is add the appropriate responsive utility classes to those. So in addition to the classes of "span4" that we just gave them, if you'll just add a space, that will allow you to insert additional classes for our additional HTML tags, and to the first let's add "visible-desktop." For the second one, we'll add "visible-tablet," and on the third one we'll add the class "visible-phone." Your code should now resemble this:

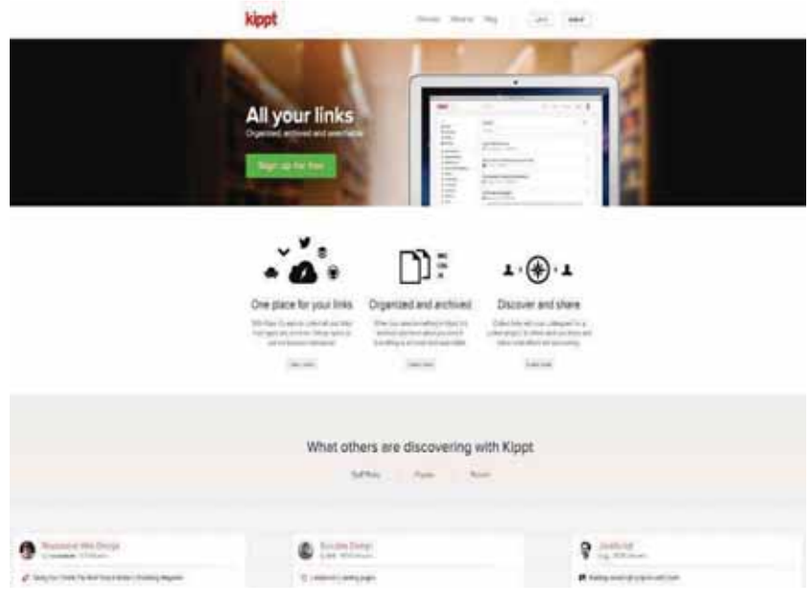
```
<section class="row-fluid">
    <div class="span4 visible-desktop">This is visible on a desktop</div>
    <div class="span4 visible-tablet">This is visible on a tablet.</div>
    <div class="span4 visible-phone">This is visible on a phone.</div>
</section>

<footer class="row-fluid">
    <h4 class="span2 offset10">@2013 FreeBoostrap.com *-* Navigation Here</h4>
</footer>
```

Go ahead and save that, and now when you go to your browser to preview this page, at the bottom you'll see **This is visible on a desktop**. As you make the screen smaller, and you pull it over a little bit, you see that text has now changed to **This is visible on a tablet**, and now as you pull the screen even further over, and resize it again, you'll see down at the bottom **This is visible on a phone**. That's the way these particular classes work. The hidden classes work similarly, and you can give those a try on your own.

ADDING A HERO-UNIT

A "hero-unit" is a large area at the top of a page, usually the main focus of a home page or a landing page, that spans the width of a web page and really grabs the user's attention. It's like having a Jumbotron on your page, which actually is what this will likely be called in Bootstrap 3.0. Here's an example of a site using a Span12 with Bootstrap for reference, that spans the entire page:



We'll be starting from the very beginning and using a brand new Bootstrap HTML5 document here. As before, we're going to start by adding a within the <body> tags with a class of "container-fluid" to have that all-inclusive wrapping that goes all the way across the web page, and then close that down at the bottom. And then, just as we've been doing all along, we have to start things with a row, so we'll have a with a class of "row-fluid" and a closing down at the bottom, and then in between that we'll go ahead and put them in your hero-unit.

Now, for this particular page, the hero-unit is really functioning as a big header at the top of the web page. It's the huge billboard that catches your attention, so let's use the <header> tag from HTML5, and give this a class of "span12," because we want it to span the page. Go ahead and close that </header> down at the bottom. Then let's go ahead and add some text to what we have so far:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>My Bootstrap Site</title>
<link href="css/bootstrap.css" rel="stylesheet" type="text/css">
<link href="css/bootstrap-responsive.css" rel="stylesheet" type="text/css">
<!-- HTML5 shim for IE backwards compatibility -->
    <!--[if lt IE 9]>
      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
    </head>

    <body>
        <div class="container-fluid">
            <div class="row-fluid">
            <header class="span12">
                <h1>My Second Bootstrap Page!</h1>
                <p>Photo booth adipisicing +1, exercitation selfies beard organic synth banh mi hashtag pug sartorial.
                    Thundercats twee disrupt plaid PBR put a bird on it authentic food truck. Fanny pack proident
                    helvetica sed high life letterpress. Magna pour-over labore food truck quis flannel.
                </p>
            </header>
            </div>

            <footer>
                <p>
                    &copy; Copyright 2013 FreeBootstrap.com
                </p>
            </footer>
        </div>
    </body>
</html>
```

Start by including an attention-grabbing headline, and put that in an <h1> tag, and then following that, let's put in a bit more of a description, so let's put in a paragraph tag , and follow that up with your text, which can be whatever you'd like for this demonstration, including just copying and pasting some Lorem Ipsum, or dummy text, (as I've done in the above example) in the area we just set up for text, and close your tag to end that paragraph.

We want to add a "Sign Up!" button at the end of all this, so let's immediately follow that up with a paragraph and a link within that using <a href="#">. We're not going to link this anywhere right now, because we only have one page of the website. Obviously, you'd put a link in there when you're doing this for real.

So, if you just save what you have so far--File > Save-and open this up in your web browser, you'll see that we have a page with a header and some text and the "Learn more" link. It's not really anything to get excited about yet, and it's just laying out some basic structure, just as we've already done in some form in the previous sections.

Now that that's in place, let's go ahead and turn it all into a hero-unit, and the way that we do that is after the "span12" class, we can simply add the class of "hero-unit." So now, we'll have: <header class="span12 hero-unit"> instead. How easy is that?

So once again, if you save that--Ctrl+S or Command+S to Save--and bring up your web browser and view it in the browser, you'll see that immediately this has now been formatted very nicely. It has a darker box behind-it to set it off, the text is nice and big, and it's definitely a little more attention grabbing.

One other thing you may want to add to the hero-unit is an image. You can add an image to this very easily, as well. If you jump back into your text editor, and go ahead and add an image just after the header, by inserting an image source. Make sure to give it an alt tag always on your live sites, for both SEO and accessibility purposes. So what you should have now is something like this:

```
<div class="row-fluid">
<header class="span12 hero-unit">
    <img src="img/mountains.jpg" alt="mountain range" class="pull-right">
    <h1>My Second Bootstrap Page!</h1>
    <p>Photo booth adipisicing +1, exercitation selfies beard organic synth banh mi hashtag pug sartorial.
        Thundercats twee disrupt plaid PBR put a bird on it authentic food truck. Fanny pack proident
        helvetica sed high life letterpress. Magna pour-over labore food truck quis flannel.
    </p>
    <p><a href="#">Sign Up!</a>
    </p>
</header>
</div>
```

You may have noticed in the above code, that I added one more class. There's a class of "pull-right" with the image source. The class of "pull-right" is simply a utility class. Not surprisingly, there's a "pull-left" as well, and, of course, it pulls things left. Essentially, these classes are designed to float an image left or right, or any other elements on your web page for that matter. So go ahead and save what you've got so far--Ctrl+S or Command+S to Save--and if you refresh this in your browser, and take a look, you should see your nice big image there to complement what you've done so far.

The last thing that we want to do here, in order to make this look like a real hero-unit, is deal with the "Sign Up!" link. We'll cover buttons in quite a bit more detail in an upcoming chapter, but for now, in your text editor, if you head over to

the <href> tag for signing up(!), we'll go ahead and assign it three classes.

Let's add a class of "**btn,**" "**btn-primary**," which actually controls the color of the button, and we'll also add "**btn-large**," which will make it a button you just can't ignore. So now, you should have a line such as: Sign Up!

Here's a chart explaining the various button colors and classes from the Bootstrap website:

Button styles can be applied to anything with the `.btn` class applied. However, typically you'll want to apply these to only `<a>` and `<button>` elements for the best rendering.

Button	class=""	Description
Default	`btn`	Standard gray button with gradient
Primary	`btn btn-primary`	Provides extra visual weight and identifies the primary action in a set of buttons
Info	`btn btn-info`	Used as an alternative to the default styles
Success	`btn btn-success`	Indicates a successful or positive action
Warning	`btn btn-warning`	Indicates caution should be taken with this action
Danger	`btn btn-danger`	Indicates a dangerous or potentially negative action
Inverse	`btn btn-inverse`	Alternate dark gray button, not tied to a semantic action or use
Link	`btn btn-link`	Deemphasize a button by making it look like a link while maintaining button behavior

Once these three classes are in place, hit Ctrl+S or Command+S to save, and off to your browser. Go ahead and refresh, and you'll see that we have a very buttony-looking "Sign Up!" link, even though that's just a regular old <href>. The styling that we've easily applied with Bootstrap gives our button a very nice look, very easily.

So, hero-units are a great way to set up your homepage to really draw the user's attention to elements of website and make users aware of what's most important on the page. In this case, we've marked it up with a <header> tag. However, it certainly doesn't have to be marked up as a <header>; it doesn't even have to occur at the top of your web page. You can use hero-units anywhere you'd like within your document: body or footer or elsewhere. You can go ahead and give this a try on your own Bootstrap websites quite easily.

CREATING A THUMBNAIL GALLERY

Thumbnail galleries are useful for displaying a bunch of images on a web page where they're the main focus. You can usually click on them to get a larger version of the image. You can also use a thumbnail gallery for displaying photos and text on a page in a more organized manner. Bootstrap includes some markup for creating a thumbnail gallery. So let's go ahead and take a look at that now.

If you go to your text editor again, and scroll on down past the with class "row-fluid," in this final , we're going to go ahead and add another row to our document.

So to do that, once again, we'll make another with a class of "row-fluid," making sure, of course, that you close that further on down the page. And inside of this particular , the markup that we're going to use for our thumbnails is an unordered list<ul>.

Let's go ahead and start that with a <ul>, and give that a class of "thumbnails." Don't forget your closing below. Most text

editors handle that for you, but just in case. Once that's in place, the next thing we need to add is add our list items <li>.

For our example, we're going to have three thumbnails on our page and we want to have them line up next to each other in three columns. Since we're working with rows and a 12column grid, simple math would reveal that we need have a span of 4. Now within that we're going to have an image, we'll have a heading, and we'll have some text. And all of those together could be considered an <article> in the world of HTML5, so we're going to use the <article> markup for that. Give it an <article> tag with a class of "thumbnail," (singular, without an 's') and inside of that we'll add our image. Choose your photo here and, again, make sure to give it an alt tag in the markup. We can go ahead and give this an <h3> for your image title, followed by your text. (You can go grab some Lorem Ipsum if necessary for playing around with this, which some text editors can insert in for you.) Then go ahead and copy that text-- Ctrl+C or Command+C to copy. Add that copied text in as a paragraph and then Ctrl+V or Command+V to paste that on in, and close it with your . So that'll be our first thumbnail in our grid. We have a single <li>; and we have an <article> inside of that. If you don't believe an <article> is appropriate to what you're doing, you may also use a or any other tag that's more appropriate to nest inside of that <li>. In the below example, I've put in my image, my <h3>, and my .

```
            <p><a href="#" class="btn btn-primary btn-large">Sign Up!</a>
            </p>
        </header>
        </div>

        <div class="row-fluid">
            <ul class="thumbnails">
                <li class="span4">
                    <article class="thumbnail">
                        <img src="img/bee.jpg" alt="">
                        <h3>Title of Image One Here</h3>
                         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                            Morbi nisi nibh, luctus mollis faucibus suscipit, lobortis at dolor.
                            Phasellus commodo tempor elit sit amet commodo. Donec tincidunt
                            rhoncus ipsum, id suscipit nulla vulputate in. Cras ultricies
                            bibendum nisl et vehicula. Suspendisse potenti.</p>
                    </article>
                </li>

                <li class="span4">
                    <article class="thumbnail">
                        <img src="img/color-swirl.jpg" alt="">
                        <h3>Title of Image Two Here</h3>
                         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                            Morbi nisi nibh, luctus mollis faucibus suscipit, lobortis at dolor.
                            Phasellus commodo tempor elit sit amet commodo. Donec tincidunt
                            rhoncus ipsum, id suscipit nulla vulputate in. Cras ultricies
                            bibendum nisl et vehicula. Suspendisse potenti.</p>
                    </article>
                </li>

                <li class="span4">
                    <article class="thumbnail">
                        <img src="img/rainbow.jpg" alt="">
                        <h3>Title of Image Three Here</h3>
                         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                            Morbi nisi nibh, luctus mollis faucibus suscipit, lobortis at dolor.
                            Phasellus commodo tempor elit sit amet commodo. Donec tincidunt
                            rhoncus ipsum, id suscipit nulla vulputate in. Cras ultricies
                            bibendum nisl et vehicula. Suspendisse potenti.</p>
                    </article>
                </li>
            </ul>
        </div>
```

Now, if you're doing this as a real thumbnail gallery in which you just have pictures and you don't have text following it the way we do here, where we have an <article> class of "thumbnail," you could just use that as an <href> which you could link to the larger version of the thumbnail.

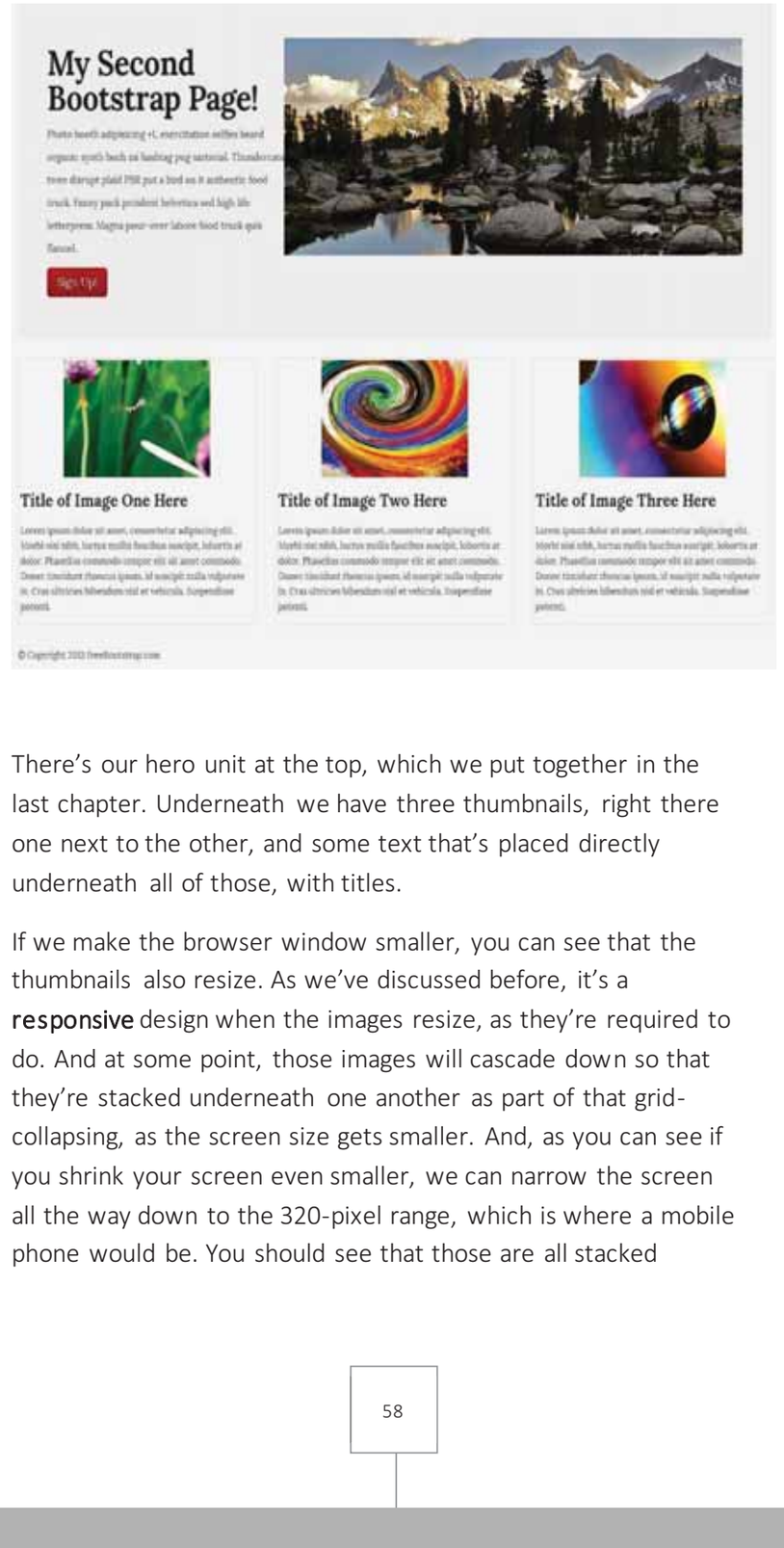
The class for that <href> would still be "thumbnail," but you wouldn't have any of the text following it. The problem with doing that here in this context is that <h3> and <p> are block level HTML tags. Meaning: you cannot nest the block level

HTML tag inside of an inline element like the <a> tag. So, in this case, we're using the <article> class for containing those block level elements. We could certainly wrap an <a> tag around the images if we wanted to make this example's images clickable.

Okay, so now that we've got that unit in place, go ahead and copy the new code snippet we've just added. Again, just highlight those lines for the <li> and then Ctrl+C or Command+C to copy, hit enter a couple of times to add some room, and Ctrl+V or Command+V to paste it in. Then set your proper indentation there. Next make a couple of changes for the next one of these s.

Insert the image for number two and change the <h3> here to the correct title. You should change your image source here and go ahead and change your Alt tag as well and include your original or dummy text. Finally, one more time, go ahead and highlight that <li>, Ctrl+C or Command+C to copy and then Ctrl or Command+V to paste it in one more time, and we'll make these changes again for image #3.

The last one that we're dealing with is image #3, so go ahead and copy that text, paste that into your text editor document, and put in the code for this image right below. And then, of course, we need to change the image source again from our images folder and change the Alt tag. Save your file, and we can go ahead and preview this in the browser, and we have something actually starting to look like a real web page.



There's our hero unit at the top, which we put together in the last chapter. Underneath we have three thumbnails, right there one next to the other, and some text that's placed directly underneath all of those, with titles.

If we make the browser window smaller, you can see that the thumbnails also resize. As we've discussed before, it's a **responsive** design when the images resize, as they're required to do. And at some point, those images will cascade down so that they're stacked underneath one another as part of that grid-collapsing, as the screen size gets smaller. And, as you can see if you shrink your screen even smaller, we can narrow the screen all the way down to the 320-pixel range, which is where a mobile phone would be. You should see that those are all stacked

underneath each other very nicely, and there's a pull-down menu button available.

So as you may imagine, the thumbnail galleries can be really useful. You can use them in a variety of different websites for many different purposes, from displaying portfolio items on a personal portfolio site, to products for sale on an e-commerce website.

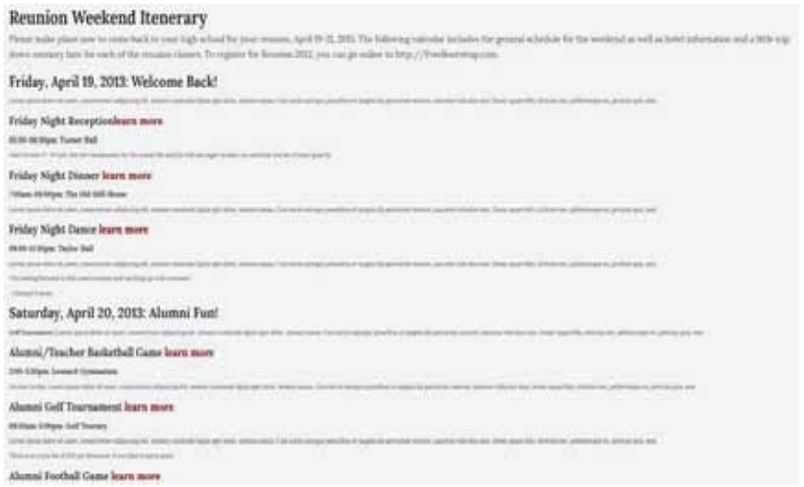
I've shown you a Thumbnail Gallery here and we added some descriptive text following each of those. You can certainly build a Thumbnail Gallery with smaller thumbnails, with images that link, and perhaps open a modal window or open in some kind of pop-up window that would show you a larger version of the picture such as a light box. All of that's possible with Bootstrap.

CHAPTER 3: STYLING

EXPLORING BASIC TYPOGRAPHY

Bootstrap comes with a number of built-in typographic styles which you can use in your site designs. There are a few in particular that you might find especially useful. This is the latest HTML document we have on our hands (I realize you need a microscope to see it on-page), which are in the example files, or you can download it at FreeBootstrap.com/resources:

This page with all the microscopic text represents calendar.html, and you see that in the browser the page is presently kind of boring looking:



We have a schedule that's divided very nicely into sections here. We have a Friday schedule, and we have a Saturday schedule. And it's easy to tell the hierarchy of the information, but still, it's pretty boring.

So let's add some styling to this.

You may notice that there's some default styling, just by virtue of being linked to the Bootstrap stylesheets. For example, the text is displayed in Helvetica and Arial, as opposed to being displayed in Times New Roman, which is the default if no styling is set (for most web browsers anyway). There's also some basic styling that's in place for the header. These "Learn more" items are links, which you can tell as you hover your cursor over them. They become underlined, and they aren't the standard blue. So there's some basic styling that's here but it's just not a lot of styling, and it's still a boring page. The markup will likely be familiar to you for the

most part. So let's see what we can do about improving this page inside of our text editor.

Open up calendar.html in your text editor and scroll on down to around line 20-- and line 20 is an introductory paragraph that summarizes what's going to be happening with this particular calendar-- it's just sort of a big summary of everything that will be going on, and begins with <p class="lead">.

One of the things that we can do to make this stick out a little bit is to assign it its own special class. Bootstrap allows this through what's called "**Lead,**" which gives your text a little more "oomph." So I've gone ahead and already given the <p> tag a class of "lead." If you take a look at what "lead" does as compared to the other paragraphs in <p> tags, you'll see that the leading paragraph has slightly larger text. It also has a little more spacing between the lines, and it stands out a little more than the rest.

Something else that we can do to this page, which is a very simple thing to do, would be to add some emphasis to text on the page. of course, is the HTML tag which many people think of as making things **bold**. But remember, it doesn't have to make things bold. Bold is only the way something looks; it doesn't have to do with the specific markup of that tag, which is to strongly emphasize something. There may be a fine line, but there is a technical distinction.

So what we'd like to do is to take these events like "A Night of Fun!" and "Day Activities," and let's emphasize those within the text by strongly emphasizing them with tags. As for the headmaster and athletes' names (Emmett Wright, Michael Jordan and Michael Vick), let's emphasize those as well with tags.

What you should now have with the changes we made should look something like this in your text editor:

