



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

Bootstrap Essentials

Use the powerful features of Bootstrap to create responsive and appealing web pages



Snig Bhaumik

[PACKT] open source*
PUBLISHING community experience distilled

www.allitebooks.com

Bootstrap Essentials

Use the powerful features of Bootstrap to create responsive and appealing web pages

Snig Bhaumik

[PACKT] open source 
PUBLISHING community experience distilled

BIRMINGHAM - MUMBAI

Bootstrap Essentials

Copyright © 2015 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: August 2015

Production reference: 1040815

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78439-517-9

www.packtpub.com

Credits

Author

Snig Bhaumik

Project Coordinator

Suzanne Coutinho

Reviewers

Paula Barcante

John Bloomer

Joe Fitzsimmons

Janko Prester

Proofreader

Safis Editing

Indexer

Rekha Nair

Commissioning Editor

Amarabha Banerjee

Graphics

Jason Monteiro

Abhinash Sahu

Acquisition Editor

Meeta Rajani

Production Coordinator

Manu Joseph

Content Development Editor

Amey Varangaonkar

Cover Work

Manu Joseph

Technical Editor

Mohita Vyas

Copy Editor

Dipti Mankame

About the Author

Snig Bhaumik is the technical director of InfoAxon Technologies, located in New Delhi, India, and Bracknell, UK. He also heads the open source evangelist team of InfoAxon. He is an active member of, and contributor to, various open source products, such as Alfresco, Liferay, and Pentaho.

Having more than 12 years of software development and architecture experience in various tools and technologies, his prime interests now lie in mobile development, social media implementations, digital governance, Internet of Things, and traditional practices such as knowledge management and business intelligence.

He authored a popular cookbook on the Alfresco Content Management System named *Alfresco 3 Cookbook*, Packt Publishing in 2011 (<https://www.packtpub.com/web-development/alfresco-3-cookbook>) and was also a part of the technical review team for *Learning Alfresco Web Scripts*, Packt Publishing (<https://www.packtpub.com/web-development/learning-alfresco-web-scripts>) in 2014.

About the Reviewers

Paula Barcante is a user experience designer, who is completing her bachelor's degree at the University of Waterloo. She started in the UX Design industry when she was 19 years old, when she began teaching herself how to code websites and web applications. She is very interested in how humans interact with technology, which has influenced the way she thinks about her work and personal life.

She has interned for two start-ups and, most recently, Amazon for 2 years in a row. She has been using Bootstrap for 3 years and is self-taught in frontend development.

I would like to thank the Packt team for giving me the opportunity to review such a resourceful and beneficial book.

Joe Fitzsimmons is a frontend web developer at SUNY Oswego, where he helps refine and grow the college's web presence. He also manages his own freelance work, where he designs web solutions for local small businesses. He is working toward obtaining a master's degree in human computer interaction. Joe is most passionate about constructing useful and clean web interfaces. He has more than 5 years of web development experience, including creating WordPress and Drupal themes using Bootstrap.

I would like to thank all my family and friends who have supported me over the years. I would also like to thank the people at Packt for the opportunity to review this book. It was truly a pleasure.

Janko Prester was born in Zagreb, Croatia. Originally trained as an economist and after finishing his master's degree, he decided to switch his career to web development. During the past 5 years, he has been working on a number of web projects, both as a freelancer and as an employee in the IT department. He is currently working as software developer in a government agency. The technologies he is most familiar with include JavaScript, CSS3, HTML5, Sass, and AngularJS and PHP, with strong reliance on the Bootstrap framework for UI development.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<https://www2.packtpub.com/books/subscription/packtlib>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

*"Dedicated to all the kind and benevolent people around me whose love and belief
made this book possible."*

Table of Contents

Preface	v
Chapter 1: The Evolution of CSS and Bootstrap	1
The mobile-first philosophy	3
Responsive design basics	4
Setting the viewport	5
Sizing your content to the viewport	5
Using media queries to achieve responsiveness	6
Responsive design patterns	6
Navigation patterns	7
Introducing Bootstrap	7
What Bootstrap includes	8
CSS	8
Components	9
JavaScript	10
Customization	11
Summary	12
Chapter 2: Getting Started with Bootstrap	13
Get Bootstrap	13
The Bootstrap file structure	14
A precompiled bundle	14
folder: css	15
folder: fonts	15
folder: js	15
A source code bundle	16
rfolder: dist	16
folder: fonts	17
folder: grunt	17
folder: js	17
folder: less	17

CSS preprocessors	17
Variables	19
Mixins	20
Operations	20
Nesting	21
How to use Bootstrap	22
The application folder structure	25
Summary	26
Chapter 3: Creating Responsive Layouts Using Bootstrap CSS	27
Basic HTML structure for Bootstrap	28
The head section	28
The body section	29
Basic HTML elements	30
Responsive classes	31
Understanding the basics	31
Controlling display of elements across devices	32
Rendering images	34
Showing responsive images in a sample application	35
The grid system	36
Constructing data entry forms	40
Making the form horizontal	41
Finalizing the Contact Us page	42
Other utility classes	46
Encapsulating everything	47
Summary	48
Chapter 4: Packaged Components in Bootstrap	49
The page header	50
Glyphicons	50
The navigation bar	51
Badges	56
Alerts	56
Toolbars and button groups	57
Panels	58
Wells	60
Jumbotron	61
Breadcrumbs	62
Paginations	63
Summary	64

Chapter 5: The JavaScript Add-ons in Bootstrap	65
Basic concepts	66
Custom data attributes	66
JavaScript APIs	67
JavaScript events	67
Packaging add-ons	67
Modal windows	67
The basic modal window	68
Example – enhancing our application using the modal dialog box	70
Tabs	72
Collapse and accordions	76
Example – showing the product categories of our store	78
Tooltips and popovers	81
This dropdown	83
Alerts	85
Carousels	87
The final preview	89
Summary	93
Chapter 6: Compiling and Building Bootstrap	95
Required tools	96
Node.js	96
Installing node.js	97
Grunt	98
Installing Grunt-cli	98
Installing Bootstrap	100
Compiling and building Bootstrap	104
Installing dependencies	104
Building Bootstrap	106
Summary	108
Chapter 7: Customizing Bootstrap	109
Customizing using the build environment	109
Customizing using Bootstrap web interface	119
Summary	122
Chapter 8: Extending Bootstrap	123
Theme extension – Bootswatch	124
Downloading the CSS files	124
Using the LESS files	125

Table of Contents

A tree-view control	129
Installing the tree-view component	131
WYSIWYG editor and Font Awesome	132
Installing and using the WYSIWYG component	137
The bootstrap-wysiwyg component	137
The jQuery hotkeys component	137
Font Awesome	137
Summary	138
Index	139

Preface

As the number of Internet users on mobile devices is growing every moment, your websites are no longer built only for the desktop machines. The mobile-first philosophy demands the sites to be fully compatible for all available and future mobile devices. Bootstrap allows and easily enables you to design and develop your websites congenial to all devices including various screen readers.

We have until now developed your websites using all those manual CSS classes and quite a number of various JavaScript libraries. Delivering the desired results and upgrading your websites has traditionally been quite a challenge. Mobile devices coming into the picture has made the task even more difficult. Bootstrap comes to your rescue here - providing all you need including CSS classes and JavaScript components in a single package.

This book covers all the theoretical and practical aspects of Bootstrap and makes you a proficient web developer for the mobile world. You will be able to download, include, and configure Bootstrap in your web project. You will understand the internal architecture and structure of Bootstrap and get fully familiar with the usage of Bootstrap CSS and components and become apprised of the JavaScript objects offered by Bootstrap. You will also be able to build and compile Bootstrap from the source code, and finally customize and extend to suit your requirements.

What this book covers

Chapter 1, The Evolution of CSS and Bootstrap, introduces you to a brief history of CSS and the birth of CSS3. It introduces the mobile-first philosophy and lays down the fundamental concepts and requirements of Bootstrap.

Chapter 2, Getting Started with Bootstrap, enables you to download and include Bootstrap in your project. You will get to know the files' and folders' structure of different distributions of Bootstrap. You will also start building a sample application using Bootstrap framework.

Chapter 3, Creating Responsive Layouts Using Bootstrap CSS, goes through all the major aspects of CSS classes offered by Bootstrap, including basic HTML elements, responsive classes, handling images, Bootstrap grid system, data entry forms, and so on.

Chapter 4, Packaged Components in Bootstrap, explores all the components that come in-built with the Bootstrap framework. For example, the Glyphicons, navigation bars, toolbars, panels, wells, and pagination. You will use all these components in the sample application to understand the practical usage.

Chapter 5, The JavaScript Add-ons in Bootstrap, makes you proficient with all the important JavaScript add-ons that Bootstrap offers. You will be able to use the modal windows, tabs, pills, accordians, tooltips, dropdowns, popovers, alerts, carousels, and so on.

Chapter 6, Compiling and Building Bootstrap, introduces you to the advanced usage of Bootstrap and how to establish the development environment of Bootstrap, how to compile the Bootstrap source code, and how to build the distributable version of the framework.

Chapter 7, Customizing Bootstrap, enables you to use the development environment and customize Bootstrap's default behavior to suit your own requirements.

Chapter 8, Extending Bootstrap, introduces you to the Bootstrap community marketplace where you will find different resources by which you can extend and broaden the Bootstrap default capabilities and features. For example, you will use Tree-view control and WYSIWYG editors, which are not a part of Bootstrap default offering. Finally, you will be able to create the sample application with almost all the major classes and components of Bootstrap along with the community controls.

What you need for this book

This book is a step-by-step and detailed guide to develop full device-friendly websites. It starts with the basic concepts of CSS, CSS3, and Bootstrap. It includes all the practical know-hows about all major features and functionalities offered by Bootstrap. It also shows you how to work with the Bootstrap main source code and how to customize and extend the framework.

An example application is accompanied by every step as an easy-to-follow practical exercise and demonstrates the usage of the features of Bootstrap.

In order to understand and work on this book as a reader, you need to be knowledgeable in HTML, CSS, and JavaScript. Knowledge of HTML5, CSS3, and jQuery will be an added advantage; however, the book is not essentially dependent on those prerequisites.

Who this book is for

Bootstrap Essentials is intended for all those web developers who design and develop the websites and pages using HTML, CSS, and JavaScript—irrespective of whether you have experience on mobile website development or not, this book will guide you to become an adroit Bootstrap web developer. For better understanding of Bootstrap and mobile website development, previous experience of HTML, CSS, and JavaScript will be helpful. Further knowledge in jQuery would be an added advantage. Advanced information about building and customizing Bootstrap are part of the book; thus, readers who want to have their own version of Bootstrap will be able to establish a full-fledged development environment of Bootstrap.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "For example, this is how HTML global element is configured in Bootstrap CSS."



A block of code is set as follows:

```
<link rel="stylesheet" media="print" href="print.css">
<link rel="stylesheet" media="(max-width: 800px)" href="max800.css">
<link rel="stylesheet" media="(max-width: 320px)" href="max320.css">
<link rel="stylesheet" media="(orientation: portrait)" href="prt.css">
<link rel="stylesheet" media="(orientation: landscape)" href="lnd.
css">
```

Any command-line input or output is written as follows:

```
npm install --prefix D:\Bootstrap grunt-cli
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "What you only need to do is to update the values of these variables in this interface and finally press the **Compile** and **Download** button at the bottom of the page."

 Warnings or important notes appear in a box like this. 

 Tips and tricks appear like this. 

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail feedback@packtpub.com, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at <http://www.packtpub.com> for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to <https://www.packtpub.com/books/content/support> and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at questions@packtpub.com, and we will do our best to address the problem.

1

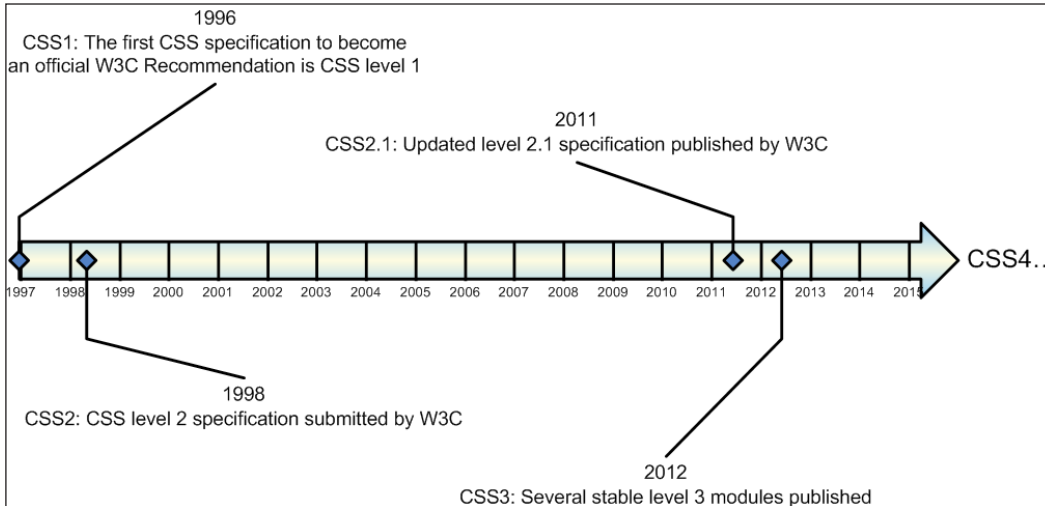
The Evolution of CSS and Bootstrap

HTML, CSS, and JavaScript—the three most important tools for you as a web developer in today's ever-evolving and ever-demanding Web 2.0 and Web 3.0 world. As the mobile world and its needs grow every day, the websites and web pages you'll be required to develop are far more challenging. Thankfully, as the appetite increases, the industry also serves with all new engines and mechanisms to meet the demand.

The following topics are going to be covered in this chapter:

- Introducing CSS3
- Understanding the mobile-first philosophy
- Learning the basics of responsive design
- Introducing Bootstrap as a responsive design framework

The purpose of this book is to understand and be proficient in using Bootstrap. Since Bootstrap is largely based on CSS3, we will spend a few minutes to see the evolution of CSS:



In late 1996, the official CSS1 specifications were published; some of the critical aspects were color of text, backgrounds, capability of margin, border, padding, positioning, various font properties, various text properties, spacing, alignment, and so on.

The CSS2 specifications were published in 1998, including a number of features such as element positioning, z-index, font shadows, and bidirectional texts.

However, CSS3 was a major shift and is currently the latest version. It added a number of powerful capabilities and boosted the mobile-first philosophy of the technology. Some of the very crucial inclusions are Media queries, Selectors, Cascading and Inheritance, Template layouts, Namespaces, MathML, Flexible and Grid layouts, Transformations and Transitions, and so on.



Several modules of CSS3 are still under consideration and in proposal stage; many older browsers of both mobile devices and desktop do not support CSS3 tags. Bootstrap uses them and is heavily based on CSS3 and HTML5; thus, as a developer, you must make sure that while using Bootstrap, you are building a future proof website, but at the same time, your website is not going to be compatible with older browsers, unless you are taking extra precautions in that aspect.

The mobile-first philosophy

As an experienced web developer, you must be aware of the mobile-first philosophy until now, and if you are not, it is high time to wake up.

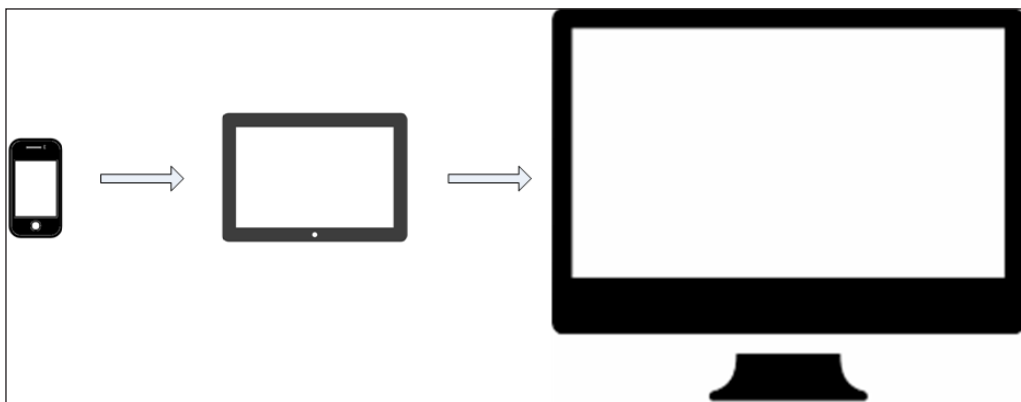
Let's consider the following few facts (all approximate values):

- There are more than 1 billion mobile users worldwide, and counting
- More than 25 percent of web users access it on a mobile only; they rarely use a desktop to browse the web
- More than 90 percent of all people on earth have a mobile phone, and more than 50 percent own a smartphone
- More than 50 percent of mobile phone users use mobile device as their primary Internet source, and not a desktop computer
- More than 70 percent of tablet owners purchase things online from their tablets each week
- By 2015 and 2016, mobiles and tablets are predicted to overtake desktop internet usage

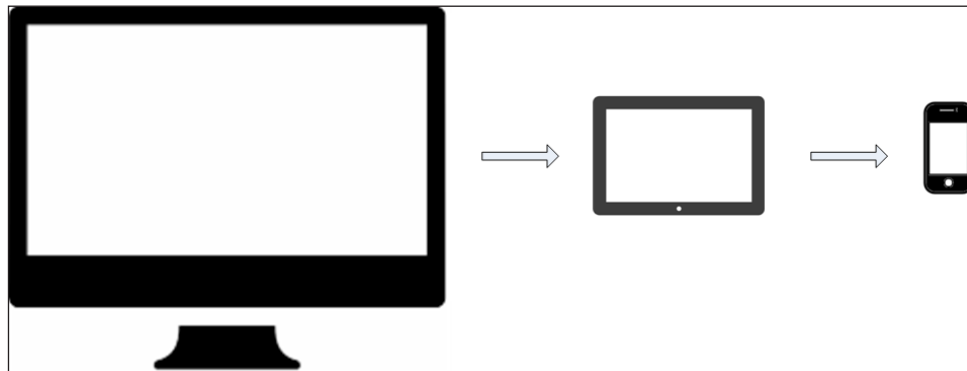
So if not now, within a couple years down the line, we will develop websites not only for a desktop machine, but mostly for the mobile devices.

There comes the mobile-first approach which goes side by side with terms such as Progressive Enhancement, Graceful Degradation, and Responsive Web Design.

In Progressive Enhancement, you design and develop your website for a mobile phone, which has the smallest real estate of them all. Then, you upgrade your site and contents to match with the available space of a tablet, and your full website for a desktop screen, which means that you are designing your website bottom-up.



On the other hand, in case of Graceful Degradation, you follow the opposite approach – the traditional top-down approach. There you design the full-fledged website for desktop users, which then gradually becomes compatible with various mobile devices with lesser space and real estate by deducting some optional contents and redesigning all the CSS styles and JavaScript to suit the mobile devices.



The Progressive Enhancement approach is getting popular day by day over Graceful Degradation – putting the mobile-first philosophy as one of the most crucial aspects of today's web design.

However, Responsive Web Design techniques are common and imperative in either strategy – irrespective of whether you go for Progressive Enhancement or Graceful Degradation. Also, as a web developer, this is one of most important skills you want to possess.

This revolutionizes the whole web design and development paradigm. Of course, the major impact is on how you build your CSS classes and styles, and designing a website for mobile devices is a responsibility to the CSS developers, rather than to the website programmers. CSS3 comes to the rescue here. As we saw, as the demand of a responsive website design has increased, so has the power of CSS. Most of the CSS3 features and specifications are in line with what you need to develop a responsive website.

Responsive design basics

In a nutshell, Responsive Web Design means designing and building your website that would properly render in almost all the devices irrespective of the device size, browser, screen size, and so on.

As we discussed earlier, the best approach is to design the site for the smallest screen size first, and then to move upward. There are few very basic steps to be followed in order to achieve responsiveness and a good user experience.



The below points are only to get underlying concepts clearer and make your elemental background ready. We will later on see in detail how Bootstrap can be used to build and generate a full functional responsive website.

Setting the viewport

Viewport is the visible area of a web page on the device or monitor the user is accessing on. The size of viewport varies with the device – for a mobile phone, it is very small; for a tablet, it is little bigger; for a desktop computer, it is bigger than that for a tablet; for a TV, it is much bigger.

Pages targeted and optimized for a variety of devices must include a `meta` viewport element in the `head` section of the document. The `meta` viewport element makes the browser understand how to control the page's dimensions and scaling for rendering different page elements.

For example,

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

The `meta` viewport value `width=device-width` instructs the browser to match the screen's width in device-independent pixels while rendering the page. HTML5 has introduced this `meta` tag to enable web developers take control over the size of the browser screen and render the web page contents as per the size. This is essentially the base of Responsive Web Design.

Sizing your content to the viewport

Irrespective of the device and browser that the user is working on, nobody prefers a horizontal scroll bar. Thus, you need to make sure that all your contents and elements are adjusted within the screen real estate you have got – with only the vertical scroll bar.

While you have set the page contained within the available viewport, it is also necessary to create and structure your content so as to fit into the space available. For example, if you put an image with fixed width 600 px, in a screen with 320 px, you will face a horizontal scroll bar to see the full image, even if you have got the `meta` viewport element added into the `head` section.

Thus, a proper responsive design requires that all the web page content is also automatically adaptive to screen viewport size.

Using media queries to achieve responsiveness

There is a crazy range of mobile devices available in the market. So what do you do in order to make sure that your page does render in all the devices and browsers efficiently without compromising on any user experience? Of course, you cannot write separate code to handle separate devices and then dynamically find out which code base is to be made executed.

Here come the CSS media queries—these are simple filters to change styles of the HTML elements, based on the characteristics and properties of the device. Consider the following example:

```
<link rel="stylesheet" media="print" href="print.css">
<link rel="stylesheet" media="(max-width: 800px)" href="max800.css">
<link rel="stylesheet" media="(max-width: 320px)" href="max320.css">
<link rel="stylesheet" media="(orientation: portrait)" href="prt.css">
<link rel="stylesheet" media="(orientation: landscape)" href="lnd.
css">
```

Responsive design patterns

Here are the few established and well-adopted patterns in Responsive Web Design:

- **Fluid design:** This is the most popular and easiest option for responsive design. In this pattern, larger screen multiple columns layout renders as a single column in a smaller screen in absolutely same sequence.
- **Column drop:** In this pattern also, the page gets rendered as a single column, however, the order of blocks gets altered. This means, if a content block is visible first in order in case of larger screen, the block might be rendered as second or third in case of smaller screen.
- **Layout shifter:** This is a complex but powerful pattern in which the whole layout of the screen contents get altered, in the case of smaller screen. This means that you need to develop different page layouts for large, medium, and small screens.

Navigation patterns

Following are some of the things you should take care while designing a responsive web page. These are essentially the major navigational elements that you would concentrate on while developing a mobile-friendly and responsive website:

- Menu bar
- Navigation/app bar
- Footer
- Main container shell
- Images
- Tabs
- HTML forms and elements
- Alerts and popups
- Embedded audios and videos and so on



You can see that there are lots of elements and aspects you need to take care of to create a fully responsive design. While all of these are achieved by using various features and technologies in CSS3, it is of course not an easy problem to solve without a framework that could help you do so. Precisely, you need a frontend framework that takes care of all the pains of technical responsive design's implementation and releases you only for your brand and application design.

Now, we introduce Bootstrap, which will help you design and develop a Responsive Web Design in a much optimized and efficient way.

Introducing Bootstrap

Simply put, Bootstrap is a frontend framework for faster and easier web development in the new standard of the mobile-first philosophy. It uses HTML, CSS, and JavaScript. In August 2010, Twitter released Bootstrap as open source.

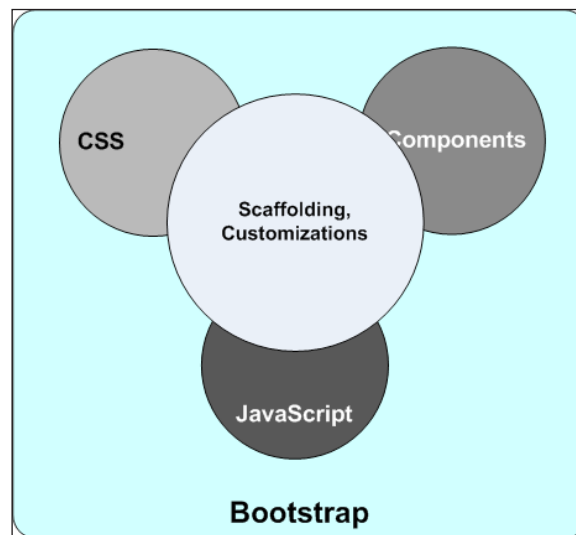
There are quite a few similar frontend frameworks available in the industry, but Bootstrap is arguably by far the most popular in the lot. It is evident when we see Bootstrap is the most starred project in GitHub since 2012.

Until now, you must be in a position to fathom why and where we need to use Bootstrap for web development; however, just to recap, here are the points in short:

- The mobile-first approach
- A responsive design
- Automatic browser support and handling
- Easy to adapt and get going

What Bootstrap includes

The following diagram demonstrates the overall structure of Bootstrap:



CSS

Bootstrap comes with fundamental HTML elements styled, global CSS classes, classes for advanced grid patterns, and lots of enhanced and extended CSS classes.

For example, this is how HTML global element is configured in Bootstrap CSS.

```
html {  
  font-family: sans-serif;  
  -webkit-text-size-adjust: 100%;  
  -ms-text-size-adjust: 100%;  
}
```

This is how a standard HR HTML element is styled:

```
hr {
  height: 0;
  -webkit-box-sizing: content-box;
  -moz-box-sizing: content-box;
  box-sizing: content-box;
}
```

Here is an example of new classes introduced in Bootstrap:

```
.glyphicon {
  position: relative;
  top: 1px;
  display: inline-block;
  font-family: 'Glyphicons Halflings';
  font-style: normal;
  font-weight: normal;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
```

Components

Bootstrap offers a rich set of reusable and built-in components such as breadcrumbs, progress bars, alerts, and navigation bars. The components are technically custom CSS classes specially crafted for the specific purposes.

For example, if you want to create a breadcrumb in your page, you simply add a DIV tag in your HTML using Bootstrap's breadcrumb class:

```
<ol class="breadcrumb">
  <li><a href="#">Home</a></li>
  <li><a href="#">The Store</a></li>
  <li class="active">Offer Zone</li>
</ol>
```

In the background (stylesheet), this Bootstrap class is used to create your breadcrumb:

```
.breadcrumb {
  padding: 8px 15px;
  margin-bottom: 20px;
  list-style: none;
```

```
background-color: #f5f5f5;
border-radius: 4px;
}
.breadcrumb > li {
display: inline-block;
}
.breadcrumb > li + li:before {
padding: 0 5px;
color: #ccc;
content: "/\00a0";
}
.breadcrumb > .active {
color: #777;
}
```



Please note that these sets of code blocks are simply snippets; we will explore all these features and functionalities in detail in later chapters.



JavaScript

Bootstrap framework comes with a number of ready-to-use JavaScript plugins. Thus, when you need to create Popup windows, Tabs, Carousels or Tooltips, and so on, you just use one of the prepackaged JavaScript plugins.

For example, if you need to create a tab control in your page, you use the following:

```
<div role="tabpanel">
  <ul class="nav nav-tabs" role="tablist">
    <li role="presentation" class="active"><a href="#recent" aria-
controls="recent" role="tab" data-toggle="tab">Recent Orders</a></li>
    <li role="presentation"><a href="#all" aria-controls="al"
role="tab" data-toggle="tab">All Orders</a></li>
    <li role="presentation"><a href="#redeem" aria-controls="redeem"
role="tab" data-toggle="tab">Redemptions</a></li>
  </ul>

  <div class="tab-content">
    <div role="tabpanel" class="tab-pane active" id="recent"> Recent
Orders</div>
```

```
<div role="tabpanel" class="tab-pane" id="all">All Orders</div>
<div role="tabpanel" class="tab-pane" id="redeem">Redemption
History</div>
</div>
</div>
```

To activate (open) a tab, you write this JS code:

```
$('#profileTab li:eq(1) a').tab('show');
```



As you may have guessed by looking at the syntax of this JavaScript line here, that the Bootstrap JS plugins are built on the top of jQuery. Thus the JS code you would write for Bootstrap are also all based on jQuery. We will explore more on this in the later chapters.

Customization

Even though Bootstrap offers most (if not all) standard features and functionalities for Responsive Web Design, there might be several cases when you would want to customize and extend the framework. One of the very basic requirements for customization would be to deploy your own branding and color combinations (themes), instead of the Bootstrap default ones. There can be several such cases where you would want to change the default behavior of the framework.

Bootstrap offers very easy and stable ways to customize the platform. We will explore more on this in the later chapters.



When you use the Bootstrap CSS, all the global and fundamental HTML elements automatically become responsive and would properly behave as the client device on which the web page is browsed.

The built-in components are also designed to be responsive, and as the developer, you shouldn't be worried about how these advanced components would behave in different devices and in different client agents.

Summary

We briefly went through the history of CSS evolution. We saw how the design approach of web portals and pages has changed toward the mobile-first approach and going to be further matured.

We have also covered the basics of Responsive Web Design and realized the need of a frontend framework that will help us develop responsive web pages.

In this context, we have been introduced to Bootstrap framework and have briefly seen the elements that are the part of Bootstrap.

In the next chapter, we will dive deep into Bootstrap, understand the structure, and see how to use Bootstrap. We will also briefly discuss some CSS preprocessor tools and languages.

2

Getting Started with Bootstrap

In the previous chapter, we understood why and where we need to use Bootstrap. We also know the main ingredients on what Bootstrap is made up of. In this chapter, we will download and start using Bootstrap. Importantly, we will start developing our own simple web application using all major features of Bootstrap.

In this chapter, we will cover the following topics:


- How to download Bootstrap
- Understanding the Bootstrap file structure
- Basic introduction of CSS preprocessors
- Start using Bootstrap with a sample web application

Get Bootstrap


Bootstrap, being an open source software (released under MIT license), full source code is available freely.

There are two ways how you can download Bootstrap.

- Get the precompiled bundle:
 1. Navigate to <https://github.com/twbs/bootstrap/releases/download/v3.3.1/bootstrap-3.3.1-dist.zip>.
 2. This will download the compiled and minified `bootstrap-3.3.1-dist.zip` file, which contains all the packaged CSS, JavaScript, and Font files.

 This book is structured with Bootstrap version 3.3.1, which is the latest version at the time of writing.

- Get the full source code:
 1. Navigate to <https://github.com/twbs/bootstrap/archive/v3.3.1.zip>.
 2. This will download the full source code bundle of Bootstrap. You have to install and compile the source code before actually using this.

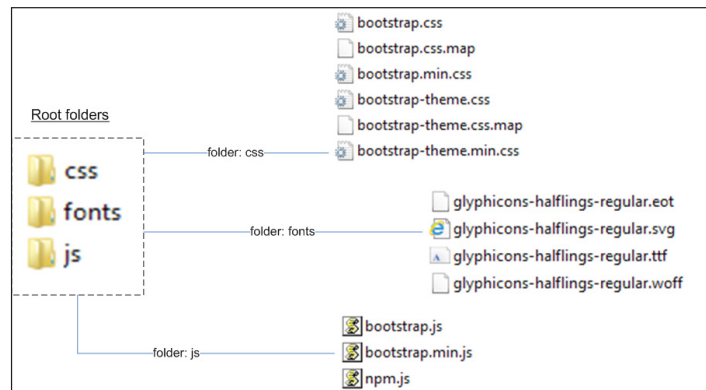
 In later chapters, we will see how to set up the development environment for Bootstrap source code and how to compile and generate the distributable files.
In case you are planning to build a development environment of Bootstrap and customizing the framework, downloading the full source code is required. Otherwise, you can very well do with the distributable pre-compiled bundle.

The Bootstrap file structure

Now that we have downloaded both the flavors of Bootstrap, let's examine what the downloaded archives contain. However, for your usage, you would download any one of them.

A precompiled bundle

A precompiled and distributable bundle comes with three folders. The following is the snapshot of the folders and their contents.




folder: css

This folder encapsulates all the Bootstrap CSS files. For development purposes, you may use the full CSS files (nonminified copies); however, for production deployment, it is always suggested to use the minified versions – such as `bootstrap.min.css` and `bootstrap-theme.min.css`. However, using the bootstrap theme is optional, as the original bootstrap CSS comes with colors and other styling tags. Having said this, it is always better and suggested to use the theme file as well because it gives you a clean way to override the color combinations and themes of original Bootstrap. Thus, when you want to implement your own color scheme and branding, you would override and change the theme file, rather than the original Bootstrap CSS file.

folder: fonts


This folder comes with all the custom fonts Bootstrap uses. One of the major utilizations of this is to generate the Glyphicons used in Bootstrap.

 We will later see how and where you would use these Glyphicons.

folder: js

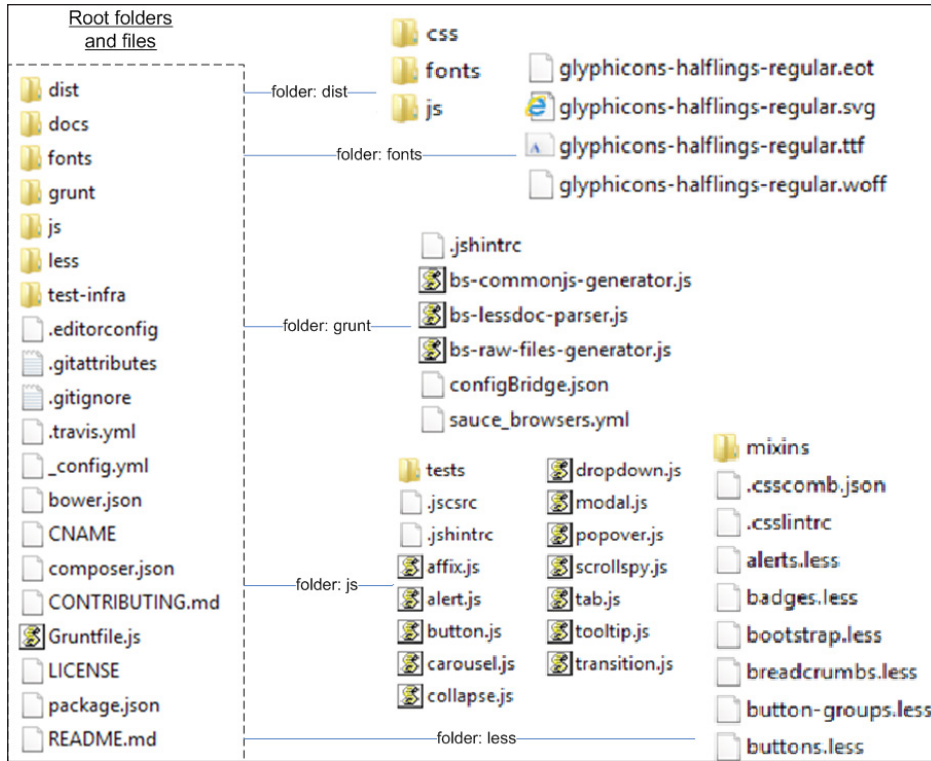
All JavaScript components and plugins of Bootstrap are stored in this folder. Again you should use the minified version – the `bootstrap.min.js` file for production deployment.

As you know, the Bootstrap JS plugins use jQuery, thus before using these JavaScript add-ons, you must include jQuery in your HTML page.

 By default, this JS file comes with all the JavaScript plugins offered by Bootstrap, however, if you require to use only some selected plugins, you can include specific JS files as well. We will explore more on this in the later chapters.

A source code bundle

Here are the files and folders that come with the source code bundle.



As you can see, the source code bundle is quite exhaustive, distributing lot many things.

However, the following are the most important of all.

rfolder: dist

This is the folder where all final distributable files are collected. If you see the contents of this folder are the same as the distributable bundle we saw in the previous section.

When we will build the development environment for Bootstrap (in the later chapters), we will see that compiled output files will be stored in this folder.

folder: fonts

Source folder and files for all the fonts used in Bootstrap.

folder: grunt

Bootstrap uses Grunt to execute all the build-, compilation-, and packaging-related tasks. This folder contains all those relevant and required files.


 We will see how to use Grunt to compile Bootstrap in detail in the later chapters.

folder: js

All the source files for the Bootstrap JavaScript plugins are stored here.

folder: less

Bootstrap uses CSS preprocessor LESS for compilation, customization, and packaging purposes. This folder contains all the LESS files to generate the final CSS files.

 There are other files and folders also, in the source code bundle; however, we will explore more on these when we will discuss about the Dev environment of Bootstrap.

CSS preprocessors

You might have noticed that we just mentioned about the LESS files in Bootstrap source bundle. In this context, before continuing further into Bootstrap, we will briefly discuss this new language and technique in CSS3.

A CSS preprocessor is a program that takes some preprocessed code and convert them into the good old CSS code that browsers understand. This model has been innovated to make CSS more dynamic, productive, multiusable, efficient, and organized.

Traditionally, CSS has not been a programming language (it is still not), and all those powers you enjoy in a language (such as .NET, Java, or PHP) are simply absent. Coding in CSS has always been Write Everything Twice (WET), now with preprocessors finally we can do **Don't Repeat Yourself (DRY)** programming as well.

To understand the scenario better, let's take some basic examples.

Here is a standard CSS segment we are in the habit of writing:

```
.text-large {
  font-family: Arial, sans-serif, verdana;
  font-weight: bold;
  font-size: 18px;
  text-transform: uppercase;
  line-height: 1.4em;
  color: #ccc;
}

.text-medium {
  font-family: Arial, sans-serif, verdana;
  font-weight: bold;
  font-size: 14px;
  text-transform: uppercase;
  line-height: 1.2em;
  color: #ccc;
}

.text-small {
  font-family: Arial, sans-serif, verdana;
  font-weight: bold;
  font-size: 12px;
  text-transform: uppercase;
  line-height: 1.0em;
  color: #ccc;
}
```

It is obvious to note that many of those attributes are duplicated. More painful it is when we want to change one attribute (for example `color`)— we need to change this in all the three places.

Enter now a preprocessor language. Our new code would be:

```
.text {
  font-family: Arial, sans-serif, verdana;
  font-weight: bold;
  text-transform: uppercase;
  color: #ccc;
}

.text-large {
```

```
.text;
font-size: 18px;
line-height: 1.4em;
}


.text-medium {
  .text;
  font-size: 14px;
  line-height: 1.2em;
}

.text-small {
  .text;
  font-size: 12px;
  line-height: 1.0em;
}
```

So what we have done is simply created a class with all the common attributes in place and included that class in all the required places. Thus, if I want to change the color, I would just do it once in the `.text` class, and all the others will be updated.

This greatly improves the reusability and efficiency of your CSS development and maintenance. Earlier we just saw the syntax of one CSS preprocessor language—LESS. There are quite a few other similar languages, such as Sass, Stylus, Turbine, and Switch CSS. Although **Syntactically Awesome Stylesheets (Sass)** and LESS are among the most popular of them.

There are quite a few interesting and powerful features being offered by these languages. Some important features are given below.

 Please note that, wherever not explicitly mentioned, all the following syntaxes are of the LESS preprocessor language.

Variables

Now, you can define variables also in the preprocessed code. For example,

```
@color-main: #444;
@color-nav: #333;
@color-text: #222;
@color-foot: #111;
body {
```



```
    color: @color-main;
  }
  div {
    color: @color-text;
  }
  .navigation {
    color: @color-nav;
  }
  .footer {
    color: @color-foot;
  }
}
```

Mixins

As per Wikipedia, "a mixin is a class that contains a combination of methods from other classes." In case of CSS, this greatly enhances reusability, since you define one class and set of rules there, and reuse that set in a number of instances elsewhere.

The first code segment earlier (the one with the `.text`, `.text-large` and other classes) is an example of mixin written in LESS. There we have defined the `.text` class once and have used that in more than one places.

Operations

Interestingly enough, now you can do mathematical operations as well using CSS preprocessors. For example, in LESS, you write this code:

```
@color-main: #111;
.header {
  color: @color-main * 3;
}
```

The resulting CSS code that would get generated will be simply as follows:

```
.header {
  color: #333;
}
```

Nesting

Consider this code segment:

```
h1 {
  font-family: Arial, sans-serif, verdana;
  font-size: 12px;
  font-weight: bold;
  line-height: 1.4em;
  a {
    color: black;
    &:hover {
      text-decoration: none;
    }
  }
}
```

This definitely looks more organized and structured than writing the following code:

```
h1 {
  font-family: Arial, sans-serif, verdana;
  font-size: 12px;
  font-weight: bold;
  line-height: 1.4em;
}
h1 a {color:black;}
h1 a:hover {text-decoration:none;}
```

The first one is the preprocessed code you would write in LESS, and the second one is a generated (or processed) CSS code ready to be rendered in the browser – which is nothing different from what we were usually writing in earlier days.

Since each of these preprocessors is just another language, they have their own syntaxes and way of writing the code – which is not essentially the syntax you are familiar with CSS. However, after processing (that is, compilation), they all generate the standard and same CSS code that the browser can comprehend.




The scope of this book does not cover full installation and working guide for CSS preprocessors, so we will restrict ourselves to the Bootstrap context only.

Bootstrap supports Sass and LESS both; in this book, we will be covering Bootstrap with LESS. In the later chapters, we will see how these features have been used in various places in Bootstrap.

How to use Bootstrap

As you might have guessed, to use Bootstrap you need to include all the CSS and JavaScript files in your HTML page. One way to do this is after downloading Bootstrap and include the CSS and JavaScript files, or another way is to directly use Bootstrap's CDN files.

 CDN stands for Content Delivery (or Distribution) Network. It is a large system of the distributed servers over the Internet across multiple locations. The objective is to provide high performance and high availability of content to the end users.

In case of a downloaded local copy of Bootstrap, we add this in the HTML HEAD section:

```
<link rel="stylesheet" href="bootstrap.min.css">
<link rel="stylesheet" href="bootstrap-theme.min.css">
<script src="bootstrap.min.js"></script>
```

Also, in case of Bootstrap's CDN:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.1/css/bootstrap.min.css">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.1/css/bootstrap-theme.min.css">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/
bootstrap.min.js"></script>
```

Here is our first HTML page using Bootstrap:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Welcome to the Online Megastore</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link href="bs/css/bootstrap.min.css" rel="stylesheet">
    <link href="bs/css/bootstrap-theme.min.css" rel="stylesheet">
    <script type="text/javascript" src="js/jquery-2.0.3.min.js"></
script>
    <script type="text/javascript" src="bs/js/bootstrap.min.js"></
script>
  </head>
  <body>
    <div class="container">
```

```

    <div class="page-header"><h1><span class="glyphicon glyphicon-
home"></span>&nbsp;&nbsp;&nbsp;Welcome to the Online Megastore</h1></div>
    <div class="row">
      <div class="col-xs-12 col-sm-12 col-md-4 col-lg-4">
        <div class="panel panel-info">
          <div class="panel-heading">Latest Additions</div>
          <div class="panel-body">...</div>
        </div>
      </div>
      <div class="col-xs-12 col-sm-12 col-md-8 col-lg-8">
        <div class="panel panel-warning">
          <div class="panel-heading">Most Popular Products</div>
          <div class="panel-body">...</div>
        </div>
      </div>
    </div>
    <div id="footer">
      <div class="container">
        <div class="text-muted pull-left"><a href="http://www.
packtpub.com" target="_blank">Bootstrap&reg; Sample App</a></div>
        <div class="text-muted pull-right">&copy; <a href="http://
about.me/snig" target="_blank">Snig Bhaumik</a> 2015</div>
      </div>
    </div>
  </body>
</html>

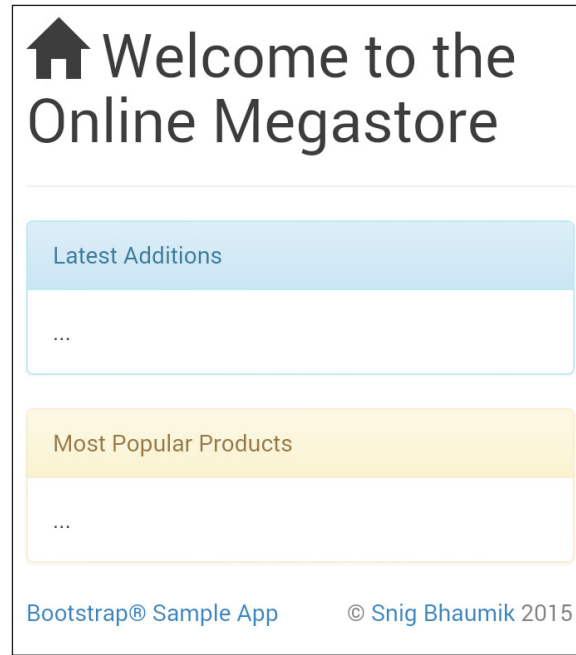
```

The following is the output would look like on a desktop screen:



Well of course, it may not be a very impressive web page, but we still have started creating a web page using the elementary classes of Bootstrap.

However, more importantly, the following is the view of the same page on a mobile screen:



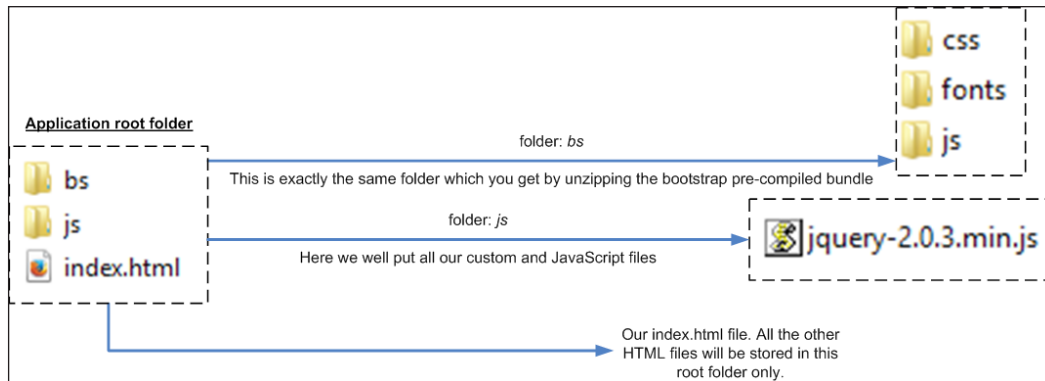
As you may have noticed, that the two blocks have been automatically adjusted and rendered in a vertical pattern, whereas in the desktop, they are rendered in horizontal pattern.

We will be creating a sample basic online store application in this book. In each of the later chapters we will explore and learn more about each Bootstrap features; and will add more functionalities in this application.

In the end, we will end up creating a full simple and responsive application user experience using Bootstrap. Here we are starting to create the base structure of the application.

The application folder structure

To start with, here is our application folder structure:



Please keep in mind that we are restricting ourselves only to create the HTML/client side part of the application; of course, the server side or data transaction part is out of the scope of this book.

We will use the downloaded copies of all the files in our application, rather than using any CDN links.

The `bs` folder is just the unzipped and unmodified version of the precompiled bundle. In this folder, we have all the required files for Bootstrap, ready to be consumed. Thus, in the HTML code, we have referenced the CSS and JS files in the following way:

```
<link href="bs/css/bootstrap.min.css" rel="stylesheet">
<link href="bs/css/bootstrap-theme.min.css" rel="stylesheet">
<script type="text/javascript" src="bs/js/bootstrap.min.js"></script>
```

The `js` folder is to have all the custom (our application-specific files) and other third-party JavaScript files. We should not merge any other JS or CSS files in Bootstrap folder; this is why, we have created a separate folder to store all those files other than Bootstrap. In this case, we have added the jQuery script file here (`jquery-2.0.3.min.js`).



Just to remind you that Bootstrap uses and is dependent on jQuery for all its JavaScript plugins. Hence, we have added jQuery before referencing the Bootstrap JS file.

The `index.html` file is the home page of our simple application – in the earlier text, you saw the code of this file. We will add more HTML files at this location as and when required by the application.

Summary

In this chapter, we got introduced to the CSS preprocessors available in the market; we also understood some of the features and advantages of using these preprocessor tools and languages.

We saw how to download Bootstrap framework and also what the flavors available in the Bootstrap package are. We learned that the files and folders are available and included in each of those distributed packages.

Finally, we learned how to create a simple web page including and using Bootstrap.

We have started creating a sample web application and created our first page using Bootstrap CSS. We also laid a foundation structure on which we will develop the sample web application.

In the next chapter, we will explore all the CSS classes and features offered in Bootstrap and will add more functionality to our web application using these classes.

3

Creating Responsive Layouts Using Bootstrap CSS

The basic idea and thumb rule in working with Bootstrap is that you shouldn't create your own CSS classes and styles; rather, you should always utilize and adapt the classes provided by the framework. This way you will be able to leverage the full power and potential offered by Bootstrap. For this, you must be aware of all the classes supplied by the framework. Of course, you can update and extend the classes as per your needs and branding; we will see how to do this later.

In this chapter we will go through most of the major CSS classes that Bootstrap is equipped with. We will cover the following:

- The Bootstrap HTML file structure
- Basic HTML elements
- The details of responsive classes in Bootstrap
- Handling images
- Understanding the all-important grid system in Bootstrap
- Building data-entry forms

Basic HTML structure for Bootstrap

In this section we will explore the basic HTML file structure when you will use Bootstrap as the front-end framework.

The head section

Just to recap, your HTML file must be marked as HTML5 `doctype`. Thus you always include this at the very beginning of your file:

```
<!DOCTYPE html>
<html lang="en">
  ...
</html>
```

Second important thing is that you must include `meta viewport` tag in your head section. For example:

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

After this, you would of course include the Bootstrap CSS and JS files. Thus, the minimalistic head section of your HTML file would look as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>...</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link href="bs/css/bootstrap.min.css" rel="stylesheet">
    <link href="bs/css/bootstrap-theme.min.css" rel="stylesheet">

    <script type="text/javascript" src="js/jquery-2.0.3.min.js"></
script>
    <script type="text/javascript" src="bs/js/bootstrap.min.js"></
script>
  </head>

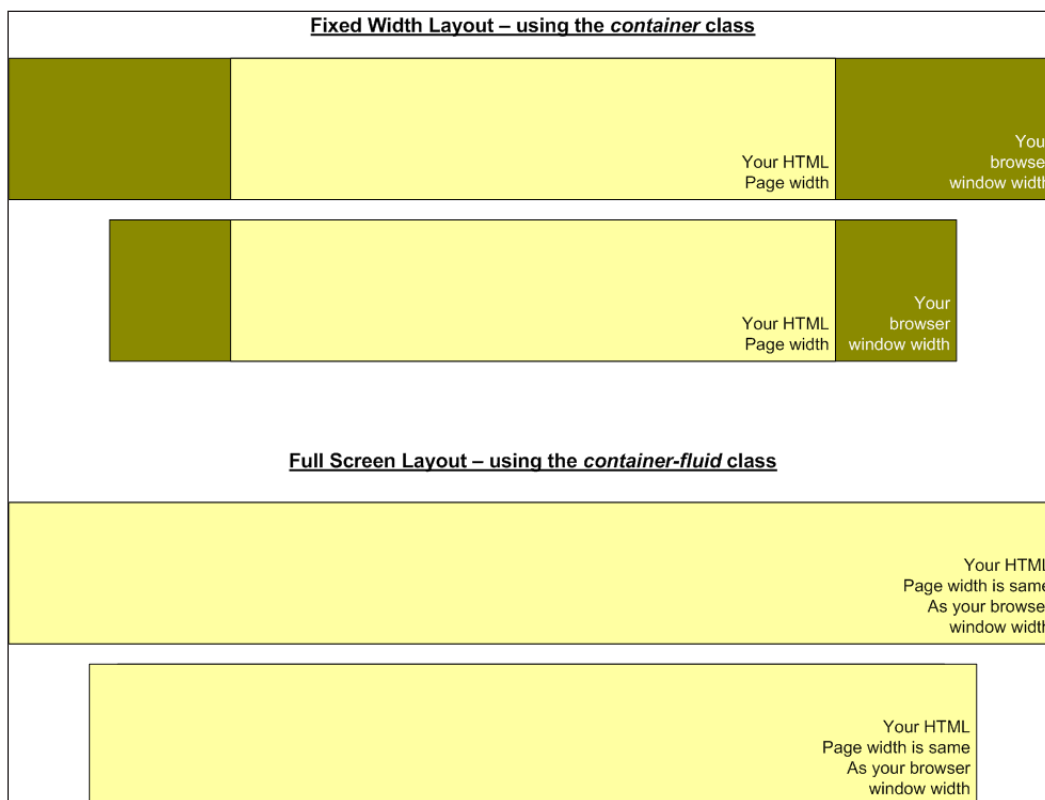
  <body>
    ...
  </body>
</html>
```

The body section

All of your HTML body section must be positioned inside a `div` element with class attached as `container` or `container-fluid`:

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

The `container` class ensures that your page will be rendered as a responsive fixed width style. While, the `container-fluid` class renders your page as a responsive full-width design. Hence, if you want your page to be of fixed width in the middle of the screen irrespective of the width of the browser window, then use `container` class. On the other hand, if you want that your page always consumes the full screen, you need to use `container-fluid` class instead. The following diagrams illustrates this:



Basic HTML elements

As you might expect, all the standard HTML5 elements are considered and styled in Bootstrap. Thus you would simply use any basic HTML elements and these will be rendered as Bootstrap specific styles and behavior.

For example, all heading tags from H1 to H6 can be used as they are. You simply write:

```
<h1>This is a Heading</h1>
```

Interestingly enough, if you want to use the same heading styles in any of your other elements, you can simply use the provided `.h1` CSS class.

All the following standard HTML5 elements are covered in Bootstrap's responsive styling set:

- Paragraph - `<p>`
- Highlight - `<mark>`
- Deleted text - ``
- Inserted text - `<ins>`
- Strikethrough - `<s>`
- Underline - `<u>`
- Bold - `` or ``
- Italic - `` or `<i>`
- Abbreviation - `<abbr>`
- Address - `<address>`
- Blockquote - `<blockquote>`
- Lists - ``, ``, ``
- Description lists - `<dl>`, `<dd>`
- Codes - `<code>`, `<samp>`, `<kbd>`, `<var>`
- Tables - `<table>`, `<thead>`, `<tbody>`, `<tr>`, `<td>`
- Form elements - `<form>`, `<label>`, `<text>`, `<textarea>`, `<password>`, `<datetime>`, `<number>`, `<email>`, `<date>`, `<month>`, `<week>`, `<time>`, `<tel>`, `<color>`, `<url>`, `<search>`, `<input>`, `<button>`

Responsive classes

Since the main focus of Bootstrap is to create mobile optimized website with a minimal effort, the framework offers a number of classes to control your desktop and mobile version of the website.

Understanding the basics

Across all the packaged classes and styles Bootstrap uses four different markers for controlling and mentioning element and device sizes. The following table lists and details these four markers:

Marker name	Target device / screen size	Element
xs	Applies to extra small devices, such as phones with < 768px width. Example class name: .hidden-xs	Will render the element with extra small size. For example, in case of buttons, font-size will be 12px, and line-height will be 1.5, padding will be 1px 5px. Use: .btn-xs
sm	Applies to small devices, such as phablets and tablets with width \geq 768px and < 992px Example class name: .hidden-sm	Will render the element with small size. For example, in case of buttons font-size will be 12px, and line-height will be 1.5, and padding will be 5px 10px. Use: .btn-sm
md	Applies to medium size devices, such as desktops with width \geq 992px and <1200px Example class name: .hidden-md	Default size
lg	Applies to large devices, such as big desktop monitors and TVs with \geq 1200px width Example class name: .hidden-lg	Will render the element with large size. For example, in case of buttons, font-size will be 18px, line-height will be 1.33, and padding will be 10px 16px. Use: .btn-lg

Controlling display of elements across devices

If you add a `div` element in your HTML code with CSS class as `.hidden-sm`, this `div` element will be hidden in all small devices (devices with width $\geq 768\text{px}$ and $< 992\text{px}$); and will be displayed in all the other devices (those that do not lie within this defined width range). Bootstrap automatically detects the size and resolution of the client device, and will render the elements correspondingly.

