*What is Bootstrap?*

Bootstrap is a free web development tool from Twitter that, with a little bit of CSS and JavaScript experience, makes building websites quick, intuitive, and fun.

There are multiple reasons to use Bootstrap:

## 1. Save Time

With Bootstrap, the first thing you do is cash in on a lot of time. The Bootstrap libraries offer readymade pieces of code that can pump life into a website. A web developer doesn’t have to spend time laboriously working out and writing code, but simply find the right piece and fit it into the structure they’re working on. In addition to this, a lot of the styling and design aspects are already taken care of, since the CSS is built with LESS.

## 2. Customizable

A great aspect of Bootstrap is that you can make it your own. You can sit down and rummage through the whole framework and keep what you need and ditch what you don’t. The whole point of Bootstrap is that it lets you factor in your own needs and tailor your development project accordingly. This is the main reason many developers feel some tools are completely useless on the Bootstrap while others revere the same.

## 3. Factor in the Design

### **The Grid Speaks**

Creating page layouts requires a decent grid. Using the platform’s grid isn’t mandatory, but it does make your job a whole lot easier. On default mode the platform provides a 16 column grid which is 940px wide. A single column holds 40px with an additional 20px acting as the gutter. Rows and spans can be manipulated as needed. Styling is completed in auto mode and a developer can get the job done just by dropping content into the HTML they’re working with. Additionally, if you need to work through columns and nesting, then you’ve got the right grid to get the job done.

### **LESS is more**

LESS has in reality gone ballistic within development circles. With its additions to the Bootstrap platform, you can use LESS mixins and CSS manipulations to customize the built-in grid. The fact that changes are automatically updated after each variable shift doesn’t hurt either. Bootstrap takes some of the most popular CSS3 utilities and brings them to life as a homogenous reaction on all browsers.

### **JavaScript**

Bootstrap comes equipped with JavaScript libraries that go above and beyond basic structural and styling. JavaScript often becomes an integral part of a web design and developers have to pull in all elements together to see the final form flourish. With Bootstrap, a developer can easily manipulate modal windows alerts, tooltips, Scrollspy, Popover, Button, Typehead, etc. The best part, however, is that Bootstrap enables you to skip writing the script altogether.

## 4. Consistency

One of the main reasons this toolset was developed was because Twitter was seeing big inconsistencies between developers working on their projects. This caused some issues both on the development front and the end-user front. Since it works with a central set of development code, Bootstrap’s results are uniform across platforms. You’ll see the same thing on Internet Explorer, Chrome and Firefox.

## 5. Updates

Take JQuery UI as an example here: updates get to users twice a year. Bootstrap, on the other hand, is ever-evolving with updates coming out on a much more regular and constant basis. As soon as web developers find a problem, the Bootstrap team starts looking to fix it.

## 6. Integration

If you’re working on a site which is already live, but still trying to iron out the 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 all you need to do from there is link up the file you are working with or hotlink to Twitter (more or less). Integration is simple, fast and easy to accomplish and once you’re done you can play with your design to your heart’s content.

## 7. Responsiveness

Bootstrap is responsive. If you shift from a laptop to an iPad and from an iPad to a Mac you won’t have to fret over your work. Bootstrap adapts to the change in platforms with super speed and efficiency.

## 8. Future Compatibility

Bootstrap comes equipped with many elements that are being considered the future of design itself. For instance both HTML5 and CSS3 are things that are going to be big in the future. Since the framework takes into account the future of design and development, it has the potential to become a yardstick for web developers in the years to come.

## 9. Competitiveness

Bootstrap isn’t the only one of its kind. It has been pitted against JQuery UI and the HTML5 Boilerplate, but a more realistic competitor is the Zurb Foundation. Bootstrap 2 has a toolset that the Foundation will have a tough time matching up to.  Third party plugins, themes, a plethora of features, codes, etc. are already up for grabs with Bootstrap – which isn’t true for the Foundation at this point. Some of you might feel that Bootstrap has a bloated feel (which the Foundation doesn’t) but that can be easily done away with through the custom Bootstrap download builder.

## 10. The Docs Talk!

Bootstrap’s documentation is impressive. The case with most newer platforms is that they don’t have proper documentation, but Bootstrap gives you a head start because you don’t have to play with everything to learn and instead can find information on just about anything through the documentation.

## 11. For the Bosses

You don’t need to spend a lot of time learning to use Bootstrap as it has taken all modular approaches and good architecture and pulled them together. From the perspective of an employer, if you give a web developer the Bootstrap framework then they won’t need to spend as much time learning, as compared to HTML and CSS. Bootstrap allows for immediate quality work, which makes it great for facilitating efficient, outcome-oriented design communication.

12 Community- There is a large, and growing, fan base of Twitter Bootstrap.

Author Michael Musgrove explores its 12-column grid layout; typography and icon libraries; fully functional components like navigation bars, buttons, and tabs, and much more. This book also shows how to add JavaScript extras like dropdown menus, modal windows, and photo carousels.

Topics include:

* + Understanding the difference between default and fluid grids
  + Nesting with fluid grids
  + Creating a thumbnail gallery
  + Adding block quotes and lists of text
  + Incorporating images and icons
  + Adding breadcrumb navigation and pagination
  + Using tabs and pills navigation
  + Making the navigation bar responsive with JavaScript
  + Adding dropdown menus to the nav bar, tabs, and pill boxes

Some Assumptions:

You should feel comfortable discussing Javascript, CSS and HTML. Although Bootstrap doesn’t require you to be a coding machine, a casual familiarity with them will make your experience with Bootstrap, and this book, much more fun. Designers the world-over long for a template or application that removes all coding, however, it’s my 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.

You need to have a text editor installed on your computer, and know how to use it. There are quite a few text editors available online that are free, and do a great job. Textmate 2, Notepad++, Sublime Text and Aptana Studio are several of the most popular. I’ve used a couple of those and have Adobe Dreamweaver CS6, and personally prefer Aptana Studio 3, which is based on Eclipse. I find it to be full-featured, highly customizable, and works cross-platform. But any text editor you choose is fine. As the cliché goes: The best editor is the one you actually use.

I’ll also assume you know the following: “Front-end” refers to the client side of a website. That’s the part of a website that’s displayed in the browser, or “viewport,” window. A framework is nothing more than a bunch of code that’s been written for you already, which you can use to customize your website.

You can currently find Twitter Bootstrap, version 2.2.2, at getbootstrap.com.

So, with all that out of the way, welcome to How to use Bootstrap!

In this book you’ll learn how to download the Bootstrap framework and integrated 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.

*Setting up your site using Bootstrap*

First things first. Direct your browser to <http://getbootstrap.com> to download Bootstrap. You also have the option in the top navigation to customize your downloaded files with 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 affects your page load speed. This also reduces the likelihood of coding errors since you don’t need to touch the code as much. For now, go ahead and click the “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.

\*\*Screenshot of Bootstrap.com home page\*\*

A few things to notice on the home page before we get started, beginning with the fact that Bootstrap (as of this writing) 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. It was recently announced, however, that it will be migrating to a home of its own at <http://GetBootstrap.com>. This reflects that as more and more users discover how easy it is to build a site using Bootstrap, it has begun to take on a life of its own. It is still managed by its original developers, Mark Otto and Jacob Thornton , however.

 Also, Bootstrap is currently licensed under [Apache License v2.0](http://www.apache.org/licenses/LICENSE-2.0). This also is about to change, and will soon be managed under an M.I.T. license, which is a little less restrictive.

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 is a JavaScript library and is processed client-side.

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 your text editor.

You may notice that I’m using Aptana Studio 3 to illustrate the coding in this particular book because I’ve worked with Aptana for a long time, and I’m familiar with it as an environment for editing code.

You certainly don’t have to use Aptana to accomplish what I’m going to show you here. You’re welcome to use any sort of program whatsoever that will allow you to edit HTML and CSS files. That’s what we’ll be doing for the most part in this book.

\*\*\*Screenshot of Aptana Studio 3 Home Page\*\*\*

List of quality text editors (most are free):

* [BBEdit](http://www.barebones.com/products/bbedit/index.shtml) (Classic Mac OS, Mac OS X, $$$)
* [Boxer Text Editor](http://www.boxersoftware.com/) (Windows)
* [Coda](http://www.panic.com/coda/) (Mac OS X, Shareware)
* [Crimson Editor](http://www.crimsoneditor.com/) (Windows, Freeware)
* [EditPad](http://www.jgsoft.com/editpad) (cross-platform)
* [EditPlus](http://www.editplus.com/) (Windows)
* [Editra](http://www.editra.org/) (cross-platform, Open Source, Free)
* [emacs](http://www.gnu.org/software/emacs/emacs.html) ([Unices, Windows, Mac OS X, Open Source, Free](http://codex.wordpress.org/Glossary" \l "Unix" \o "))
* [Fraise](http://www.fraiseapp.com/) (Mac OS X, Open Source, Free, based on Smultron)
* [gedit](http://www.gnome.org/projects/gedit/) ([Unices](http://codex.wordpress.org/Glossary" \l "Unix" \o "))
* [JEdit](http://jedit.org/) (cross-platform)
* [Kate](http://kate.kde.org/) ([Unices](http://codex.wordpress.org/Glossary" \l "Unix" \o "))
* [Komodo Edit](http://www.activestate.com/komodo_edit/) (cross-platform, Open Source, Free)
* [Kwrite](http://www.kde-apps.org/content/show.php?content=9901) ([Unices](http://codex.wordpress.org/Glossary" \l "Unix" \o "))
* [Notepad++](http://notepad-plus.sourceforge.net/) (Windows, Open Source, Free)
* [phpDesigner](http://www.mpsoftware.dk/phpdesigner.php) (Windows)
* [pico](http://www.washington.edu/pine/) ([Unices](http://codex.wordpress.org/Glossary" \l "Unix" \o "))
* [PSPad](http://www.pspad.com/) (Windows, Free)
* [Smultron](http://smultron.sourceforge.net/) (Mac OS X, Open Source, Free/$) Smultron 4 (req OS-X Lion) (v.cheap in MacApp store)
* [SubEthaEdit](http://www.codingmonkeys.de/subethaedit/) (Mac OS X, $)
* [Sublime Text 2](http://www.sublimetext.com/2) (Windows, Linux, Mac OS X)
* [TextEdit](http://www.apple.com/education/accessibility/technology/textedit.html) (comes with Mac OS X)
* [TextMate](http://www.macromates.com/) (Mac OS X, $)
* [TextPad](http://www.textpad.com/) (Windows)
* [TextWrangler](http://www.barebones.com/products/textwrangler/index.shtml) (Mac OS X, Free)
* [UltraEdit-32](http://www.ultraedit.com/)
* [vim](http://www.vim.org/) ([Unices, Windows, Mac OS X, Open Source, Free](http://codex.wordpress.org/Glossary" \l "Unix" \o "))
* [Notepad2](http://www.flos-freeware.ch/notepad2.html) (Windows, Freeware)
* [WebTide Editor](http://webtide.eu/) (Windows, Linux, Mac OS X, Freeware, Java)

So here we are in Aptana, I’m in the default designer view, as you take a look at the screen here.

The very first thing to do is to make a new site in a local folder. We’ll name it “Bootstrap.”

If you unzipped your downloaded Bootstrap files to your Desktop, go there and add a new file to the folder by right clicking and choosing, and inside of that are those 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 HTML page type, with the file extension of .html or .htm (your choice, but be consistent in your programming).

For the doc type, make sure you choose HTML 5, because that is what we will

be working with for Bootstrap, which is specifically designed to work with HTML 5 You should see a document very similar to this on your web page:

\*\*\*Screenshot of formatted index.HTML\*\*\*

We need to insert a meta tag for our responsive design. This is a meta tag indicating that the viewport

starts with an initial scale of 1.0. This is important in responsive design. Go ahead and copy the following snippet by doing Ctrl+C or Command+C to copy and then Ctrl+V or Command+V to paste that line of code right into place, just after where it says meta character set of UTF-8:

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

Now you’ll want to change your title from Untitled Document to your site name, whatever that may be. For our examples, I ‘m just going to use “How to Use Bootstrap” which is just the title of the book.

The next thing we need to do is attach the style sheets that we’ll be using in this particular

document. Locate your Bootstrap CSS folder, and inside of that, you’ll

find that there’s actually four stylesheets that are available to you, and you’re probably

wondering what the differences are between these.

First of all, you’ll see that there are two sets of two, really, bootstrap and bootstrap.min,

bootstrap-responsive and bootstrap-responsive.min. The differences here are the .min files are what are called “minified” and they’ve been processed so that all the space that is normally present in a stylesheet file has been taken out. That will help the files download more quickly, 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, 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, go ahead and link to the full version of these

stylesheet files in case we ever want to go and take a look at them, which is highly probable.