```
<article class="span12">
  <h1>Reunion Weekend Itenerary</h1>
  <p class="lead">Please make plans now to come back to your high school for your reunion, April 19-21, 2013. The following calendar includes the g
  <h2>Friday, April 19, 2013: Welcome Back!</h2>
  <p><strong>A Night of Fun!</strong> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum
  <hgroup>
    <h3>Friday Night Reception <a href="#">learn more</a></h3>
    <h4>05:30-06:30pm: Turner Hall</h4>
  </hgroup>
  <p>Meet <em>Emmett H. Wright</em>, the new headmaster for the school He and his wife are eager to meet you and hear stories of years gone by.</p>
  <hgroup>
    <h3>Friday Night Dinner <a href="#">learn more</a></h3>
    <h4>7:00am-08:00pm: The Old Mill House</h4>
  </hgroup>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magn
  <hgroup>
    <h3>Friday Night Dance <a href="#">learn more</a></h3>
    <h4>08:00-11:30pm: Taylor Hall</h4>
  </hgroup>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magn
  <p>&quot;I&rsquo;m looking forward to this year&rsquo;s reunion and catching up with everyone.&quot;</p>
  <p>- Michael Tyson</p>
  <h2>Saturday, April 20, 2013: Alumni Fun!</h2>
  <p><strong>Day of Activities</strong> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.
  <hgroup>
    <h3>Alumni/Teacher Basketball Game <a href="#">learn more</a></h3>
    <h4>2:00-5:30pm: Leonard Gymnasium</h4>
  </hgroup>
  <p><em>Michael Jordan</em>, Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis
  <hgroup>
    <h3>Alumni Golf Tournament <a href="#">learn more</a></h3>
    <h4>09:30am-5:00pm: Golf Tourney</h4>
  </hgroup>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magn
  <p>There is an extra fee of $50 per foursome, if you plan to participate. </p>
  <hgroup>
    <h3>Alumni Football Game <a href="#">learn more</a></h3>
    <h4>2:30-4:30pm: Kenan Field</h4>
  </hgroup>
  <p><em>Michael Vick</em> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis nat
  <h3>Baker Hall</h3>
  <p>Look for these three new additions to the school in Baker Hall</p>
```

INCLUDING BLOCKQUOTES

In the middle of the page we've been working on, between the two days of events, there's a testimonial from Michael Tyson. It's easy to miss, because it's pretty much buried within all of the text that's going on around it on the page. Fortunately, Bootstrap has some built-in styling that you can apply to a quote like this to make text like that a little more noticeable.

Back to your text editor and we're going to scroll down the page to around line 38, and you'll see that we have the quote, "I'm looking forward to this year's reunion and catching up with everyone." That's a paragraph , followed by a paragraph with a person who said it, Michael Tyson, and we're going to go ahead and give this a little bit of styling. It's not really marked up correctly anyway, since this is a quote from someone, then it should really be contained within a <blockquote> tag. So we'll start with that. Give it a <blockquote> tag and indent the quote, indent the name of the person who said it, and </blockquote> will close that. Now save your file, File > Save, Ctrl+S or Command+S to save and then view it in the browser, scrolling down to see it, you'll see that just by putting it in the <blockquote> tag and marking it up correctly, the text is automatically larger. You'll also notice that we have a new grey border over on the side which also makes it stand out a little, so that's a nice detail as well.

The next thing that we want to do is de-emphasize the person's name. We want the quote to be the focus rather than the person's name, so the text should be a bit bigger, and the person's name should be a bit smaller. Simple.

A quick and easy way that we can do that is with a <small> tag. A <small> tag is a tag that's come back in HTML5, so let's go ahead and add that now. Right around the name of the person, we can put in a <small> tag and </small> on the other side of the person's name and, if we save that and we head back to the browser and refresh the page, you'll see that the person's name in fact becomes small, and it's also a lighter grey and scaled back in emphasis just a little. You'll also notice that it puts a character in front of that person's name, that long em dash (--) that's there. So the tilde that we had in front of the name previously when we typed in the text

really isn't necessary anymore, so we can get rid of that, and the styling for that quote is a little better.

Here are the markup and the results:

<blockquote> "I’m looking forward to this year’s reunion and catching up with everyone." <small>Michael Tyson</small> </blockquote>

Will yield:

> "I'm looking forward to this year's reunion and catching up with everyone."
> – Michael Tyson

One of the additional things that we can add to the <blockquote> tag would be the ability to float this quote left or right on the page. I mentioned "Floating" earlier which, as you may know, is a CSS manipulation, and a technique you should familiarize yourself with, if you aren't already HTMLDog.com is a good place for brushing up on your HTML and CSS, and Chris Coyier's CSS-Tricks.com is a font of CSS knowledge (and one of my favorite sites).

So in your text editor, in the <blockquote> tag go ahead and add a class of "pull-right." This will float your quote to the right of the

page up against the side. You can also add a class of “pull-left” to your blockquotes, and control their appearance that way as well

Remember, “pull-left” and “pull-right” are two classes that you can insert pretty much anywhere inside of a Bootstrap document that will cause that element to float left or right. This quote that we floated right obviously would need a bit more styling if we decided to use it that way here. Of course, it’s totally possible to do custom styling inside of Bootstrap. We’ll cover soon in an upcoming section in this chapter.

That block quote looks a little strange in this example over on the right, so I’m going to go back and pull that class out in my text editor. I just wanted you to know that it exists in case you ever happen to want to do that. It’s a nice built-in styling feature here inside of Bootstrap.

INCLUDING LISTS

Down at the bottom of the page where our calendar mentions three new additions to Baker Hall, we have a list. Right now it’s a simple single bulleted list and then under that we have a sentence that says to visit three new instructors who will be available, and then it lists the course they teach.

Currently, those are marked up as individual paragraphs, and that’s actually rather incorrect markup. It’s still a list of things; Miles Coxe is the instructor; French One and Two is his topic. They go together, but they’re not really individual list items. The best way to mark up particular lists of things like we have down at the bottom is really a **definition list**.

A definition list isn’t something that is typically utilized a whole lot inside of web pages right now, but they’re very useful. It’s got a two-part list, so there’s a definition term and a definition itself, so things may be paired together inside of a list. We need to mark that particular group of items up correctly, so let’s take a look at some styling that Bootstrap offers for dealing with lists.

Inside of your text editor, if you scroll on down to the bottom of the page, let’s take a look at that bulleted list.

```
58      <h3>Baker Hall</h3>
59      <p>Look for these three new additions to the school in Baker Hall:</p>
60      <ul>
61        <li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. </li>
62        <li>Maecenas nec odio et ante tincidunt tempus. Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. </li>
63        <li>Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet.</li>
64      </ul>
65      <p>Be sure to meet three new instructors, who also went to school here, and will be available to meet and greet on Saturday.</p>
66      <p>Miles Coxe</p>
67      <p>French 1 and 2</p>
68      <p>Ralph Smalley</p>
69      <p>American History</p>
70      <p>Ted Blaine</p>
71      <p>AP English</p>
72    </article>
```

As you can see, we can display those three lines of Lorem Ipsum right now with individual bullets, if we wish. Or we don’t have to display the bullets at all by taking the <ul> and giving it a class of “unstyled,” and just by adding that class of “unstyled” we’ll get rid of the bullets in the list.

However, let’s go ahead and mark this up correctly. Any time you see something like this, you should use a definition list. We start with wrapping the block of paragraphs with the <dl> tags, which, not surprisingly at this point, stands for **definition list**.

Then down underneath the end of that list, we’re going to wrap it up, so of course that </dl> is required.

Then we're going to have our first instructor here, Miles Coxe. We'll change his paragraph tag <p>to a <dt>, which stands for **Definition Term**. And French 1 and 2, we'll change to a <dd> which is the actual definition. And although we're not thinking of these as dictionary terms with literal "definitions," these are, again, related pieces of content that are tied together, and by using <dt> and <dd> we're showing the association between those two items. So now let's mark this all up as a definition list:

```
<dl>
<p>Be sure to meet three new instructors, who also went to school here, and will be available to meet and greet on Saturday.</p
<dt>Miles Coxe</dt>
<dd>French 1 and 2</dd>
<dt>Ralph Smalley</dt>
<dd>American History</dd>
<dt>Ted Blaine</dt>
<dd>AP English</dd>
</dl>
```

So if we save this again, File > Save, and put it into a browser, you'll notice that we have a very nice style here, where we have Miles, Ralph and Ted presented in bold, and then underneath them their course are slightly indented. It's pretty clear that French 1 & 2 goes with Miles Coxe, for example, so that's a very tight list.

Be sure to meet three new instructors, who also went to school here, and will be available to meet and greet on Saturday.

Miles Coxe
French 1 and 2
Ralph Smalley
American History
Ted Blaine
AP English

There's another style that Bootstrap offers that you may want to make use of here as well. If you go back to your text editor, and for the <dl>, simply add the class of "**dl-horizontal**" to it. What that'll do, as you may predict, is place the name (or <dt>) over on the left, and the course (or <dd>) over on the right, in horizontal rows.

Okay, so Bootstrap has a few nice treatments for lists. These are lists that are just your basic content lists that I've shown here. There are additional options that are available for navigation, which is often marked up as a list. We'll be covering those in the upcoming chapter on navigation.

STYLING BUTTONS

You may have noticed the "Learn more" link next to the names of the activities on this program. They aren't really styled at all. It would be a lot better if they looked a little more interesting, and certainly more compelling for the user to click on them.

Friday Night Reception learn more

05:30-06:30pm: Turner Hall

Meet Emmet W. Wright, the new headmaster for the school He and his wife are eager to meet you and hear stories of years gone by.

As you may have expected, Bootstrap comes with a number of styles that will make those links more interesting. In fact, we can make all of those links look exactly like buttons, if we want to, and we do. Let's apply some treatments to these "Learn

more" links to make them look more interesting, more clickable, and pop-off the page a little bit more than they currently do.

Back to the text editor and scroll on down to the first instance of these "Learn more"s, which should be around line 24:

<h3>Friday Night Reception Learn more</h3>.

You'll see that we have the <a> tag with a "Learn more" link inside of it. There are a number of things that we can do to give this a really nice effect. One thing that we could do is simply use that <small> tag which we just went over in a previous section, and you can just wrap the <small> tag around the link itself. If you try that and just save it, Ctrl+S or Command+S to Save, and Refresh this inside of your browser, you'll see that we've now made the text slightly smaller next to the names, which is good. It doesn't look like it's competing with the name of the event anymore as it did before. If you take a look at "Friday Night Reception," it looks complementary to the text, instead of text just running on down the line as it did.

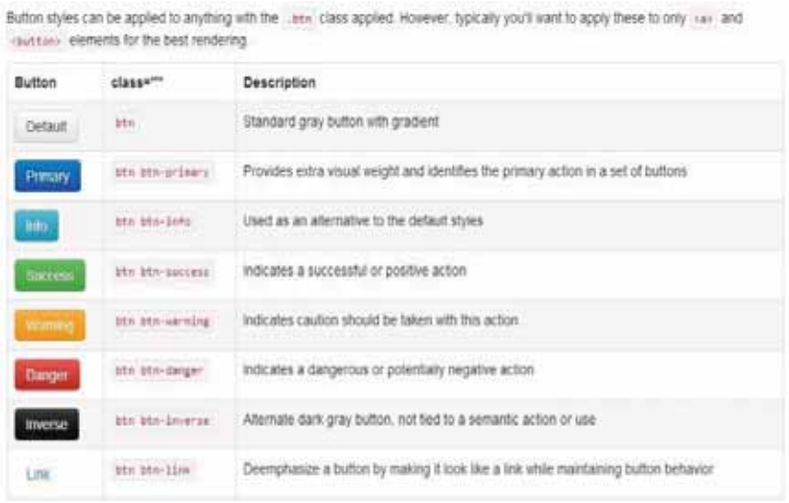
So this is a little better, but we can make it a lot better using Bootstrap's button classes. So run back to your text editor, and let's pull that <small> tag out. Instead, let's go ahead and apply some classes to style these links up. At the most basic, we can apply a class of "btn" for button inside of the <href> tag there, and save that. Take a look at it in the browser, and refresh one more time. You'll see that the "Learn more" link now actually looks like a button; it's squared, it has rounded corners, and it has the words

"Learn more" within it. When you hover your cursor over it, it looks as if the button is being slightly pressed. That's definitely an improvement over what we have for the other "Learn more" links on the page right now.

One of the other things you'll probably be interested in doing is adding some kind of color to the button. Those are available for you to apply as well, which we glossed over earlier in the book.

Back to your text editor of course, as an additional class that we can add on here, there is "btn-info." That will make this button a blue color, which is sort of the default styling for buttons on a Bootstrap web page. Go ahead and save that, and refresh the page inside of your browser, and you'll see that the button has become blue. We've seen this before in the Hero-unit Section, and when you mouse over it, you'll see that the color still changes a little bit. However we still have a few more options.

If you scroll on down the page to the next instance of this, and give this a class of “btn” and “btn-success,” this will make it a green button. And you can keep on going. Here are the colors again and their associated classes:



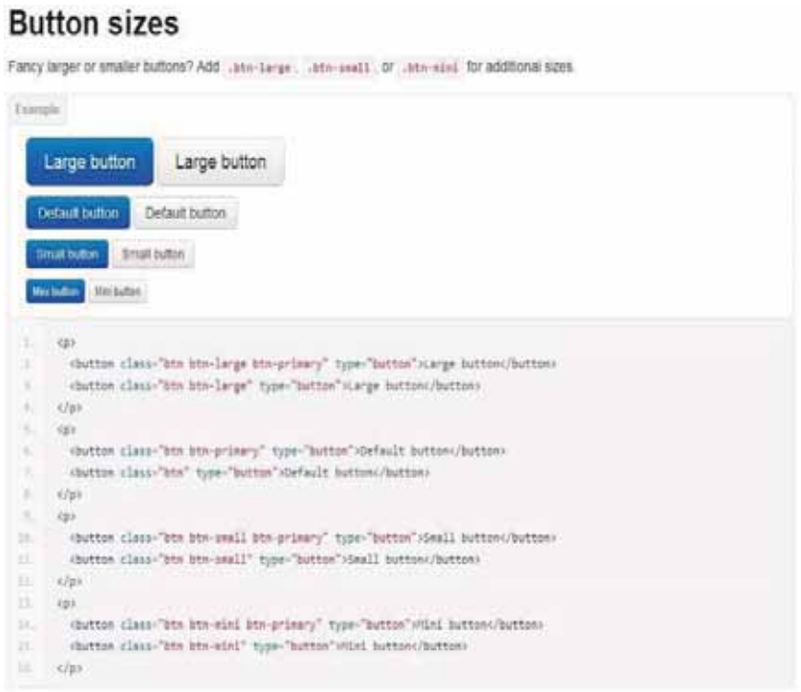
Button styles can be applied to anything with the `.btn` class applied. However, typically you'll want to apply these to only `<a>` and `<button>` elements for the best rendering.

Button	class=""	Description
Default	`btn`	Standard gray button with gradient
Primary	`btn btn-primary`	Provides extra visual weight and identifies the primary action in a set of buttons
Info	`btn btn-info`	Used as an alternative to the default styles
Success	`btn btn-success`	Indicates a successful or positive action
Warning	`btn btn-warning`	Indicates caution should be taken with this action
Danger	`btn btn-danger`	Indicates a dangerous or potentially negative action
Inverse	`btn btn-inverse`	Alternate dark gray button, not tied to a semantic action or use
Link	`btn btn-link`	Deemphasize a button by making it look like a link while maintaining button behavior

Notice the names of the classes that you have for these: “button-primary,” “button-info,” “button-success,” “button-warning,” “button-danger,” and “button-inverse.” None of these are tied to a color name, and that’s done intentionally so that you can override these colors with your own colors. You should think of these buttons in terms of their functionality, rather than color. The primary button styling that you’re going to have is an informational button; a green button indicating success, an orange button indicating a warning, and so on. That’s why these names exist, and that’s why there are so many of them inside of Bootstrap.

One final thing we can do with our buttons is we can change their size. Yes, back to your text editor again here for a moment, and let’s go back on up to where we have that first button.

There are different sizes of buttons that come with Bootstrap. We’ve been seeing the default size here, but there are three other sizes that are currently available as well.



Button sizes

Fancy larger or smaller buttons? Add `.btn-large`, `.btn-small`, or `.btn-mini` for additional sizes.

Example

Large button Large button

Default button Default button

Small button Small button

Mini button Mini button