The following table lists these classes and corresponding renditions:

Class name	Extra small devices, such as phones with $< 768\text{px}$ width	Small devices, such as phablets and tablets with width $\geq 768\text{px}$ and $< 992\text{px}$	Medium sized devices, such as desktops with width $\geq 992\text{px}$ and $< 1200\text{px}$	Large devices, such as big desktop monitors and TVs with $\geq 1200\text{px}$ width
<code>.hidden-xs</code>	x	✓	✓	✓
<code>.hidden-sm</code>	✓	x	✓	✓
<code>.hidden-md</code>	✓	✓	x	✓
<code>.hidden-lg</code>	✓	✓	✓	x

For example, see the following code:

```
<div class="hidden-xs">This DIV will be hidden in extra small devices,
and will be displayed in all other devices.</div>
```

Similarly, if you want that your `div` element should be displayed only in case of large devices and not in smaller devices (a very common use case when you do not want a heavy media file to be rendered in a mobile phone), you just use the class `.visible-lg-block`:

Class name	Extra small devices, such as phones with $< 768\text{px}$ width	Small devices, such as phablets and tablets with width $\geq 768\text{px}$ and $< 992\text{px}$	Medium size devices, such as desktops with width $\geq 992\text{px}$ and $< 1200\text{px}$	Large devices, such as big desktop monitors and TVs with $\geq 1200\text{px}$ width
<code>.visible-xs-block,</code> <code>.visible-xs-inline,</code> <code>.visible-xs-inline-block</code>	✓	x	x	x

Class name	Extra small devices, such as phones with <768px width	Small devices, such as phablets and tablets with width ≥768px and <992px	Medium size devices, such as desktops with width ≥992px and <1200px	Large devices, such as big desktop monitors and TVs with ≥1200px width
.visible-sm-block, .visible-sm-inline, .visible-sm-inline-block	x	✓	x	x
.visible-md-block, .visible-md-inline, .visible-md-inline-block	x	x	✓	x
.visible-lg-block, .visible-lg-inline, .visible-lg-inline-block	x	x	x	✓

For example, see the following code:

```
<div class="visible-lg-block">This DIV will be displayed only in large devices, and will be hidden in all other devices.</div>
```

Thus you can control the visibility of your HTML elements depending on the target device the end user is using with the help of the preceding classes. In the previous example, the `div` element will only be displayed in large devices, not in any smaller or medium devices.

If you want your HTML element to be shown in both medium and large devices, but not in smaller devices, you can add both the classes `.visible-md-block` and `.visible-lg-block`.

Rendering images

As you know, controlling the image rendering as per the available space and screen size is a challenge, especially in case of responsive web sites.

Bootstrap again comes to the rescue – you just add `.img-responsive` class into your `IMG` HTML element, and this's all.



It is important to note that, except the images, most of the other HTML elements respond to the target device automatically when you include the Bootstrap CSS in your page. However, in case of images, you need to add this class manually.

This simply adds the following CSS styles to your image:

```
.img-responsive
{
display: block;
max-width: 100%;
height: auto;
}
```

Thus, the image captures full width as per its container element size, and the height of the image is scaled as per its aspect ratio, this happens irrespective of the actual size of the image.



As you can see, the responsive image class consumes 100 percent width of the container element, it is up to you to structure the parent elements properly.

For example, if you fix the width of the parent element with hard coded size, then whole purpose of having this `.img-responsive` class is voided. You always need to make sure that you are not putting any fix width on any of your HTML elements in your page in order to make your web page properly responsive.

There are a few utility classes as well that come handy for styling your image. For example, if you want your image to be rendered with soft rounded corners (a pattern that is most popular after iOS design), you use the class `.img-rounded`. It does nothing but add a 6px border radius to the image:

```
.img-rounded {
border-radius: 6px;
}
```

Also if you want your image to be rendered as a circle (or oval, depending on the aspect ratio of the image), you use the class `.img-circle`:

```
.img-circle {
  border-radius: 50%;
}
```

Finally, if you want a small outline border for your image, use the class `.img-thumbnail`. Thus, to create a rounded responsive image you use both the classes `.img-rounded` and `.img-responsive`:

```

```



One exception is `.img-thumbnail` class; you need not include the `.img-responsive` class if you are using the `.img-thumbnail` class, since most of the responsive stylings are included in the thumbnail class itself.

Showing responsive images in a sample application

Let's now add some product images in our shopping cart web application. Here's the code:

```
<ul class="list-inline">
  <li></li>
  <li></li>
  <li></li>
</ul>
```


The output will be similar to the following screenshot:



Please note that the following points:

- We have used the `.list-inline` class of UL in order to show the product images in a single horizontal line
- However, in case of mobile phones with very small width, the images will be stacked vertically.
- The **You recently visited** caption does not come from the preceding code; we will see the full code soon in the coming sections.

The grid system

One of the basic thumb rule of designing a responsive website is to get rid of traditional HTML tables as far as possible and use fluid div elements instead. Bootstrap offers very strong framework of grid system with extensive use of `div` elements which is dynamically compatible with all the devices and screens currently available in the market.

Basic architecture of the grid system of Bootstrap is that the screen is horizontally divided into maximum 12 columns (logically). And Bootstrap has provided classes to create each combination of columns – for each device sizes. The following matrix illustrates this:

Class Names	Description
<code>.col-xs-1, .col-sm-1, .col-md-1, .col-lg-1</code>	Classes to use to consume one column in the corresponding device size.
<code>.col-xs-2, .col-sm-2, .col-md-2, .col-lg-2</code>	Classes to use to consume two columns in the corresponding device size.
<code>.col-xs-3, .col-sm-3, .col-md-3, .col-lg-3</code>	Classes to use to consume three columns in the corresponding device size.

Class Names	Description
.col-xs-4, .col-sm-4, .col-md-4, .col-lg-4	Classes to use to consume four columns in the corresponding device size.
.col-xs-5, .col-sm-5, .col-md-5, .col-lg-5	Classes to use to consume five columns in the corresponding device size.
.col-xs-6, .col-sm-6, .col-md-6, .col-lg-6	Classes to use to consume six columns in the corresponding device size.
.col-xs-7, .col-sm-7, .col-md-7, .col-lg-7	Classes to use to consume seven columns in the corresponding device size.
.col-xs-8, .col-sm-8, .col-md-8, .col-lg-8	Classes to use to consume eight columns in the corresponding device size.
.col-xs-9, .col-sm-9, .col-md-9, .col-lg-9	Classes to use to consume nine columns in the corresponding device size.
.col-xs-10, .col-sm-10, .col-md-10, .col-lg-10	Classes to use to consume ten columns in the corresponding device size.
.col-xs-11, .col-sm-11, .col-md-11, .col-lg-11	Classes to use to consume eleven columns in the corresponding device size.
.col-xs-12, .col-sm-12, .col-md-12, .col-lg-12	Classes to use to consume all twelve columns in the corresponding device size.

We will take a few example scenarios to make this clear. In our first scenario, suppose we want to create three equal columns – irrespective of all device sizes. The code for this is as follows:

```
<div class="row">
  <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
    Column 1: col-xs-4 col-sm-4 col-md-4 col-lg-4
  </div>
  <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
    Column 2: col-xs-4 col-sm-4 col-md-4 col-lg-4
  </div>
  <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
    Column 3: col-xs-4 col-sm-4 col-md-4 col-lg-4
  </div>
</div>
```

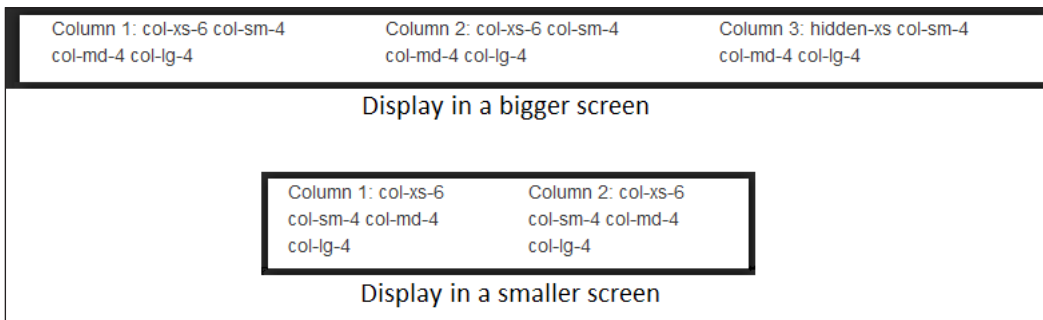
Here is the output (as seen in all the devices):

Column 1:	Column 2:	Column 3:
col-xs-4	col-xs-4	col-xs-4
col-sm-4	col-sm-4	col-sm-4
col-md-4	col-md-4	col-md-4
col-lg-4	col-lg-4	col-lg-4

In our second scenario, suppose we want to display only first two columns in smaller screens, third column will not be displayed and in bigger screens we retain the same design. Our code is as follows:

```
<div class="row">
  <div class="col-xs-6 col-sm-4 col-md-4 col-lg-4">
    Column 1: col-xs-6 col-sm-4 col-md-4 col-lg-4
  </div>
  <div class="col-xs-6 col-sm-4 col-md-4 col-lg-4">
    Column 2: col-xs-6 col-sm-4 col-md-4 col-lg-4
  </div>
  <div class="hidden-xs col-sm-4 col-md-4 col-lg-4">
    Column 3: hidden-xs col-sm-4 col-md-4 col-lg-4
  </div>
</div>
```

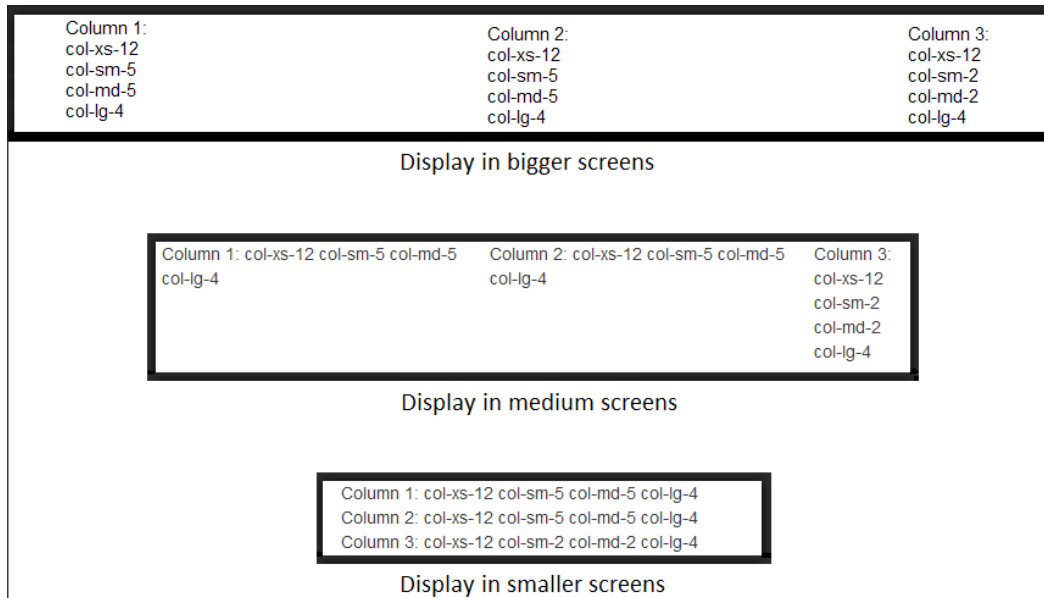
So the output is as follows:



In the third scenario, suppose in smaller screens we need all the columns to be vertical. In medium screens first two columns should consume more space while in large screens they should be equal. Here is our code:

```
<div class="row">
  <div class="col-xs-12 col-sm-5 col-md-5 col-lg-4">
    Column 1: col-xs-12 col-sm-5 col-md-5 col-lg-4
  </div>
  <div class="col-xs-12 col-sm-5 col-md-5 col-lg-4">
    Column 2: col-xs-12 col-sm-5 col-md-5 col-lg-4
  </div>
  <div class="col-xs-12 col-sm-2 col-md-2 col-lg-4">
    Column 3: col-xs-12 col-sm-2 col-md-2 col-lg-4
  </div>
</div>
```

And the output is as follows:



As you can see, the width (and visibility) of the three columns are being automatically controlled by the classes as per the target device and screen size.

To create a placeholder of one column width, you use the `.col-*-1` class, for two columns width, you use the `.col-*-2` class; and so on. For a placeholder which consumes all 12 columns of space, you use `.col-*-12` class. Here `*` stands for the corresponding device size you are targeting (such as `xs`, `sm`, `md` and `lg`).

One of the most important points here is that the total size of the columns must always be 12 – if it is less than 12, some space will be left unused, if it is more than 12, the last placeholder will be wrapped in the next line.

This is why in the first scenario, we have used classes with column size of four for all the device sizes as in `col-xs-4`, `col-sm-4`, and so on. Since we have three placeholders in a row, each placeholder becomes of size 4 ($3 \times 4 = 12$).

In the second scenario, for extra small devices we have made first two placeholders each with `col-xs-6` while the third placeholder is hidden ($6+6 = 12$).

In case of the third scenario, we have made the first placeholder spanning across the full width for smallest devices (`col-xs-12`). For medium devices, we have made first two columns consuming most space ($5+5+2 = 12$). And for large devices, all placeholders are equal.

Thus you can see, Bootstrap's grid system enables you to create tabular layouts for all the device sizes quite effortlessly, and without any extra complex CSS/JavaScript coding.

Constructing data entry forms

Forms are integral parts of a web page – whenever you need to capture any information from the user you need to create a web form in your page. Bootstrap offers a variety of ways to design and create HTML forms for both mobile devices and desktops.

Let's create a *Contact Us* form for our web application. Here is our code:

```
<form>
  <div class="form-group">
    <label for="yourName" class="control-label">Name</label>
    <input type="text" class="form-control" id="yourName"
placeholder="Your name please">
  </div>
  <div class="form-group">
    <label for="yourEmail" class="control-label">Email address</label>
    <input type="email" class="form-control" id="yourEmail"
placeholder="Your Email Id">
  </div>
  <div class="form-group">
    <label for="yourComments" class="control-label">Tell us</label>
    <textarea class="form-control" id="yourComments" placeholder="Your
comments" rows="3"></textarea>
  </div>
  <div class="checkbox">
    <label><input type="checkbox">Subscribe Me !!!</label>
  </div>
  <button type="submit" class="btn btn-primary">Post It</button>
</form>
```

Here is what the form would look like:

Name

Email address

Tell us

Subscribe Me !!!

Making the form horizontal

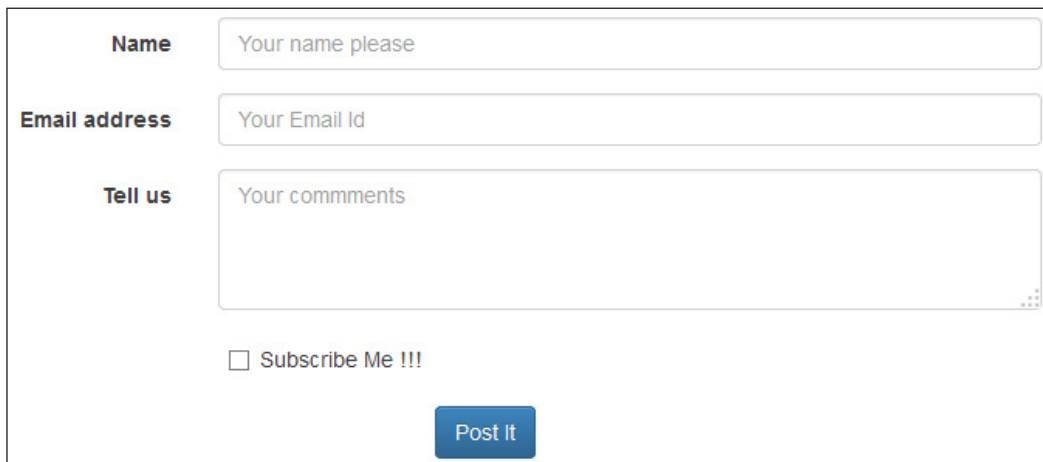
Suppose that now we want to show the labels horizontally alongside the controls.