We want the more human readable format, so link directly to just the regular Bootstrap file here first. Your text editor may warn you that it’s going to use a file path until I’ve saved my document, which we’ll save in just a moment, so go ahead and say OK for that and OK again.

So this has gone ahead and attached that bootstrap.css file to my document.

Right after that, we are going to attach the second stylesheet, and we want the bootstrap-responsive stylesheet. \*!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.

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.

\*\*screenshot of file tree

Go ahead and call this index.html, always a good named to call an HTML document, and

you’ll see that the paths here inside of your text editor adjust accordingly and correctly to take

a look at those two stylesheets.

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 a

extra space here inside of your <head>.

You’ll see that I 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 backwards compatibility in Internet Explorer browsers. This itself is not part of Bootstrap, but it will help make your websites more compatible

with Internet Explorer, and you can just drop this right into your Bootstrap document

to improve backwards compatibility, particularly with IE.

<!-- HTML5 shim for IE backwards compatibility -->

<!--[if lt IE 9]>

<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>

<![endif]-->

So your index.html page should resemble this:

<!doctype html>

<html>

<head>

<meta charset**=**"utf-8">

<meta name**=**"viewport" content**=**"width=device-width, initial-scale=1.0">

<title>Untitled Document</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>

<script src**=**"http://code.jquery.com/jquery-latest.js"></script>

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

</body>

</html>

So go ahead and highlight that code, do a Ctrl+C or Command+C to copy, and inside

of your text editor Ctrl+V or Command+V to paste that on in. Again, this is an optional bit of code, but I’m recommending that you add it. I think it’ll save you some frustration later on as you do more with Bootstrap.

Alright, so now if you scroll on down your page a little bit you’ll see that we have

a starting <body> tag and an end </body> tag. Put a few spaces in between those and just before

the end 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 /script tag. This itself will load the latest version of jQuery, which is a JavaScript framework from the jQuery website.

In order for this line of code to work, make sure that you are on the Internet so

that it is able to download this code. If you’re not on the Internet, you may want to go to jquery.com and download the latest version of jQuery, creating that as a file and linking that to your file, but if you use this

specific line of code, even if jQuery continually changes its version, the latest version (whatever

it is) will always be located here, and that’s a great way to keep your Bootstrap site up to date.

The next thing that we’re going to add is a line of code for the specific Bootstrap JavaScript,

and if you take a look in your local files, you’ll see once again we have two versions of Bootstrap JavaScript.

One is bootstrap.js, and the other is bootstrap.min.js. Once again, this is the minified version of the code.

You can see the difference between 55K for the bootstrap.js and 25K for the minimized version of this.

By using the .min versions, you’ll definitely save some download time for mobile phone users

in particular, and certainly those on dial-up as well.

Since we’re not going to be taking a look at JavaScript in great detail in this book,

I’m going to go ahead and link directly to the min version of the JavaScript file.(Decide if I want to append/include some JS resources for the reader)\*\*

Once again, include <script src=””>, and insert the path to the .min version of the JavaScript. Close the script tag: </script>.

So this basic document that you’ve put together is a great point for starting any Bootstrap web page.

It’s got everything that you need here; it’s got the links to the stylesheets in the right

order, it has HTML5 shim for backwards compatibility with Internet Explorer, it has a meta tag,

that’s important for responsive design, and down here, towards the bottom of the page,

we’ve linked to both of the JavaScript pieces that are required to make Bootstrap work.

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 difference between the default and fluid layouts**

Bootstrap ships with a 12-column grid system that comes in two varieties. If you take a look at the Bootstrap website at getbootstrap.com, and you click on the Scaffolding tab on the top of the page, and you click on the link on the left side for the grid system, you’ll see the Bootstrap’s default grid system.

This is the default 940 wide grid:

\*\*\*INSERT IMAGE OF 12 COLUMN GRID HERE\*\*

It’s designed to adapt somewhat via adaptive layout to different screen widths. The numbers indicate how many columns wide each box is. In the first row, there is only one column per box, the next row is grouped into a few columns. If you roll your mouse over each box in the documentation at getbootstrap.com, there is a tooltip that will come up to tell you how wide each box is. That tooltip only works in the documentation, it’s not going to work on your live site.

Now you might be wondering, I just told you this is a 12-column grid, but yet when you

look at this example here on the Bootstrap website, you’ll see that it’s only 9 columns, why is that? Well, the whole entire getbootstrap.com site is built in Bootstrap itself. The left column on this particular website is three columns wide, leaving nine columns left over for the main content area.

In this example, for how the grid HTML works, it says 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 if you click on the link for the Fluid grid system--we’ll jump down the page

to that particular example--the fluid grid system is a true responsive grid. Note that it is 12 columns wide, 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 the columns of 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 of 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 defined by Ethan Marcotte in an article published at alistapart.com in 2010.

Marcotte said that responsive design contains three specific aspects: 1) fluid grids, 2) flexible images, and 3) media queries.

1) A fluid grid is one that’s built using a relative unit of measure 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 big-screen, and they scale down as the size reduces.

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 described on the alistapart.com

website in 2006, by Marc den Dobbelsteen. This uses a fixed width grid instead. In the particular implementation we’re looking at, the layout of the screen resizes once specific pixel widths are reached.

This design may not fill the entire screen at all times the way a fully-responsive layout does.

Many people, including Jeffrey Zeldman, dubbed “The King of Web Standards” by *BusinessWeek,* would like to see a broader definition of responsive design; one that will encompass adaptive design and adaptive layouts. However, these two approaches are 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 always, it depends on what your goals are for your website.

In general, most designers, including this writer, seem to be 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 primary 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 right here inside of the body tag building out some of the grids,

so you can see how this works.

\*\*\*SCREENSHOT of current index.HTML here that we’re beginning with\*\*

So in general, when you start a HTML web page inside of Bootstrap, you are going to start

it with a div, which is pretty standard when constructing a website.

You’ll have some kind of wrapper that goes around the entire web page, and inside of Bootstrap

that would be a div with a class of **container-fluid**. Let’s start by going ahead and putting that in place along with a closing div down at the bottom. So that’s going to encompass your entire web page; that’s the outside wrapper. Then the next thing I am going to add to this is the start of my row.

If you have a div with a class of row-fluid, this 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 might have another couple of divs, so let’s say you have a div with a

class of span3. In other words: span3 columns and put something inside of it.

So we could say, for example, here’s a paragraph, here is a left column on this web page, and

end that particular div. Then I can have a second div inside of this row. In this case, we have nine columns left over. So we’ll put in a span9, in order to always add up to a total of 12. Remember 12 columns. Incidentally, Boostrap was originally a 16-column, 960px layout, but was changed.MAY WANT TO INCLUDE WHY BOOTSTRAP CHANGED-LOOK ON QUORA

So span1 through span12 are all the classes that are available to you, and each one of

those classes indicate how many columns it should expand over. Alright!

So in your html file, have a starting div and an ending div, and then inside of that you’ve got some text.

Here is the main content area of this web page:

\*\*SCREENSHOT OF CURRENT HTML FILE HERE\*\*

I put just a simple paragraph inside of each one of these divs 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 divs. I am making this very simple as the first example.

So now, if we go ahead and save this web page, File > Save, go ahead and view

this inside of Firefox. You may very well (and should) have a way to do that in your text editor by typically clicking on a Globe icon on the top and clicking “Preview in Firefox,” or Chrome, IE, Opera, Safari, or whichever is the browser you’re interested in and have configured.

And you will see here that we have the start of a grid on the website, we have our left

column, and we have a main content area. The main content area doesn’t look that large because I only put in one sentence, but if you put in a more extensive chunk of text, you would see that it stretches all the way across the web page.

Remember at this point that Bootstrap was made 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.

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

I have done this with divs to start with because most developers/designers understand divs, and

there are number of you who are just learning HTML5 at this point in time.

So one of the things that we might want to have in our particular web page is to include a header at the top of our web page. Pretty standard for a website, right? So inside of our div with a class of container-fluid, I’m going to go ahead and put in a header for this particular web page, and in HTML5, the tag that you use for this 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.

So, no extra div 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>, set this to have a class of span12 because we want it to go all the way across the web page.

\*\*INCLUDE SCREENSHOT OF THIS HERE!\*\*

Now we have better markup there for our header. Let’s go ahead and fix what we did before with all of those divs in what is now the second row, we have the div with a class of **row-fluid**.

That might be a section of your web page, so we can change the div class of **row-fluid**

to a section with a class of **row-fluid**, and then inside of that we 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 particular area, it might be

an **aside**, so an **aside** with a class of span3, and it’s possible and quite likely, in fact,

that whatever goes on the right side of this page is an **article**. So this might be the article tag, an article with a class of span9/article.

So, now that you have two rows in your particular web page (you have a header, and you have a section) go ahead and save this web page and see what it looks like in a browser.

\*\*PLACE SCREENSHOT OF WHAT WE HAVE WITH HTML5 TAGS IN BROWSER HERE\*\*

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

Go back to your text editor, just to show you how that might work. Perhaps you want to add a row that has a very large left column and a very small right column. You’ll add a section, once again, to the class of row-fluid, and inside of that, perhaps 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. Here is another article, /article and maybe this time we put the aside over on the right side, so aside class of span3, so maybe we

have some more information about that article over on the aside, and close that, and you can close the section.

\*\*SCREENSHOT OF WHERE WE ARE HERE\*\*

So once again, just go to File > Save, and if you take a look at my web browser, you’ll

see that now you have a second row of information. We have our article 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. It’s in this case

displaying in a right column.

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 footer tag for doing this.

So in this case, <footer> will have a class, and just give this a span of 12

so that it goes all the way across the page, and you may want to include some text in here, such as your Copyright, and some bottom navigation.

And then you should make sure that your closing div is down there at the bottom. That /div 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 have a

footer that’s in place at the bottom of the web page.

\*\*\*SCREENSHOT OF FOOTER IN PLACE\*\*

Finally, you might be wondering about the HTML5 markup that you just put into place, and

That’s currently being used in the web page that you just built. Bootstrap, as a system and a framework, indicates that it’s mostly backwards compatible to IE7. However, sometimes you are going to find backwards compatibility issues with older browsers, particularly if you’re working with these new HTML5 semantic tags that you just used, such as header, footer, aside, article, and others.

**Modernizr** is a JavaScript solution that can help make your Bootstrap website HTML5 backwards compatible with older browsers. It’s not perfect, nor is it the only solution to make this work, but it is a solution that’s widely used, and one you might want to try if you find your Bootstrap work is producing

bugs in older browsers that you wind up having to support. You can find this by going to modernizr.com, and that website will explain everything that you need to know about Modernizr. Modernizr itself is beyond the scope of this book, but you can find it covered fairly comprehensively online. Modernizr is, of course, compatible with Bootstrap.

*Understanding Fluid Offsetting*

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

\*\*SCREENSHOT OF CURRENT WEBSITE\*\*

Let’s say what you really wanted to have happen is you wanted to have a left column of 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 I had the articles all lined up one after the other.

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.

We can set this up such that you can have the two articles appear underneath each

other and not have a left column in the second row. (Remember that every row has to add up to 12.) There has to 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 make a couple of quick changes to your page.

Just to make your articles stand out a little bit more here in the second row, add an <h2>, and call this **Article title**, and then down in the second row, get rid of the aside with class of span3. Also, clean up the markup in this particular article a little bit as well, so it has a heading, another article, and of course, underneath that would be a paragraph with whatever text happens to be contained in that article.

SCREENSHOT OF CLEANED-UP HTML IN TEXT EDITOR\*\*

So, now what we need to do is inside of this third row that you see here where we have a span9,

but we don’t have the other three columns accounted for just yet. So, the way to do this is to use the offset. The way to utilize it is: span9 **offset3**, all one word.

These offset classes, again, 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 columns on the left side of the screen.

That will be blank, followed by the article, which will occupy the 9 columns on the right side.

Go ahead and save this page, File > Save, and I put this 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 a whole lot better on the page, or at least it looks a whole bunch better than it did before.

We may also want to do something with this copyright while we’re here. Right now the copyright is located down in the bottom of this web page. It’s got a span of 12 on it, so it goes all the way across the bottom.

Perhaps what you really want to do is shorten that up and have it appear just on the right side of the page.

You can do that also with offsets. So go 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 alt+169), 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 have 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 are going to contain some kind of content.

*Nesting with Fluid Grids*

On occasion, 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:

\*\*\*PLACE SCREENSHOT OF WHERE WE ARE HERE\*\*\*

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, because one will be four columns

wide and one would be five columns wide.