```
<p>
  <button class="btn btn-large btn-primary" type="button">Large button</button>
  <button class="btn btn-large" type="button">Large button</button>
</p>
<p>
  <button class="btn btn-primary" type="button">Default button</button>
  <button class="btn" type="button">Default button</button>
</p>
<p>
  <button class="btn btn-small btn-primary" type="button">Small button</button>
  <button class="btn btn-small" type="button">Small button</button>
</p>
<p>
  <button class="btn btn-mini btn-primary" type="button">Mini button</button>
  <button class="btn btn-mini" type="button">Mini button</button>
</p>
```

If, as a third class, you add to this tag, “btn-large,” you’ll make a large button. You may remember we did that in the hero-unit in the previous chapter. Or if you insert “btn-small” you’ll

have a smaller button, and we can go even smaller using "btn-mini," which will make a very small button.

Go ahead and save those and take a look at it in the browser, and you'll see we have a "large" button, we have a "small" button, and we have a "mini" button. You can compare those with the orange button at the bottom, which is the standard size.

Although I'm sure the urge is strong, I recommend not overriding any of these colors just yet.

Doing that with CSS is straightforward in Bootstrap, but there's a way that I'd like for you to do that. Just a couple more sections, and we'll cover how to override the core CSS with your own code.

INCORPORATING IMAGES

So our page is looking better than when we started, but it's still lacking in a lot of color and much interest, and one of the things that we could do to make the page more vibrant and break up the text a little bit, of course, would be to add some images.

In addition to that, I'll show you some cool image treatments that ship with Bootstrap that are really fun. You can take a square image and apply a style to that, but we'll also make it appear circular. There's an effect that will give you rounded corners on your images, and there is also something called the "Polaroid" treatment, which will put a gray line around your image, as well as a white border, like a Polaroid photograph.

I've included three images for you to use, which I'll be using as well. You certainly don't have to use these images. However, make sure you specify the correct path with the right file name. The file names for the images I've provided are sunflowers.png, stripes.png and polaroid.png.

The first place we'll add an image is right after our <h3>, "Welcome Back!" In fact, right inside of the Lorem Ipsum paragraph describing "Welcome Back!" we'll go ahead and add the image tag <img src="">. Also be sure to stick an alt tag in there and a description, if your image is to be more than purely decorative, so screen readers can read it. I'm leaving it out on our example.

```
15  <body>
16  <div class="container-fluid">
17    <div class="row-fluid">
18      <article class="span12">
19        <h1>Reunion Weekend Itenerary</h1>
20        <p class="lead">Please make plans now to come back to your high school for your reunion, April 19-21,
21        <h2>Friday, April 19, 2013: Welcome Back!</h2>
22        <p><img src="img/stripes.png" alt="" /><strong>A Night of Fun!</strong> Lorem ipsum dolor sit amet, c
23        <hgroup>
```

Then, after our block quote and just following the <h2> containing "Saturday, April 20, 2013: Alumni Fun!" let's insert another picture in that paragraph, which will be sunflowers.png.

And then finally, let's scroll down the page to just before "Look for these three new additions to the school in Baker Hall:" directly after the "Baker Hall"<h3>, I'm going to go ahead and add an <img src=""> tag right here inside of this paragraph, and this particular image that I'm going to put in is img/polaroid.png, and once again, a blank <alt> tag.

Okay, so we've just added three images to our document: stripes, sunflowers and a Polaroid (guess which effect it gets). If we go ahead and save the file and pop into our browser and take a look at our page and refresh, you'll see that we have these images, but they're vertical and just sort of stacked there, pushing text out of the way, and unattractive. Let's add those additional styles that we discussed!

Switch back to your text editor and scroll back on up the page to the first image, so we can add some classes do this. The big class

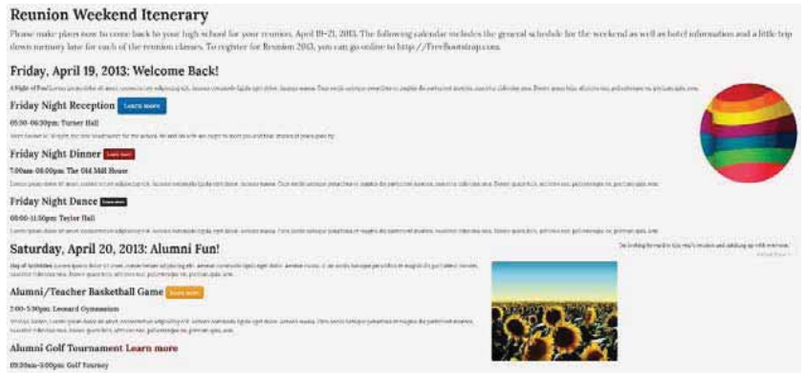
we'll add to that first image of the stripes is "img-circle," which will give us a very cool circular image effect, with nice clean edges.

So if you just do that much and save your page and take a look at it in your browser, you'll see that, lo and behold, your squared-off image has now become a circle with a nice clean line, and it looks great. Let's also float this on over to the right side of the page. It's not really working over there on the left side.

So, inside of your text editor, let's add a second class to "img-circle". It's our old friend "pull-right," which will float the image over to the right side of the page. Go ahead and save that again, take a look at it in a refreshed browser, and you'll see we have a nice round image pushed to the right side of the page, and it looks much better. So that's just one of the treatments that are available to you in Bootstrap. "img-circle" will give you cool circular images, and remember: "pull-left" and "pull-right" will do exactly that to the object/class you've targeted.

Now if we go back to our text editor again, and we scroll on down the page, back to our second image here with the image of the sunflowers, let's go ahead and add another class, and let's give this a class of "img-rounded," which will give it rounded corners, and in addition to that, while we're here, we'll also do the same thing again with a "pull-right" to make it float to the right side of the page.

Save this again, and if you refresh it inside of your browser, you'll see that we have our image of our sunflowers over there floating out to the right side of the page. You'll see that the image corners on this are also rounded, which is a nice effect.



Lastly, we're going to give this Polaroid picture down there a treatment, so scroll on down in your HTML document to the Polaroid, and for this one we'll go ahead and add a class for it, and the class is, brace yourself… "img-polaroid," and once again, let's add a "pull-right" to that particular class and go ahead and Save that, and we'll Refresh that inside of the browser, and you'll see that we now have our Polaroid picture over here on the right side

of the page. You see that we have a nice gray border going around that with some white in between. If we had a different color background, that white in between the light gray line and the picture would stay white, which helps explain the name "Polaroid."

So there we go; three simple but effective image treatments here inside of Bootstrap. You can make use of these very simply with just a single class of the Polaroid treatment, the Rounded corner treatment, and of course, the Circle treatment up there on the top.

INCORPORATING ICONS

Bootstrap also comes with a full set of icons that have been contributed by Glyphicons, which are updated regularly. There are currently 140 glyphicons available for free, but it you want to scale up, there are options to buy the license from Glyphicons to use 420 Glyphicons, plus 160 smaller ones, all vector. They are nice, but if you want a variety of free vector icons, I highly recommend Dave Gandy's Font-Awesome. To get a full listing of available Glyphicons that are included with Bootstrap, you can visit getbootstrap.com and click on the link on the top for "Base CSS" and then click on the link on the left side for icons by Glyphicons. You'll see that you have a whole list of 140 choices of all different shapes and sizes, actions and all types of things. They come in both black and white.

With the rising popularity of Bootstrap, you can find some decent third-party resources online. FuelUX.com is a great site to find extra custom JavaScript components for your Bootstrap site, and BootstrapCDN.com will help you tremendously to get your Bootstrap site up and running quickly. Font-Awesome is a growing repository of useful font icons that are made and optimized especially for Bootstrap. There are presently 240 icons there. The site is definitely worth bookmarking and utilizing, (if not following outright on GitHub.) The benefits of using font icons are great, and they are certainly worth taking a close look at. For example, you can increase or decrease their size with no loss of quality or sharpness, their colors may be modified, even with a gradient on some browsers, and their file size is tiny, since they're just a typeset character. Their use is currently a trend in web design, so you have probably seen them used on nav menus and elsewhere to enhance buttons and simply illustrate words on a website. The

usage of them is similar across the board, so for our purposes of demonstration here, we'll focus on Glyphicons, since that's what currently comes with Bootstrap.

I try to copy/paste as much as I can to reduce the likelihood of typographical errors on my part, so if you're the same you can point your browser to the Glyphicons part of Bootstrap's website and grab the calendar icon. Simply highlight the words "icon-calendar" and hit Ctrl+C or Command+C to copy that.

If you're more confident than I with your typing skills, you can just type icon-calendar, now that you know which we'll use.

We're going to incorporate that into the web page by putting an icon in front of the text. So, what you'll need to do now is jump back into your text editor, and scroll on down to the first instance of a scheduled event, which is 05:30-06:30pm: Turner Hall, and that's where I'd like to go ahead and put this particular calendar icon.

The way you do this is by adding a tag with a class of whatever class is associated with that particular icon in the list. Here's the current collection with classes:

You're going to follow that immediately with a closing , so you can see here the <span class "icon-calendar"> is the name of the class which you see on the Glyphicons web page. The names on the right are all of the names of the classes to get those particular icons to display.

Go ahead and save--Ctrl+S or Command+S to Save-and if you put this page into your browser, you'll see that our icon shows up right there on the page in front of the time.

So that's a nice little feature that Bootstrap ships with. But what if you don't want black? What if you want white instead? That's possible to do as well, as you'll see inside the documentation at the bottom of the page, you can use "icon-white" as a part of that tag which will give you the icon as well, but in the white format instead of in the darker format.

So, if we scroll on down to the next instance here, let's put in that tag again of "icon-calendar" "icon-white" which will give you a white version of the icon followed by a .

Once again (I told you we'd be doing this a lot), go ahead and Save this and refresh your browser, and you'll see the icon is there. It's sort of hard to see, since it's a white icon on a white background, but you can kind of see it a little bit, because the edges of the icon have been treated to blend into a dark background.

So, that's the basic way that you would go about placing icons into your document. You can go ahead and repeat this process for the rest of the page to continue incorporating icons--as many as you like. These icons are being pulled from is inside of your file structure. Inside of your image folder, you'll notice that there are a couple of these images: "glyphicons-halflings," as well as a white version.

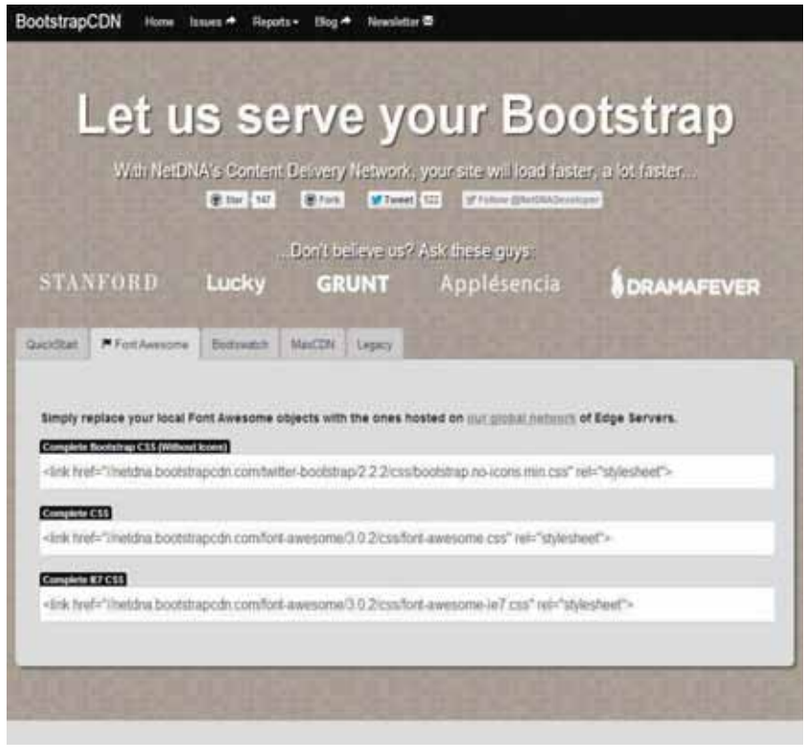
In fact, what you have here is an image sprite, so there are actually 140 tiny images in one PNG file, which is a really great way of handling images for page load considerations. Sprites are often used for buttons, and sprite generators can easily be found online. Rather than having to pull down 140 individual images, in this particular case, we're only referencing a single image in that .PNG and then, using CSS, we target only the images that we want shown on that page.

The drawback to doing it this way is that since these are image sprites and they're rasterized small images, as opposed to vector or .eps, and you're not able to blow these up and use them as large-format pictures within your document. Also, even though the sprite file size is relatively small, you still are dealing with a larger file than you would if you only needed a few of the icons. Page-load speed is important, not only for the user

experience, but it's becoming more of a factor in SEO, and if you have an e-commerce site, you could even be losing customers from a slow website.

There is a solution, which I personally prefer to Glyphicons, and that's using Font-Awesome, which I've mentioned already. Font-Awesome has taken care of the most common complaints of Glyphicons by having their icons in vector format and they're optimized for Bootstrap and for running a lean website. Font-Awesome's icon fonts are used in the same manner as our Glyphicons, except with using an <i> tag, yet are only the file size of a single character. Also, having the icons in vector format makes them scalable without any loss of quality. Being font-icons we can dictate their colors, size and even animate them. Font-Awesome also offers the ability to subset your icons to make your file size even smaller. They're also screen-reader compatible, support IE7, and best of all, possibly, is that they're free.

The way that I recommend using Font-Awesome fonts if you want, instead of Glyphicons, is by going to Bootstrapcdn.com and clicking on the Font-Awesome tab at the top.



Next, copy and paste the Complete CSS, and Complete Bootstrap CSS (Without Icons) and paste them in place in your header, as you would a non-Font-Awesome stylesheet. We now have a file that has no Glyphicons in order to avoid an icon war on our screen.

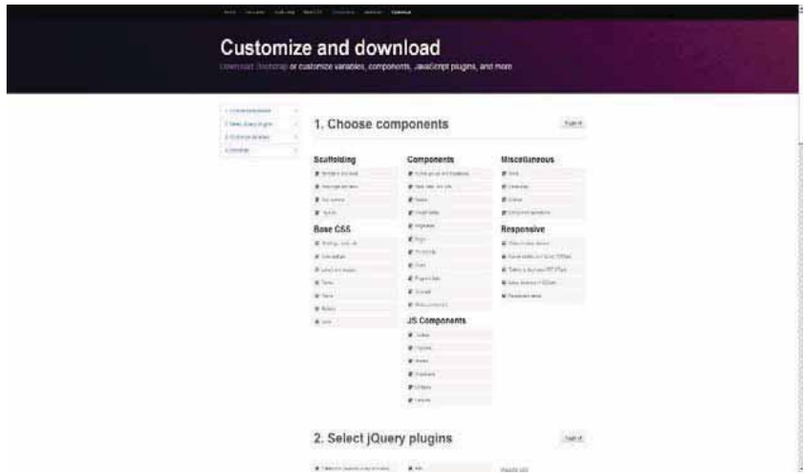
Insert your tag `<i class="icon-twitter"></i>`, substituting "icon-twitter" with your chosen Font-Awesome icon anywhere you'd like an icon in your HTML, style it up with a CSS class, and you have icon magic. Since you've targeted it with a class, you can change its appearance easily.

OVERRIDING CORE BOOTSTRAP CSS

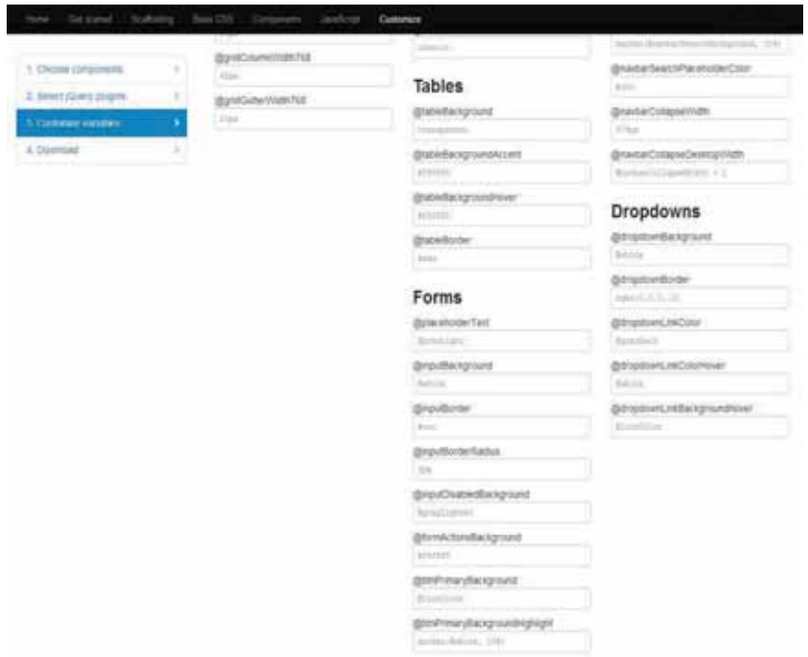
Finally, the part we've all been waiting for: how do you override the core CSS in Bootstrap?

Well, there are two possible ways that you can do this.

If you go to the getbootstrap.com website, and click on the "Customize" link, you'll be taken to the page where you can customize Bootstrap to do exactly what you need it to do, and nothing more. You can choose exactly which components you want to include and exclude, and download customized files. If you take a look at the customization page, you may recognize the Scaffolding which we reviewed back in chapter one, but much of the rest of this we haven't talked about yet, We'll cover all this over the next couple of sections, but quickly, these are JavaScript-driven widgets. Some of the styling that's in CSS can be chosen for inclusion (or not), you can choose to include specific plug-ins that we rely on jQuery to handle, and you can customize these variables as well.



Bootstrap makes use of the LESS (LEaner cSS) framework, which is a CSS preprocessor, and makes your CSS code leaner and meaner, and you can enter different values for these variables. The easiest one to understand is under "Colors." For example, you can decide on the exact shade of blue you'd like to have where blue occurs inside of your document; what exact shade of green, and so forth. You can customize all these variables just by typing in your own values and customize them easily with Bootstrap's web app. You simply download your customized file by clicking on the giant button at the bottom of the page, and that will download exactly what it is that you want to have for your website.



However, there are some major downsides to customizing your Bootstrap Installation this way, particularly where the CSS is concerned. Firstly, Bootstrap releases new versions frequently (version 3.0 is under way). We're currently on v. 2.3.0, but that will of course change in the future, which usually is a good thing to be happy about. Some people find a version they like and stick with it. But if you want to update your website to the latest version of Bootstrap,

right now what you would do is simply download the latest version, replace the existing CSS and JavaScript files with the new versions that you would download from the Bootstrap site. That's great until you realize that any customization that you've done to bootstrap.css or bootstrap-responsive.css will instantly be lost, because you'll overwrite those files with the new ones that you've downloaded. Hmmm…What to do?

So now, as part of that process, you'll either need to keep track of where all those customizations are and retype them, or you'll have to keep track of what they are and re-enter them when you upgrade your version of Bootstrap. Personally, that's a little too involved for me. I'd never remember what I've changed and what I haven't changed.

I think a far better approach is to create a custom stylesheet and put any changes that you're going to include there. Anything you're going to override or any additional styling that you're going to include, put it all in that stylesheet, and that will make your website much more easy to maintain. It'll be easy to upgrade to the latest version of Bootstrap, because then you'll just copy those new files in over the old files and your customizations are all still intact.

So, let me show you how to do that process now. Back in your text editor, let's go ahead and create a new document, (File > New,) and this time it's going to be a stylesheet. Create a blank stylesheet document and be sure to save it with the extension .css. A common file name for this is "custom.css." Since this is just for demonstrative purposes, we'll just make a really quick change so you can get the idea. Let's go ahead and change the body background color to an unforgettable shade of green. So, as

with CSS, we'll enter a hex background-color of #bada55, and save that, File > Save. So let's go to your Bootstrap folder, then into your CSS folder, and save this as "**custom.css**," so that's now your custom stylesheet. Sweet!

Now the next, and very important, thing we have to do is attach it to the document we've been working on, so head over to calendar.html, which is the document we've been working on in this chapter. Underneath the existing stylesheets we reference, just above our HTML5 shim, add the following (assuming you saved your custom.css file in your Bootstrap css folder:

```
<link href="css/custom.css" rel="stylesheet" type="text/css">
```

Again (this is important) you want it to occur last in the chain of stylesheets, because remember, anything that you put in your custom.css stylesheet is supposed to override what comes before it. So if you put this custom.css stylesheet somewhere further up the line, closer to the <title> tag in the code that we're working with here, for example, it may not necessarily override the default styling from Bootstrap, thanks to the cascading aspect of CSS.

Okay, once you've got that in place, File > Save, we can go ahead and take a look at this in the browser, and if you Refresh the page that you're working on, you can now see that we have a serious shade of green behind our text, and the icons that we inserted in the previous lesson. You can see that we have a nice white calendar icon. We have a white border associated with our block quote, and if we scroll on down the page a little more, take a look at the image that had the "image-polaroid" class applied to it. You can now see that it has a nice white stripe around in addition to that thin light gray stripe that goes around it as well.

So, that's how you add custom styles to your document. Pretty easy, and very valuable info as it frees Bootstrap to become any type of site you'd like.

Just make sure you put your customizations in a separate stylesheet that you create and name. I usually call my file "custom.css," because it's pretty clear what that is, and then make sure you link it to your document after any other style declarations that are on your page so that it'll override things correctly.

CHAPTER 4: NAVIGATION SYSTEMS

ADDING BREADCRUMBS

The next segment of the Bootstrap framework that we'll examine includes the various methods of creating website navigation systems. Nav menus have historically been a challenge for some designers and developers, but they're a snap with Bootstrap.

Bootstrap offers many different widgets for our navigation, including tabs, pills, buttons, dropdowns, lists, and full navigation bars, as well as helper systems like breadcrumbs and pagination, which we'll learn here as well.

As you'll see in the next chapter, in many cases these menus can be combined with JavaScript to include Dropdowns. But for now, let's start with taking a look at breadcrumbs. For our purposes, "breadcrumbs" are a trail of links that you might see at the top or bottom of a web page that indicates where you are in the website's hierarchy and structure. These links can be a path, location or attribute.

If you've shopped online, you've probably run into breadcrumbs before. They look like this on Apple's site, but generally are a horizontal "trail" of web pages you've traversed to get to where you are on the site.

This indicates that you're on the page where you configure your new 15 inch MacBook Pro (with Retina), and before that you were on the page about MacBook Pros and before that you were on a page about Macs and just before that, the home page. We can create this exact same navigation system inside of Bootstrap.

Bootstrap comes with a system for styling breadcrumbs, and we can take a look at that now. If you'll return to your text editor, and you open up the breadcrumbs.html document that's inside of your exercise files, you'll see that this is without any styling.

If you view this in your web browser right now, you'll see that it's just a bulleted list with some links included in some of the s, as well as a separator which is that small double-angled bracket. So now we need to add a little bit of styling to this and a little bit of extra markup to make this a presentable breadcrumb.

First of all, since we have this displayed on a web page, we should utilize the grid system to lay this out. If you choose to do that in

these examples, which is what we're about to do, you would start with a with a class of "container-fluid," as we've done a few times before, with a closing , of course. And you can follow that with the call to your class of "row-fluid." And then the last part of this would be what's inside of that row. This is how you'll begin most of your web pages.

At this point we'd usually incorporate some kind of tag that would have the class inside of it, which would indicate how many rows this particular aspect of the website will span, and we want this to span to all 12 columns. The question is what kind of markup should we use to do that? There's actually some debate about the best kind of markup for breadcrumbs going on in online communities. My personal feeling on it, for whatever that's worth, is that the best tag to use is the <nav> tag, since I view breadcrumbs as a type of navigation around a site, albeit linear. Some people disagree with this and say that the <nav> tag should be used only in cases of major navigational elements, such as the main navigation on a web page, and nothing more. So, what our argument would be is that we're going to use the <nav> tag to mark up elements on our web page that are certainly related to navigation.

So we're going to go ahead and use the <nav> tag here to indicate this is where our breadcrumb will be going. We'll also put in a <nav> tag with a class of "span12" because we'd like for this to span the entire page. And of course, we'll close that down there at the bottom.

```
</head>

<body>
<div class="container-fluid">
    <div class="row-fluid">
        <nav class="span12">
            <ul>
             <li><a href="http://www.freebootstrap.com/" title="FreeBootstrap.com">FreeBootstrap.com </a>» </li>
             <li><a href="http://www.freebootstrap.com/home/authors.aspx" title="Authors">authors </a>» </li>
             <li>Michael B. Musgrove </li>
            </ul>
        </nav>
    </div>
</div>

<script src="http://code.jquery.com/jquery-latest.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

You may need to adjust your HTML and put in a few tabs (or spaces, whichever is your preference), just so this code is a little easier to manage So now that we've got that set up, you may realize that we've actually done very little with our list at this point. All we've done is insert some additional markup to structure our page, and this isn't any particularly special bulleted list at this point in time. It's just a plain bulleted list. So, if we just save this page right now and we take a look at it again inside of the browser, you'll see that actually nothing has really changed a whole lot. You may have seen it jump a little bit over from the left column; and that has to do with the grid system and its margins and spacing.

So now let's add the actual markup that makes this a breadcrumb inside of Bootstrap. The quickest and simplest way to do that is right inside of the <ul> tag. We'll add a new class, and this will be a class of "breadcrumb" (singular; 1 crumb. Make sure not to use "breadcrumb**s**," plural here, which is an easy mistake.)

The second thing that we're going to add to make this a breadcrumb trail is we're going to indicate which item is active. So

on the previous Apple.com example, we were on the "Configure" page. So "Configure" would be the active aspect of the breadcrumb indicating where you are. "Active" is a class that is used all over Bootstrap to indicate where you are. We'll be using it many times in this chapter on navigation.

So, let's add a class right on this last <li> of "active," and that will indicate where we are in the hierarchy. The last thing that we're going to do is create our particular spacers between the breadcrumbs, and we can do that right in front of the little double-arrows.

```
</head>

<body>
<div class="container-fluid">
    <div class="row-fluid">
        <nav class="span12">
            <ul class="breadcrumb">
             <li><a href="http://www.freebootstrap.com/" title="FreeBootstrap.com">FreeBootstrap.com </a><span class="divider">»</span> </li>
             <li><a href="http://www.freebootstrap.com/home/authors.aspx" title="Authors">authors </a><span class="divider">»</span> </li>
             <li class="active">Michael B. Musgrove </li>
            </ul>
        </nav>
    </div>
</div>
```

As you can see above, near the end of the first two links, we'll add spans with classes of "divider" wrapped around the double arrows, and close them with a . That indicates that the little character that we're using here, which is that double arrow, is what's going to separate our breadcrumbs. Of course, you could use any character that you want for that purpose. Go ahead and save what you have, Ctrl or Command+S to Save, and let's take a look at it again in the browser. You'll notice that it now looks very different.

Now we have a decent-looking breadcrumb trail. It's shaded slightly gray, and has very nice spacing to it. You can see that the character in between the breadcrumbs is nicely styled as well so that you can see that there's a separator, but it's not dominating the look of this particular breadcrumb. There's also another effect that we could give to this if we wish.

In one of the previous sections, we reviewed the Glyphicons that come with Bootstrap, and we could use one of those as our divider, which could really enhance a website's nav system, and the user's experience.

So, what we do if we want to change out the little characters for an icon instead, is inside of the "divider" class, just add "icon-arrow-right" (for example.) That will give us a little arrow icon instead of the HTML characters that we were using here. Be sure to get rid of the HTML characters that you were using, if you choose to do that. Let's save it again, and we'll take a look at it inside of the browser. Be sure to refresh the page.

Gives us this:

So, you see here we now have our arrow icon that's separating the items in our breadcrumb. And remember, of course, this is a dark icon. If you prefer to use the white, remember just like we reviewed in the Glyphicon section, you can simply change the class to "icon-white" and that will load the inverse image. Of course, it actually looks better dark on a light background as we have here. But if you had been playing around with styling here and you changed some of the colors, white could be a better choice.

So, now you've seen how to incorporate a breadcrumb into your web page. It's very simple; we just add a few classes to style up our breadcrumb. You can use all kinds of different HTML characters as separators in between the navigational elements inside of your breadcrumb, or of course, you could use one of the Glyphicons that ship with Bootstrap or an icon from FontAwesome.

ADDING PAGINATION FOR PAGING THROUGH CONTENT

Occasionally at the bottom of the page, there may be a lot of blog posts. Pages listing search results or a page that displays a bunch of products usually have a list of additional pages listed at

the bottom. (Think about the bottom of a Google Search Results page.)

If you head to Google's search engine and do a Search for FreeBootstrap.com, you'll see that we get a list of all kinds of stuff here that pertains to Bootstrap, and then down at the bottom, we have this long Goooooogle at the bottom which has a series of numbers plus the Next button.

If we go to the "Next" page of results and scroll down to the bottom, again, we have a "Previous" and "Next" button down there, as well as the numbers.

This is called "Pagination;" specifically the numbers that are there in the middle, as far as Bootstrap is concerned. The "Next" and "Previous" buttons are treated slightly differently; they're called "**pager**," and those are coming up in the next section.

So let's take a look at how we might include something like those numbers at the bottom of the screen in our Google Search Results. Inside of your text editor, you should still have a bulleted list. It's wrapped inside a <nav> because, as we covered, this is our navigation that helps us move around within the web site.


```
<nav>
  <ul>
    <li><a href="#">&lt;&lt;</a></li>
    <li><a href="#">1</a></li>
    <li><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
    <li><a href="#">5</a></li>
    <li><a href="#">&gt;&gt;</a></li>
  </ul>
</nav>
```

If we view this right now inside of the browser, you won't see a whole lot that was incredibly interesting. It's a bulleted list with some numbers, and some characters that point you in either direction. So how can we spice this up?

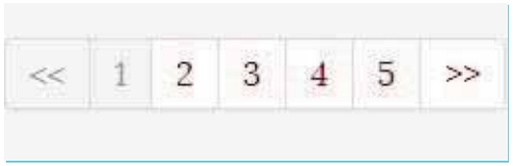
Well, Bootstrap has some styles for dealing with these. If we go to our <nav> and add a class of "pagination," that would be a great way to start by lining our numbers up horizontally across the page. If you Save that and preview that inside of the browser, you'll see that you have a series of numbers that are now in boxes, stretched out on the page.

Now, if you noticed on the Google Results Page, when we landed on one of the resulting pages, there was an indicator showing us exactly which page we were on. If we were on Page

1, for example, there would be nothing before that, so the number 1 might be grayed out, and so might the double caret on the left side of the Pagination. We couldn't move beyond Page 1 in that direction.

Bootstrap also has a way of doing that.

So in your text editor, if you add to the first <li> a class of "disabled" (in other words, grey this out, because there's nothing that comes before the Search Results), and for the number 1, if we add a class of "active," in another words, this is the current page of results, there will be an indication that we're on page 1, and that there's nothing before this.



So, once again, if we Save this and Refresh this in the browser, you'll see that it looks like your double caret is grayed out, and when you hover your cursor over it, it looks like it's no longer clickable. Likewise for the number 1.

Now, it's true that it looks like the double caret and the number 1 are not clickable, and the UI would indicate that they're not clickable. CSS has been used to change the appearance of the mouse cursor to make sure that it doesn't look like a pointing hand, the way it does if you roll over number 2. However, rest assured that the number 1 and the double caret are still very much clickable items. If you want to make sure that they're truly disabled, you may want to look utilizing at some kind of JavaScript voodoo. Also, you could use a content management system like

WordPress or some type of programming language which will remove the links so that they aren't actually there at all.

So just be careful about that. Just because you use a class of "disabled" doesn't necessarily mean that there's no link that exists on an item anymore. If you take a quick look at our HTML, there's definitely a link that appears there, and it's still clickable, even though it doesn't look like it is. May not be a big deal to you, but it's a good thing of which to be aware when you're creating a site.

So one final thing to show you. If you're worried about design, which you should be, you're probably wondering how you can center this pagination on your page. You're may think we need to do that with a custom style, but there's no need to do that. Bootstrap comes with the ability to easily center pagination on the page.