However, there is another approach that we can take to this.

We could **nest** another row inside of the row so that we would have 12 columns to work with,

and then divide those 12 columns into 2 pieces of 6 columns each. This is called a **nested row** inside of Bootstrap.

If you go back into your text editor and open up your HTML web page, and scroll on down past that second article, we can add some more code. Add this as a **section** with a class of **row-fluid**, and put it

in a div with a class of span9 offset3, and you may want to close those so you don’t lose track of those two particular tags.

If you went ahead and put it in your boxes here, at this point we could have one that

was four columns wide and one that was five-sized columns. But let’s say what you want are two evenly-sized boxes. Do that, just go with an inside div with a class of span9, offset3. Go ahead and add another row.

So inside of this div, go ahead and add another div with a class of row-fluid, just exactly the way we’ve done before, and then inside of that particular div you’ll put in your two information boxes.

\*\*SCREENSHOT OF HTML WITH NESTED ROW HERE\*\*

So you could put in here a div with a class of span6, and inside of that you can put an h3,

and we’ll call that Info Box 1, and perhaps you want an image to appear here. \*\*You will need to have copied these images into your images folder via FTP or whichever is your preferred method of file-handling. If you’re looking for an FTP program, the one I use and recommend is **FileZilla,** which can be found and downloaded at **FileZilla.com**.

So you can go ahead and add those here with an image source tag <img src=””>, and you can browse for those files to put the path between the quotation marks. Inside of your Bootstrap images folder, you could also put in a paragraph of information about the image. You can copy this one little snippet of

code here (this div with a class of span6) highlight and Ctrl+C to copy it, and

right after that closing div, paste that on in place. You can add another image here, so what we have looks like this:

\*\*SCREENSHOT OF ADDDING IMAGES HERE\*\*

So we have Info Box number 2, and it will be your 2nd image instead of your 1st, of course.

Now we’ve gone ahead we’ve added an additional row to our grid and inside of that row we’ve

nested another row, and so we have to make sure all of those divs are closed and everything

is looking good. Go ahead and do a File > Save, 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 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 go further inward you’ll see that the grid continues to adapt, so what you see going on 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.

\*\*SCREENSHOT OF SHRINKING THE BROWSER HERE\*\*

Notice that those images are continuing to resize as well, and if you scroll on down the

page, you can see that everything is appearing stacked on top of each other, which is 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 create any rows in it, and we’ve taken a look

at fluid offsetting--so the concept of having some items shifted over a bit on the page--

and finally, we’ve seen how we can nest a row inside of another row to achieve even finer control on that grid.

*Exploring Bootstrap Media Queries and device-specific styling*

One of the last things you might want to do working with your grids in responsive design

inside of Bootstrap is that you may wish to have some particular parts of the layout visible

only on a desktop, or visible only on a tablet or only on a phone, or you may want to

hide on one of those devices, and Bootstrap includes some classes that will allow you

to do exactly that.

Head to getbootstrap.com, and under the Scaffolding tab under Responsive design, if you’ll just scroll on down a little bit on this page, you’ll see a very useful chart here under Responsive utility classes.

\*\*INCLUDE CHART OF RESPONSIVE UTILITY CLASSES HERE\*\*

So this explains to you exactly how these particular classes work and what particular

devices using those classes will produce content that’s visible or invisible on certain devices. We’re going to put those to work here in just a minute.

So, what we would like to do now is to set up a row on the web page that we’ve been working

on so far, and set up such 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 hint here is these are CSS, so they are testing the width of the

window of the browser. They aren’t actually testing the device, which is something very

different that actually comes over as part of the header in the request process for getting web pages. This is just testing the width of the screen, so it’s calling things phone, tablets, and desktops, but just by making your screen wider or making your screen narrower, you’ll be able to simulate phones, tablets, and desktops, and they will work just fine.

So again, remember that phone, tablet, and desktop actually refer to specific widths of

the screen, and those widths are detailed in that chart. Phones are 767 pixels and below, tablets

are 768 pixels to 979 pixels, and desktops are the default, something wider than 979 pixels.

So inside of your text editor, if you scroll it down to just before the footer--let’s

go ahead and add another part of this web page. Let’s add a section with a class

just as we’ve been doing all along here of **row-fluid**, and inside of here add three divs, so you’ll have a div with the class of span4. This is visible on the desktop. Now add a second div with the class of span4. This is visible on a tablet, and then finally, another div with a class of span4. This is visible on the phone.

Now what you’ll need to do is add classes here. So in addition to this class of span4, if you just put a space, that will allow you to play more than one class per particular HTML tag, and let’s add “visible-desktop,”

on the second one we’ll add “visible-tablet,” and on the third one we’ll add “visible-phone.”

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 the 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 the tablet**, and now as I pull the screen even further over,

and it resizes again, you’ll see down here at the bottom it says **This is visible on the phone**. That’s the way this particular classes work.

The hidden classes work very much the same way, and you can give those a try on your own.

*Adding a Hero Unit*

A hero unit is a large area usually spanning the width of a web page that intends to attract the user’s attention. Here’s an example:

\*\*ADD EXAMPLE OF A HERO\_UNIT HERE\*\*

So, we’re starting from the very beginning of a Bootstrap document here. So as we’ve

done before, we’re going to start by adding a div with a class of container-fluid to have that big wrapping div that goes all the way around the web page, and then close that div down at the bottom.

And then just as we’ve been doing all along, we have to start a row, so we’ll have a div

with a class of row-fluid and a slash div down at the bottom, and then in between that we’ll go ahead and put them in your hero unit.

\*\*SCREENSHOT OF WHAT WE HAVE HERE\*\*

Now, in this particular page, the hero unit is really functioning as a big header at the top of the web page. It’s the huge thing that catches your attention, so let’s use the <header> tag from HTML5, and let’s give this a class of span12, because we want it to go all way across the page. Go ahead and close that header down at the bottom. Let’s go ahead and add some text to this.

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<p>, and follow that up with your text, which can be whatever you like for this demonstration, including just copying and pasting some Ipsum-Lowrem. Include link to Ipsum Lorem to the web page, and close you <p> tag to end that paragraph.

Let’s follow that up with a paragraph and a link (href). We’re not going to link this anywhere, because this is just one page of the website. Obviously, you’d put a link in there if you’re doing this for real, and we’ll put in a “learn more” button at the end of all of this.

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 in the “Learn more” link. It’s not really anything exciting yet, and it’s not anything that we haven’t already done in some of the previous sections.

Alright, so now that that’s in place, let’s go ahead and turn this into a hero unit, and

the way that we do that is after the span12 we can simply add the class of “hero-unit.”

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’s got a screen black box around-is this what it really looks like? it, the text is nice and large, it’s definitely attention grabbing.

One other thing you might want to add to this would be a picture. You can certainly

add an image to this very easily. If you hop back into your text editor, and go ahead and

add an image just after the header, you can put an image source. Make sure to give it an alt tag always, always for SEO and accessibility.

Actually, before we close it, let’s put in one more thing: a class of “pull-right.” The

class of pull-right is a utility class. There’s actually a pull-left, and there is a pull-right.

Basically, 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 you take a look, there should be your nice big image there to complement what you’ve done so far.

\*\*INCLUDE SCREEN SHOT OF IMAGE INCLUDED\*\*

The last thing that we want to do here, in order to make this look like a real hero box, is deal with

that “Learn more” link, and we’ll cover buttons in much more detail in a later chapter. For now in your text editor, if you head over to the href tag, we’re going to go ahead and assign three classes.

So let’s add a class of **btn** or **button**, **btn-primary**, which(review this) controls the color of the button actually, and we’ll also add “**btn-large**,” which will make it a very large button.

\*\*INCLUDE WHAT COLORS FOR WHICH BUTTONS< PLUS OTHER BTN FEATURES\*\*

So, once these three classes are in place, 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 “Learn more” link,

even though that’s just a regular old href type of thing. The styling that we’ve applied with Bootstrap gives our button a very nice look.

So hero units are a great way to set off your homepage to really draw the user’s attention

to the website and make users aware of what’s most important on the web page. In this case, we’ve marked it up with a header tag. It certainly doesn’t have to be marked up that way, it doesn’t even have to occur at the top of your web page. You could use this hero unit anywhere you’d like within your document, so you can go ahead and give this a try on your own Bootstrap websites.

*Creating a Thumbnail Gallery*

Thumbnail galleries are useful for displaying many images on a web page where the image is the main focus. You can usually click on these to get a larger version of the image. You can also use a thumbnail gallery for displaying photos and text on a page in an organized manner. Bootstrap does include some markup for creating a thumbnail gallery. So let’s go ahead and take a look at that now.

If you hop back into your text editor again, and scroll on down, past the div class row-fluids, in this final/div, we’re going to go ahead and add another row to our document.

\*\*INCLUDE SCREEENSHOT OF WHERE WE ARE HERE\*\*

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

Let’s go ahead and start that with a <ul>, and give that a class of “thumbnails.” Of course, don’t forget your closing </ul> somewhere down at the bottom. Most text editors handle that for you, but just in case. Once that’s in place, the next thing we need to add our list items <li>.

For our example li, we’re going to have three thumbnails on our page. We want to have them all next to each other in three columns. Since of course, we’re working with rows and a 12-column grid, that would indicate 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,”(no ‘s’) and inside of that we’ll put in our image. Choose your photo here, and again, make sure to give it an Alt tag. We can go ahead and give this <h3> for markup with your image title, followed by some text. You can go grab some Lorem Ipsum if necessary for playing around with. Go ahead and copy that text-- Ctrl+C or Command+C to copy.

Put this is in as a paragraph <p>and then Ctrl+V or Command+V to paste that on in and close it with your </p>. So that’s our first thumbnail in our grid. We have a single li; we have an article inside of that.

If an article isn’t appropriate to what you’re doing, you could use a <div> or any other tag

That’s appropriate to nest inside of that <li>. In the below example, I’ve put in my image, my h3, and my p.

\*\*INSERT HTML WIT HIMAGE< TITLE< ETC>>> HERE\*\*\*

Now, if you’re doing this as a real thumbnail gallery in which you just have pictures, you

don’t have text following it the way I do here, where I have an article class of “thumbnail,” which you could use that as an href which you might 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 this. The problem with doing that here in this type of context is that <h3> and <p> are block level HTML tags.

You cannot nest the block level HTML tag inside of an in-line element like the <a> tag. So, in this particular case, I’m using the article class here for containing those block level elements. I could certainly wrap an <a> tag around my image if I wanted to do that to make my image clickable.

Okay, so now that we’ve got that unit in place, let’s go ahead and copy what we’ve just done. Just highlight those lines for the <li> and then Ctrl+C or Command+C to copy, hit enter a couple times, 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 <li>.

Insert the image for #2 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.

The last one that we’re dealing with is image #3, so go ahead and copy that text

here, paste that into your text editor document, and put in this image’s right here. And then, of course, we need to change the image source and 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.

\*\*INSERT WHAT WE HAVE AT THIS POINT HERE\*\*

There’s that 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 right underneath all of those.

If we make the browser window smaller, you can see that these thumbnails do 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 each other as part of that grid collapsing as the screen size gets smaller. And as you can see if you shrink your screen, 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 nicely.

\*\*INSERT IMAGE OF ME SHRINKING SCREEN SIZE\*\*

So the Thumbnail Galleries are very useful. You can use them in a variety of different websites for different puposes, from displaying a portfolio to on en e-commerce website. I’ve shown you a Thumbnail Gallery here and we’ve had some descriptive text following each of these. 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*

## List of components

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

* 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

In future guides, we may walk through these components individually in more detail. Until then, look for each of these in the documentation for information on how to utilize and customize them.

**Exploring Basic Typography**

Bootstrap comes with a number of built-in styles which you can use in your site designs. There are a few in particular that you might find quite useful. This is the latest HTML document we have on our hands:

\*\*FIND OUT WHICH FILE SCREENSHOT TO PLACE HERE\*\*

This is schedule.html, and you see that the page is kind of boring looking:

We’ve got a schedule. It’s obviously a schedule. It’s divided very nicely into sections here. We have a Monday schedule, and we have a Tuesday schedule. And it’s easy to tell the hierarchy of the information, but it’s not terribly exciting to look at. So we’re going to add some styling to this.