What you'll need to do is inside of the <nav> class of "pagination," simply add the class of "pagination-centered" and if you Save that and Refresh that inside of your browser, you'll see that your pagination is now centered on the page. The option also exists for right-aligned pagination. If you change the class to "pagination-right" you can make right-aligned pagination as well. It doesn't get much easier than that.

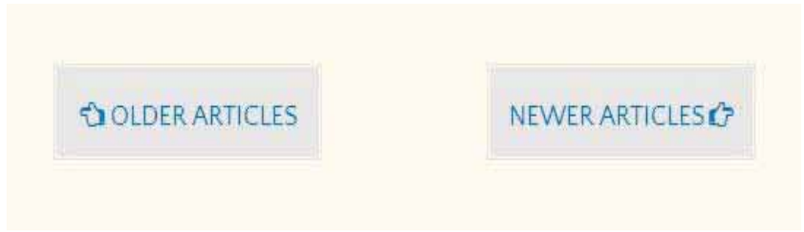
So that's the story behind pagination and treatments for it. Remember, pagination is usually a result of a content management system (CMS) or some kind of database-driven application for the bottom of a web page to tab through items. I'm showing you examples inside of your text editor, so if you were to use this kind of pagination in your websites, you would, of course, need to hand-code all of your links to separate HTML documents. That takes an awful lot of time.

Pagination is really a widget that's best linked to something which has a database behind it where the pagination is generated from the database. Therefore, it's not likely that's something that you would use it just on pages of a static HTML website. You're most likely going to use it much more often in something with the database behind it, such as blog posts on a content management system like WordPress.

ADDING A PAGE FOR NEXT AND PREVIOUS LINKS

One of the navigational elements that many blogs use, including the FreeBootstrap.com blog is down at the very bottom of the web page. If you scroll all the way down to the bottom of the page, you'll notice there are 2 buttons that says "Older Articles" and "Newer Articles."

This is a very common type of navigation that you see in blogs, so if you click the link that says "Older Articles," and go all the way down to the bottom of the page, again, you'll notice that it now it says "Older Articles" and "Newer Articles."

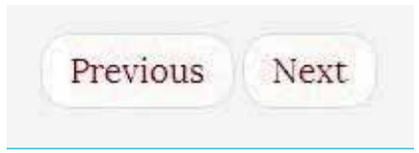


This type of navigation lets you page back and forth between parts of the blog just by clicking the links. We can go back a page in time, or go forward a page in time. This is called "**Pager"** inside of Bootstrap. A "Pager" allows you to style these links for "next" and "previous" or "older" and "newer" like you see at the bottom of these pages.

So, back to the trusty text editor and you can see this in action with really simple markup; this is just a two-bullet list:

```
<ul>
    <li><a href="#">Previous</a></li>
    <li><a href="#">Next</a></li>
</ul>
```

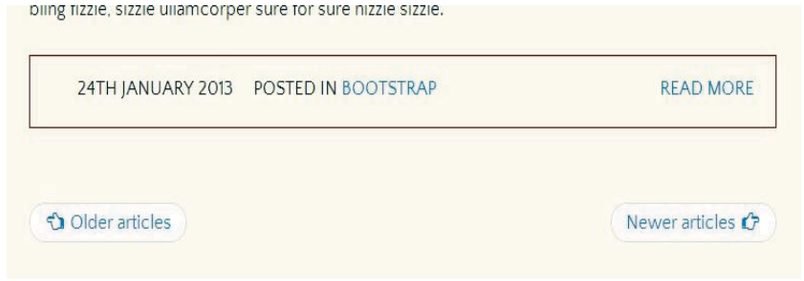
This simple code is just a <ul> with the two bullets that say "Previous" and "Next." Styling this is just as easy, so we're going to start by simply adding up on the top a class of "pager" inside of the <ul> tag. That will give us the default styling for our "previous" and "next" buttons.



If you go ahead and save that document, and preview it inside of your browser, you'll see now that we have two links, just as above. You can hover over them and see that they have a very nice hover state as well. So that's great!

But you may be wondering, "what if I want to spread these out so that they're at the edges of whatever area I've included these particular buttons?" So, for example, at the bottom of the

freebootstrap.com blog we've pushed "Older" and "Newer" Entries to the very edges of the blog there.



What if you want to push those two buttons apart as well? Well, that's completely possible. If you go back into your text editor again, we'll add two classes. The first <li> for "Previous," we're going to add a class of "previous." Pretty easy. And for the next button labeled "Next," we just add a class of "next."

```
 <ul class="pager">
   <li class="previous"><a href="#"> Older
articles</a></li>
   <li class="next"><a
href="#">Newer articles</a>
</li>
 </ul>
```


So just by doing that, this will push these two buttons to the very edges of the web page, and "Previous" would indicate the side where you want it to have "older" or further back in time, the left side of the page, and the one where it says "Next," you want to push to the right side of the page, or going into the future. You may notice I've used some of Font-Awesome's pointing hand icon fonts as well, which is simple to do. The above code doesn't include those, but it's just a matter of inserting the <i> attribute.

So if we refresh our web page, there we go. Now you see that we've pushed these all the way to the edges of the page. Since we don't have any of the Scaffolding included inside of this HTML document, they're going to be pushed all the way to the edges of the window. Of course, these will be pushed to the edges of whatever its containing element is, if you had some additional markup here. So if you have these in the bottom of some kind of <div,> let's say, with a span4, they'll be pushed to the edges of that.

In other words, about a third of the screen width, instead of the entire screen width, as you see here.

So that's great, but what if there are no previous entries? What if you would like to have that "Previous" button sort of grayed out? Well, just as saw before in the last section, you can easily add a class of "disabled" to the class of "previous." "Disabled" will gray that button out, just as before. So if you Save that one more time and Refresh your web page, you'll see now that the "previous" button is grayed out, there's no hover state, and the arrow stays when you roll your mouse over it, implying that this is not a clickable button.

Remember, however, that as we learned in the previous section, it actually is still a clickable button. We've just used CSS to make it

appear as if it isn't. If you really don't want this button to be clickable, you're going to have to find some way of getting rid of that link, or using more JavaScript tricks, or using some kind of coding magic to make this truly a dead button. The next button, of course, is going to remain clickable.

Alright, so this is some very simple markup for including "Next" and "Previous" or "Older" and "Newer" buttons inside of your web pages. If you want to flip forward a page or back in time, they're a commonly-used solution, and often found at the bottom of blogs and product pages of online catalogues. And you can combine this with a variety of classes to give this some nice styling and some great functionality.

USING TABS AND PILLS NAVIGATION

Over the next few sections, we're going to take a look at the humble but very versatile bulleted list and a number of different ways that we can change the classes to give our list look all sorts of different effects inside of Bootstrap.

We'll first take a look at pills and tabs. You're probably already familiar with tabs, and pills are sort of more brick-looking buttons and not-so-much pills, to me at least. They have rounded corners and you can also use them to make awesome-looking navigation bars. They look just like this:

So what we have right now in our default markup is pretty boring. If we quickly look at this inside of our browser, you'll

see that what you've got is just your basic bulleted list; very functional, but hardly stylish:

```
<li><a href="#">Home</a> </li>
<li><a href="#">About</a></li>
<li><a href="#">Blog</a></li>
<li><a href="#">Contact</a></li>
<li><a href="#">Store</a></li>
```

So let's begin by adding some classes to see how we can make this look a little better or, at least, a little different. Inside of your text editor, the very first step is to add a class of "nav" to the <ul>. This is the most basic markup. Just by adding this <nav> you've indicated that this is some navigation and some basic elements are about to be included here.

If you save and refresh this new code, you'll see that the margin and padding associated with <ul> have disappeared, and that the bullets are gone, and we're now crammed over into the side of the web page. The reason we're leaning up against the side of the web page is because we're not making use of Bootstrap's Scaffolding system yet. We've kept our markup very simple; the <nav> tag is located inside of the <body> tag, and it's being pushed all the way out to the very edges of the website's body. If we had this <nav> tag located somewhere else inside of the Bootstrap Scaffolding, this navigation wouldn't look like it's being pushed over into the edge.

But as you roll your mouse over these, you can see that there's a nice hover state that now occurs. The reason that hover state travels across the page is because that's how wide the containing element for these particular links is. If you employ the Scaffolding system, of course, this could be much narrower, depending on which of the span classes you choose to put your navigation list in.

Alright, so that's it! The most basic first steps of styling. Now let's add something to spice it up. If we add a class of "nav-tabs," that will give us more of a "tabbed" appearance. It'll cause our links to become horizontal instead of stacked as well. If we refresh the page in the browser, you'll see that has in fact happened. We have a series of links with a line underneath, and everything is looking great, but it still doesn't really look like tabs yet.

The reason it doesn't look like tabs is because we haven't indicated what page we're on. Which page is the active page? So we need to

insert our "active" class again. So let's say we're on the Home page, so if we put in a class of "active" right here inside of our <li> and save that one more time and put that into the browser, you'll see that we have a very nice looking tab here. Of course, we can put some content underneath of this, and then it would be really obvious where we are.

```
<nav>
    <ul class="nav nav-tabs">
      <li class="active"><a href="#">Home</a> </li>
      <li><a href="#">About</a></li>
      <li><a href="#">Blog</a></li>
      <li><a href="#">Contact</a></li>
      <li><a href="#">Store</a></li>
    </ul>
</nav>
```

Gives us this:

So, that's tabs in a nutshell, and they're a pretty simple, but very effective structural element. Make sure that you indicate your **active** state so that the user is oriented on the page and within your site.

Now you might be wondering what else could you do with this type of tab styling? There's another class that makes menu items vertical. So what would happen if we took this tab styling, and we actually made it vertical? Well, it's a little surprising, but very cool.

After “nav tabs”, if you add another class of “nav-stacked” and save that page one more time, and Refresh in your browser, you’ll see that we have a very nice stack of links. The links are vertical again, but it’s definitely a step up from what we had with just the “nav” class on that <ul>. We have a little more margin, a bit more padding, we have some nice borders going on between these, we have a great hover state, and we still have our active state (Again, it only looks like it’s grayed out, as if you can’t click on it, even though you can. Remember, that’s just a CSS trick.)

Of course, these links continue to look as if they go completely across the page because there’s no Scaffolding in place. Add your scaffolding, and then you would see these look narrower.

So, what about the “pills” that I mentioned before? Well, if you go back into your text editor and remove “nav-stacked” and we change “nav-tabs” to “nav-pills” and Save that and Refresh your web page in your browser, you’ll see that the navigation has again become horizontal. Also, the active page that we were on has this nice red pill surrounding it. In this case, a red box with rounded corners and, of course, there’s a very nice hover state here as well. As you hover over these links, you’ll see that they have a lovely

grayed box also with rounded corners. You can use custom CSS to change these colors or tweak it to the way you’d like them styled as well.

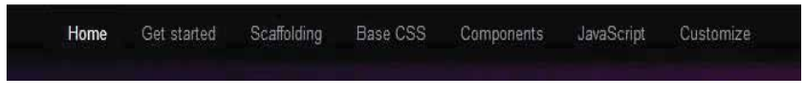
Just as we saw with tabs, you can make this pilled structure vertical as well, and if we go back to the text editor again and add “nav-stacked” just the same way we did with the tabs, and save and look at this in the browser again, you’ll see that now our navigation has become vertical. We still have our current page clearly marked here with these big, blue rounded cornerbacks, and we still have our hover state in place as well. Pretty nice!

So. That was an introduction to tabs and pills and how we can easily manipulate them to give us at least three or four different unique looks for our navigation on a web page, with just a handful of classes using the Bootstrap framework.

ADDING THE BASIC NAVIGATION BAR

Bootstrap comes with an actual navigation bar (nav bar) that you can use in your web pages as well. This nav bar comes with many options for customization and styling which will set your website apart.

If you head over to getbootstrap.com, and if you take a look at that black navigation bar at the top of the page, that's the nav bar that we're going to work on coding next.



So, if you take a look at your text editor document for this section, you'll notice the code looks very similar to what we just reviewed in the previous section.

```
<nav>
    <ul>
      <li><a href="#">Home</a> </li>
      <li><a href="#">About</a></li>
      <li><a href="#">Blog</a></li>
      <li><a href="#">Contact</a></li>
      <li><a href="#">Store</a></li>
    </ul>
</nav>
```

It's again just our simple navigation structure. You can see that we have another bulleted list here<ul>. So that's our starting

point, and we'll go ahead and begin by adding a few classes to create our awesome new navigation bar.

As with any navigation bar that you make in Bootstrap, the very first step is to set your <ul> to have a class of "nav" and then afterwards we'll need to pick an active link in our list, so the <li> will have a class of "active." We'll just set that on the "home" link for simplicity's sake. We've seen this before:

```
<nav>
    <ul class="nav">
      <li class="active"><a href="#">Home</a> </li>
      <li><a href="#">About</a></li>
      <li><a href="#">Blog</a></li>
      <li><a href="#">Contact</a></li>
      <li><a href="#">Store</a></li>
    </ul>
</nav>
```

Step number two is to take the <nav> tag that we have on the outside of this, and we're going to add another class to that as well. So, we'll add the class of "navbar" and we need to add a that goes around this also, which is important to the styling. Let's add a with a class of "navbar-inner" and of course, don't forget to close that down at the bottom.

```
<nav class="navbar">
    <div class="navbar-inner">
        <ul class="nav">
          <li class="active"><a href="#">Home</a></li>
          <li><a href="#">About</a></li>
          <li><a href="#">Blog</a></li>
          <li><a href="#">Contact</a></li>
          <li><a href="#">Store</a></li>
        </ul>
    </div>
</nav>
```

If we take a look at what we have so far by saving and refreshing--so Ctrl+S or Command+S to save--and then we preview this page in the browser, you'll see that we have a navigation bar that will appear across the top of our page and very nice hover states on the links that are there. The currently occupied page is indicated with a nice gray box around the word "Home," so everything is looking fine and dandy.

Branding

One thing that we're missing is some type of branding or signage for what website we're currently on, and that's also easily created within the specification for the navigation bar, so we can add that now as well.

Inside of your text editor, just after the with the class of "navbar-inner" we're going to add an anchor<a> with a class of "brand" and an href link. You, of course, can make this link to anywhere that you'd like, probably back to your home page, and you're going to put in the text that you want to appear as the identification of the website. For this example I'll just put "How to Bootstrap." You can put whatever you'd like.

```
<nav class="navbar">
    <div class="navbar-inner">
      <a class="brand" href="#">How to Bootstrap</a>
        <ul class="nav">
          <li class="active"><a href="#">Home</a></li>
          <li><a href="#">About</a></li>
          <li><a href="#">Blog</a></li>
          <li><a href="#">Contact</a></li>
          <li><a href="#">Store</a></li>
        </ul>
    </div>
</nav>
```

If we save that again and we take a look at our web page in the browser, you'll see that we have some nice identification, which is clickable, and it will take us directly to the homepage (That's the way that I would probably code it), and we have a navigation bar that appears.

Nice!

Now, if we minimize our browser and you start to pull the corner in to shrink the page, you'll notice that the styling kind of falls apart some. You'll notice that the navigation wraps onto the next line, apart from the "brand," which is nice. It's degrading somewhat gracefully here, but as we get really small, say mobile phone small, you can see that this is not as pretty as perhaps it could be. That's because we haven't added a responsive aspect to this particular navigation bar just yet, but we do have some basic styling for the navigation bar so far, at least.

One last thing that you probably noticed is that the navigation bar inside of Bootstrap was black, but here inside of our example it's white, and it's sort of difficult to see on the screen with that big white background behind it, but you can actually make this navigation bar black quite easily as well.

Inside of your text editor, up where we have the <nav> with the class of "navbar," go ahead and make that "navbar-inverse" and this will give you the inverted (black) color. You may recall this makes buttons black as well. If you refresh the page again, we'll now have a black nav bar closer to what you saw on that Bootstrap

page. This is the most basic markup for the navigation bar that comes with Bootstrap.

```
<nav class="navbar navbar-inverse">
    <div class="navbar-inner">
      <a class="brand" href="#">How to Bootstrap</a>
        <ul class="nav">
          <li class="active"><a href="#">Home</a></li>
          <li><a href="#">About</a></li>
          <li><a href="#">Blog</a></li>
          <li><a href="#">Contact</a></li>
          <li><a href="#">Store</a></li>
        </ul>
    </div>
</nav>
```

Will quite easily give us this:

So now we've reviewed some simple things like setting up the general structure of a nav bar, we've added some branding to it so the user will know what website he or she's on, and we also changed the color of it to black, so we have a white version and a black version of this particular navigation bar. Pretty basic. Of course, you'll probably want to eventually override these colors or styles with your own, which you may easily do by using a custom stylesheet.

CHAPTER 5: JAVASCRIPT EFFECTS

UNDERSTANDING HOW JAVASCRIPT WORKS IN BOOTSTRAP

Bootstrap currently ships with 13 custom **jQuery**-based plug-ins. jQuery is a very popular JavaScript library, and Bootstrap has included some components that we can easily use from this powerful framework. They include

Modal Windows, Dropdowns, Tabs, Tool Tips, Alerts, Popovers, an Image Carousel, and much more.

Together, the **Components** and **JavaScript plugins** sections provide the following interface elements:

LIST OF COMPONENTS

- Button groups
- Button dropdowns
- Navigational tabs, pills, and lists
- Navbar
- Labels
- Badges
- Page headers and hero unit
- Thumbnails
- Alerts
- Progress bars
- Modals

- Dropdowns
- Tooltips
- Popovers
- Accordion
- Carousel
- Typeahead

You may want to walk through these components individually in the documentation in more detail for all the options available and ways to customize them.

If you're familiar with JavaScript and jQuery, you can modify anything in Bootstrap using the provided API. If you're not familiar with JavaScript or jQuery, don't worry, there are a ton of effects that are created for you which you can easily put to use immediately. You'll just need to very carefully copy some code from the documentation pages into your work.

Head to GetBootstrap.com again, and click on the JavaScript link at the top of the page, and then on this page you'll see the 13 effects that use jQuery, and have been integrated into Bootstrap. They're listed over in a stack on the left side of the page.

There are also some items under the "Components" menu item in the Bootstrap navigation bar that also require the use of JavaScript or jQuery for them to work. Be very sure that you've linked to the **bootstrap.min.js** file, as well as the **latest version of jQuery**! The way to do that is by placing the following snippet just before the closing </body> tag:

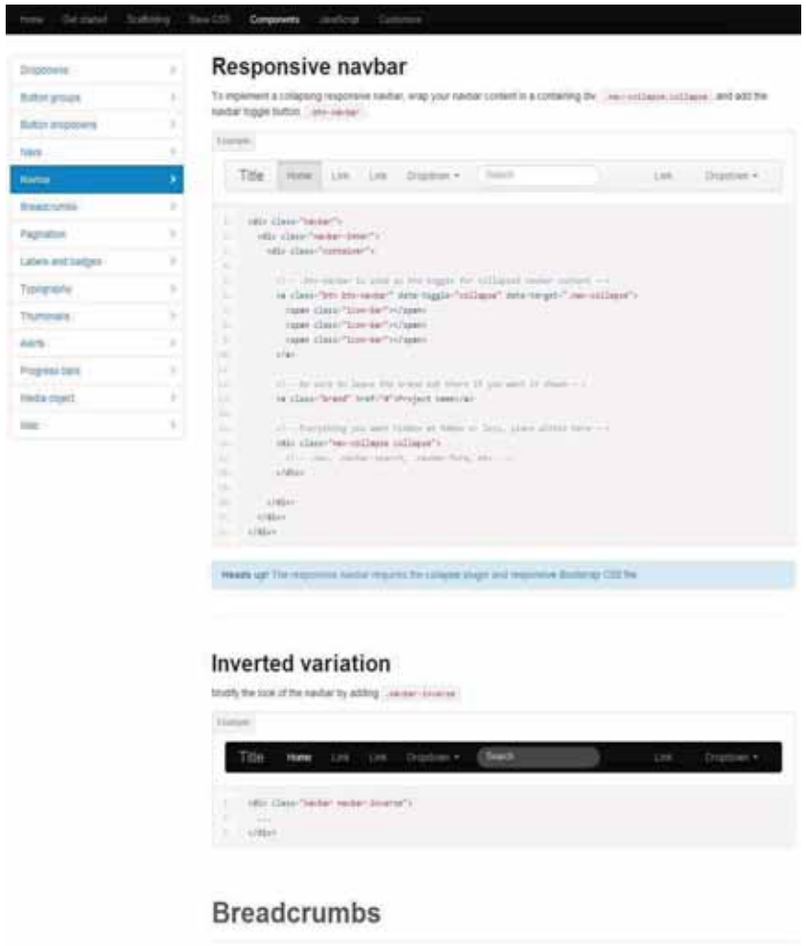
```
<script src=http://code.jquery.com/jquerylatest.js></script>
```

<script src="js/bootstrap.min.js"></script>

Also, if you're creating these files on your own, be sure you link to both of these files, otherwise these JavaScript effects won't work.

Remember that the .min files contain the same information as the regular JavaScript files, but the .min files have all the white space removed from between the characters to make the file sizes smaller, and your pages load faster. Although we can't read them, computers can, so always try to minify your code, if possible. There are web apps that can crunch your code for you, such as minifycss.com (which can minify CSS and JavaScript) and jscompress.com.

Be aware there are a few simple custom HTML5 attributes you should be familiar with that are going to be used in Bootstrap, and you'll see that in a number of places over the next few sections. Go ahead and click on the link for "Navbar" under "Components," and then scroll down to find the section about the Responsive navbar.



There on line 6 you'll see two items working together. The **data-toggle** is applied to a button or a link, or basically anything that has to be clicked or rolled over to trigger an event or to make something happen. The **data-target** is something that takes the action from the controlling element. So, in the case of tabs that will change content when clicked, the data-toggle is the link on the tabs, while the data-target is a containing the text that will display when one of the tabs is clicked.

So, let's go through a few of these now in detail with Bootstrap and JavaScript so that you can see how data-targets and data-toggles work together in Bootstrap, and you can become comfortable with some of the basic JavaScript inside of Bootstrap.

MAKING THE NAV BAR RESPONSIVE WITH JAVASCRIPT

In the last chapter of this book, we created a navigation bar for our website. And it looked great while it was viewed on a desktop computer and the browser stays maximized, but as we reduce the size of the web browser, the navigation bar wraps on to another line, which is fine for a little while, but when we get down to a browser size that would simulate the proportions of a mobile phone screen, you'll see that the navigation bar can become awkwardly large, and that's probably not really an effect that you'd like to have. What should we do?

Well, it's possible to add some JavaScript to make the navigation bar more responsive. It's not the only approach to making a nav bar more responsive, some may want to debate. Some may say using a JavaScript solution isn't even the best approach. But this is a great effect that comes with Bootstrap that will give you the option of having your nav bar shortened as the screen sizes gets smaller. And I'll show you exactly how that works now.

Back to the old text editor document inside your exercise files, and we'll look at our file from where we left off from Chapter Four. We're going to add this short bit of additional code to it:


```
<a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
</a>
```

This is the first code snippet that we're going to add to your text editor page here. Inside of your text editor page, just after but before the "brand" line we inserted earlier, Ctrl+V or Command+V what's on the clipboard to Paste that in place. What you have should now resemble this:

```
<nav class="navbar navbar-inverse">
    <div class="navbar-inner">
        <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </a>
      <a class="brand" href="#">How to Bootstrap</a>
        <ul class="nav">
          <li class="active"><a href="#">Home</a></li>
          <li><a href="#">About</a></li>
```

So what's this code snippet that we just placed in here do, exactly?

Well, first of all, you'll see that the three s are calling Glyphicons. There's an icon that's called the "icon-bar" so we're putting three of them in here.

The key that makes this interactive is what's inside of the anchor <a> tag, which will be a clickable link. You'll notice that this has been assigned several classes, both the "btn" and "btn-navbar,"

and then you see that this JavaScript calls for data toggle collapse and then data- target of .nav collapse.

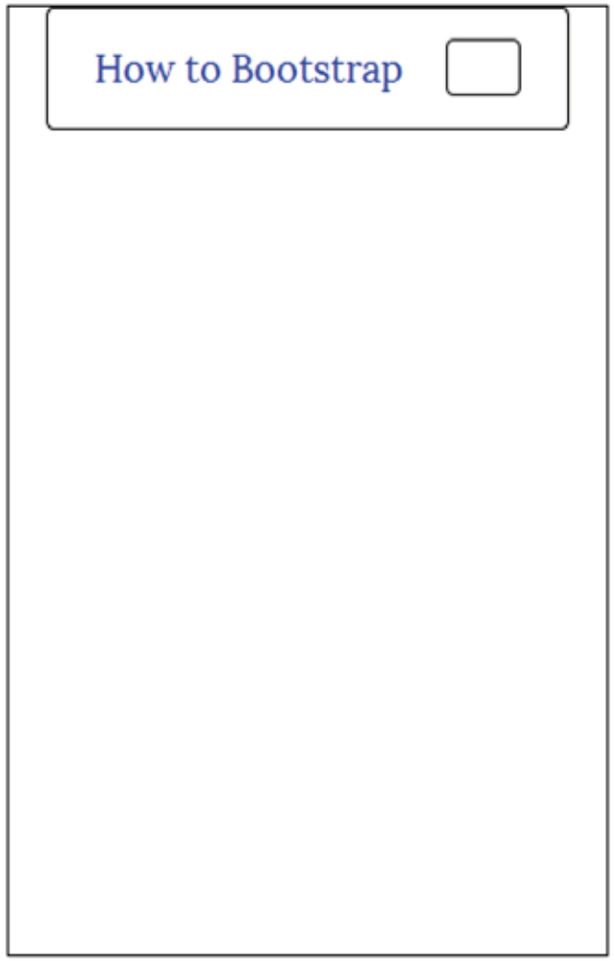
What these are doing is incorporating the JavaScript into this web page to make our icons expand when they're clicked inside of the bar. So if you copy in that code where indicated, that will make your navbar responsive. And of course, you can certainly switch out those three "icon-bar" lines here for another icon, or even a sprite image. You can put in a word like "menu," for example, or anything so there's something there that's clickable, and you wouldn't change the functionality of what's happening here.

The second part to what we have to do is show what's going to be hidden as the nav bar collapses in size. Typically we wouldn't want to hide the brand, but we do want to hide all of the navigation here. So what we'll do is add another tag there, and we'll give it a class of "nav-collapse." Simple enough. So this particular , which we'll close immediately after the bulleted list, is indicating this chunk of code will be hidden as the navigation bar gets smaller. And the media query will trigger to display this navigation bar with just the brand and the little button, because the rest of this, which is inside this of "nav-collapse," will be hidden when that transition happens.


```
<nav class="navbar navbar-inverse">
    <div class="navbar-inner">
        <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </a>
      <a class="brand" href="#">How to Bootstrap</a>
        <div class="nav-collapse">
            <ul class="nav">
              <li class="active"><a href="#">Home</a></li>
              <li><a href="#">About</a></li>
              <li><a href="#">Blog</a></li>
              <li><a href="#">Contact</a></li>
              <li><a href="#">Store</a></li>
            </ul>
        </div>
    </div>
</nav>
```

So go ahead and Save this page-- Ctrl+S or Command+S to Save, and if we put this into the browser and we Refresh the page, immediately we see nothing special, if your browser's maximized on a desktop or laptop computer.

But if we begin to squeeze the page in, you can see we have an icon that now appears. That's the Glyphicon with the three bars stacked together. And as we make the page progressively smaller, then the navigation bar is resizing itself nicely. When you click the little button we created, you'll notice that we get the full menu with things oriented in a vertical manner. Any content that you have on the page will be pushed down underneath the black box.



So this is how you can go about making your site's navigation bar responsive just by adding these two simple bits of code.

So, we've added a snippet of code that will trigger the JavaScript to cause the open and close effect on the web page, as we click

to open and click to close. We also added some markup that indicates what's supposed to show and hide when that button is clicked.

ADDING A DROPDOWN MENU TO THE NAV BAR

So over the last few sections we've built a fully responsive navigation bar. We know that it will respond to smaller screen sizes by hiding the navigation where required. You can access it through JavaScript, which will make a nice, easily-accessible navigation bar for smaller devices and appliances.

But you may be wondering about an aspect of this navigation bar that hasn't been mentioned yet. That is there's only one level of navigation here. Just the primary navigation is present. What about secondary or tertiary navigation for submenus? How do we show that on our web page?

Well, certainly you could always take the approach of plugging in a left or right column on this web page or even another horizontal bar that might contain those secondary or tertiary links. But a popular and more reasonable approach these days is to include dropdowns, and new devs want to know how to build dropdowns all the time. Well, finally, I'm going to show you how to do it.

I do feel obligated to mention that as of this writing, which coincides with version 2.3.0, there is discussion among the developers and the more vocal users about whether to support

dropdown navigation menus in version 3. While that version isn't due out for some time, the planning and work is underway, and version 3 of Bootstrap is going to be "mobile-first," as more and more people are using mobile phones to access webpages, instead of (or along with) desktop and laptop computers. Bootstrap's developers think including dropdowns unnecessarily adds to the file size and is a functionality that isn't used enough to warrant dropdown inclusion, and must be given special considerations when coding, as we've just seen. There are some users, on the other hand, that use dropdowns in their navigation all the time and would very much miss having it. While dropdowns can be added via a plugin, some feel it should at least be an option on Bootstrap's customization page. Time will tell and we'll see what the final verdict is when version 3 arrives. For now, they're available, so we'll learn to utilize them.

Dropdowns are very easy to add with Bootstrap, and I'll show you how to do that now. Head back to your trusty text editor, add this bit of code to your file immediately after the <li> of "About":

```
<ul>
  <li><a href="#">History</a></li>
  <li><a href="#">Misson statement</a></li>
  <li><a href="#">Employment</a></li>
</ul>
```

So there are the links for our "About" page. Notice that the "About" code begins with an <li>, but doesn't close it at the end of that line. We then have our <ul> and then we have our <a> tag with the and closing . But the closing doesn't occur until further down, right after that. Remember that these lists are actually nested within other <li>. That's the correct way to mark them up.

So we have our <ul> here inside of that <li> with a further down, and we have three bullets inside of that. What we'd like to do is have this become a dropdown menu. So let's take a look at where we are at this particular point.

```
<a class="brand" href="#">How to Bootstrap</a>
  <div class="nav-collapse">
      <ul class="nav">
        <li class="active"><a href="#">Home</a></li>
        <li><a href="#">About</a>
          <ul>
              <li><a href="#">History</a></li>
              <li><a href="#">Misson statement</a></li>
              <li><a href="#">Employment</a></li>
          </ul>
          </li>
        <li><a href="#">Blog</a></li>
        <li><a href="#">Contact</a></li>
        <li><a href="#">Store</a></li>
      </ul>
```

If you'll open this document in your browser, you'll notice that we indeed have a bulleted list that appears right under the "About" section, which is great. But it's not really a dropdown because, of course, it's showing everything right there on the page and it's static. So we need to add some additional styles in order to make it an actual dropdown menu.

The first thing we'll do inside of the text editor is, for the specific <li> belonging to the word "About," we'll add a class of "dropdown," which of course indicates that we're about to include a dropdown within this <li>.

Then over in the <href> side of things we'll add a class of "dropdown-toggle," and then we're going to use that data-toggle, which is what's helping to integrate the JavaScript: data-toggle="dropdown" just like this:

```
<a class="brand" href="#">How to Bootstrap</a>
  <div class="nav-collapse">
      <ul class="nav">
        <li class="active"><a href="#">Home</a></li>
        <li class="dropdown"><a href="#" class="dropdown-toggle" data-toggle="dropdown">About</a>
          <ul class="dropdown-menu">
              <li><a href="#">History</a></li>
              <li><a href="#">Misson statement</a></li>
              <li><a href="#">Employment</a></li>
          </ul>
          </li>
```

So we've added a class of "dropdown-toggle" and a data-toggle of "dropdown" to a tag that's associated with the word "About." Lastly, what we'll do, as seen above, is add to our <ul> a class of "dropdown-menu," which will give us some styling for the dropdown menu.

So if you go ahead and save that--Ctrl+S or Command+S to save--Refresh the page inside of your browser. Now you see that we have a navigation bar. When you move your mouse over the word "About" and click, you'll notice that we get a dropdown menu.

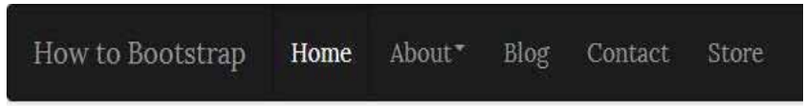
So there are our History, Mission Statement, and Employment links, and all these are clickable and will take you to those particular pages. How did we know that "About" was the one that had the dropdown menu? Well, we knew only because we coded it. But how is a visitor of our web page going to know that "About" has the dropdown menu? Great question.

It's likely that a visitor isn't going to know that there's a dropdown menu below "About," just by looking at it. So we need to indicate somehow that "About" has more information behind it than just a single link.

Back to your text editor, and we can add some additional code which will give us a tiny triangle that will appear right next to that word "About" that will indicate that a dropdown menu is located there. So right here after the word "About," before the closing , go ahead and put in a span class of "caret" immediately followed by a closing .

```
<a class="brand" href="#">How to Bootstrap</a>
  <div class="nav-collapse">
      <ul class="nav">
        <li class="active"><a href="#">Home</a></li>
        <li class="dropdown"><a href="#" class="dropdown-toggle" data-toggle="dropdown">About<span class="caret"></span></a>
          <ul class="dropdown-menu">
              <li><a href="#">History</a></li>
              <li><a href="#">Misson statement</a></li>
              <li><a href="#">Employment</a></li>
          </ul>
          </li>
```

This little bit of code is going to put in that tiny little triangle arrow that will indicate that a dropdown lives there. Save your page, Ctrl+S or Command+S, refresh the page inside of your browser, and now the word "schedule" has the little triangle right next to it, which would indicate that if you click on it you now have some additional options to explore.



So dropdowns are pretty straightforward to add to our navigation toolbar. Actually, they're pretty easy to add to tabs and pills as well. So let's learn how how to do that in this next section.

ADDING A DROPDOWN MENU TO TABS AND PILLS

In the last chapter we built a pill-based menu, and we'll take a look at expanding upon one now.

If you'll recall, the pill-based menu had rounded, cornered red rectangles indicating where you were on the page, and when you moused over the navigation you had a light gray one that goes across the tab. Your menu may be blue; this is just how my browser's default settings are configured.

We've added to that a secondary navigation right under "Blog," which is where it appears right now. And we'd like this to be a dropdown menu, but at the moment it's displaying on this page in a static manner. So let's take a look at how to code this to make this a dropdown menu.

```
<nav>
<ul class="nav nav-pills">
  <li><a href="#">Home</a> </li>
  <li class="active"><a href="#">About</a></li>
  <li><a href="#">Blog</a>
     <ul>
        <li><a href="#">Personal</a></li>
        <li><a href="#">Professional</a></li>
        <li><a href="#">Volunteer</a></li>
     </ul>
  </li>
  <li><a href="#">Contact</a></li>
  <li><a href="#">Store</a></li>
</ul>
</nav>
```

If you have a look at the above code in your text editor, you can see that I've added a bulleted list of sub-topics for our blog. They're located inside of the <li> for "Blog," which, as you now know, is the correct way to mark that up.

Note that you start with the opening <ul> tag. In between you have the bulleted list, each with its own bullet. So to make this a dropdown menu, there are just a couple of things we need to do to make that happen.