You’ll notice there’s some styling that’s here by default, just by virtue of being linked to the Bootstrap style sheets. For example, the text is displayed in Helvetica and Arial, as opposed to being displayed in just Times New Roman, which is the default if no styling is set (for most web browsers anyway). There’s some basic styling that’s in place for the headers. These “Learn more” items are links, which you can see as you roll your mouse over them, they become underlined, and they aren’t the standard blue. So there is some basic styling that’s here. It’s just not a lot of styling, and it is still a very boring page. So let’s see what we can do about improving this inside of your text editor.

Open up this document 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 schedule-- it’s just sort of a big summary of everything that happens to be going on.

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 something called “Lead.” So this particular paragraph tag, let’s give it a class of Lead. If you go ahead and save this, File > Save, and you open this on up in your browser of choice and take a look at what you’ve done, you’ll see that that leading paragraph has a little bit bigger text. It also has a little bit more spacing between the lines and it stands out a little bit more than it did previously.

Something else that we can do to this page--it’s a very simple thing to do--would be to add

some <strong> emphasis to the page. <Strong> of course, is the HTML tag which many people think of as making things bold. Remember, of course, it doesn’t have to make things bold. Bold is the way something looks; it doesn’t have to do with the actual markup of that tag, which is to strongly emphasize something.

So what we’d like to do is to take these names like Art in Full Color and Art

in Full Bloom, and I’d like to emphasize those within the text by making them strongly emphasized. And as for the artist names, I’d like to emphasize those.

We can highlight the words that we’d like to strongly emphasize, and click the B button down here in the bottom that actually adds a strong tag, as you can see here in this little line of HTML tags. I call it the breadcrumb, where you can see that the strong tag has been added.

\*\*Explain how to add strong and em tags here\*\*As we go a little bit further down the page, we’ll see LaVonne L. LaRue. I like to emphasize that, so I’m going to click the I tag down here. Notice that gives it the emphasis tag. And if we flip over to code view, you’ll see that it is has in fact put in that markup there correctly. Both on line 22 here, it’s strongly emphasized Art in Full Color, and it’s emphasized LaVonne L. LaRue. If I just save that, and I take a look at it again in Firefox, you’ll see that those words Art in Full Color are bold, at least appearing bold here, and LaVonne L. LaRue is emphasized here, and you can see that in the form of an italic name.

Go through the rest of your document and emphasize Art in Full Color where it appears. Also emphasize Water in Art where it appears in the second section of the page. And we’re going to emphasize all of the artist names that appear inside of this document.

*Including Blockquotes*

In the middle of the page we’ve been working on, between the two days of events, there’s

a testimonial. It’s really, really easy to miss it, because it’s pretty much buried in 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 your text pop out just a little bit more.

So if you head over to your text editor we’re going to scroll down the page here to line 38, and you’ll see that \*\*EXPLAIN HOW TO MAKE A BLOCK QUOTE\*\*we have the quote, “This is simply an amazing conference. I’m inspired by it each year I attend.” That’s a paragraph, followed by a paragraph with a person who said it, Katie Cross, and we’re going to go ahead and give this a little bit of styling. It’s not really marked up correctly anyway, because since this is the quote from somebody, it should really be contained within a <blockquote> tag. So we’ll start with that, we’ll 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 bigger. You’ll also notice that we have a little border over on the side which makes it stand out just a little bit, so that’s a nice treatment as well. The next thing that we want to do is deemphasize the person’s name. We want the quote to be more important, so the text should be a bit bigger, and

the person’s name should be a bit smaller.

One of the ways that we can do that is with a <small> tag. A small tag is one of these

tags that’s come back in HTML5, so we can go ahead and add that now. Right around the name of the person, instead of just leaving it the way it is, we can put in <small> tag and </small> on the other side of the person’s name and, if we save that and we hop back to the browser and refresh the page, you’ll see that the person’s

name becomes in fact small, and it’s also a lighter gray, and scaled back in emphasis just a little bit. 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 I’d put in front of the name before when we typed in the text really isn’t necessary anymore, so we can get rid of that, so the styling for that quote is a little better.

\*\*INCLUDE SCREENSHOT OF THIS BLOCKQUOTE\*\*

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, and so in your text editor, if in the <blockquote>

tag go ahead and add a class of either “pull-left” or “pull-right.” This will float your quote on the page.

Save that again and take a look at it in your browser. You’ll see here that the quote is here, and it is indeed floated left. It’s caused the heading here for Tuesday, March 6, to be pulled up on the page, which is what happens when you have a float that isn’t cleared afterwards. So that’s a treatment that will work on some pages.

It certainly doesn’t work on the page that we’re looking at here, but it may be something

that you want to do.

Remember, pull-left and pull-right are two classes that you can insert pretty much anywhere

inside of a document that will cause that element to float left or float right. This one obviously would need a bit more styling if we decided to use it here. And of course it’s possible to do custom styling inside of Bootstrap. We’ll cover that in a later section in this chapter: the correct way of doing custom styling with Bootstrap to keep your website maintainable.

So I don’t really like the way that looks. I’m going to go back and pull that class out in Your text editor.

So I’m just going to get rid of that class of pull-left, but just know that it exists

in case you happen to want to do that.

So this is a nice built-in styling feature here inside of Bootstrap.

We’ve got some start for styling the quote and making it stand out of the page a little bit. You can certainly enhance that with your own styles to give it a little bit more polish and stronger treatment if you wish.

*Including Lists*

Down at the bottom of the page where it talks about Exhibit Halls A & B, we have a list. Right now, it’s a single bulleted list that’s about smart displays that will be occurring in Exhibit Halls A & B throughout the week. And then under that we have a sentence that says to visit three local photographers who will be displaying their works, and it says something like Sam Gilmore: Ocean photography.

Currently, those are marked up as individual paragraphs, and that is actually rather incorrect markup. It’s still a list of things; Sam Gilmore is the photographer; Ocean photography is his topic.

They go together, but they’re not individual list items the way Area high school students

or Roux Academy students or Conference participants are. The best way to mark up that particular laundry list of stuff down there at the bottom is really a **definition list**.

A definition list is something that is not typically used a whole lot inside of web pages

Right now, but they’re incredibly useful. It’s got a two-part list, so there’s a definition term and a definition itself, so things can 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 you the bottom of the page. I’m at line 62. First of all, let’s take a look at that bulleted list.

\*\*\*SHOW WHERE I AM ON PAGE WITH SCREENSHOT\*\*

As you can see, those are displayed right now with individual bullets, if we wish. We don’t have

to display the bullets at all by taking this <ul> and adding a class of “unstyled,” and by adding that class of “unstyled” we’ll get rid of the bullets in the list.

So now if you refresh your web page, you see there’s your list under Exhibit Halls A & B,\*\*?what to do about this…\*\* and the bullets on that list no longer exist.

Now I don’t think that works at all. I liked it much better as a bulleted list. I thought

it was much easier to read that way; however, I wanted to be sure to mention unstyled lists,

which is a class that you can use on both ordered lists, the <ol> tag and unordered lists, which is the <ul> tag. You, of course, could then layer other styling on top of your list once you’ve turned the bullets off via the unstyled tag. Also note that it’s gotten rid of the margin or the padding that goes around that list as well. All in all, that’s a pretty handy style to know about. I’m not going to use it on this page, but now you know that it exists.

Okay, so back to your text editor. Take that class of “unstyled” out again, and then down at the bottom is that group of photographers and their specialties. 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 the <dl> tag, which 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.

\*\*INCLUDE ILLUSTRATION OF WHEN YOU’d WANT TO USE A DL\*\*

Then we’re going to have our first one here, Sam Gilmore.

We’ll change this paragraph tag <p>to a <dt>, which stands for **Definition Term**.

And Ocean photography we’ll change to a <dd> which is the *actual* definition. And although we’re not thinking of these as dictionary terms with 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:

\*\*\*SCREENSHOT OF A <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 the Sam Gilmore presented as a bold name and

then underneath of it Ocean photography is slightly indented. It’s pretty clear that Ocean photography goes with Sam Gilmore, for example, so that’s a very nice-looking list.

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.” What that will do, as you might predict, is place the name over on the left and the occupation or interest over on the right. Again, if you save this page and put it into your browser and refresh, you’ll see at the bottom of the page now that we have a list of photographers, and they’re still bold in with their interests over on the right side, which is also a very, very nice presentation.

So Bootstrap has a few treatments for lists. These are lists that are just your basic content list 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 session names on this program.

They aren’t styled very nicely. It would be so much better if they looked more interesting

and certainly more compelling for the user to click on them.

\*\*SHOW ABOVE LEARN MORE LINK\*\*

Fortunately, 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 wanted to. 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 do right now.

Back to the text editor and scroll on down to the first instance of these “Learn

More”s, which I think is around line 24, Art in Unexpected Places. 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 treatment. One thing that we could do is simply use that <small> tag which we covered in a previous chapter, and you can wrap the <small> tag around the link itself, and if I just save that, 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 session anymore as it did before. If you take a look at Art in Full Bloom, it looks like it’s Art in Full Bloom learn more as opposed to Art in Full Bloom, learn more.

\*\*SHOW EXAMPLE OF USING<small> TAG HERE\*\*\*

So this is pretty good, but we can make it a lot better using Bootstrap’s button classes. So back into your text editor, and let’s going to pull that <small> tag out. Instead, let’s go ahead and apply some classes. At the most basic, if we apply a class inside of the <href> tag here, you set a class of “btn” for button, and save that and 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, it has the words “Learn more” in it. When you roll your mouse over it, it looks as if the button is being slightly depressed. 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 might be interested in doing is adding some kind of color to it.

Those are available to you as well. Back to your text editor of course, as an additional class that we can add on here, there is “btn-primary.” That will make this button a blue color, sort of the primary styling for buttons on a Bootstrap web page. Go ahead and Save that, and refresh this inside of your browser, you’ll see that the button becomes blue. We’ve seen this before in the Hero unit, and when you roll over it, you’ll see that the color does change a little bit, but we 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-info,” this will make a light-blue button. And you can keep on going. Here are the colors and the associated classes:

\*\*PUT GRAPH OF COLORS AND CLASS NAMES HERE WITH EXAMPLES\*\*

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 very deliberately so that you can override these colors with your own colors. You can think of these buttons in terms of their functionality rather than color. The primary button,

treatment that you’re going to have is an informational button; a green button indicating success, a yellow? Button indicating a warning, and so on. That is 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 the top page, back to line 24 where we have our first button.

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

\*\*\*INSERT BUTTONS WITH SIZES HERE\*\*\*

If, as a third class, you add to this tag, “btn-large,” you’ll make a large button. You may remember we did that with the Hero treatment 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.

So once again, go ahead and save those, 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 this yellow button at the bottom, which is the standard size.

I recommend not overriding any of these colors just yet. I’m sure you’re more than capable

of doing that in CSS, 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.

*Incorporating Images*

So our page is looking better than when we started, but it’s still lacking in a lot of

color and a lot of interest, and one of the things that we could do to make the page more

interesting and break up the text a little bit, of course, would be to add a few images.

In addition to that, I’ll show you some great image treatments that ship with Bootstrap that are really fun. You can take a square image and apply a style to that. We will make it appear circular on your web page. There’s a treatment that will give you rounded corners on the 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.

So, there are some images inside your exercise files, so make sure you go ahead and copy

those into your Bootstrap site, and we’re going to hop in Your text editor here, we’ll go

ahead and add those images.

\*\*INSERT THE ABOVE 3 or 4 DIFFERENT PHOTO TREATMENTS HERE\*\*\*

The first place we’ll add an image is right after Art in Full Color. In fact, right inside of the paragraph describing Art in Full Color, we’ll go ahead and add the image tag <img src=””>. Also be sure to stick an alt tag in there and a description.

\*\*IMAGE OF WHERE WE ARE HERE\*\*

Then let’s insert another picture, and then finally, let’s scroll down the page to just before Exhibits A & B,so this is around line 60. I’m going to go ahead and add an image tag right here inside of this paragraph, and this particular image that I’m going to put in is the picture of the road, and once again, a blank alt tag.

Okay, so we’ve just added three images to our document. 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 pictures, but they look a little bit chunky and blocky, so we’ll add some additional styles to these to give them the treatments that we discussed.

\*\*IMAGE HERE OF INSERTED PHOTOS\*\*

Switch back to your text editor and scroll back on up the page here to the first image, so we can add some classes do this. The big class we’ll add to this is image is “img-circle,” which will give us a

treatment of a circular image.

\*\*IMAGE HERE OF CIRCULAR PHOTO\*\*

So if you just do that much and save your page and take a look at it in Firefox, you’ll

see that, lo and behold, your squared-off image has now become a circle, and it looks great.

I am now also going to float this on over to the right side of the page. I don’t think

it really works over here on the left side.

So, inside of Your text editor, I’m going to add a second class, it’s our old friend pull-right,

which will float the image over to the right side of the page.

So we’ll go ahead and save that again, take a look at it in Firefox, and you’ll see we

have a nice round image over here pushed to the right side of the page, looks a lot better.

So that’s one of the treatments that’s available to you in Bootstrap image, img-circle will

give you the circular images.

Now if we go back to Your text editor again, and we scroll on down the page, back to our second

image here with the image of the artist, I’m going to go ahead and add another class, and

I’m going to give this a class of img-rounded, which will give it rounded corners, and in

addition to that I’m also going to do the same thing again, I’m going to do a pull-right

to make it float to the right side of the page.

Save this again, and if you refresh it here inside of Firefox, you’ll see that we have

our image of our artist over here 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 treatment.

Last, we’re going to give this road picture down here a treatment, so scroll on down in

your document to the road picture, and for this one we’ll go ahead and add a class for it,

and the class is img-polaroid, and once again, I’ll add a pull-right to that particular

class and go ahead and Save that, and we’ll Refresh that inside of Firefox, and you’ll

see that we now have our image of the road 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 I had a different color background, that white in between the light gray line and the

picture would stay white, which explains the name Polaroid.

So there we go, three image treatments here inside of Bootstrap.

You can make use of these very, very simply with a single class, the Polaroid treatment,

the Rounded corner treatment, and of course, the Circle treatment up here on the top.

*Incorporating Icons*

Bootstrap also comes with a full set of icons that have been contributed by glyph icons.

To get a full listing of these icons, if you go to getbootstrap.com, 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 these, 140 choices of all different shapes and sizes

and indications of things.

I’m going to grab this calendar icon right here, I’m just going to highlight the words

icon-calendar and Ctrl+C or Command+C to copy that.

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 hop into your text editor, and scroll on down here to the first instance of where this happens, which is here, at 9:30 to 10:30am: Elizabeth Hall, and that’s where I’d like to go ahead and put this particular icon. (Figure out how to show where and what icon to put here)

The way you go about doing this is you’re just going to put in a span tag with a class

of whatever class is associated with that particular icon. Those are all displayed on Bootstrap’s web page and new ones are added often, but here is the current collection:

\*\*\*ADD ICONS HERE\*\*\*

It’s definitely worth mentioning that there is another wonderful collection of font-icons at Font-Awesome.com.

\*\*\*FONT\_AWESOME INFO HERE\*\*

You’re going to follow that immediately with a slash span, so you can see here the span

class “icon-calendar” is the name of the class which we saw here on the web pages. Those 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 here that our icon shows up right here 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, it will indicate that you can use icon-white, which will give you the icon as well, but in the white format instead of in the dark format.

\*\*INSERT WHAT I’m REFERENCINZG HERE WITH WHITE ICONS\*\*

So, now if we scroll on down to the next instance here, let’s put in that span tag again of “icon-calendar” “icon-white” which will give you a white version of the icon followed by a slash span. Once again, go ahead and Save this and and refresh your browser, and you’ll see the icon is there. It’s very hard to see, of course, 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 way that you would go about incorporating 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. Where these icons are being pulled from is inside of your file structure within your text editor. Inside of your image folder, you’ll notice that there are a couple of these images here, “glyphicons-halflings” and a white version of those.

In fact, what you have here is an image sprite, so there are 140 tiny little pictures that

occur in one PNG, which is a really great way of handling these tiny images for mobile phones. Rather than having to pull down 140 individual images, in this particular case, we’ve only pulled down a single image and then using CSS, we target the picture that we want directly on that image, so that is where those are located.

The drawback to doing it this way is that since these are image sprites and they’re rasterized small images, you’re not able to blow these up and use them as large-format pictures within your document.

However, as I mentioned, you can download those kind of graphics elsewhere on the Internet (such as at Font-Awesome)and incorporate those into your Bootstrap site, but these glyph icons that ship with Bootstrap, they are image sprites, and that’s the way that they work.

*Overriding Core Bootstrap CSS*

Finally, the part you have 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. We’ve reviewed the Scaffolding, much of the rest of this we haven’t talked about yet, but these are JavaScript-driven widgets that we’re going to cover in a couple of sections. Some of the styling that’s in CSS you can choose to include it or not, you can choose to include specific

plug-ins that we rely on jQuery, and you can customize these variables.

Bootstrap makes use of the LESS (LEaner cSS) framework, which is a CSS preprocessor, and you can type in different values for these things. The easiest one to understand is under “Colors.” You can decide what exact shade of blue would you like to have where blue occurs inside of your document, what exact shade of green and so forth. You can customize all these variables and type in your own values and then customize

it easily. You simply download by clicking on the giant button at the bottom of the page that will download exactly what it is that you want to have for your website.

\*\*SHOW IMAGE OF CUSTOMIZATIO PAGE WITH GIANT BUTTON(may need to shrink page)\*\*

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. We’re currently on v. 2.2.2, but that will change in the future, which usually is a good thing. 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, which is 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.

So you’ll now, as part of that process, 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 and personally, that’s just too much work for me. I can’t 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 so much more maintainable. It’ll be easy to upgrade 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, (so 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. Since this is just a proof of concept thing, we’re going to make a really quick change here. Let’s go ahead and change the body background color to a lovely shade of green,

so we’ll enter background-color is #bada66, and save that, File > Save.

So let’s go to your Bootstrap folder, then to your CSS folder, and save this as **custom.css**, so that is your custom stylesheet.

Now the next thing we have to do is attach it to the document we’ve been working on,

so here’s your schedule.html, which is the document we’ve been working on in this chapter.

Go ahead and insert your style sheet where indicated below:

\*\*INSERT SCREENSHOT OF THE ABOVE\*\*

.

So, then choose the Custom Style Sheet here from the list, go ahead and say OK. It’ll go ahead and write in the code to include your custom style sheet right here inside of your document. (show this exactly)

Again, you want it to occur last in the string of style sheets, because remember, anything

that you put in your custom style sheet you’re going to want to override what comes before

it, and 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, it may not necessarily override the

default styling from Bootstrap.

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 lovely shade of green behind our text, and the icons that we inserted

in the previous lesson. You can see that we have a nice dark icon and we also have a white

icon. We have some white here associated with our block quote, and if we scroll on down the

page a little farther, you can see the image that had that class applied to it, **image-polaroid.**

You can 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 is how you add custom styles to your document.

Make sure you put them in a separate style sheet, and call it whatever you like. I usually

call mine 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 aspect of the Bootstrap framework that we’ll examine includes the various methods

of creating navigation systems.

Bootstrap offers many different widgets for doing this, including tabs, pills, buttons,

dropdowns, lists, and full navigation bars, as well as helper systems like Breadcrumbs and Pagination.

As you’ll see in the next chapter, in many cases these systems can be combined with JavaScript to include

Dropdowns.

But for now, let’s start with taking a look at Breadcrumbs. A breadcrumb is a trail of links that you might see at the top of a web page that indicates where you are in the website’s hierarchy.

For example, if you’re at freebootstrap.com and you take a look at my author page, you’ll see

that the breadcrumb is located up at the very top of the web page just under the Navigation Bar.

\*\*INSERT EXAMPLE OF BREADCRUMBS HERE\*\*\*

This indicates that you’re on the page talking about Jen Kramer, and to get there you had

gone through the author’s page, which would list all of the authors on the website.

And before that you were at the home page, in this case, the freebootstrap.com homepage. We can create this exact same navigation system inside of Bootstrap.

Bootstrap comes with a system for styling breadcrumbs, and can take take a look at that now. If you return to your text editor, and you open up the document that’s inside of your exercise

files, you’ll see that this is actually the breadcrumb we just saw in freebootstrap.com but

without any styling applied to it.

If we view this in your web browser right now, you’ll see that it’s just a bulleted list with some links included and some of the <li>s, as well as a separator which is that little double-angle bracket. (What is this, exactly?-mm) So now we need to add a little bit of styling to this and a little bit of extra markup

to make this a breadcrumb.

First of all, since we have this displayed on a web page, we can certainly incorporate the grid system into laying this out. If you choose to do that anywhere along the way in these examples, what you would do is, of course, you’d start with a <div> with a class of “container-fluid,” as we’ve done before. (And don’t forget to close that <div>.) And you can follow that with the call to your “row-fluid.” And then the last part of this would be what’s inside of that row.

\*\*SCREENSHOT OF WHAT WE”VE DONE HERE\*\*

At this point we usually incorporate some kind of tag that would have the span class inside of it, that would indicate how many rows this particular aspect of the website will span, and I’d like this to span to all 12 columns. The question is what kind of markup should we use to do that? There is actually a little bit of debate about what the best kind of markup is for breadcrumb in the online commuity. My feeling on it is that the best tag to use is the <nav> tag, because breadcrumbs are a form of navigation. Some people disagree with this and say that the <nav> tag should be used only in cases of major navigation elements, like the main navigation on a web page. So, what our argument would be is we’re going to use the <nav> tag to mark up elements on

our web page that are related to navigation.

So we’re going to go ahead and use the <nav> tag here to indicate this is where our breadcrumb is going. We’ll also put in a <nav> with a class of “span12” because we’d like this to stretch all the way across the page. And of course, we’ll close that <div> down there.

Let’s put in a couple of tabs just so that this code is a little bit easier to -- three over for the <ul>, and four over for each of the <li>s. So now that we’ve got that setup, we’ve actually done very little with what we’ve incorporated here at this point. All we’ve done is incorporate some additional markup with a bulleted list, and this is not any particular special bulleted list at this point in time. It’s just a 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 too much. You might have seen it twitch a little bit over from the left column; that has to do

with the grid system and its spacing and margins and so forth.

So now let’s add the actual markup that makes this a breadcrumb inside of Bootstrap. The fastest and simplest way to do that is right inside of the <ul> tag. We’re going to add a class, and this will be a class of breadcrumb (singular--that would be the part that I would fumble and call it breadcrumb**s**, plural.)

The second thing that we’re going to add to make this a breadcrumb is we’re going to indicate

which item is active. So on the previous freebootstrap.com example, we were on the Michael Musgrove author page. So Michael Musgrove 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. You’ll see it used many times in this navigation chapter.

So, we are going to add a class right on this <li> of “active,” and that will indicate

where we are in the hierarchy. The last thing that we’re going to do is specify particular little spacers, and

we can do that right in front of the little double-arrow guys.

\*\*SCREENSHOT OF WHERE WE ADDED THIS CLASS\*\*

We’ll add a span with a class of “divider,” and close that with a </span>, and we’ll apply

that to the second divider as well, along with the </span>. That indicates that the little character that we’re using here, which is this double arrow, would be what’s going to separate our breadcrumb. You, of course, could use any character that you like in that process. Go ahead and save this, Ctrl or Command+S to Save, and let’s take a look at it again in the browser. You will notice that this looks very different.

\*\*INSERT SCREENSHOT OF WHAT WE HAVE SO FAR IN BROWSER\*\*

Now we have a nice-looking breadcrumb. It’s shaded black just slightly, and has very nice spacing to it. You can see that the character in between is nicely styled as well, screen blackened just a little bit so that you can see that there’s a separator, but it’s not dominating the look of this particular breadcrumb. There’s another treatment that we could give to this if we wished.

In one of the previous sections, we reviewed the glyphicons that come with Bootstrap, and we could incorporate one of those as our divider, if we wished.

So, what we would do if we chose to change out the little characters instead for an icon is inside of the class where it says divider, go ahead and just add “icon-arrow-right” (for example.) That will give as a little arrow icon instead of the HTML character that we are using here.(using where?-MM) Of course, you will need to get rid of the HTML character, if you choose to do that. For the below example I’m going to leave the other one the way we had it set up so that you have an example of each treatment. Let’s save that again, and we’ll take a look at it inside of the browser. Be sure to Refresh the page.

\*\*\*INSERT SCREENSHOT OF THE NEW ICON WITH OLD ONE AS WELL\*\*

So you see here weI have our icon now 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 saw in the glyphicon section, you can 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 played around with styling here and you changed some of the colors, white might 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 ships with Bootstrap.

*Adding Pagination for Paging through Content*

Occasionally at the bottom of the page, there are many blog posts. On a page with search results or a page displaying many products, you’ll see a list of additional pages listed at the bottom. (Think about the bottom of the Google Search Results page.)

\*\*INCLUDE SCREENSHOT OF GOOGLE SEARCH FOR FREEBOOTSRAP\*\*

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.

\*\*INSERT SHOT OF GOOGLE SEARCH PAGINATION HERE\*\*

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 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,(WHAT? Double check this mm) and we’re going to cover those in the next section.