First of all, find the <li> which contains the word "Blog," we're going to need to add a class of "dropdown." We'll also need to add a class to the <ul> located inside of that <li>. We're going to give that a class of "dropdown-menu."

```
<ul class="nav nav-pills">
  <li><a href="#">Home</a> </li>
  <li class="active"><a href="#">About</a></li>
  <li class="dropdown"><a href="#">Blog</a>
     <ul class="dropdown-menu">
        <li><a href="#">Personal</a></li>
        <li><a href="#">Professional</a></li>
        <li><a href="#">Volunteer</a></li>
     </ul>
```

However, just adding those classes isn't going to do much for us. What we need to do now is make sure that the JavaScript is integrated into this, and where that will be applied is inside of the <href>, which goes around the word "Blog." So here we'll add a class of "dropdown-toggle," and then we're going to add that data-toggle, which as you may remember has to do with the JavaScript "dropdown."


```
<ul class="nav nav-pills">
  <li><a href="#">Home</a> </li>
  <li class="active"><a href="#">About</a></li>
  <li class="dropdown"><a href="#" class="dropdown-toggle" data-toggle="dropdown">Blog</a>
     <ul class="dropdown-menu">
        <li><a href="#">Personal</a></li>
        <li><a href="#">Professional</a></li>
        <li><a href="#">Volunteer</a></li>
     </ul>
```

And those two little items, the class of "dropdown toggle" and the data-toggle of "dropdown" will give us all that's required to make this work. So go ahead and Ctrl+S or Command+S to save the page and refresh this in the browser, and you'll see that we have the menu. It doesn't look like there's a secondary navigation, but when we hover the cursor over the word "Blog" and click, you'll see that results in a dropdown. So that's pretty cool.

But again, the only reason we knew that we could click on the word "Blog" and get a dropdown is because we coded it. We need to have some sort of indicator that the dropdown lives there under our "Blog." So to do that, we, again, could add a little triangle that points downwards, and that would be a

great indicator to the user that "Blog" is interactive and has a dropdown associated with it.

So, right after the word "Blog," we're going to put a space followed by a class of "caret" and a . It's exactly what we just did in the navigation bar dropdown chapter. So go ahead and save that, and if you refresh in your browser, you'll see that the word "Blog" now has a triangle by it. That triangle's clickable and our dropdown menu appears underneath. Success!

So based on the code that I've shown you here, you can add as many dropdown menus as you want to your web pages. You can continue to add those through every navigation item that's on this page. You could even apply them to some of these secondary navigation items.

You could give them a nested list as well and then have another dropdown appear from there.

TABBING WITHIN THE SAME PAGE

On occasion, you may wish to tab between blocks of content within the same web page, usually through a folder-

like interface with a series of tabs that you can click on and see different kinds of information without the page having to reload, or make another call to the server to switch between pages. Bootstrap has a great JavaScript-enabled interface for creating this kind of effect.

So our starting HTML looks like this right now:

```
<body>

    <ul>
      <li><a href="#">Harvard</a></li>
      <li><a href="#">Stanford</a></li>
      <li><a href="#">Princeton</a></li>
    </ul>


    <h3>Harard</h3>
    <p>Fingerstache echo park put a bird on it iphone thundercats. Intelli
    <h3>Stanford</h3>
    <p>Fingerstache echo park put a bird on it iphone thundercats. Intelli
    <h3>Princeton</h3>
    <p>Fingerstache echo park put a bird on it iphone thundercats. Intelli


  <script src="http://code.jquery.com/jquery-latest.js"></script>
  <script src="js/bootstrap.min.js"></script>
</body>
```

We have a bulleted list at the top with some school names, and underneath we have some "Hipster Ipsum" text. It's just marked up with an <h3> for a title and a paragraph of dummy text which appears underneath.

So there's our plain-old regular bulleted list, and then a bunch of <h3>s and paragraphs. So what we'd like to do is turn this

into an array of tabs that we can click on and see each of these schools' information displayed individually, rather than all together on one page, in a list.

To make that happen, there are a couple of things that we need to do. First of all, we're going to need to do some markup on the tabs. So we're going to start by taking the <ul>, and we're going to add classes of "nav" and "nav-tabs." Then we need to indicate which one of these tabs is going to start off as the active one. So we may as well just make the first the active one, so go ahead and give that a class of "active." Obviously, we could have started with any of the other tabs as the active one, if we wished. Doesn't matter which, but one needs to be the active tab.

Then we're going to add a data-toggle to the <a href> tags for all of these, and the data-toggle, as we now know, is what's going to pull in the JavaScript to make this page interactive. So add data-toggle="tab" to each. Since we need to apply that to all three of these links, it may be easier and safer to just going to highlight that, Ctrl+C or Command+C to copy, Ctrl+V or Command+V to paste in back in twice more, so that it now appears in all three of these links.

```
<ul class="nav nav-tabs">
  <li class="active"><a href="#" data-toggle="tab">Harvard</a></li>
  <li><a href="#" data-toggle="tab">Stanford</a></li>
  <li><a href="#" data-toggle="tab">Princeton</a></li>
</ul>
```

Finally, what we need to do is connect the specific link for Stanford, for example, with the bit of text that will be accompanying each school name. The links that we have up there

right now are just sort of dummy links; they don't actually go anywhere. So somehow we need to differentiate between these.

So let's give these names. Right there with the first one, we'll call it "#Harvard," and leave the pound sign (#) there in front, and the second one will be #Stanford, and the third one is #Princeton. So now what we have should look like this:

```
<ul class="nav nav-tabs">
  <li class="active"><a href="#Harvard" data-toggle="tab">Harvard</a></li>
  <li><a href="#Stanford" data-toggle="tab">Stanford</a></li>
  <li><a href="#Princeton" data-toggle="tab">Princeton</a></li>
</ul>
```

This is exactly what we would do if we were using anchors further down on the page with links on the top. The way these links were coded, they would skip down to a link that has a name of "Princeton," for example. It would jump on down the page to wherever that happened to be placed.

We're going to do something similar here, except directed by JavaScript. So we have the markup that we need for the list at the top. Then underneath we're going to need to add some markup here as well.

First of all, let's add a <section> that wraps around all of this with a class of "tab-content." The <section> tag is important because it's defining that this is the actual content for the web page here, so that's why we're using the <section> tag. Then go down and

close that <section> at the bottom. Then after the <section> tag, let's add an <article> tag with a class of "tab-pane."

So then this <article> tag is going to get wrapped around what effectively is an "article," the <h3>, and the paragraph <p>. Go ahead and close that <article> down at the bottom, and we can clean up our code if necessary and indent this <h3> and the <p> because now they're inside of the <article> tag.

In most text editors you're able to just highlight all those lines and then hit the "Tab" key. It's a very useful feature for the code-weary. So let's go ahead and add the same <article> class of "tab pane" to the top of each one of those little mini articles, and always of course, make sure you close your tags, and then as before, tab everything over if needed to show that it's inside of the <article>.

One more time, paste the <article> class of "tab-pane," and we'll include our </article> down there at the bottom, and then one last time, tab over this group of code if you need to.

So we have three articles with a class of "tab-pane." We also need to add to the same <article> tag and a couple of other pieces of markup, and one is an id. The id element in this particular case for this first one will be "Harvard." Some of you may think about ids exclusively in terms of CSS, if you're used to working only with HTML and CSS. But ids are also used in JavaScript, as they are here.

So what we're doing now is connecting the link at the top of "#Harvard" to an id down below, the id of "#Harvard." So whatever you called your links up there on the top, you're going to need to take that and identify them as ids beneath, inside of these three articles.

Go ahead and add an id of "Stanford" and further down an id of "Princeton." And there's one quick final thing that we need to do. Since "Harvard" is the tab that we want to appear by default when the page loads, which we specified earlier by giving it's <li> a class of "active," we also need to identify which piece of content is going to come up by default underneath, and that, of course, is the article here about Harvard. So we'll indicate that Harvard should come up when the page loads by adding a class of "active" right to this <article>. So we have classes of "tab-pane" and then also "active" for the <article> tag, while the other <article> classes are only "tab-pane."

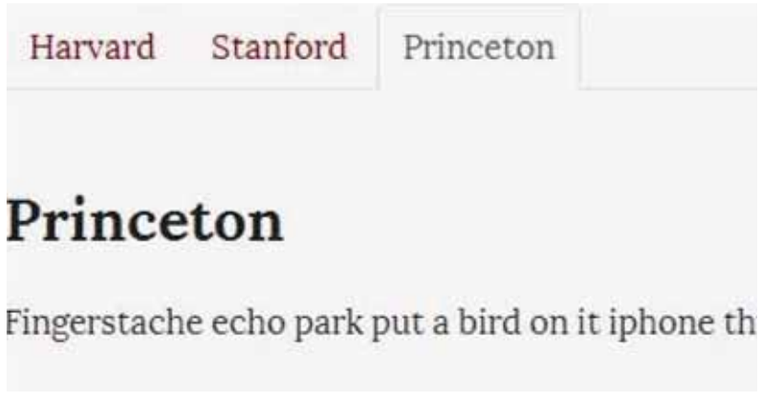
I know that was a lot to follow and absorb, but you should now have something in your text editor that looks like this:

```
<ul class="nav nav-tabs">
  <li class="active"><a href="#Harvard" data-toggle="tab">Harvard</a></li>
  <li><a href="#Stanford" data-toggle="tab">Stanford</a></li>
  <li><a href="#Princeton" data-toggle="tab">Princeton</a></li>
</ul>

<section class="tab-content">
    <article class="tab-pane active" id="Harvard">
        <h3>Harvard</h3>
            <p>Fingerstache echo park put a bird on it iphone thundercats. Intel
    </article>
    <article class="tab-pane" id="Stanford">
        <h3>Stanford</h3>
            <p>Fingerstache echo park put a bird on it iphone thundercats. Intel
    </article>
    <article class="tab-pane" id="Princeton">
        <h3>Princeton</h3>
            <p>Fingerstache echo park put a bird on it iphone thundercats. Intel
    </article>
</section>
```

Now that we have all of this markup in place, go ahead and save your page, Ctrl+S or Command+S to save, and Refresh the page in

your browser, and you'll see that we have tabs now going across the top of the page. When we click on "Stanford," you can see that Stanford's dummy-text loads here inside of the tab. The little dotted box is up there because we've selected it at the moment. If we click away from that tab, that box goes away, and here's Princeton, and you'll see that the information for them is listed here as well. So we can easily click around in these tabs without extra page loads. Notice that the URL for the page never changes. We're just displaying different sections of the page inside of these tabs.



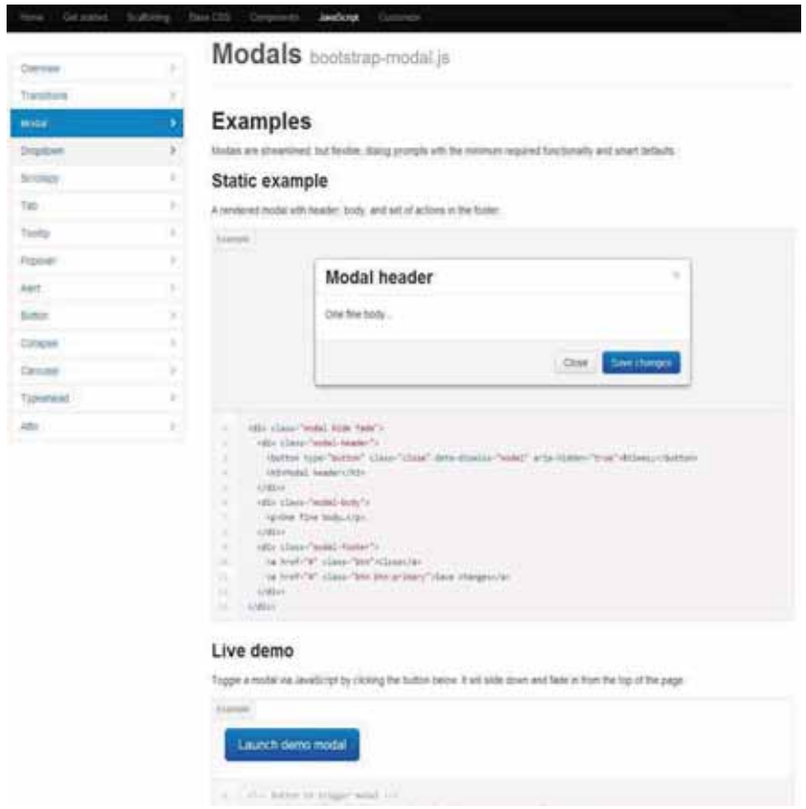
So this is a really handy effect that's a positive UX in many circumstances. You'll see it used on a number of websites these days. It's a little more stylized and subtle than having long lists of links or skipping things down the page like we used to do. This is a lot more fun, a lot more interactive, and certainly more attractive.

You may be wondering about Search Engine Optimization now that we've separated these out between the tabs like this, however SEO shouldn't be impacted at all. Why is that?

Well, if you take a look at the raw HTML for this web page, the raw HTML is all right there laid out for search engines to find in the web page. So Google, Bing, Yahoo! and all the other search engines will be able to see that when they visit your website. They can read all of the content there despite how it's displayed to you and me. They won't see the JavaScript and they won't see the JavaScript effects; they'll see the links at the top of the page, they'll see that those link further on down the page where the ids are, but the page is very readable to search engines, and all of the tabs can be indexed very easily by search engine spiders.

CREATING MODAL WINDOWS

A modal window is a familiar effect, but it's nothing more than a that's embedded within your web page that appears when called by a mouse click or hover state or some other event. It appears on top of your page on a darkened background, and it looks kind of like a popup window. However, it's not an actual pop-up because it doesn't appear in a separate browser window. You may have seen this on many different websites. It's a non-intrusive way to relay to or request additional information from the user, and very handy to have in your toolbox. Some examples are given at GetBootstrap.com, then click on the "JavaScript" tab at the top, and then the "Modal" button on the side.



If you scroll down a little bit there, you'll see that you have a link to a live demo.

If you click that big blue button that says "Launch demo modal," you'll see how the display darkens, and a small window seems to scroll in to place. The window is comprised of three parts:

1. We have the modal heading up on the top with a clickable "Close" button over on the side (the little X).
2. We have a scrollbar on the right of the text to keep the text from overflowing the modal window, and

3. Down at the bottom we have two buttons: one button to close the window and another button to save our changes.

The "Save Changes" button may seem a little out of place to you because you're wondering "what can you possibly change within this window?" And you're right; there's nothing in that window you can actually change and save.

But, if this window was hooked up to a database, you might be able to make some changes within the Modal window and save those changes. Many content management systems frequently use a feature like this.

So let's close that and close the tab. The next html file is our modal.html file, which may be found in your exercise files at FreeBootstrap.com/. This is our starting point for how this page is going to be laid out.

Up at the top of the page look for the text that reads "click me!" That's going to be our big button that we'll have on the web page. The rest of this will end up being the actual Modal window, including our bacon ipsum.

```
<body>
<p><a href="#">click me!</a></p>
<section>
  <header>
    <button type="button">&times;</button>
    <h3>The Kentucky Derby</h3>
  </header>
  <h4>About The Event</h4>
  <p>Bacon ipsum dolor sit amet rump ham sausage,
  <footer>
    <button>Close</button>
  </footer>
</section>
```

And you'll see the button with the X on it will wind up being that little X that appears in the corner. We have some bacon ipsum dummy text in the window, and then down at the bottom, there's a button to close the window. So now we need to get all styled-up, and we need to apply some JavaScript to make the window appear and vanish. Let's do it!

Head back over to your text editor and you should be looking at the above code, which will give us our modal window and accompanying features.

You should have the "click me!" link at the top, and we'll apply quite a bit of styling and JavaScript to that in a little bit. Then we have a section for the web page. That section is comprised of three parts, similar to the demo we saw on Bootstrap's website, and not unlike a regular web page, with a header, body and footer:

1. There's a header which contains that little X button as well as the heading.
2. We have some dummy text, and then down at the bottom,
3. We have the footer with our "Close" button.

So, let's walk through how we're going to set up the styling for this.

First, starting up top with the <header>, we'll give this a class of "modal-header." We'll wrap a around the text there that's in the middle, which will be our modal text. Then we're going to give that a class of "modal-body." You may need to straighten up your code some after this to keep it manageable, and to make the code a little easier to read. And then close the down below.

Then, you may not be surprised to learn that for the footer, we give it a class of "modal-footer." And there we go! We've identified the three sections of our Modal window: a header, a body, and a footer.

So that's all in place. Now, let's apply some styling to the buttons.

The button down at the bottom (the "Close" button) needs to receive a class, so we're going to give it a class of "button" ("btn") and we've covered the styling of buttons in an earlier section in the CSS chapter.

Let's say we want to make this button blue, so, we'll give it another class of "btn-primary," which will give it an attractive blue color.

For the button that's further up on the page, the little "X," we'll do something slightly different. We're just going to give this a class of "close." Now, for both of these buttons, the functionality behind

them is exactly the same, and that's when they're clicked, the Modal window should just go away. And there's an attribute that we can add that will make that magic happen and talk to the JavaScript. There's nothing more to that than adding "data-dismiss," to that same class, with a value of(=) "modal."

Then add that to the other button at the bottom as well: data-dismiss="modal." So after all that, you should have some code that looks similar to this:

```
<body>
<p><a href="#">click me!</a></p>
<section>
  <header>
    <button type="button">&times;</button>
    <h3>The Kentucky Derby</h3>
  </header>
  <h4>About The Event</h4>
  <p>Bacon ipsum dolor sit amet rump ham sausage,
  <footer>
    <button>Close</button>
  </footer>
</section>
```

So now that we have all that in place, let's style the link that's up at the top of the page, the one that says "click me!"

We'll go ahead and give that a style. We'll give it a class of "btn" for button, because we want it to look like a button. Just to be different, let's give it a class of "btn-success", which will make it

green. We also want to make it really large, so let's give it a class of "button-large."

```
<body>
<p><a href="#" class="btn btn-success btn-large">click me!</a></p>
<section>
  <header class="modal-header">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
```

So, there's our basic styling. The next thing we need it to do is make sure that when the button is clicked, that something happens to the section that contains the Modal window information. And the way to do that is to change the <href> there from just the pound sign(#) to something like "launch." So up in the section, let's give this a matching id of "launch."

Remember that the id here for the section, whatever it is that you named that, should then be called via pound sign with that same id name within the <href>.