So let’s take a look at how we might style something like those numbers at the bottom

of the screen in our Google Search Results. Inside of your text editor, you should have a bulleted list. It’s wrapped inside a <nav> because, of course, this is navigation that’s there on the web page.

And if we just take a look at this right now inside of our browser, you wouldn’t see a whole lot that was terribly interesting. It’s a bulleted list with some numbers and some characters that point you in either direction. So how we can make this look a little bit more interesting?

Well, of course, Bootstrap has some styles for dealing with something like this. If we go to the <nav> and add a class of “pagination,” that would be a great way to start by making our numbers go horizontally across the page. If you go ahead and 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.

\*\*SCREENSHOT OF WHAT I JUST DESCRIBED WITH PAGINATION\*\*

Now, if you noticed on the Google Results Page, we landed on one of the pages, and there was

an indication of which page we are 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, and Bootstrap has a way of doing that as well.

So in your text editor, if you add to this first <li> a class of “disabled” (in other words, gray this out, because there is 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 there’s nothing before this.

\*\*SHOW SCREENSHOT OF NUMBERS I JUST DESCRIBED\*\*

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 run my mouse over it, it looks like it’s no longer clickable. Likewise for the number 1.

Now, it is 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 cursor here to make sure that it doesn’t look like a finger, the way it does if you roll over number 2. However, most assuredly 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’re going to have to do that with some kind of JavaScript magic or maybe you’ve hooked this up to a content management

system or some kind of programming language, which will spit this back so the links aren’t actually there at all.

So be careful about that. Just because you use class of “disabled” doesn’t necessarily mean that there’s no link that exists on an item anymore, because if you take a look at the HTML, there is most certainly a link that appears there, and it is still clickable, even though it doesn’t look like it.

\*\*MAYBE INCLUDE A SCREENSHOT HERE OF WHAT IM TALKING ABOUT ABOVE\*\*

So one final thing I want to show you. You designers are probably wondering how you can make your pagination centered on the page. You are probably thinking about doing that with a custom style. There’s no need to do that. Bootstrap comes with the ability to easily center the pagination on the page.

What you need to do is inside of the <nav> class of “pagination,” add the class of “pagination-centered”

and if you Save that and Refresh that inside of your browser, you’ll see that now your pagination

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.

So that’s the story behind pagination and treatments for it. Remember, pagination usually comes out of a content management system or some kind of database-driven application at the bottom of a web page for tapping through things. 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 these links to separate HTML documents. It takes a lot of time.

Pagination is really a widget that’s best linked to something that has a database behind it where the pagination is generated out of the database. Therefore it’s not likely it’s something that you would use on just a static HTML website. You’re most likely going to see it used more often in something with the database behind

it, such as a content management system like WordPress.

*Adding a Page for Next and Previous Links*

I’m here on the freebootstrap.com blog at freeboorstrap.com.

One of the navigation elements that the freebootstrap.com blog uses is down at the very

bottom of the web page. If you scroll all the way down to the bottom of the blog, you’ll notice down at the

bottom there’s this little link that says “Older Entries.”(make sure this is effective and available-mm)

This is a very common type of navigation that you see in blogs, so if you click the link

that says “Older Entries,” and go all the way down to the bottom of the page, again, you’ll notice that it now it says “Older Entries” and “Newer Entries.”

\*\*SHOW SCREENSHOT OF THIS\*\*

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, so a Pager allows you to style these links for “next” and “previous” or “older” and “newer” that you see at the bottom of these pages.

So if you head over to your text editor, you can see this in action with really simple markup; this is just a two-bullet list.

\*\*INCLUDE THE HTML OF THE LIST I JUST MENTIONED\*\*

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 adding up on the top a class of “pager” inside of the <ul> tag.

\*\*SHOW WHAT I MEAN HERE\*\*

And that will take care of the styling for our “Previous” and “Next” buttons with Bootstrap’s defaults. If you just go ahead and save that document, and preview it inside of your browser, you’ll see now that we have two links: “Previous” and “Next.” You can roll over them and see that they have a very nice hover state as well. So that’s great!

You may be wondering, “what if I want to space these out so that they’re at the edges of whatever area I’ve included these particular buttons in?” 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.

\*\*SHOW WHAT I’m TALKING ABOUT ABOVE\*\*

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’re going to 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,” I’m sure it’ll be shock to learn we add a class of “Next.”

So just by doing that, this will push these two buttons to the 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.

\*\*SCREENSHOT HERE OF ABPVE METHOD\*\*

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. Because we don’t have any of the Scaffolding included inside of this HTML document, these are pushed all the way to the edges of the window. Of course, these will be pushed to the edges of its containing element 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 span of four, it will be pushed to the edges of that.

\*\*SCREENSHOT OF THIS HERE\*\*

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 I would like to have that Previous button sort of grayed out?

Just as we’ve seen before in the last section, you can add a class of “disabled.” “Disabled”

will gray that button out. So if you go ahead and 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.

However, just as I pointed out in the previous section, it most assuredly is a clickable button. We’ve just used CSS here to make it look like it’s not. If you really, truly 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 programming language to make this truly a not clickable button. The next button, of course, still remains 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 very commonly used at the bottom of blog entries. And you can combine this with a couple 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 simple bulleted list and a number

of different ways that we can change the classes to make such a list look very, very different inside of Bootstrap.

We’ll first take a look at pills and tabs. Tabs you’re probably familiar with, and pills are sort of round, oval sort of shapes. They have curved edges that you can also use to make very interesting-looking navigation bars. They look just like this:

\*\*INSERT PILLS EXAMPLE\*\*

What we have right now in our default markup(What is our markup? Where did we get it?-mm), though, isn’t terribly interesting. If we very quickly look at this inside of the browser, you’ll see that what you’ve got is your basic bulleted list; very functional, not terribly pretty.

So let’s start adding some classes to see how we can make this look a little bit more attractive or a little bit different. Inside of your text editor, the very first step is to add a class of <nav>. This is the most basic markup. Just by adding this <nav> you’ve indicated that this is navigation and some basic things are going to happen here.

When you refresh you’ll see that the margin and padding associated with <ul> has disappeared, and

you’ll see that the bullets have gone away, and we’re now crammed into the side of the web page. The reason we’re crammed into the side of the web page is because we’re not making use of Bootstrap’s Scaffolding system at this point. We’ve kept our markup very, 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 web page.

If we had this <nav> tag located somewhere else inside of the Bootstrap Scaffolding, this

navigation wouldn’t look like it’s being pushed into the edge.

\*\*IMAGE OF WEB PAGE WITHOUT SCAFFOLDING\*\*

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

Alright, so that’s it! The very, very basic first steps of styling. Now let’s add something to make it a little bit more interesting. If we add nav-tabs, this will of course create more of a very “tabbed” type of look. It’ll make our links become horizontal instead of vertical as well. If we Refresh the page in the browser, you’ll see that that has in fact happened. We have a series of links with a line underneath, and everything is looking great, but it doesn’t really look like tabs yet.

And the reason it doesn’t look like tabs is because we haven’t indicated what page we’re on. What page is the active page? There’s that active class again, so let’s say we’re on The XXX 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 in 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.

So, that’s tabs, and they’re a pretty simple structure. Make sure that you have an **active** state so that the tab meme is a little bit more obvious on the page.

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 bit surprising.

After “nav tabs”, if you add “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. It’s definitely a step up from what we had with just the “nav” class on that <ul>. We have a bit more margin, a bit more padding, we have some nice borders going on between these, we have a great hover state, we still have our active state (it looks like it’s grayed out, as if you can’t click on it, even though you can. Remember, that’s just a trick that the CSS is playing on you.)

Of course, these links continue to look as if they go all the way across the page because there’s no Scaffolding in place. Hide scaffolding, then you would see these look a bit narrower.

So, what about that pill thing that I mentioned before? Well, if we go back into your text editor, take out “nav-stacked” and we change “nav-tabs” to “nav-pills” and if you 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 lovely blue pill surrounding it, in this case, a blue 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 could use custom CSS to change these colors or tweak the way that these are styled as well.

Just as we saw with the tabs, you can make this pilled structure also vertical, and if we hop back in 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 gone horizontal. 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.

This is an introduction to tabs and pills and how we can work with them to give us at

least three or four different unique looks for our navigation on a web page, with just a handful of styles using the Bootstrap framework.

*Adding the Basic Navigation Bar*

Bootstrap ships with a navigation bar (nav bar) that you can use in your web pages as well. This nav bar comes with many, many options for customization and styling.

Head to getbootstrap.com, and if you take a look at that black navigation bar that’s at the top of the web page, this is the navigation bar that we’re going to work on coding next.

\*\*SHOW WHAT THE FINISHED NAV BAR LOOKS LIKE HERE\*\*

So if you take a look at your text editor document, you’ll notice the code looks very

familiar compared to what we just had out in the previous section.(again-where are we getting the code?-mm)

It’s a simple navigation bar. You can see that we have a nice bulleted list here. So this is our starting point, and we’re going to go ahead and start adding a few classes to create our awesome navigation bar.

With any navigation bar that we make in Bootstrap, the very first step is to set our <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.

\*\*SHOW SCREENSHOT OF WHAT WE”RE DOING HERE\*\*

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 <div> that goes around

this as well, which is important to the styling. Let’s add a <div> with a class of “navbar-inner” and of course, don’t forget to close that <div> down at the bottom.

If we take a look at what we’ve got so far, just go ahead and save--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.

One thing that we’re missing is some type of indication for what website we’re on, and that’s also included in the specification for the navigation bar, so we can add that.

Inside of your text editor, just after the <div> with the class of “navbar-inner” we’re going to

add a class of “brand.” You, of course, can make this link anywhere that you’d like, probably back to your homepage, and you’re going to put in the text that you want to appear as the identification for the website. In this example I’ll put “XXXXX.” You can put whatever you’d like.

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, will take us back to the homepage (That’s the way that I would probably code it), and we have a navigation bar that appears. Nice!

\*\*SHOW NAV BAR WITH BRANDING\*\*

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 a little bit here. 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 particularly 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 you have some basic styling for the navigation bar so far, at least.

One last thing that you probably noticed is that when we took a look at the navigation

bar inside of Bootstrap, it was black, but here inside of this example it’s white, and

it’s actually perhaps a little bit difficult to see here on the screen with this big white

background next to it, but you can actually make this navigation bar black as well.

\*\*SHOW WHITE NAV BAR WE ENDED UP WITH\*\*

So inside of your text editor, up where we have the <nav> with the class of “navbar” go

ahead and add “navbar-inverse” and this will give you the inverted (black) color. If you Refresh the page again, we 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 ships with Bootstrap.

We’ve gone through some simple things like setting up the general structure of a nav bar, we’ve

added some branding to it so that we know what website we’re 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. Of course, you’re more than welcome to override any of these colors or styles with your own, using a custom style sheet.

*Chapter 5: Javascript Effects*

Understanding how Javascript works in Bootstrap

Bootstrap ships with a number of **jQuery** based plug-ins. jQuery is a JavaScript framework, and Bootstrap has written some effects using this powerful framework. They include Modal Windows, Dropdowns, Tabs, Tool Tips, Alerts, Popovers, an Image Carousel, and much more.

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

Head to GetBootstrap.com(hyperlink these links-mm) again, and click on the JavaScript link at the top of the page, and on this page you’ll see 13 effects that use jQuery, and have been coded to work within Bootstrap. They’re all listed over on the left side of the page.

There are also some items under the components part of the documentation that also require

the use of JavaScript or jQuery in order for them to work. Be very sure that you’re linked to **the bootstrap.min.js** file, as well as the **latest version of jQuery**!

The exercise files that I’ve given you contain links for that at the bottom of the pages

just before the slash body tag. Also, if you’re creating these files on your own, make sure you link to both of these files, otherwise these JavaScript effects will not work.

Remember that the .min files contain the same information as the regular JavaScript files, but the .min files have the white spaces removed to make the file sizes smaller, and your pages load faster.

There are a few custom HTML5 attributes you should be familiar with that are going to

be used in Bootstrap, and you’ll see that in many places over the next few sections. If you click on the link for “Navbar” under “Components,” and scroll on down to the section that talks about the Responsive navbar.

There on line 6 you’ll see these two items working together. The data-toggle is applied to a button or a link, 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 change content when they’re clicked, the data-toggle is the link on the tabs, while the data-target is a <div> containing the text that should display when a tab is clicked.

So, let’s go through a number of these in detail now with Bootstrap and JavaScript so that you can see how these data-toggles and data-targets work, 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 looks great while it’s running on a Desktop and the browser is maximized, but as we reduce the size of the web browser, this 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, you’ll see here that the navigation bar can get quite long and fat, and that’s probably not really an effect that you’d like to have.

It’s possible to add some JavaScript to this to make the navigation bar a bit more responsive. It’s not the only approach to making a navigation bar more responsive, many would argue. Same 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 navigation bar shortened as the screen sizes get a bit smaller. And I’ll show you how that works now.

Back to the old text editor document inside your exercise files, this is the

finish point for the last video from the last chapter. We’re going to add some additional code to this.

\*\*\*SHOW CODE WE NEED TO HAVE TO WORK ON\*\*

Also inside your exercise files is a little text document here that contains some code.

\*\*PUT THAT CODE HERE\*\*

So this is the first little bit of code that we’re going to add to your text editor page here. You may be able to just copy this bit of code using Edit > Copy. Then inside of your text editor page, just after the navigation tag but before the brand, let’s Ctrl+V or Command+V what’s on the clipboard to Paste that in place.

We’re going to give this just a little bit of formatting here,(what was the formatting?-mm) just so we can make it a little bit easier to read. So what is this little chunk of code that we just copied in here do?

Well, first of all, you’ll see these three lines, lines 20 through 22, these are calling the glyphicons. So there’s an icon that’s called the icon bar. It’s just a little line, and we have three of them here stacked on top of each other. That’s all that part is.

\*\*SHOW WHAT I JUST TALKED ABOUT\*\*

The interesting part that makes this interactive is what’s inside of the <a> tag; the clickable link there. So you’ll notice that this has been assigned a few classes, both the button and “btn-navbar” and then you see this JavaScript calls for toggle collapse and data-target of .nav collapse.

What these are doing is integrating the JavaScript into this web page to make our little icon expand when it’s clicked inside of the bar. So if you copy in this code(what code?-mm) shown below, this is going to make your navbar responsive. You can certainly swap out these three lines here where you have the three icon bars.

\*\*PLACE ABOVE\_REFERENCED CODE HERE\*\*

You can put in a word like “menu,” for example, or you could put in a different icon if

you wished so that there’s something different 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 navigation

bar collapses in size. We probably don’t want to hide the brand, but we do want to hide all of the navigation here.

So what we’re going to do is we’re going to add another <div> tag here, and we’re going

to give it a class of “nav-collapse.” Simple enough. So this particular <div>, which we’ll close immediately after the bulleted list, this particular <div> 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 <div> of “nav-collapse” will be hidden when that transition happens.

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 nothing special with the browser maximized on a Desktop or Laptop computer.

But if we make the page smaller, you can see that long around(?-MM), actually quite wide, this icon now appears. Remember, that’s the glyphicon. We have the three bars that are stacked on top of each other. And as we make the page progressively smaller, then the navigation bar is resizing very nicely. When you click the button, 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.

\*\*SHOW WHAT I JUST DESCRIBED\*\*

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

So we had a bit of code that triggered the JavaScript to cause the effect of this open and close 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 course of the last few sections we’ve built this 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-accessed navigation bar work great on a mobile phone.

But you might be wondering about one aspect of this bar that we haven’t explored yet. That is there’s only one level of navigation here, just the primary navigation is present. What about secondary navigation or tertiary navigation? How do we show that on our web page?

Well, certainly you could always take the approach of plugging in a left column or

right 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.

Dropdowns are very easy to add within Bootstrap, and I’ll walk you through how we can go about doing that now.

Back in your trusty text editor, add this bit of code to your particular page:

\*\*ADD CODE HERE< IN WHAT HTML PAGE SHOULD LOOK LIKE\*\*

So there is the link for schedule. Notice that starts with an <li> then we have our <a> tag with the </a>. But the closing </li> doesn’t occur until further down. Remember that these lists are actually nested within the <li>. That’s the correct way to mark them up.

So we have our <ul> here inside of that <li> with a </ul> 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.

\*\*SCREENSHOT OF HTML DOC NOW\*\*

If you just open that document(what document?-mm) and then take a look at it inside of your browser, you’ll notice that we have indeed a bulleted list that appears right under the word “schedule,” which is great. But it’s not really a dropdown because, of course, it’s showing right there on the page right now. So we need to add some additional styles in order to make this a dropdown.

The first thing we’ll do inside of the text editor is, for this specific <li>, the <li>

belonging to that word “schedule,” we’re going to add a class of “dropdown,” which of course

indicates that a dropdown will follow within this <li>.

Then in the href side of this we’re going to add a class of “dropdown-toggle,” and then we’re going to use that data-toggle which is what is helping to integrate the JavaScript: data-toggle=”dropdown” just like this:

\*\*EXAMPLE OF WHAT WE JUST DID\*\*

So we’ve added a class of “dropdown-toggle” and a data-toggle of “dropdown” to a tag

that’s associated with the word “schedule.” Finally, what we want to do is add to the <ul> a class of “dropdown-menu” which is going to give us some styling here 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 “schedule” and click, you’ll notice that we get a dropdown menu.

So there’s Day One, Day Two, and Day Three, and all these are clickable and will take

you to those particular pages.

How did we know that schedule was the one that had the dropdown menu? Well, we knew because we coded it.

But how is a visitor of our web page going to know that schedule is the dropdown menu? An excellent question. It’s likely that a visitor isn’t going to know that that’s a dropdown menu. So we should probably indicate somehow that “schedule” has got more behind it than just a single link.

Back within your text editor, we can add some additional code which will give us a little triangle that will appear right next to that word “schedule” that would indicate that a dropdown menu is located there. So right here after the word “schedule,” before the </a>, go ahead and put in a span class of “caret”

immediately followed by a </span>.

\*\*SHOW THIS BIT OF CODE\*\*

This little bit of code is going to put in that tiny little triangle arrow that would indicate that the dropdown lives right 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 that 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. I’ll show you how to do that in the next section.

*Tabbing within the same page*

On occasion, you may wish to tab between blocks of content within the same web page,

usually through a tab-like interface where you can click on various tabs within the page,

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 interface for making this kind of effect.

So our starting web page looks like this right now:

\*\*SHOW WEBSITE AS\_IS\*\*

We have a bulleted list at the top with some names, and underneath we have some text; it’s just marked up with <h3> for a title and a paragraph of text that appears underneath. And we can see that here inside of your text editor.

So there’s our bulleted list, and then underneath there’s just a bunch of <h3>s and paragraphs. And what we’d like to do is turn this into a bunch of tabs that we can click on and see each of these artists’ bios displayed individually, rather than altogether on a page like a laundry list.

So to make that happen, there are a couple of things that we need to do. First of all, we are going to need to do some markup here on the tabs. So we are going to start by taking the <ul>, and we’re going to add a class of “nav” and “nav-tabs.” Then we need to indicate which one of these tabs is going to start off being the active one. So we may as well make the first the active one, so we 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.

Then we’re going to add a data-toggle to the a href tag for all of these, and the data-toggle,

as we know, is what’s going to help pull in the JavaScript to make this page more interactive, data-toggle=”tab.”

And we need to apply that to all three of these links, so we’re just going to highlight that, Ctrl+C or Command+C to copy, Ctrl+V or Command+V to paste, so that it now appears in all three of these links.

\*\*SHOW WHAT I JUST DID\*\*

Finally, what we need to do is connect the specific link for Constance Smith, for example, with the chunk of text that’s going to be below. 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, so right here with the first one we’ll call this rewington, leave the Pound sign there in the front, and the second one we’ll call #smith, and the third one I’d call #ta.

\*\*SHOW THE ABOVE\*\*

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 would skip on down to a link say, a name ta, for example. It would jump on down the page wherever that happened to be placed.

We are going to do something similar here, although JavaScript driven. All right!

So that’s 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, I’m going to add a section to go around all of this with a class of tab-content.

That section class is important because it’s defining that this is the actual content for

the web page here, and that’s why I am using the section tag.

So I’m going to go down and close that at the bottom.

Then after the section tag, I’m going to add an article tag here, article with a class of tab-pane.

So this article tag is going to get wrapped around what effectively is an article,

the h3, and the paragraph.

So I am going to go ahead and end that article down here, and I can indent this h3 and the p

because now they are inside of the article tag.

So what I did is just highlighted all those lines and just hit the Tab key.

So I am going to go ahead and add the same article class of tab pane to the top of

each one of these little mini articles, and always of course, make sure you close your

tags, and I will Tab over the content here to show that it’s inside of the article.

One more time, paste again the article class of tab-pane”, and we’ll put in our /article down here in

the bottom, and we’ll tab over this chunk of code. Okay.

So we have three articles with a class of tab-pane.

We also need to add to the same article tag a couple of other pieces of information, one is an id.

The ID in this particular case for this first one will be Rewington.

Those of you who think about IDs in terms of CSS almost exclusively, if you’re a front-end

developer who just works with HTML and CSS, IDs are also used in JavaScript.

And so what we’re doing here is we are connecting the link at the top with #rewington to an

ID down here underneath, the ID of rewington. We’re connecting these two together.

So whatever you called your links up here on the top, you’re going to need to take that

and identify them as IDs beneath inside of these articles.

So I’ll go ahead and add an ID of smith and further down an ID of ta.

And one final thing that we’re going to need to do, since Riley Rewington is the tab that

I want to have come up by default when the page loads, I have already specified that

up here on line 18 as having an li with a class of active.

I’m also going to need to identify which piece of content is going to come up by default

underneath, and that of course is the article here about Riley Rewington.

So I’m going to indicate that that should come up when the page loads by adding a class

of active to this article.

So we have tab-pane and active here for the article class, while the other article classes are just tab-pane. All right!

Now that we’ve got all 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 “Constance Smith,” you can see that her information 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 Xhou Ta, and you’ll see that the bio for him is listed here as well. So we can click around in these tabs.

Notice that the URL for the page never changes. We are just displaying different sections

of the page inside of these tabs.

So this is a really handy effect. I see it used on a number of websites these days. It’s a little bit more interesting than those long laundry lists or skipping things down the page we used to do way back when. This is a lot more fun, a lot more interactive, and certainly more attractive.

Some of you might be wondering about Search Engine Optimization now that we’ve separated these out between the tabs, and the answer to that is Search Engine Optimization should not 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 spelled out right there in the web page. So Google, Bing and the other search engines would be able to see that when they pay a visit to your website. It can read all of the content there. They won’t see the JavaScript and it won’t see the JavaScript effects; they’ll see the links at the top of the page, they will see that those link down to further into 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.

*Adding a dropdown menu to the navigation bar*

Back in the last chapter we built a pill-based menu, and we’re taking a look at a format of that right now.

The pill-based menu had this rounded cornered blue rectangle indicating where you were on the page, and when you rolled over the navigation you had this light gray one that goes across the tab.

\*\*SHOW WHAT I JUST MENTIONED\*\*

I’ve added to this a secondary navigation right under schedule, 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 the code and see what we need to do to make this truly a dropdown menu.

So if you take a look inside of your text editor, you can see that I have added this bulleted

list right here, lines 23 to 27. They are located inside of the <li> for “schedule.” That’s the correct way to mark that up.

Note that you start with the opening tag here on line 22 and the closing tag on line 28. In between you have the bullet list, each with its own bullet. So to make this a dropdown menu, there are several things we need to do to make that happen.

First of all, find the <li> which contains the word “schedule,” 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.”

\*\*SHOW WHAT WE JUST DID IN CODE\*\*

So 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 “schedule.” So here we will add a class of “dropdown-toggle,” and then we’re going to add that data-toggle

which as I told you has to do with the JavaScript of dropdown.

And those two particular items the class of dropdown toggle and the data-toggle of dropdown

will give us the functionality that is 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 roll over the word “schedule” and click, you’ll see that we get a dropdown that results. So that’s pretty cool.

But how did I know that I can click on the word “schedule?” Again, only because we coded it. We need to have some sort of indicator that the dropdown lives there under schedule. So to do that, we could add a little triangle that points downwards, and that would be a great indicator of that schedule is interactive and has a dropdown associated with it.

So, right after the word “schedule,” we’re going to put a space followed by a span class of “caret” and a </span>. 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 “schedule” now has a triangle by it. That triangle’s clickable and our dropdown menu appears underneath.

\*\*SHOW AN EXAMPLE OF THE ABOVE\*\*

So based on the code that I’ve given you here, you can add as many dropdown menus as you

wish 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 have another dropdown appear from there as well.

*Tabbing within the same page*

On occasion, you may wish to tab between blocks of content within the same web page,

usually through a tab like interface where you can click on various tabs within the page 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 interface for making this kind of effect.

So our starting web page looks like this right now.

\*\*SHOW WHAT PAGE LOOKS LIKE NOW\*\*

We have a bulleted list at the top with some names, and underneath we have some text. It’s just marked up with an <h3> for a title and a paragraph of text that appears underneath. And we can see that inside of your text editor.