Then, finally in the <href>, we'll add a data toggle with a value of "modal." That will call up the Modal window when the button is clicked, and that data-toggle will be talking to the JavaScript that makes that happen.

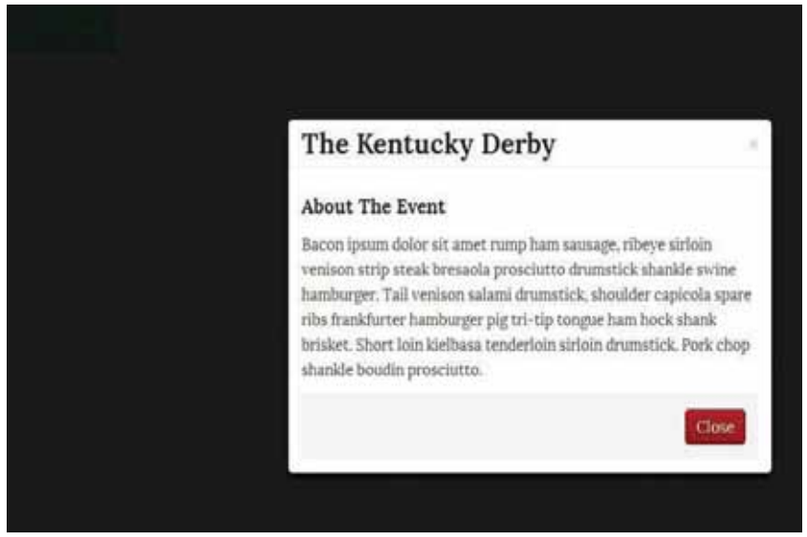
Last but not least, let's style the <section> with the id of "launch." We'll give it a class of "modal" as well. That'll control the appearance and disappearance of the window. We'll add to that class "hide" and "fade." "Hide" and "fade" will give the modal window a nice smooth slide as it comes in and as it goes away again.

So, between the closing</head> tag and the <script> tags at the bottom, you should have something like this:

```
<body>
<p><a href="#launch" class="btn btn-success btn-large" data-toggle="modal">click me!</a></p>
<section id="launch" class="modal hide fade">
  <header class="modal-header">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h3>The Kentucky Derby</h3>
    </div>
  </header>
  <div class="modal-body">
  <h4>About The Event</h4>
  <p>Bacon ipsum dolor sit amet rump ham sausage, ribeye sirloin venison strip steak bresaol
  <footer class="modal-footer">
    <button class="btn btn-primary" data-dismiss="modal">Close</button>
  </footer>
</section>
```

Now that we've got all of that in place, go ahead and Ctrl or Command+S to save, and off to the browser, and just hit your Refresh button. You'll see that we have a big green button at the top of our page that says "Click me!" Go ahead and click it!

Sure enough, here comes your fancy Modal window exactly as you would expect it to appear. Notice that when we click the "Close" button at the bottom, it goes away, and the screen returns to full brightness. If you click it again, you can click the little "X" in the upper right-hand corner, and it behaves exactly the same way as the "Close" button.



Modal windows may be very helpful to you in your web design, and they're easy to implement using Bootstrap. Certainly more complicated-looking than they actually are. They're also nice for SEO because all of the text that's within that Modal window is available for search engines to read, as opposed to pop-up windows which are separate HTML documents.

So, even though we're using JavaScript, it shouldn't detract from your search engine optimization results, and you now have everything that you need to put Modal windows to work inside of your Bootstrap website.

CREATING A PHOTO CAROUSEL

The last JavaScript effect we'll review is the Image

Carousel, and this is really cool. People love carousel effects, and they're very effective for spotlighting several large images on your website with an attention-grabbing effect.

Here we have a series of large, landscape-esque photos that are scrolling on by with some text underneath each one. We have little buttons so that we can click to the next image as fast or as slowly as we want. You can also go backwards in the carousel. These carousels are pretty much everywhere on the web these days, and there are plugins and a thousand different ways to code them. Of course, we'll learn the Bootstrap way of doing it, which I think renders a really nice effect.

So take a look at your Carousel.html document and you'll notice that we have some robot photos here. Then there's some robot-ipsum text that goes underneath each of the images. Then down at the bottom some little left and right arrows, and if we take a look at

this in the text editor, you'll see that all this code is exactly the same.

```
<body>
      <img src="img/robbie.jpg" alt="Robbie Robot ">
        <h4>Robbie Robot</h4>
        <p>Trigger point Brackenridge inverse kinematics wires beep.
      <img src="img/robot.jpg" alt="white robot">
        <h4>iRobot</h4>
        <p>Trigger point Brackenridge inverse kinematics wires beep.
      <img src="img/retro.jpg" alt="retro toy robot">
        <h4>Retro Robot</h4>
        <p>Trigger point Brackenridge inverse kinematics wires beep.

  <a href="#">&lsaquo;</a> <a href="#">&rsaquo;</a>
```

We have some images, we have some headings with little bit of text underneath; and then down at the bottom we have little left and right arrows. Now what we need do is mark this up to make it all work for our carousel. If you're using different images, be sure to link them up accordingly, changing out the included robot image file names with your own.

So first of all, the easiest things to add are some s here which will designate where the captions will appear for our carousel. So just after the first image, but before the <h4>, we're going to add a with a class of "carousel-caption." And then, of course, after the last closing paragraph, make sure you close the . We'll be doing this for all the images.
Do that three times here: with a class of "carousel-caption," and close out your s.

```
<img src="img/robbie.jpg" alt="robbie robot">
  <div class="carousel-caption">
      <h4>Robbie Robot</h4>
          <p>Trigger point Brackenridge inverse
  </div>
<img src="img/robot.jpg" alt="white robot">
    <div class="carousel-caption">
        <h4>iRobot</h4>
            <p>Trigger point Brackenridge inverse
  </div>
<img src="img/retro.jpg" alt="retro toy robot"
    <div class="carousel-caption">
        <h4>Retro Robot</h4>
            <p>Trigger point Brackenridge inverse
      </div>
```

So we've marked off where those carousel captions are going to appear, and now we have three units here: the image plus the carousel captioning is all grouping itself. Let's give each of the three image groupings another . These are s with a class of "item:"


```
<body>
    <div class="item">
      <img src="img/robbie.jpg" alt="robbie robot">
        <div class="carousel-caption">
            <h4>Robbie Robot</h4>
                <p>Trigger point Brackenridge inverse kinematic
        </div>
    </div>

    <div class="item">
      <img src="img/robot.jpg" alt="white robot">
        <div class="carousel-caption">
            <h4>iRobot</h4>
                <p>Trigger point Brackenridge inverse kinematic
        </div>
    </div>

    <div class="item">
      <img src="img/retro.jpg" alt="retro toy robot">
        <div class="carousel-caption">
            <h4>Retro Robot</h4>
                <p>Trigger point Brackenridge inverse kinematic
        </div>
    </div>
```

You may notice that once again, as with several other sections in this book, we're not using the grid system here for displaying and containing this carousel. Of course, you could take the code that we're going to put between the <body> tags, and copy that into one of the columns inside of your grid to display on your site. That's a very easy thing to do with Bootstrap.

The other thing we want to add here, for the very first item, is that we want to designate this as an "active" item. In other words, we want the carousel to start with the image we want to be first, just as we've been doing with our navigation links in previous chapters.

Now we just need to add two more s around the second and third items. Then we're going to wrap everything that you see here in another with a class of "carousel-inner," then scroll all the way down to the bottom, and we're going to close that tag just before the <href> with our arrows down at the bottom of the html page.

Then just above that with a class of "carousel-inner," as you might expect since that's called "inner," is that have an "outer" wrapper as well. So let's add one final here to wrap it all up nice and tight. This final will be a with a class of "carousel." And be sure to close that just after the <href>s with the arrows down at the bottom, which are small directional arrows pointing either way.

Finally, in that with the class of "carousel," also give it a class of "slide." This will cause the images to have that cool sliding effect from one to the next across the screen. Without the class of "slide," the images will simply flash onscreen, one after another. Then give that an id of "robots."


```
<body>
   <div class="carousel slide" id="robots">
   <div class="carousel-inner">
    <div class="item active">
      <img src="img/robbie.jpg" alt="robbie robot">
        <div class="carousel-caption">
            <h4>Robbie Robot</h4>
                <p>Trigger point Brackenridge inverse
        </div>
    </div>

    <div class="item">
      <img src="img/robot.jpg" alt="white robot">
        <div class="carousel-caption">
            <h4>iRobot</h4>
                <p>Trigger point Brackenridge inverse
        </div>
    </div>

    <div class="item">
      <img src="img/retro.jpg" alt="retro toy robot">
        <div class="carousel-caption">
            <h4>Retro Robot</h4>
                <p>Trigger point Brackenridge inverse
        </div>
    </div>
  </div>
 <a href="#">&lsaquo;</a> <a href="#">&rsaquo;</a>
</div>
```

Whew! So that's the entire markup that we need for the actual content of the carousel. Now we're going to scroll down to the bottom of the page where we have the links with the left and

right arrows, and we're going to apply some additional classes and information down there.

So, let's go ahead and apply some styling to the left arrow. We'll give it a class of "left" as well as "carousel-control." As you would expect, that's going to apply the left arrow and cause the carousel to go to the left side of the images, and then do the same for the right arrow. Go ahead and give that the similar classes of "right" and "carousel-control." You should now have two lines of code like this:

```
<a href="#" class="left carousel-control">&lsaquo;</a>
<a href="#" class="right carousel-control">&rsaquo;</a>
```

Also, rather than just the <href> with a pound sign (#) in that line referencing nothing, we're going to give this an <href> of "#robots." Just as we've seen before, the id above needs to match the <href> so that the JavaScript can work correctly.

And last but not least is the action that we want to have on this <href>, and in this case that's going to be "data-slide," with "prev"(not the entire word "previous") for the first one, and then data-slide="next" for the last one. Now you should have some code towards the bottom that looks like this:

```
<a href="#robots" class="left carousel-control" data-slide="prev">&lsaquo;</a>
<a href="#robots" class="right carousel-control" data-slide="next">&rsaquo;</a>
```

Alright! So now that we have all the information we can fit packed into these <a> tags down at the bottom of the screen, we're ready to save our document. You know the drill by now: Ctrl+S or

Command+S to save the document, and then go on over to your browser and refresh your page, and you should see a fully-formatted carousel.

As to be expected, the carousel stretches all the way across the page again in this instance, and that's because, the containing element for this particular page is the <body> tag. We don't have Bootstrap's scaffolding in place to make this carousel smaller or larger. If you click your arrows, you can see that the images slide on in and out just as we wanted. Very cool.

And we have the text underneath, describing each one of our images as well.

So the image carousel is a great way to showcase images on your website, add a little flair and interactivity, and they're very fun and engaging for the user.

AFTERWORD

I hope this book has helped you become more comfortable working with Twitter Bootstrap and within a grid system framework.

You can follow Bootstrap on GitHub and the Bootstrap blog to stay up-to-date with what's going on with Bootstrap and where it's headed. I believe Bootstrap has a very bright future, and will only improve over time as more and more people discover what a useful tool it is. I would encourage you to become involved in the Bootstrap community to really gain a sound foundation with Bootstrap and to get the most out of what it has to offer.

Really, as with any new undertaking, you must develop your fundamental skills and become familiar with the particulars in order to take your knowledge to the next level. You'll reach a point where you're able to develop an efficient and highly effective workflow, and Bootstrap is only different in that getting the basics down doesn't take very long, relatively. Trying out the different components and learning what tools are available, where they are, and how to use them to their full potential alone or with a CMS or anything you can dream up is the key, and, for most designers and developers, I think just trying it out for yourself and spending some time experimenting on a local server is very effective. Your text editor should be able to connect you via an ftp service like FileZilla so that you can see exactly what your creation will look like when it's live. Also, the resources I've included are among the best that are currently available. I'd recommend following the main players on Twitter and subscribe to their RSS feeds, or whatever is the best way is for you to stay in the loop. As I mentioned, there is a lively and growing community using and talking about Bootstrap, and you may even be able to influence

future releases while becoming engaged in its future development.

While this book was meant to get you off to a quick start, it would really get you up to speed fast if you use it in conjunction with some quality online videos, like those at NetTuts, just for example, and go through all the documentation on Bootstrap's website. If you're new to programming, there are some great, free, online courses and materials you can use to be coding in no time. You'll also pick up on coding best practices when using Bootstrap, which is an unintended bonus.

One of the great things about Bootstrap is that is updated regularly. Bugs get ironed out fast, and Mark Otto(@mdo) and Jacob Thornton(@fat), the developers, listen to the users about where to take Bootstrap and seek input from you. As mentioned, Version 3 is currently being worked on, and there are some big changes on the horizon, the biggest being that Bootstrap 3.0 will be mobile-first. Submenus may not be supported any longer, btn-info will be history, the hero-unit will then be known as "jumbotron" and so on. But, of course, nothing is final until it's final. You can follow the progress of Bootstrap online on GitHub and the Bootstrap blog.

So, like Bootstrap is a living creation, so is this book. If there is anything that you would like covered in the future, or anything that you think should be amended or removed, please let me know. I want this book to be as helpful as I can make it, to be a great resource for anyone who uses, or wants to use, Twitter Bootstrap. I believe that the more people that know about Bootstrap and use it regularly, the better it will be. You can email me at Info@FreeBootstrap.com or via Twitter @GetBootstrap. Thanks again for buying and reading the book!

~ Michael

BOOTSTRAP RESOURCES

Additional resources for your enjoyment and use may be found at:

FREEBOOTSTRAP.COM

Bootstrap has been around for only a little over a year, but in that time has attracted a lot of attention. Free and premium themes are popping up everywhere if you're looking for a quick fix, and with Bootstrap being the most-forked project on GitHub, there is plenty of support around the web, which is growing rapidly in both quality and quantity.

There are a few noteworthy third party Bootstrap projects that are worth taking a look at, and maybe using in your everyday web design and development workflow. This isn't an exhaustive list of course, but it's most of the better material I've come across that's available to you as you work with Bootstrap.

RESOURCES ON BOOTSTRAP ITSELF:

- ✓ Bootstrap Documentation Archive:

  http://bootstrapdocs.com/

- ✓ Thoughts on Building Twitter Bootstrap by Mark Otto (a creator of Bootstrap):

http://www.alistapart.com/articles/buildingtwitter-bootstrap/

- ✓ BootstrapHero.com: This is a great website dedicated to all things Bootstrap. There is a curated and comprehensive (and perhaps most importantly, updated) list of Bootstrap resources broken down into functionality. Rather than recite them all here (Bootstrap is all about efficiency, right?), I'm going to punt on this one. While on this site, make sure you visit:
  THE BIG BADASS LIST OF 263 USEFUL TWITTER BOOTSTRAP RESOURCES

- ✓ There is also a Reddit Bootstrap Group, which is active and growing: (Full URL: http://www.reddit.com/r/bootstrap)

- ✓ Love Bootstrap: http://lovebootstrap.com/ A collection of websites using Bootstrap

- ✓ Bootstrap Croc: http://bootstrapcroc.com/
  Bootstrap tutorials and resources

- ✓ Series of highly recommended videos on beginning Bootstrap:
  http://webdesign.tutsplus.com/series/twitterbootstrap-101/

BOOTSTRAP GOODIES:

- ✓ Font-Awesome: If you use Bootstrap, use Font-Awesome. It is indeed awesome.
- ✓ An easy way to use Font-Awesome is found at BootstrapCDN.com
- ✓ Color Picker & Date Picker for Bootstrap: http://www.eyecon.ro/colorpickeranddatepicker-for-twitter-bootstrap.htm
- ✓ Custom Bootstrap Button Generator(with FamFamFam icons): http://www.plugolabs.com/custom-twitterbootstrap-button-generator-withfamfamfamicons/
- ✓ WordPress Bootstrap: http://320press.com/wpbs/
  A free WP theme that integrates with BS
- ✓ Style Bootstrap: http://stylebootstrap.info/
  Customize the styles of Bootstrap components
- ✓ Initializr: http://www.initializr.com/ Set up your HTML5 Boilerplate/Bootstrap template in seconds

CSS

- ✓ A really nice CSS3 button generator: http://charliepark.org/bootstrap_buttons/
- ✓ CSS3 Gradient Generator: http://gradients.glrzad.com/

- ✓ CSS3 generator: http://css3gen.com/

HTML

- ✓ HTML ipsum: http://html-ipsum.com/ Most commonly-used HTML code snippets to quickly grab
- ✓ ReuzeMe: http://reuze.me/ HTML generator

JAVASCRIPT:

- ✓ FuelUX has some great JavaScript tools: (Full URL: http://exacttarget.github.com/fuelux/)
- ✓ JavaScript Fundamentals: http://jqfundamentals.com/chapter/javascriptbasics
- ✓ Bootstrap-wysihtml5:
  http://jhollingworth.github.com/bootstrapwysihtml5/ is a JavaScript plugin that makes it easy to create simple, beautiful WYSIWYG editors with the help of wysihtml5 and Twitter Bootstrap
- ✓ Free JS Course with interactive problems: http://www.codecademy.com/#!/exercises/0
- ✓ Learn by doing; http://www.codeschool.com/ Many JS courses reside here.
- ✓ Principles of writing consistent, idiomatic JS: https://github.com/rwldrn/idiomatic.js
- ✓ Community-made JavaScript checker: http://jshint.com/

- ✓ Generate documentation from code:
  http://code.google.com/p/jsdoc-toolkit/
- ✓ A large database of current JavaScript tutorials:
  http://pineapple.io/resources/tagged/javascript?type=tutorials&sort=all_time
- ✓ Lots of articles and video on JavaScript:
  http://javascript.crockford.com/
- ✓ Moo Tools for Bootstrap:
  http://anutron.github.com/mootools-bootstrap/

DIRECT LINKS TO EXERCISE FILES:

Resource Directory: http://freebootstrap.com/resources/

Proof

Printed By Createspace

createspace™
an Amazon.com company

Digital Proofer