So there’s our bulleted list, and then underneath there’s just a bunch of <h3>s and paragraphs. What we’d like to do is turn this into a bunch of tabs that we can click on and see each of these artists’ bios displayed individually, rather than altogether on a page like a laundry list. So to make that happen, there are a few things that we need to do.

First of all, we are going to need to do some markup on the tabs. We’re going to start by taking this <ul>, and we’re going to add a class of “nav” and “nav-tabs.” Then we need to indicate which one of these tabs is going to start off being the active one. So, again, we might as well make the first the active one, so let’s 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.

Then we’re going to add a data-toggle to the a href tag for all of these, and the data-toggle, as we now know, is what’s going to help pull in the JavaScript to make this page more interactive. Data-toggle=”tab.”

And we need to apply that to all three of these links, so just highlight that, Ctrl+C or Command+C to copy, Ctrl+V or Command+V to paste, so that it now appears in all three of these links.

And finally, what we need to do is connect the specific link for Constance Smith, for example, with the chunk of text that’s going to be below. 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.

We’re going to give these names, so with the first one we’re going to call this rewington, leave the Pound sign there in the front, the second one I’ll call #smith, and the third one I’d call #ta. This is exactly what we would do if we were using anchors further down on the page with links on the top.

\*\*SHOW EXAMPLE OF THE ABOVE CODE HERE\*\*

The way these were coded would skip on down to a link say, a name ta, for example. It would jump on down the page wherever that happened to be placed. We’re going to do something similar here, although JavaScript-driven. So that’s the markup that we need for the list at the top. Then underneath we’re going to need to add some markup there as well.

First, let’s add a section to go around all of this with a class of “tab-content.” That section class is important because it’s defining that this is the actual content for the web page, and that’s why we’re am using the <section> tag. So let’s make sure to go down and close that at the bottom.

Then, after the <section> tag, we’re going to add an <article> tag here, an <article> with a class of “tab-pane.” This <article> tag is going to get wrapped around what effectively is an article, the <h3>, and the paragraph.

So we’re going to go ahead and close that <article> down at the bottom, and we can indent the <h3> and the <p> because now they are inside of the <article> tag.

\*\*SHOW CODE FORMATTED\*\*

All I did is just highlight all those lines and just hit the Tab key. So we’re am going to go ahead and add the same article class of “tab pane” to the top of each one of the little mini articles, and always of course, make sure you close your tags, and tab over the content to show that it’s inside of the article.

One more time, paste again the article class of “tab-pane,” and we’ll put in our </article> down in

the bottom, and we’ll tab over our chunk of code. So we have three articles with a class of “tab-pane.” We also need to add to the same <article> tag to a few other pieces of information, and one is an id.

The id in this particular case for this first one will be Rewington.

For those of you who think about IDs only in terms of CSS, if you’re a front-end

developer who just works with HTML and CSS, will like to know IDs are also used in JavaScript. So what we’re doing here is we’re connecting the link at the top with #rewington to an ID underneath, the ID of rewington. We’re simply connecting these two together.

So whatever you called your links up at the top, you’re going to need to take that and identify them as IDs beneath inside of these articles. So let’s go ahead and add an ID of smith and further down an ID of ta.

And one final thing that we’re going to need to do, since Riley Rewington is the tab that we want to have appear by default when the page loads, is to specify that up on line 18 as having an <li> with a class of “active.”

We’re also going to need to identify which piece of content is going to come up by default underneath, and that, of course, is the article here about Riley Rewington. So we’ll indicate that that should come up when the page loads by adding a class of “active” to this <article>.

So we have “tab-pane” and “active” here for the article class, while the other article classes are just “tab-pane”. Perfect! Now that we’ve got all of our 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 you click on Constance Smith, you can see that her information loads here inside of the tab. The little dotted box we have up there because we have selected it at the moment.

\*\*SHOW TABBED NAV WE JUST MADE\*\*

If we click away from that tab, that box goes away, and there’s Xhou Ta, and you’ll see that

the bio for him is listed here as well. So we can click around in these tabs. Notice that the URL for the page never changes. We are just displaying different sections of the page inside of the tabs.

So this is a really handy effect. You’ll see it in use on a number of websites these days. It’s a little bit more interesting than those long grocery lists or skipping things down the page like we used to do back in the dark ages of the internet. This is a lot more fun, a lot more interactive, and certainly more attractive.

Some of you might be wondering about Search Engine Optimization now that I’ve separated

these out between the tabs, and the answer to that is Search Engine Optimization should

not 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 spelled out right there in the web page.

So Google would be able to see that when it comes to visit your website.

It can read all of the content there.

It won’t see the JavaScript, it won’t see the JavaScript effects, it will see the links

at the top of the page, they will see that those link down to further into 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. \*\*DUPLICATE?\*\* I think I just wrote this a page or two up!

*Creating Modal Windows*

A Modal window is a <div> that’s embedded within your web page which appears on top of your page on a grayed background, and it looks kind of like a pop-up window. It’s not a true pop-up in that it doesn’t appear in a separate browser window. You may have seen this on many different websites.

\*\*SHOW EXAMPLE OF MODAL WINDOW\*\*

Head on over to the Bootstrap website at getbootstrap.com. Click on the JavaScript Tab, and then Modal over on the side.

\*\*SHWO WHERE TO CLICK AND WHAT TO LOOK FOR\*\*

If you scroll down a little bit here, you’ll see down here we have a link to a live demo.

\*\*Image of button

If you click this big button here that says “Launch demo modal,” you’ll see how the screen

grays back a bit, and this window seems to scroll in to place. The window is comprised of three parts:

1) We have a heading up here on the top with a little Close button over here on the side, the little X.

2) There’s some main text to keep the Modal window that’s very long from scrolling off the web page.

(Notice there’s a scrollbar over on the side.)

3) Then down at the bottom, we have a button to close the window or a button to save changes.

The “Save Changes” button may be a little bit distracting to you because you’re wondering

“what can you possibly change within this window?” And you’d be correct; there’s nothing in that

window you can actually change.

However, if this window was hooked up to a database, you might be able to make some changes

within a Modal window and save those changes. Many content management systems use a feature like this.

Let’s close that and close the tab.

And the other tab is our modal.html file, which is in your exercise files. This is our starting point for how this is going to be laid out.

Up at the top of the page look for the text, “click me!” That is going to be our big button that we’ll see on the web page. The rest of this will end up being the Modal window.

\*\*SHOW CLICK ME THEY NEED TO FIND\*\*

And you see here the button with the X on it will wind up being that little X that appears in the corner. We have some text that we’ll be talking about what’s going on in the window, and then down

at the bottom, there’s a button to close the window. So we need to get all styled-up, and we need to apply some JavaScript to make the window appear and vanish.

Head on over to your text editor, here’s all of the code here that I just described:

\*IINSERT CODE HERE\*\*

You’ll see the 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:

1) There’s a header which contains that little X button as well as the heading.

2) We have some text that describes the event. This is just the blah-blah that’s there inside of the window, and then down at the bottom,

3) we have the footer with the “Close” button.

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

First, starting up at the top with the header, we’re going to give this a class of “modal-header.”

We’ll wrap a <div> around the text there that’s in the middle. We’re going to give that <div> a class of “modal-body.” You may need to tab these over a little bit to make the code a little easier to read. And then close the <div> down below.

Then, of course, for the footer, we’ll give it a class of “modal-footer.” So, 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 these buttons in an earlier section in the CSS chapter.

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 go away. And there’s an attribute that we can add that will make that magic happen and talk to the JavaScript that is data-dismiss, and that is modal is the value.(What?-mm)

So, we need to add that to both of those buttons, data-dismiss=”modal.”

\*\*SHOW WHAT WE JUST DID WITH THE CODE\*\*

Voila! 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, well, we want it to look like a button. Just to be different, let’s give it a class of “btn-warning”, which will make it orange. We also want it really big, so let’s give it a class of “button-large.”

\*\*SHIOW CODE\*\*

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 call that, should then be called

via pound that ID name within the href.

Then finally in the href area, we’ll add a data toggle: data-toggle=”modal.” And that will mean that it 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 add a little bit of styling to the section.

So, we’re going to give this a class of “modal.” That’s going to control the appearance and disappearance of the window. We’ll also give it “hide” and “fade.” Hide and fade will help drive that nice slide as it comes in and as it goes away again.

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 orange button that says “Click me!” on this page. Go ahead and click it.

Lo and behold, here comes your 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. So Modal windows may be very helpful to you.

They’re nice for search engine optimization because all of the text that’s within that

Modal window is available to search engines to read, as opposed to a pop-up window which

is a separate HTML document.

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

*Creating a Photo Carousel*

The last JavaScript effect I am going to show you is the Image Carousel, and this is really cool.

People love these carousel-kind of effects. You see it in action right here on this web page(what web page?-mm)

\*\*SHOW EXAMPLE OF PHOTO CAROUSEL\*\*

Here we have a series of photos that are scrolling on by with some text underneath. We have little buttons so that we can click to the next image as fast or as slow as we prefer. You can also go backwards.

These carousels are pretty much everywhere on the web these days, and there’s a thousand

different ways to code them. We’re going to learn the Bootstrap way of doing it.

So take a look at your starting Carousel document and you’ll notice that we have some photos here. They have a little bit of text that go underneath of all of the images. Then down at the bottom some tiny little left and right arrows, and if we take a look at this in the text editor, you’ll see that that code is exactly the same.

\*\*SHOW CODE I JUST DESCRIBED\*\*

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.

So first of all, the easiest thing to add are some <div>s here which will designate where the captions for this carousel are going to appear. So just after the image, before the <h4>, we’re going to add a <div> with a class of “carousel-caption.” And then, of course, after the last closing paragraph, make sure you close the div.

Do that three times here: <div> with a class of “carousel-caption,” and close out your <div>s

\*\*SHOW WHAT THIS CODE LOOSK LIKE\*\*

So we’ve marked off where those carousel captions are going to appear, and now we have a unit here, the image plus the carousel caption, that is all a grouping itself.

So let’s add another <div> for that, this is a <div> with a class of “item,” and you may want to put a little bit of spacing between each one of these groupings, just to make things a little bit easier to read.

\*\*SHOW WHAT I MEAN HERE\*\*

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

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 first one. So we’re going to designate it as such. Now we need to add just two more <div>s. We’re going to wrap everything that you see here in another div with a class of carousel-inner, scroll all the way down to the bottom, and we’re going to close that tag just before the href down at the bottom of the screen.

\*\*SHOW WHERE WE ARE\*\*

Then just above that div with a class of “carousel-inner,” as you might expect since it was called inner,

we should have an outer one, right? So let’s add one final <div> here to the mix. This will be <div> with a class of “carousel.” And then we’ll just close that just after the hrefs all the way down in the bottom, which will be the little directional arrow pointing either way.

\*\*SHOW THIS\*\*

Okay, in that <div> with a class of “carousel,” also give it a class of “slide.” This will cause

the images to have that wonderful sliding effect from one to the next. Without the class of “slide,” what will happen is they will simply appear one after the other. Then we’re going to give this an ID of “artists.”

So that’s all the markup that we need for the actual content of the carousel. Now we’re going to scroll on 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 here.

So for our left quote, let’s go ahead and apply some styling to this. We give it a class of “left” as well as “carousel-control.” As you would expect, that’s going to apply to the left arrow and cause it to go to the left

side of the images, and likewise for the next arrow, let’s go ahead and give that a class of “right carousel-control.”

\*\*SHOW THIS CODE\*\*

Instead of the href of just pound sign (#) here, we’re going to give this an href of #artists. Just as we’ve seen before, the ID above needs to match the href so that the JavaScript can work correctly.

And lastly is the action that we have to have on this href, and in this case it’s data-slide, and that will be previous, P-R-E-V ,for the first one and data-slide=”next” for the last one.(WTF-mm)

\*\*SHOW THIS\*\*

Alright! So now that we have got all that information packed into these a tags down at the

bottom of the screen, we are ready to save our document. You know the drill by now: Ctrl+S or Command+S to save the document, go on over to your browser and refresh your page, and you should see a fully-formatted carousel. It does stretch all the way across the page, and that’s because, of course, the containing element

for this particular page is the <body> tag. We don’t have the scaffolding in place to make this carousel smaller or larger. If you click your arrows, you can see that the images jump in between just fine. Very cool. And we have the text underneath, describing each one of our images just as we wanted.

So the image carousel is a great way to show off photos on your website, add a little bit

of flair and interactivity, and they are very fun and interactive for the user.