The code to do this is as follows:

```
<form class="form-horizontal">
  <div class="form-group">
    <label for="yourName" class="col-sm-3 control-label">Name</label>
    <div class="col-sm-9">
      <input type="text" class="form-control" id="yourName"
placeholder="Your name please">
    </div>
  </div>
  <div class="form-group">
    <label for="yourEmail" class="col-sm-3 control-label">Email
address</label>
    <div class="col-sm-9">
      <input type="email" class="form-control" id="yourEmail"
placeholder="Your Email Id">
    </div>
  </div>
  <div class="form-group">
    <label for="yourComments" class="col-sm-3 control-label">Tell us</
label>
```

```
<div class="col-sm-9">
  <textarea class="form-control" id="yourComments"
placeholder="Your comments" rows="3"></textarea>
</div>
</div>
<div class="col-sm-9 col-sm-offset-3">
  <div class="checkbox">
    <label><input type="checkbox">Subscribe Me !!!</label>
  </div>
</div>
<div class="clearfix">&nbsp;</div>
<div class="col-sm-12 text-center">
  <button type="submit" class="btn btn-primary">Post It</button>
</div>
</form>
```

The output is as shown here:



Finalizing the Contact Us page

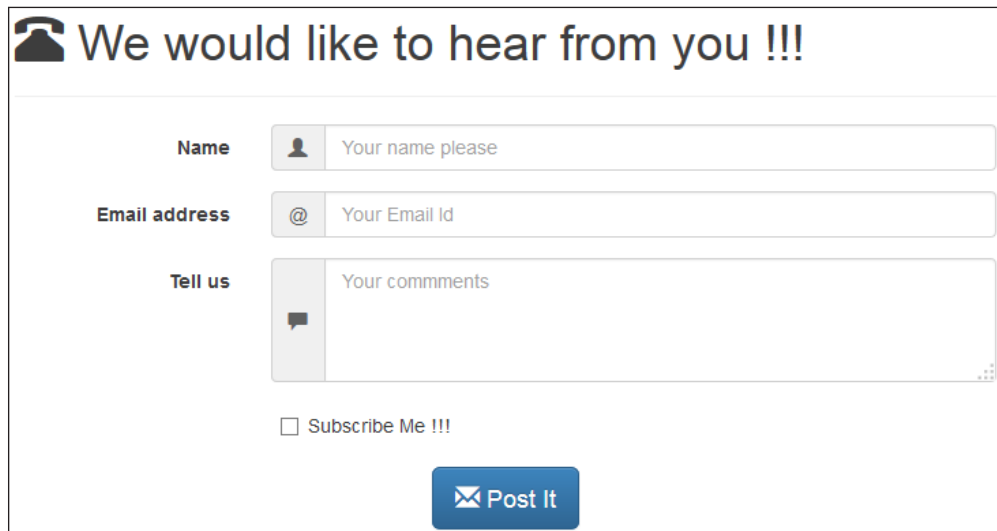
Let's beautify the form a little bit more by adding some contextual icons for each of the text controls. We will create a full *Contact Us* page in our application. Here is the code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Contact Us</title>
```



```
        <span class="input-group-addon glyphicon glyphicon-
comment"></span>
        <textarea class="form-control" id="yourComments"
placeholder="Your comments" rows="3"></textarea>
    </div>
</div>
<div class="col-sm-9 col-sm-offset-3">
    <div class="checkbox">
        <label><input type="checkbox">Subscribe Me !!!</label>
    </div>
</div>
<div class="clearfix">&nbsp;</div>
<div class="col-sm-12 text-center">
    <button type="submit" class="btn btn-primary btn-lg"><span
class="glyphicon glyphicon-envelope"></span>&nbsp;&nbsp;Post It</button>
</div>
</form>
</div>
</body>
</html>
```

The following is how the page would look:



We would like to hear from you !!!

Name

Email address

Tell us

Subscribe Me !!!

Let's now look at what has happened behind the scenes:

- The `.form-group` class is used to encapsulate multiple controls in a group – just as we have done for labels and corresponding text boxes.
- The `.control-label` and `.form-control` classes are used to style the labels and form elements respectively.
- If you want to create a form with horizontal labels and controls – use the `.form-horizontal` class and place the labels and controls as grid columns. The `.control-group` class will act as each row in the form.
- Using the `.input-group` class you can associate multiple controls in adjacently.
- You can create inline forms using the `.form-inline` class, instead of the `form-horizontal` class.
- You can use `disabled` or `readonly` state for the controls – Bootstrap will associate necessary styles automatically, however, all the controls must have `.form-control` class added.
- Use `.has-success`, `.has-warning`, `.has-error` classes to reflect various validation states.
- In order to control the sizes you can use `.input-lg`, `.input-sm` classes. For controlling the size of a group you can use `.form-group-lg` or `.form-group-sm` classes.
- For showing up any help texts, you can use `.help-block` class in a separate label. This will show the texts in a new line.
- We have styled our button with `.btn-primary` class. Apart from this the other available classes are `.btn-success`, `.btn-warning`, `.btn-info`, `.btn-danger` and `.btn-link`.
- We have used Glyphicons to beautify the form. This concept is detailed in later chapters.

Other utility classes

Bootstrap offers a number of other utility CSS classes that you can use for a number of occasions and purposes:

- For web sites that should follow the Web Accessibility standards (<http://www.w3.org/WAI/>), contents must be compatible with screen readers and other equivalent devices. For that, you use `.sr-only` and `.sr-only-focusable` classes. This would ensure that the content is not rendered for normal people and screens; however, it will be spelt out by screen readers.
- For showing or hiding a content you should use `.show` and `.hide` classes respectively.
- For horizontal alignment of content, available classes are `.pull-left` and `.pull-right`.
- Bootstrap provides five different color classes for foreground – `.text-primary`, `.text-danger`, `.text-warning`, `.text-info` and `.text-success`. Similarly, for background you have `.bg-primary`, `.bg-danger`, `.bg-warning`, `.bg-info` and `.bg-success`.
- For creating tables, you would use a standard `<table>` tag; however, along with this we should also use `.table` class for the element. For rendering borders around the tables and cells, the class is `.table-bordered`; and for displaying alternate colored rows you should use `.table-striped` class. Similarly, for changing colors of the rows while mouse hovering, you would use the `.table-hover` class.
- Importantly, in order to create responsive tables, you should use the `.table-responsive` class. Please note that this class only affects the display on devices less than 768px, for larger devices there would be no effect.

Encapsulating everything

We now rewrite our sample application home page using the classes and styles we have discussed in this chapter.



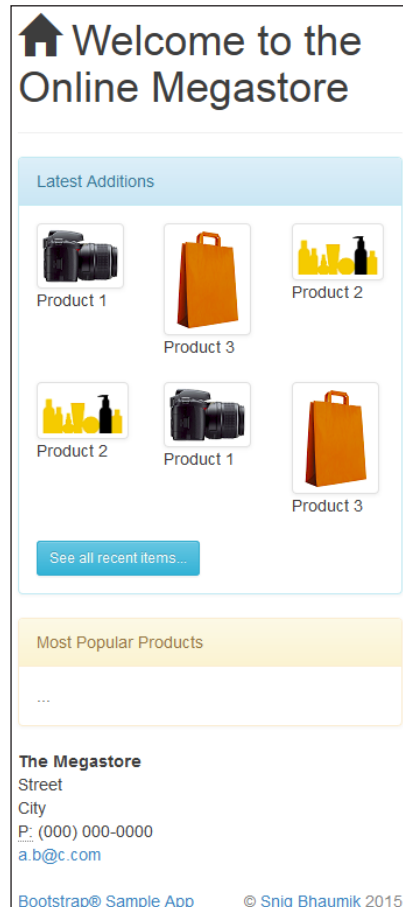
The full source code can be downloaded from Packt Publishing website.

The page now looks like this on a desktop:

The screenshot shows a desktop application home page with the following layout:

- Header:** A home icon followed by the text "Welcome to the Online Megastore".
- Latest Additions:** A light blue section containing six product cards in a 2x3 grid. The top row shows Product 1 (black camera), Product 3 (orange bag), and Product 2 (yellow bottles). The bottom row shows Product 2 (yellow bottles), Product 1 (black camera), and Product 3 (orange bag). A blue button labeled "See all recent items..." is at the bottom.
- Most Popular Products:** A yellow section with the title "Most Popular Products" and a single product card (orange bag).
- You recently visited:** A light green section containing three product cards: black camera, yellow bottles, and orange bag.
- Footer:** Contact information for "The Megastore" (Street, City, P: (000) 000-0000, a.b@c.com) and a copyright notice "© Snig Bhaumik 2015".

And in case of devices with smaller screen size the page is look as follows:



Summary

In this chapter, we have learned about the CSS classes and features offered in the Bootstrap framework. Of course this does not cover all the classes and functionalities that Bootstrap offers in terms of CSS, one chapter may not be enough to cover all the entire range of capabilities that Bootstrap provides. However, we have still covered almost all the major features; and will carry on with the other functionalities in next chapters. We have also added a new page in our sample application as *Contact Us* and restyled our home page with some new layouts and pretty sections.

In the next chapter we will learn about the packaged components that come with Bootstrap.

4

Packaged Components in Bootstrap

Components in Bootstrap can be understood as a collection of related HTML elements each associated with predefined CSS classes in order to generate a particular functionality and UI affect. In the previous chapter, we saw how each HTML element is styled by Bootstrap CSS; in this chapter, we will see how a few elements, each powered by some CSS classes, together can create a new control and generate a full functionality – a new complex component.

For example, you have always created components in HTML such as navigation bar, pagination control, progress bar, alerts, and notifications. Also, you know that each of these are actually a collection of a number of HTML elements grouped together. Most of the times you have used some third-party libraries, or have built the component yourself using lots of HTML, CSS, and JavaScript code.

Now it is the time to use Bootstrap components that are pretty straightforward, simple to use, and most importantly with lesser number of code lines to implement. Of course, all are by default mobile friendly.

In this chapter, we will go through some major components packaged and provided by Bootstrap framework:

- Working with Glyphicons
- Creating the navigation bar
- Designing the badges, toolbars, and button groups
- Showing up alerts and warnings
- Organizing contents with panels and wells
- Controlling navigation with pagination and breadcrumbs

The page header

First thing you would want in your page is a good-looking header at the top of the page. This is a simple HTML heading wrapped on a `div` element.

Here is the code we have used in our application's home page:

```
<div class="page-header"><h1>Welcome to the Online Megastore</h1>
</div>
```

If you fancy a message along with the header, you can use the `<small>` element inside. For example, have a look at the following code:

```
<div class="page-header">
<h1>Welcome to the Online Megastore <small>the one-stop shop</small></
h1>
</div>
```

Glyphicons

As per <http://glyphicons.com/>, the original vendor of Glyphicons, "*this is a library of precisely prepared monochromatic icons and symbols, created with an emphasis on simplicity and easy orientation.*" In simple terms, these are icon fonts to depict and symbolize easy and common behaviors and contexts. For example, if you want to have a hyperlink for home page, you can add an icon depicting this link. Shopping cart has a most common icon; similar examples are login button/link or logout button/link.

These icons are very common in today's web design, and many a times only icons suffice, you may not need to write a text after that. Until now, you will have used small icon images for this purpose; however, now we would use Glyphicons that are very small in size, quick to render, and efficient in mobile devices as well. In addition, they can scale and change color easily, minimizes number of HTTP calls because we are replacing image icons by using these.

In order to use Glyphicons in Bootstrap, you just simply add a `span` element with the CSS class that would be appropriate for your icon. For example, have a look at the following code:

```
<span class="glyphicon glyphicon-home"></span>
```

If you note that we have used this in our page header on the home page, and it has generated the **Home** icon ahead of the page header text. There is a wealth of Glyphicons available in Bootstrap library, which are listed at <http://getbootstrap.com/components/#glyphicons>.

You might recall that in the folder structure of Bootstrap, there was a `fonts` folder—files of this folder are responsible for all functionalities of Glyphicons.

The navigation bar

After the page header, the next important thing is to have a navigation bar for the user to browse through different pages of your website.

Our application pages, as of now, do not have a navigation bar. We have though two pages in our application—let's have a navigation bar for these.

This is a basic and preliminary code for the navigation bar:

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#the-menu">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand brand" href="/index.html">Megastore</a>
    </div>

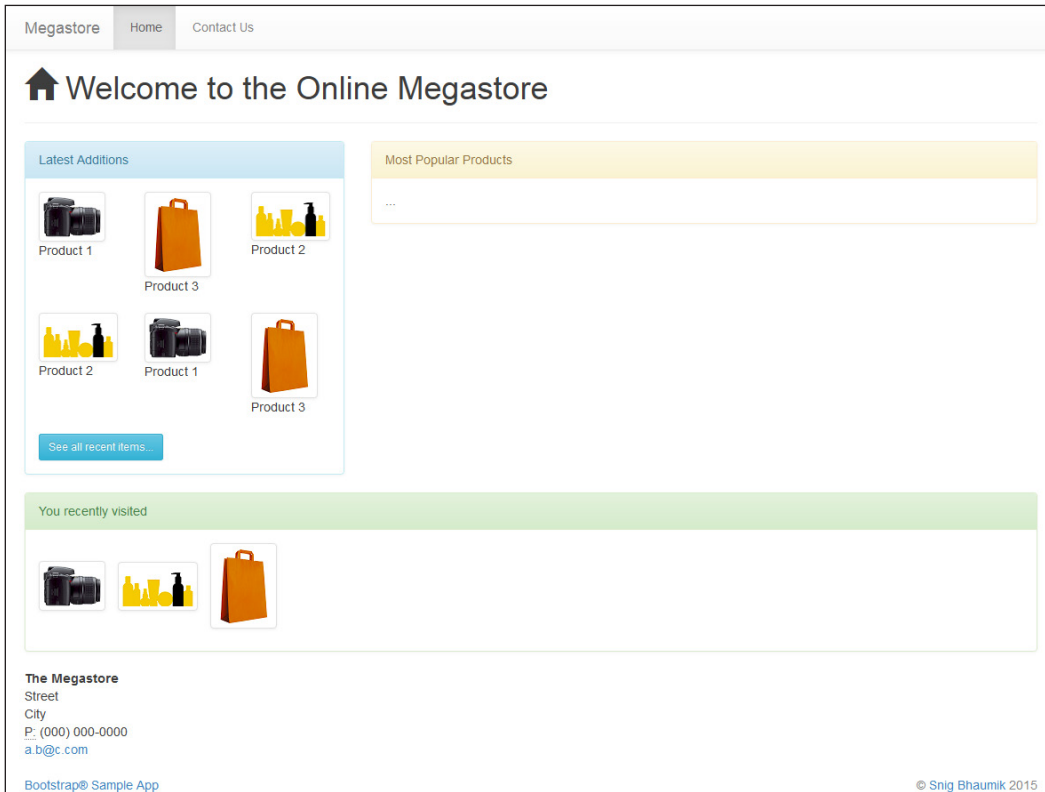
    <div class="collapse navbar-collapse" id="the-menu">
      <ul class="nav navbar-nav">
        <li class="active"><a href="/index.html">Home</a></li>
        <li><a href="/contact.html">Contact Us</a></li>
      </ul>
    </div>
  </div>
</nav>
```

Downloading the example code

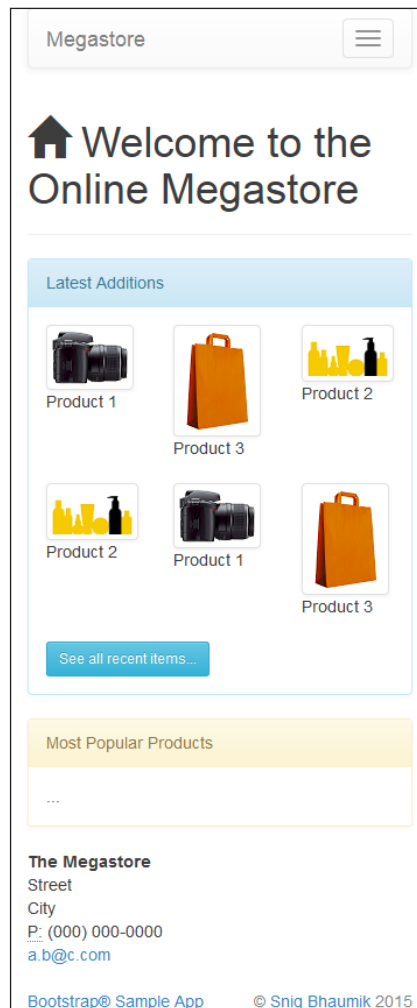


You can download the example code files from your account at <http://www.packtpub.com> for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

We include this code in our index page just under the main `div` container. This renders as follows:



In case of smaller devices such as mobile phones, it would look as shown in the following screenshot:




Most important thing to note here is that the navigation menu has automatically transformed into the hamburger or sandwich icon, as this being the most popular and standard way of displayed menus in mobile devices.

Now, let's add a few more interesting features to our navigation bar. For example, if you vertically scroll down the home page, the navigation bar disappears at the top. We will just add another Bootstrap class to make it fixed at the top.

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

This will fix the menu bar at the top of your browser and will always be visible to the user even if you have a long page and user scrolls down.

 However, you will note that the actual body contents are being overlaid under this fixed navigation bar. In order to rectify this, we need to add a certain top padding to the body element. We have added this in the body tag, however, it is of course suggested to put this in a CSS class:

```
<body style="padding-top: 30px;">
```

Next thing we want to do is to beautify our menu with some icons. For that, we will add Glyphicons for each of the menu items, using the following code:

```
<ul class="nav navbar-nav">
  <li class="active"><a href="/index.html"><span class="glyphicon glyphicon-home"></span>&nbsp;Home</a></li>
  <li><a href="/contact.html"><span class="glyphicon glyphicon-earphone"></span>&nbsp;Contact Us</a></li>
</ul>
```

We also need a login link in the top menu so that the user can log in/log out from the menu itself. We just add another UL link in the navigation menu:

```
<ul class="nav navbar-nav navbar-right">
  <li><a href="#"><span class="glyphicon glyphicon-log-in"></span>&nbsp;Sign In</a></li>
</ul>
```

Finally, a search box to enable users to search for any products from our store is as follows:

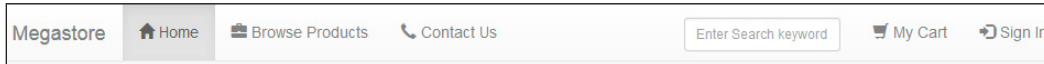
```
<form class="navbar-form navbar-right visible-md-inline visible-lg-inline">
  <div class="form-group" style="margin-top: 4px">
    <label class="sr-only" for="keyword">Keyword</label>
    <input type="text" class="form-control input-sm" id="keyword"
placeholder="Enter Search keyword" />
  </div>
</form>
```

If you notice, we have added this search box only for medium- and large-sized devices. Here is the final source code of the navigation bar. We have added another link (`browse.html`) for a page to display all the products we have in our store. We will create this page later. We have also added a link to see the user's shopping cart:

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#the-menu">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand brand" href="/index.html">Megastore</a>
    </div>

    <div class="collapse navbar-collapse" id="the-menu">
      <ul class="nav navbar-nav">
        <li class="active"><a href="/index.html"><span
class="glyphicon glyphicon-home"></span>&nbsp;Home</a></li>
        <li><a href="/browse.html"><span class="glyphicon glyphicon-
briefcase"></span>&nbsp;Browse Products</a></li>
        <li><a href="/contact.html"><span class="glyphicon glyphicon-
earphone"></span>&nbsp;Contact Us</a></li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        <li><a href="#"><span class="glyphicon glyphicon-shopping-
cart"></span>&nbsp;My Cart</a></li>
        <li><a href="#"><span class="glyphicon glyphicon-log-in"></
span>&nbsp;Sign In</a></li>
      </ul>
      <form class="navbar-form navbar-right visible-md-inline visible-
lg-inline">
        <div class="form-group" style="margin-top: 4px">
          <label class="sr-only" for="keyword">Keyword</label>
          <input type="text" class="form-control input-sm"
id="keyword" placeholder="Enter Search keyword" />
        </div>
      </form>
    </div>
  </div>
</nav>
```

Here is the current look of our navigation bar:

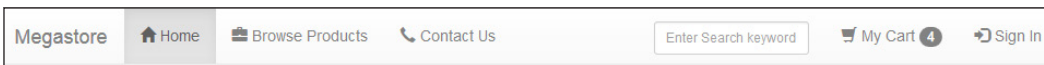


Badges

Badges are simple indicators to show count of the items you want to highlight. Very simple use cases are showing number of unread notifications, e-mails, messages, and many more. We will add this in our navigation bar to show the number of items in the shopping cart. This is the updated line of code:

```
<li><a href="#"><span class="glyphicon glyphicon-shopping-cart"></span>&nbsp; My Cart <span class="badge">4</span></a></li>
```

It looks like in the following way:



As you can see, you just simply add a `span` element with `badge` class, and put the text inside in order to highlight it.

Alerts

You can use alerts to provide communicative information to the user; for example, showing the status of an operation user is trying to do, showing some system information, showing user any warning messages, showing any error messages or status, and much more. Showing basic alerts is pretty simple—just create a `div` element with the `.alert` class in Bootstrap. For example, have a look at the following code:

```
<div class="alert alert-info">You last visited us on ...</div>
```

You can use any of these classes to show contextual information—`alert-info`, `alert-warning`, `alert-success`, or `alert-danger`.

Putting links inside `alert` would require you to add the `.alert-link` in the anchor tag. This will ensure that the links will also be displayed in the corresponding contextual color. As an example, we have added this alert into the last section of our home page:

```
<div class="alert alert-info">You last visited us on Thursday, 20th
March 2015. Please <a href="/contact.html" class="alert-link">let us
know</a> in case of any issues or feedback.</div>
```



There are some other features offered in the Bootstrap alert module, which we will discuss in the next chapter.

Toolbars and button groups

For grouping a number of buttons together, we would use the `.btn-group` class. This class creates a group of buttons; however, each of these buttons must have the `.btn` class associated. For example, use the following code:

```
<div class="btn-group">
  <button type="button" class="btn btn-success">Add</button>
  <button type="button" class="btn btn-warning">Edit</button>
  <button type="button" class="btn btn-danger">Delete</button>
</div>
```

This is what the preceding code will generate:



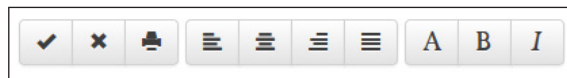
We can club a number of button groups together to generate a toolbar. Let's be a little innovative and use the Glyphicons in the buttons (instead of button text). Here is the code for this:

```
<div class="btn-toolbar">
  <div class="btn-group">
    <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-ok"></span></button>
    <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-remove"></span></button>
    <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-print"></span></button>
  </div>
</div>
```

```
<div class="btn-group">
  <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-align-left"></span></button>
  <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-align-center"></span></button>
  <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-align-right"></span></button>
  <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-align-justify"></span></button>
</div>

<div class="btn-group">
  <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-font"></span></button>
  <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-bold"></span></button>
  <button type="button" class="btn btn-default"><span
class="glyphicon glyphicon-italic"></span></button>
</div>
</div>
```

This is how our toolbar looks like:



You can also use button groups, toolbars, and dropdowns together to generate richer user experience. We will explore those in the next chapter.

Panels

You, of course, want to segregate your web page and contents in to different sections and subsections. It is like you see in normal portal frameworks or was popularized during the days of iGoogle. We have used this model in our homepage also where we had created three different sections – **Latest Additions**, **Most Popular Products**, and **You Recently Visited**.

In Bootstrap, panels can be used to create these sections. Panels in Bootstrap have two separate placeholders – namely header and body – with classes `.panel-heading` and `.panel-body`, respectively.

Just like many other components in Bootstrap, panels also have five different modes— `.panel-info`, `.panel-success`, `.panel-primary`, `.panel-warning`, and `.panel-danger`.

Thus, in our code, we have used:

```
<div class="panel panel-warning">
  <div class="panel-heading">Most Popular Products</div>
  <div class="panel-body">...</div>
</div>
```

It is also advised to use the `.panel-title` class inside `.panel-heading` in order to retain pre-styled headings and display contextual colors of hyperlinks.

Similar to the panel heading, you can also use `.panel-footer` to create the footer section of the panel. Thus, a full panel code can appear as follows:

```
<div class="panel panel-warning">
  <div class="panel-heading">
    <h2 class="panel-title">some title</h2>
  </div>
  <div class="panel-body">contents</div>
  <div class="panel-footer">some footer</div>
</div>
```

One interesting aspect is to show tables inside a panel. Bootstrap is intelligent enough to merge all the paddings and borders and will unify these two controls seamlessly.

Here is the code:

```
<div class="panel panel-warning">
  <div class="panel-heading">
    <h2 class="panel-title">some title</h2>
  </div>
  <table class="table table-hover table-striped">
    <thead><td>caption 1</td><td>caption 2</td></thead>
    <tr><td>line 1</td><td>line 1</td></tr>
    <tr><td>line 1</td><td>line 1</td></tr>
    <tr><td>line 1</td><td>line 1</td></tr>
    <tr><td>line 1</td><td>line 1</td></tr>
  </table>
  <div class="panel-footer">some footer</div>
</div>
```


The output is:

some title	
caption 1	caption 2
line 1	line 1
line 1	line 1
line 1	line 1
line 1	line 1
some footer	


As you can see, the table and its borders are seamlessly adjoined with the panel, giving a cohesive look and feel.

Wells

Wells are almost similar to panels, but with very less features and functionalities. This will generate a container element for you so that a group of logically related HTML contents can be rendered in that container.

As an example, just recap our **Contact Us** page, which we created in the last chapter. Let's put this form inside a well. This is how the page looks like now:

The screenshot shows a web page for 'Megastore' with a navigation bar containing 'Home', 'Browse Products', and 'Contact Us'. A search bar and 'My Cart' are also visible. The main content area features a heading 'We would like to hear from you !!!' and a contact form. The form is contained within a light gray well. It includes input fields for 'Name' (with a person icon), 'Email address' (with an @ icon), and a text area for 'Tell us' (with a speech bubble icon). Below the text area is a checkbox for 'Subscribe Me !!!' and a blue 'Post It' button. The footer contains contact information for 'The Megastore' and a copyright notice for '© Snig Bhaumik 2015'.

 The full source code for the page can be downloaded from <https://www.packtpub.com/>.

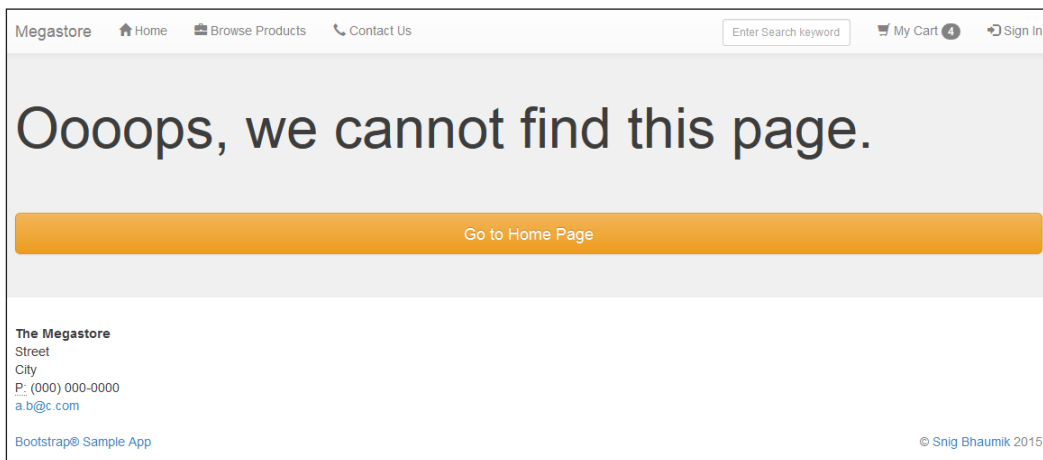
Apart from adding the navigation bar, we can also add a new `div` element with class `.well` and wrap the **Contact Us** form inside that well:

```
<div class="well well-sm">
  <form class="form-horizontal">
```

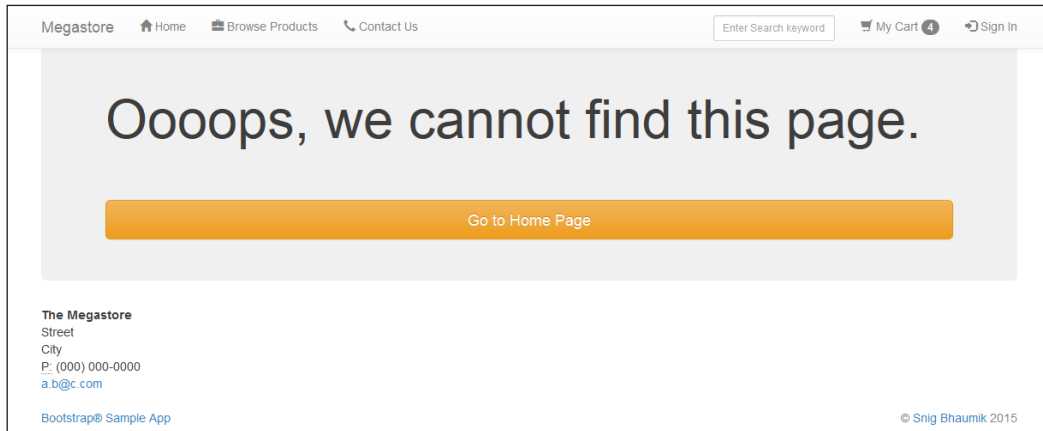
We will now have an entirely different look of our page.

Jumbotron

An interesting addition in Bootstrap is **Jumbotron**. This is a component that optionally captures the whole viewport and renders the contents inside. We can use this component when we want to showcase some contents, or in some pages with very less content but important message to be displayed. One good example use of this can be a 404 page:



As you can see, the section has captured full width of the available screen space. However, if you don't want that, here is another variation:



The code is:

```
<div class="jumbotron">
  <div class="container">
    <h1 style="padding-bottom: 50px">Oooops, we cannot find this
page.</h1>
    <a href="/index.html" class="btn btn-warning btn-lg btn-block">Go
to Home Page</a>
  </div>
</div>
```

The difference between the two variances is only that, in the first case, the `jumbotron` code block is placed outside the main container element, and in the second case, the code block is placed inside the main container element.

Hence, in the first case, it was capturing the full width of the screen, and in the second variation, it is rendered as a fixed width view.

Breadcrumbs

This is a simple component to generate the breadcrumb in your web page. Just add the `.breadcrumb` class in an `OL` element, and all the `LI` elements inside it will be rendered as each breadcrumb stage. Here is the code:

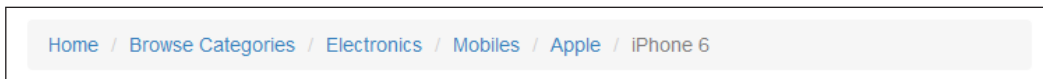
```
<ol class="breadcrumb">
  <li><a href="#">Home</a></li>
```

```

<li><a href="#">Browse Categories</a></li>
<li><a href="#">Electronics</a></li>
<li><a href="#">Mobiles</a></li>
<li><a href="#">Apple</a></li>
<li class="active">iPhone 6</li>
</ol>

```

Here is the generated breadcrumb:



Paginations

There are two patterns of pagination controls offered by Bootstrap, shown as follows:



Here is the code:

```

<!-- Pattern #1 -->
<nav>
  <ul class="pagination">
    <li><a href="#">&laquo;</a></li>
    <li><a href="#">1</a></li>
    <li><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
    <li><a href="#">5</a></li>
    <li><a href="#">6</a></li>
    <li><a href="#">&raquo;</a></li>
  </ul>
</nav>

```

```
<!-- Pattern #2 -->
<nav>
  <ul class="pager">
    <li><a href="#">&larr;&nbsp;&nbsp;&nbsp;Previous</a></li>
    <li><a href="#">Next&nbsp;&nbsp;&nbsp;&rarr;</a></li>
  </ul>
</nav>
```

As you can see, both the patterns are simple UL controls with different Bootstrap classes.

Summary

In this chapter, we saw how Bootstrap can be extended by using several CSS classes in a group of HTML controls to create different packaged components. By that process, we have almost all the commonly used controls that are required to design a flexible web page. All these components are built using CSS classes only. In the next chapter, we will explore the JavaScript components of Bootstrap.


5

The JavaScript Add-ons in Bootstrap

One of the most powerful features in the Bootstrap framework is JavaScript add-ons. In this library, we have almost all the required frontend components, such as modal window, carousel, dropdown, tooltip, alert, navigation tab, accordion, and many more. Along with the previously discussed CSS classes and components, these JavaScript add-ons make Bootstrap a complete suite and framework to develop your HTML pages – both for desktops and mobile devices.

In this chapter, we will go through these add-ons in depth and will enhance our example application quite a bit using these components.

As we have briefly discussed earlier, Bootstrap JavaScript components are built using, and on top of, jQuery (<https://jquery.com/>). Thus, all the syntaxes and behaviors of these add-ons are similar to jQuery.

 We assume that you have a basic working knowledge on jQuery as we will be using jQuery-based scripts in all our further implementation and explanations. Those who are unfamiliar with jQuery can refer to online documentation at <https://api.jquery.com/>.

A basic concepts

Before diving deep into the implementation of JavaScript add-ons, let's clarify a few basic concepts first.

Custom data attributes

As per W3C specification, custom data attributes can be defined as follows:

"Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements."

(http://www.w3.org/html/wg/drafts/html/master/dom.html#embedding-custom-non-visible-data-with-the-data-*-attributes). Thus, in HTML5, you can add custom attributes in any HTML element. The name of the attribute should start with `data-*`; here, `*` stands for the actual name of your attribute. The value of the attribute like any other HTML attribute is string. This feature has been introduced in HTML5 to enable site developers to store custom values corresponding to HTML elements, that is, values that cannot be adjusted in any of the standard attributes.

In Bootstrap JavaScript add-ons, some of such custom data attributes are heavily used:

- `data-target`
- `data-toggle`
- `data-dismiss`
- `data-spy`
- `data-placement`
- `data-content`
- `data-container`
- `data-animation`
- `data-loading-text`
- `data-complete-text`
- `data-parent`
- `data-slide`
- `data-ride`

All these `data-*` attributes are purely custom and are used by the Bootstrap framework. We will later see how these data attributes have been used in various JavaScript components. In a nutshell, without even writing a single JavaScript line in your page, you can develop the functionalities around these add-ons.

JavaScript APIs

Even though we can develop many of the functionalities using the powerful data attributes used in Bootstrap, we still need the standard way of doing stuff – calling methods, handling events, setting properties dynamically, and much more.

JavaScript events

Like any other standard JavaScript component library, Bootstrap add-ons also support typical way of triggering and handling events. Handling those events could be very useful whenever you want to perform some actions whenever any particular event occurs. For example, you want to invoke an AJAX call whenever a modal window is closed.

Packaging add-ons

As you might have noted in the source code we have written in previous chapters, we have included the `bootstrap.min.js` file in our HTML page. This file includes all the JavaScript add-ons Bootstrap offers. Thus, when you include this file in an HTML page, you would be able to use any and all of the add-ons.

Should you want to include and use only a few of the available components, then you can create and use a subset of this file as well. We will see how to do this in later chapters.

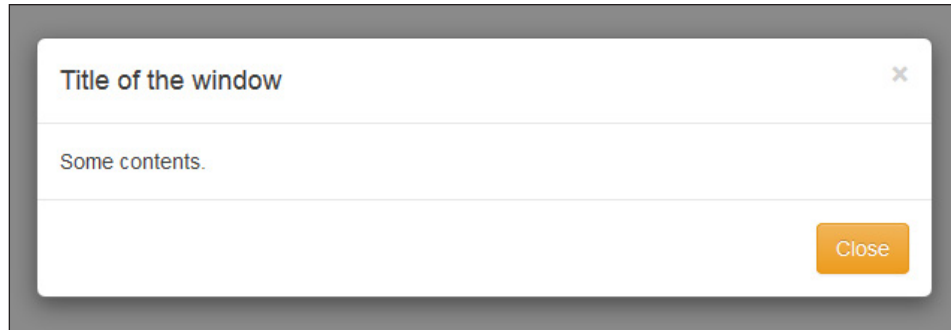
Modal windows

Modal windows is perhaps one of the most widely used custom component. The days of standard JavaScript alert and prompts are gone; you should implement custom modal windows in order to interact with the user.

Bootstrap offers a rich set of functionality and features that would suit almost all requirement variations around modal windows.

Basic modal window

Here is the most basic version of a modal window:



Here is the code to create it:

```
<div class="modal" id="theModal" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-
dismiss="modal"><span>&times;</span></button>
        <h4 class="modal-title" id="theModalLabel">Title of the
window</h4>
      </div>
      <div class="modal-body">
        Some contents.
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-warning" data-
dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
```

Of course, you need a button or link to open this window. Here is the code:

```
<button type="button" class="btn btn-success btn-lg" data-
toggle="modal" data-target="#theModal">Open Modal</button>
```

By putting just these two code blocks, you should be able to create a modal window and a button to open it. As you can see, we haven't written any JavaScript code yet, but still could generate a modal window and a control to open it.

The most important things to note here are the `data-toggle` and `data-target` attributes of the `button` element. By putting `modal` as the `data-toggle` value, we are informing Bootstrap that this button will invoke some modal window, and which modal window it should invoke is determined by the `data-target` attribute. Hence, you can see that the value of the `data-target` attribute is the `modal` – this is actually the ID of the `div` element, which works as the modal window. Now, let's note the button we have in the `modal-footer` section. This button has a `data-dismiss` attribute that says that it will close the current modal window.

This is the common way of working of most of the JavaScript add-ons Bootstrap has distributed. Of course, there are several other data attributes being used in the other components, but the development and usage method is the same.

Finally, we can have the modal dialog box opened by normal JavaScript as well, in case you do not want to use the data attributes for some reason – for example, if you want to open the modal window when any particular event occurs, such as an AJAX callback being invoked. Here is our traditional way of scripting:

```
<button type="button" class="btn btn-primary btn-lg"
onclick="callMe()">Open Modal by Script</button>
```

The script is as follows:

```
function callMe() {
$('#theModal').modal();
}
```

This method is essentially a short form of following other overloaded variants:

- `modal('show')`: This opens up the modal dialog box.
- `modal('hide')`: This closed the dialog box.
- `modal('toggle')`: This toggles the state – closes if the dialog is open and opens if it is closed.
- `modal(options)`: Some of the commonly used options are `keyboard` (`true/false`), `show` (`true/false`), and `backdrop` (`static/true/false`). By putting the `keyboard` value as `false`, you can ensure that the dialog box won't close if *Esc* key is pressed. Putting `static` as `backdrop` value would restrict the dialog box from closing when mouse is clicked.



One important thing to be noted here is if you want to use any of the preceding options (such as `keyboard`, `show`, `backdrop`), you might need to use the JavaScript API, rather than invoking the dialog box by data attributes.

Example – enhancing our application using the modal dialog box

Now, let's enhance our application using the modal dialog box. You might recollect that we had added a link in the top navigation bar for **Sign In** in the previous chapter. Let's have a popup for users to log in to our site. Here is our code of the modal box:

```
<!-- Sign In Dialog Box -->
<div class="modal fade" id="logonBox" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-
dismiss="modal"><span>&times;</span></button>
        <h4 class="modal-title">Sign In</h4>
      </div>
      <div class="modal-body">
        <form>
          <div class="form-group">
            <input type="text" class="form-control" id="userName"
placeholder="User Name">
          </div>
          <div class="form-group">
            <input type="password" class="form-control" id="password"
placeholder="Password">
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-warning" data-
dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary"
id="doLogon">Sign In</button>
      </div>
    </div>
  </div>
</div>
```

This is the code for the **Sign In** link in the navigation bar:

```
<li><a href="#" id="logonLink"><span class="glyphicon glyphicon-log-
in"></span>&nbsp;Sign In</a></li>
```

Finally, this is our JavaScript:

```
<script>
  $('#logonLink').on('click', function(e) {
    $('#logonBox').modal({
      keyboard: false,
      backdrop: 'static'
    });
  });
  $('#doLogon').on('click', function(e) {
    alert('Thank you for Signing In');
    $('#logonBox').modal('hide');
  });
</script>
```



You can have the full-page source code from the downloaded source code bundle.

This is how our **Sign In** dialog box appears:

In the script, of course, you would add your code for actually signing the user in, which may be a submit call or an AJAX call to the server.

In addition, Bootstrap modal windows also support a few events to enable you to fire any action in case any of these events occur.

For example, have a look at the following code:

```
$('#logonBox').on('hidden.bs.modal', function (e) {  
    alert('You have cancelled the Sign In operation');  
})
```

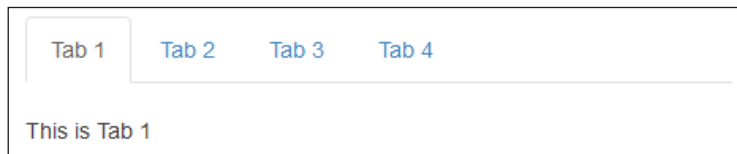
This will show this browser alert whenever the modal window has been fully closed. The other exposed events are — `show.bs.modal`, `shown.bs.modal`, `hide.bs.modal`, `hidden.bs.modal`, and `loaded.bs.modal`.

Tabs

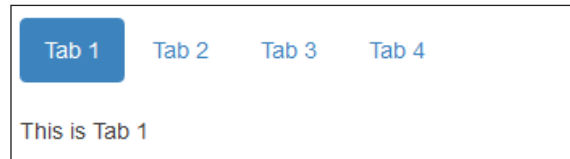
Tabs are another popular and pretty standard way of navigation in today's websites. In Bootstrap, it is very easy to design a tab-based navigation model; all you have to do is to create a UL element with `.nav-tabs` CSS class. Here is a sample code:

```
<div>  
  <ul class="nav nav-tabs">  
    <li class="active"><a href="#tab1" data-toggle="tab">Tab 1</a></li>  
    <li ><a href="#tab2" data-toggle="tab">Tab 2</a></li>  
    <li ><a href="#tab3" data-toggle="tab">Tab 3</a></li>  
    <li ><a href="#tab4" data-toggle="tab">Tab 4</a></li>  
  </ul>  
  
  <div class="tab-content">  
    <div class="tab-pane active" id="tab1"><br/>This is Tab 1</div>  
    <div class="tab-pane" id="tab2"><br/>This is Tab 2</div>  
    <div class="tab-pane" id="tab3"><br/>This is Tab 3</div>  
    <div class="tab-pane" id="tab4"><br/>This is Tab 4</div>  
  </div>  
</div>
```

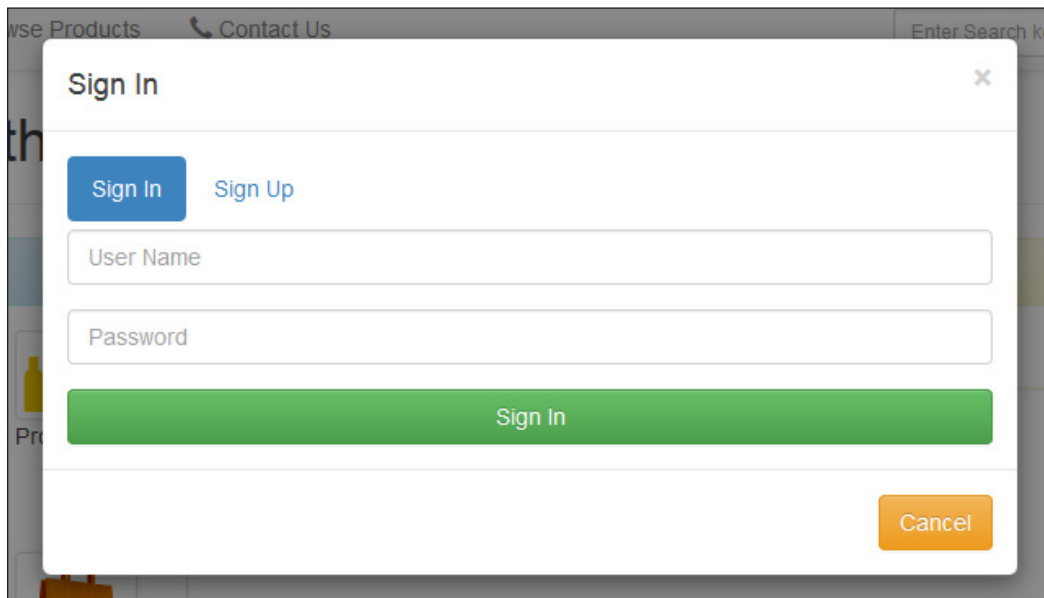
Here is the simple tab display:



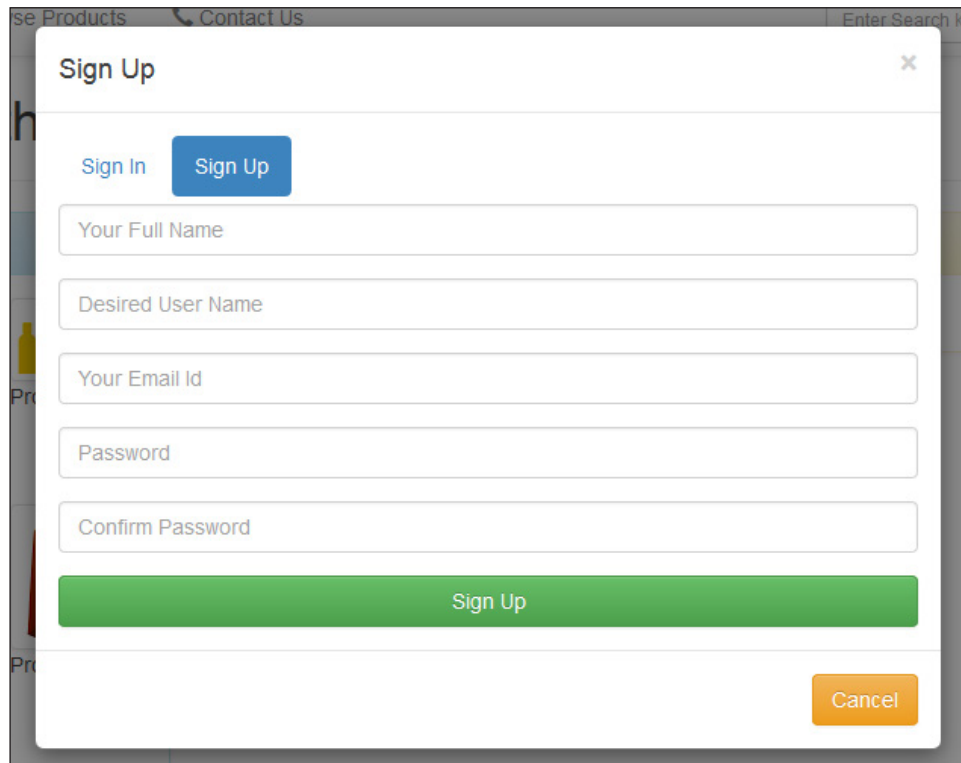
There is another way of displaying tabs, which is termed as **pills**. Just replace the class `.nav-tabs` with `.nav-pills`. Here is the rendition:



Now, let's use this tabbed display in our application. We want to have a **Sign Up** form as well (until now, we have only **Sign In**). We will add this **Sign Up** form in the same modal dialog box where the login form is displayed. Here are the screenshots of the same:



The following is a screenshot of the **Sign Up** tab:



Now, let's look at the code for the previous screenshots:

```
<div class="modal fade" id="logonBox" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-
dismiss="modal"><span>&times;</span></button>
        <h4 class="modal-title">Sign In</h4>
      </div>
      <div class="modal-body">
        <div>
          <ul class="nav nav-pills">
            <li class="active"><a href="#signin" data-toggle="tab"
data-caption="Sign In">Sign In</a></li>
            <li ><a href="#signup" data-toggle="tab" data-
caption="Sign Up">Sign Up</a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
</ul>
<div class="tab-content">
  <div class="tab-pane active" id="signin">
    <form style="padding-top: 5px">
      <div class="form-group">
        <input type="text" class="form-control"
id="userName" placeholder="User Name">
      </div>
      <div class="form-group">
        <input type="password" class="form-control"
id="password" placeholder="Password">
      </div>
      <button type="button" class="btn btn-success btn-
block" id="doLogon">Sign In</button>
    </form>
  </div>
  <div class="tab-pane" id="signup">
    <form style="padding-top: 5px">
      <div class="form-group">
        <input type="text" class="form-control"
id="yourName" placeholder="Your Full Name">
      </div>
      <div class="form-group">
        <input type="text" class="form-control"
id="yourUserName" placeholder="Desired User Name">
      </div>
      <div class="form-group">
        <input type="text" class="form-control"
id="yourEmail" placeholder="Your Email Id">
      </div>
      <div class="form-group">
        <input type="password" class="form-control"
id="yourPwd" placeholder="Password">
      </div>
      <div class="form-group">
        <input type="password" class="form-control"
id="confirmPwd" placeholder="Confirm Password">
      </div>
      <button type="button" class="btn btn-success btn-
block" id="doLogon">Sign Up</button>
    </form>
  </div>
</div>
</div>
</div>
```



```
    </div>
    <div class="modal-footer">
      <button type="button" class="btn btn-warning" data-
dismiss="modal">Cancel</button>
    </div>
  </div>
</div>
</div>
```

Here is the JavaScript code:

```
$( 'a[data-toggle="tab"]' ).on('shown.bs.tab', function (e) {
  var data = $(e.target).data('caption');
  var modal = $('#logonBox');
  modal.find('.modal-title').text(data);
});
```

The HTML markup is quite self-explanatory; however, the small JS code is interesting. What this piece of code does is, whenever the user navigates from one tab to another (from **Sign In** to **Sign Up**, and vice versa), the caption (title) of the dialog box gets changed correspondingly.

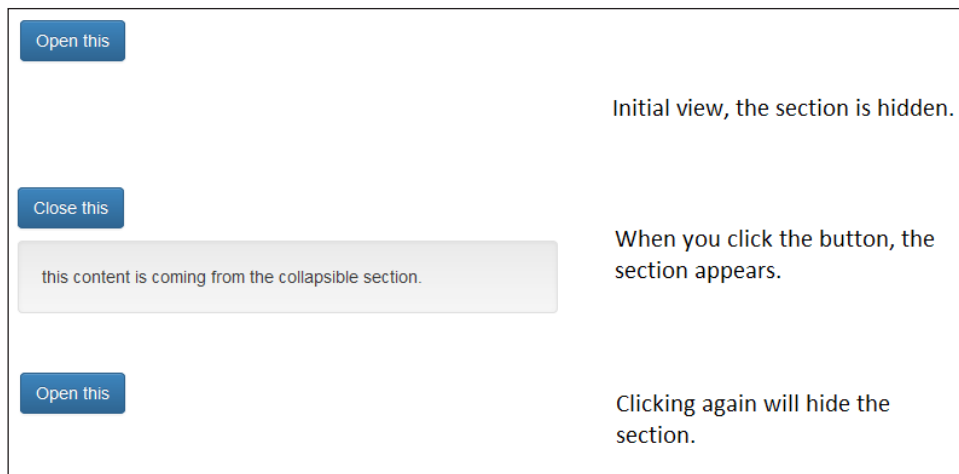
The few points worth noting here are:

- The function will get invoked in the event of a new tab being selected (the `shown.bs.tab` event). This function is essentially the event handler.
- The `e.target` object represents the newly opened tab.
- In the HTML markup, you can see that we have added a custom data attribute `data-caption`; this attribute is used to set the title of the modal window.

Similar to this `shown.bs.tab` event, other exposed events are — `show.bs.tab`, `hide.bs.tab`, and `hidden.bs.tab`.

Collapse and accordions

Bootstrap collapse component is used to simply show and hide different sections and subsections of your web page. This is useful when you don't want the sections to be visible initially, but will toggle when clicked on a button or hyperlink. Here is a simple example:



Here is the HTML code for the preceding screenshot:

```
<button class="btn btn-primary" type="button" data-toggle="collapse"
data-target="#theSection" id="theButton">Open this</button>
<div class="collapse" id="theSection">
  <div class="well">
    this content is coming from the collapsible section.
  </div>
</div>
```

The JavaScript code for the preceding screenshot is:

```
// When the collapsible element is shown, change the button's text to
'close this'
$('#theSection').on('show.bs.collapse', function () {
  $('#theButton').text('Close this');
});
// When the collapsible element is collapsed, change the button's text
to 'Open this'
$('#theSection').on('hide.bs.collapse', function () {
  $('#theButton').text('Open this');
});
```

Again, using the `data-target` and `data-toggle` attributes, the collapse behavior has been achieved. Also, we have used the `show.bs.collapse` and `hide.bs.collapse` event handlers to change the text of the button correspondingly.

Example – showing the product categories of our store

You might remember that we had added a link in our navigation bar in the last chapter. Let's create a page for it and use this collapse component as an accordion in that page to show the product categories of our store. Here is the markup for the categories using accordion pattern:

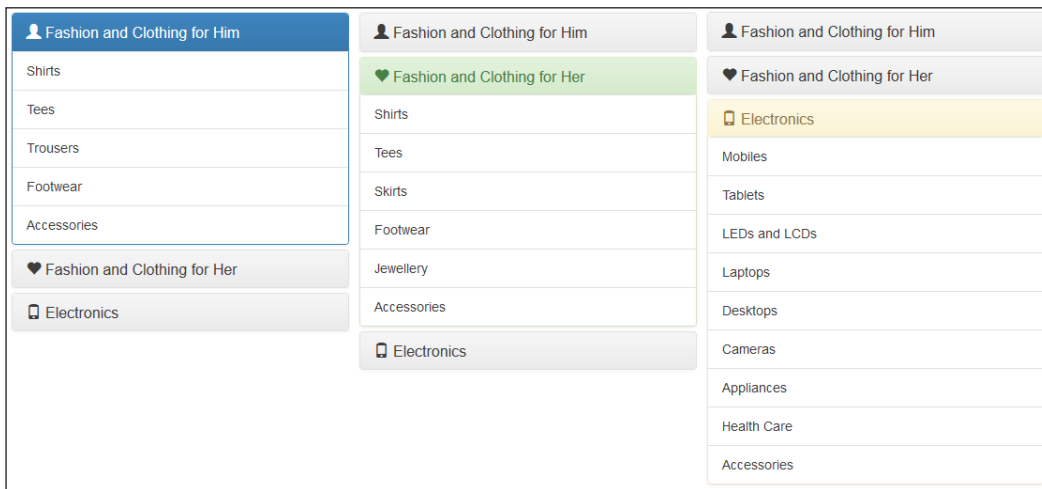
```
<div class="panel-group" id="catList">
  <div class="panel panel-primary" id="cat1Head">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-parent="#catList"
href="#cat1"><span class="glyphicon glyphicon-user"></
span>&nbsp;Fashion and Clothing for Him</a>
      </h4>
    </div>
    <div id="cat1" class="panel-collapse collapse in">
      <ul class="list-group">
        <li class="list-group-item">Shirts</li>
        <li class="list-group-item">Tees</li>
        <li class="list-group-item">Trousers</li>
        <li class="list-group-item">Footwear</li>
        <li class="list-group-item">Accessories</li>
      </ul>
    </div>
  </div>

  <div class="panel panel-default" id="cat2Head">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-parent="#catList"
href="#cat2"><span class="glyphicon glyphicon-heart"></
span>&nbsp;Fashion and Clothing for Her</a>
      </h4>
    </div>
    <div id="cat2" class="panel-collapse collapse">
      <ul class="list-group">
        <li class="list-group-item">Shirts</li>
        <li class="list-group-item">Tees</li>
        <li class="list-group-item">Skirts</li>
        <li class="list-group-item">Footwear</li>
      </ul>
    </div>
  </div>
</div>
```

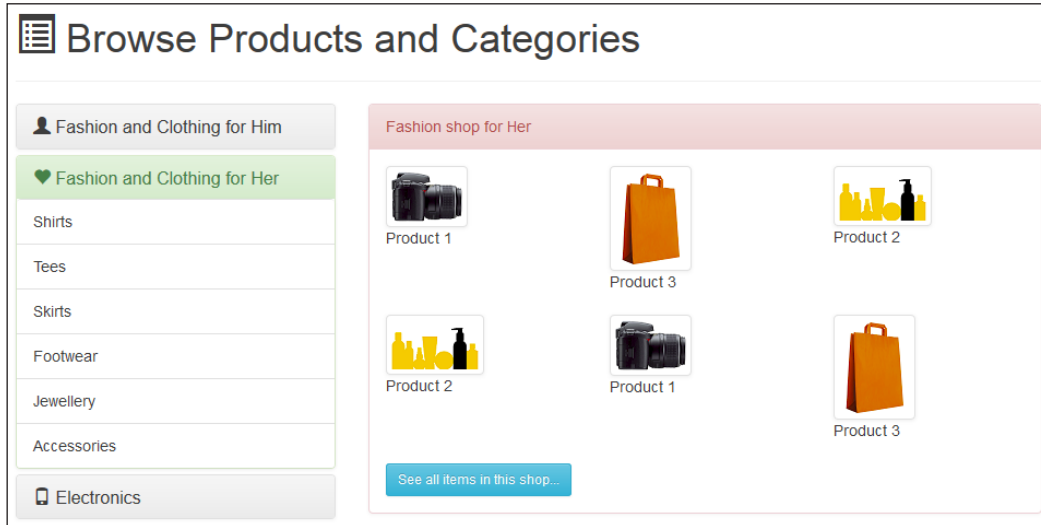



```
$('#cat2Head').removeClass('panel-default').addClass('panel-success');
$('#cat3Head').removeClass('panel-warning').addClass('panel-default');
});
$('#cat3').on('show.bs.collapse', function () {
    $('#cat1Head').removeClass('panel-primary').addClass('panel-default');
    $('#cat2Head').removeClass('panel-success').addClass('panel-default');
    $('#cat3Head').removeClass('panel-default').addClass('panel-warning');
});
```

This is how our category browser looks like. We have created three separate section for three main categories. The following is the screenshot of all three categories merged into one:



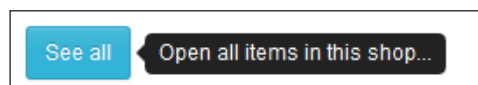
Finally, this is the way our browse page looks:



[ Full source code for this page is available for download.]

Tooltips and popovers

You would of course need tooltips for every user-friendly site you develop. Tooltips are more advanced versions of the `title` attributes you normally use in case of images. Here is a simple tooltip:



The HTML markup for this is shown as follows:

```
<a class="btn btn-info btn-sm" href="#" role="button" data-
toggle="tooltip" data-placement="right" title="Open all items in this
shop...">See all</a>
```

Of course, the `data-toggle`, `data-placement`, and `title` attributes options are only relevant in this context. You have four options (four directions essentially) for `data-placement` – top, right, bottom, and left. The tooltip will be displayed as per this attribute value.

However, you need to initialize the tooltip functionality in the page where you want to use it. This is because, unlike any other component, Bootstrap does not initialize this component by default. Here is the code for this – this you can include in the HTML page JS section itself:

```
$(document).ready(function () {
  $('[data-toggle="tooltip"]').tooltip();
  $('[data-toggle="popover"]').popover();
});
```

This script will enable all the tooltips and popovers you have in the current page. The next advanced option of tooltip is popover. In tooltips, you cannot have complex and lengthy data. Also, you can not show HTML content in a tooltip. Popovers are more flexible and advanced. Suppose, in the list of the products, that we want to show the brief details on mouse click. The following example contains two types of popover creation – one by data attribute and another by JavaScript. Here is our HTML code:

```

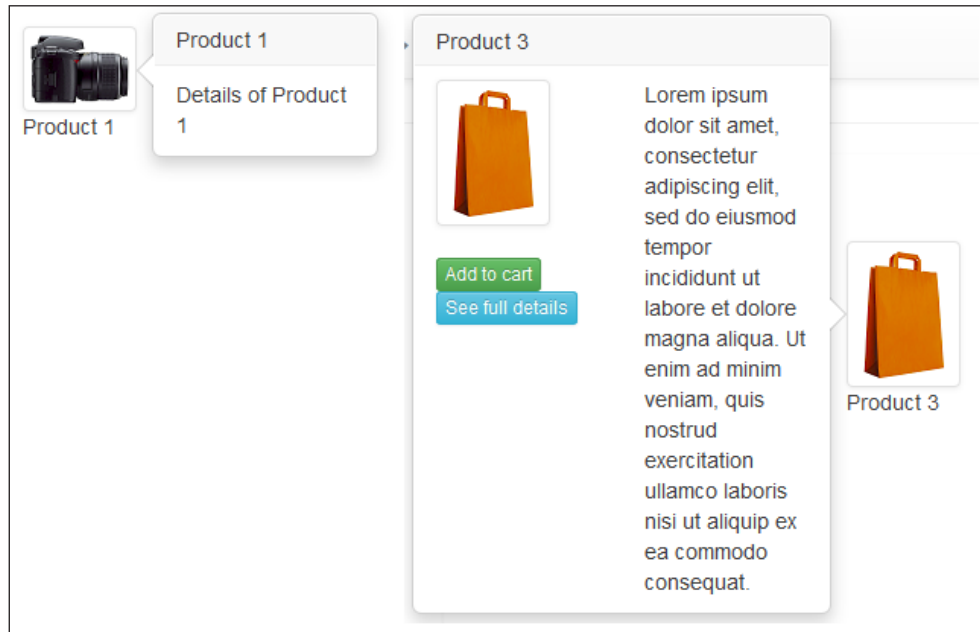
```

For product 2, HTML markup and JavaScript are as follows:

```


$('#prod3').popover({
  html:true,
  placement:'left',
  trigger:'click',
  container:'body',
  content:function() {
    return '<div class="row"><div class="col-xs-12 col-sm-12 col-
md-6 col-lg-6"><br/>&nbsp;<br/><button
type="button" class="btn btn-xs btn-success">Add to cart</
button><br/><button type="button" class="btn btn-xs btn-info">See full
details</button></div><div class="col-xs-12 col-sm-12 col-md-6 col-
lg-6">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.</div></div>';
  }
});
```

This is the way the popovers render (two popover screenshots merged into one):

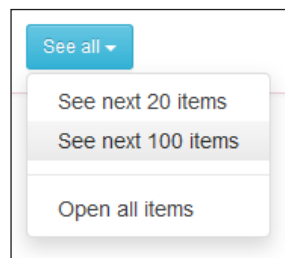


You can see that in case of the JavaScript initialization of the popover, you can pass a variety of other parameters that gives you more control and flexibility. As for the content value, you can always call another function, which, in turn, invokes an AJAX call and fetches data from your database.

Dropdown

The Bootstrap drop-down components can be used in a number of cases, for example, in a menu, button, link, navigation bar, and so on.

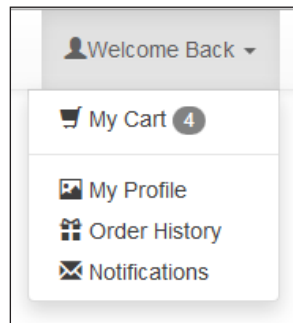
Suppose that we need to have a few options in dropdown in the **See all** link we had created in the product list panel. Here is what we want:



This is the markup to implement this:

```
<div class="dropdown">
  <a class="btn btn-info btn-sm" href="#" data-toggle="dropdown"
  title="Open items in this shop...">See all&nbsp;<span class="caret"></
  span></a>
  <ul class="dropdown-menu">
    <li><a href="#">See next 20 items</a></li>
    <li><a href="#">See next 100 items</a></li>
    <li class="divider"></li>
    <li><a href="#">Open all items</a></li>
  </ul>
</div>
```

In a similar way, we can add a dropdown in the navigation menu also. Let's say that we need to have a drop-down menu in the navigation bar for user's account, profile, shopping cart, and so on. This is the expected result:



To achieve this, we change the line:

```
<li><a href="#"><span class="glyphicon glyphicon-shopping-cart"></
  span>&nbsp;<span>My Cart <span class="badge">4</span></span></a></li>
```

to the following:

```
<li class="dropdown">
  <a data-toggle="dropdown" href="#"><span class="glyphicon glyphicon-
  user"></span>Welcome Back&nbsp;<span class="caret"></span></a>
  <ul class="dropdown-menu">
    <li><a href="#"><span class="glyphicon glyphicon-shopping-cart"></
    span>&nbsp;<span>My Cart&nbsp;<span class="badge">4</span></span></a></li>
    <li class="divider"></li>
    <li><a href="#"><span class="glyphicon glyphicon-picture"></
    span>&nbsp;<span>My Profile</span></a></li>
```

```

    <li><a href="#"><span class="glyphicon glyphicon-gift"></span>&nbsp;<a href="#">Order History</a></li>
    <li><a href="#"><span class="glyphicon glyphicon-envelope"></span>&nbsp;<a href="#">Notifications</a></li>
  </ul>
</li>

```

Thus, you can see that Bootstrap dropdowns are very generic in nature, and they can be used in a number of places and for number of purposes.

Alerts

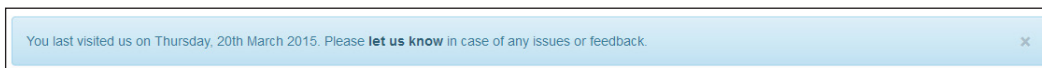
We had visited this component briefly in the previous chapter. To extend on that functionality, let's add a simple close button to dismiss the alert box:

```

<div class="alert alert-info">
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
  You last visited us on Thursday, 20th March 2015. Please <a href="/contact.html" class="alert-link">let us know</a> in case of any issues or feedback.
</div>

```

So, we have got close button here:



The alert component does offer JavaScript interface as well, because most of the time you would want the alert messages to be displayed dynamically as per the user's operation.

If you remember, in the popover components, we had added a popover on a product click in order to display the product details. There we had an **Add to cart** button as well. Let's say, whenever the user clicks on this button an alert box should appear confirming to the user that the product has been added to the shopping cart. We change the popover a little bit for this:

```

$('#prod3').popover({
  html:true,
  placement:'left',
  trigger:'click',

```

```
    container: 'body',
    content: function() {
        return '<div class="row"><div class="col-xs-12 col-sm-12 col-
md-6 col-lg-6"><br/>&nbsp;<br/><button
type="button" class="btn btn-xs btn-success" onclick="addToCart()">Add
to cart</button><br/><button type="button" class="btn btn-xs btn-
info">See full details</button></div><div class="col-xs-12 col-sm-12
col-md-6 col-lg-6">Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat.</div></div>';
    }
};
```

The only change we have done here is that we have added an `onclick` event handler for the **Add to cart** button.

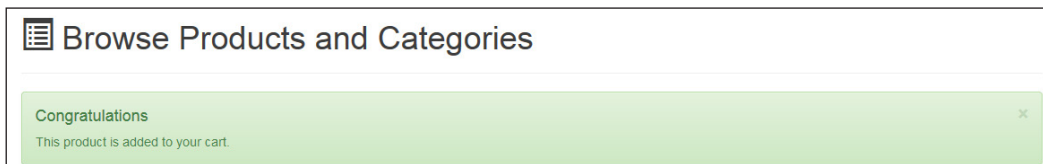
We have also added a blank placeholder in the page (just after the page header) for the alert messages to appear:

```
<div class="page-header"><h1><span class="glyphicon glyphicon-list-
alt"></span>&nbsp;<br/>Browse Products and Categories</h1></div>
<div id="msg"></div>
```

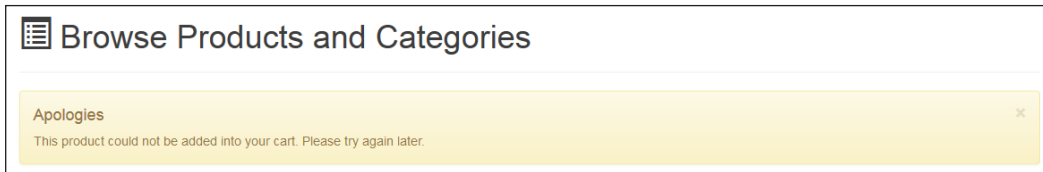
Finally, the JavaScript function for the `onclick` event handler:

```
function addToCart() {
    var txt = '<div id="alertdiv" class="alert alert-success alert-
dismissable"><button class="close" data-dismiss="alert">&times;</
button><h4>Congratulations</h4>This product is added to your cart.</
div>';
    $('#msg').html(txt);
    $('#prod3').popover('hide');
};
```

Thus, whenever the **Add to cart** button will be pressed, this alert message will appear:



Of course, in this function, you will add your AJAX calls in order to actually add the product in the shopping cart. Depending on this execution result, the user will be notified. Here is another sample in case the product could not be added in the cart for some reason:



The updated code is of course in the `addToCart` JavaScript function:

```
var txt = '<div id="alertdiv" class="alert alert-warning alert-dismissable"><button class="close" data-dismiss="alert">&times;</button><h4>Apologies</h4>This product could not be added into your cart. Please try again later.</div>';
```

Thus you can see, the alert component is quite flexible and generic enough in order to be used in a number of circumstances.

Carousels

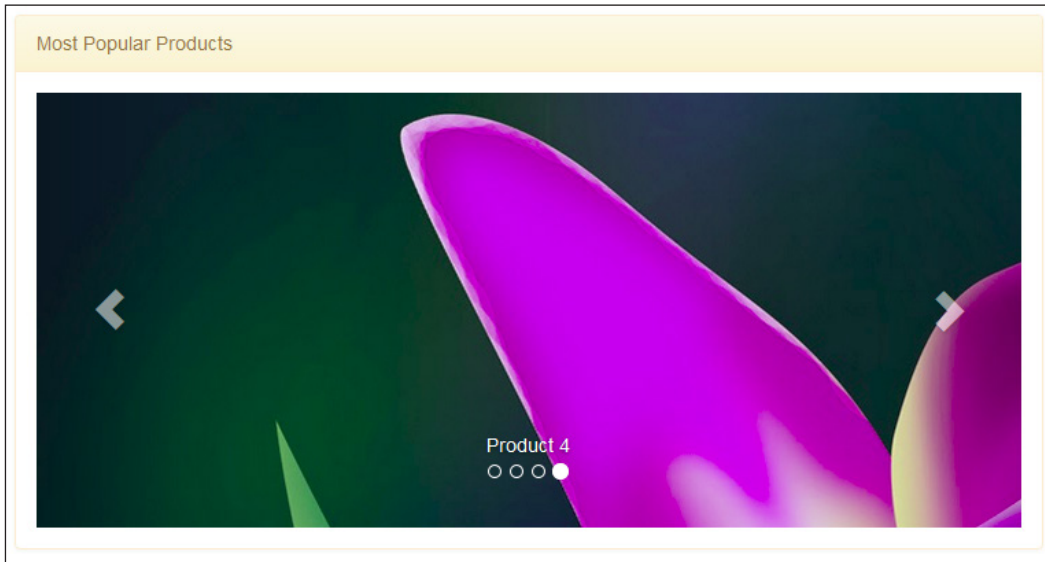
Carousels are popular slideshow components and are also available for use in Bootstrap. In our home page, let's display the most popular products as a carousel. Here is the HTML markup:

```
<div id="bestSellers" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
    <li data-target="#bestSellers" data-slide-to="0" class="active"></li>
  </ol>
  <ol class="carousel-indicators">
    <li data-target="#bestSellers" data-slide-to="1"></li>
    <li data-target="#bestSellers" data-slide-to="2"></li>
    <li data-target="#bestSellers" data-slide-to="3"></li>
  </ol>
  <div class="carousel-inner">
    <div class="item active">
      
      <div class="carousel-caption">Product 1</div>
    </div>
    <div class="item">
      
      <div class="carousel-caption">Product 2</div>
    </div>
  </div>
</div>
```

```
<div class="item">
  
  <div class="carousel-caption">Product 3</div>
</div>
<div class="item">
  
  <div class="carousel-caption">Product 4</div>
</div>
</div>

<a class="left carousel-control" href="#bestSellers" data-
slide="prev">
  <span class="glyphicon glyphicon-chevron-left"></span>
</a>
<a class="right carousel-control" href="#bestSellers" data-
slide="next">
  <span class="glyphicon glyphicon-chevron-right"></span>
</a>
</div>
```

Let's place the preceding piece of code in the **Most Popular Products** section in our home page. The carousel renders like the following screenshot:



Like other components, the carousel component is also available via JavaScript APIs. For example, in the following code if we had to invoke carousel component via scripts:

```
$('#bestSellers').carousel({
  interval: 3000,
```

```

    pause: 'hover'
  });

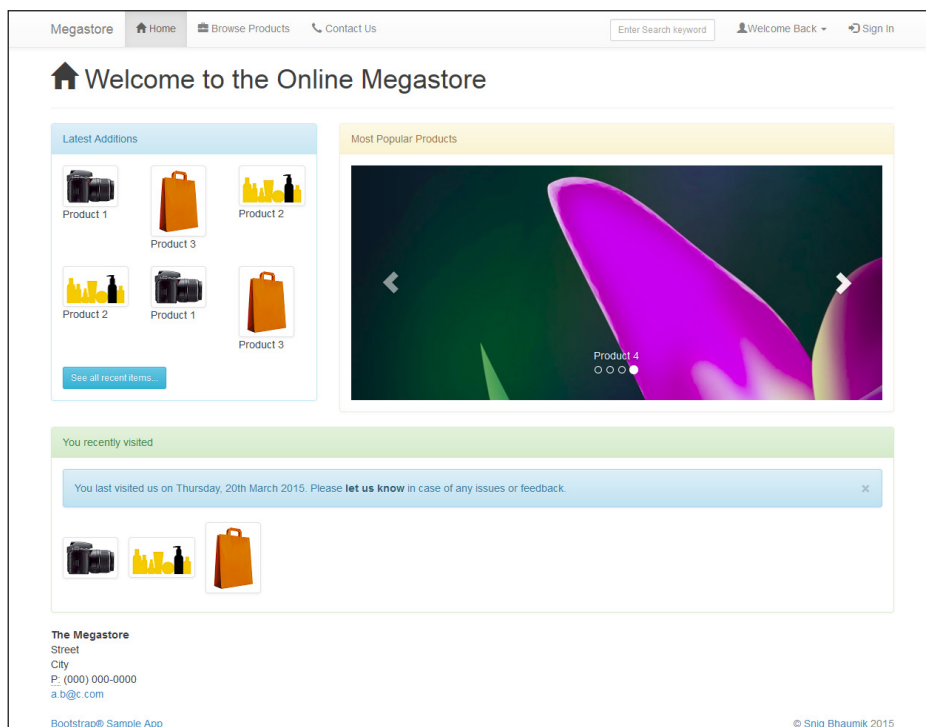
```

The `interval` value defines the amount of time in milliseconds for each slide to stay and then move ahead. Mentioning the `pause` as `hover` instructs Bootstrap to stay on the current slide until the mouse is moved out of the container. The important parts of the carousel component are as follows:

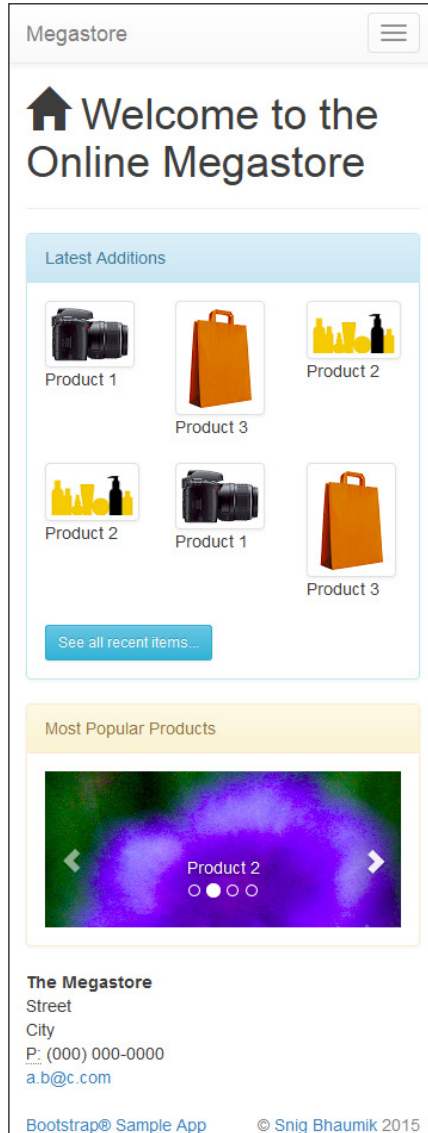
- `carousel-indicator`: These controls are used to open a particular slide in the set of slides in the carousel
- `carousel-inner`: These sections are actual contents of the each of the slides in the carousel
- `carousel-control`: These are essentially left and right arrows to navigate to previous or next slides in the carousel

The final preview

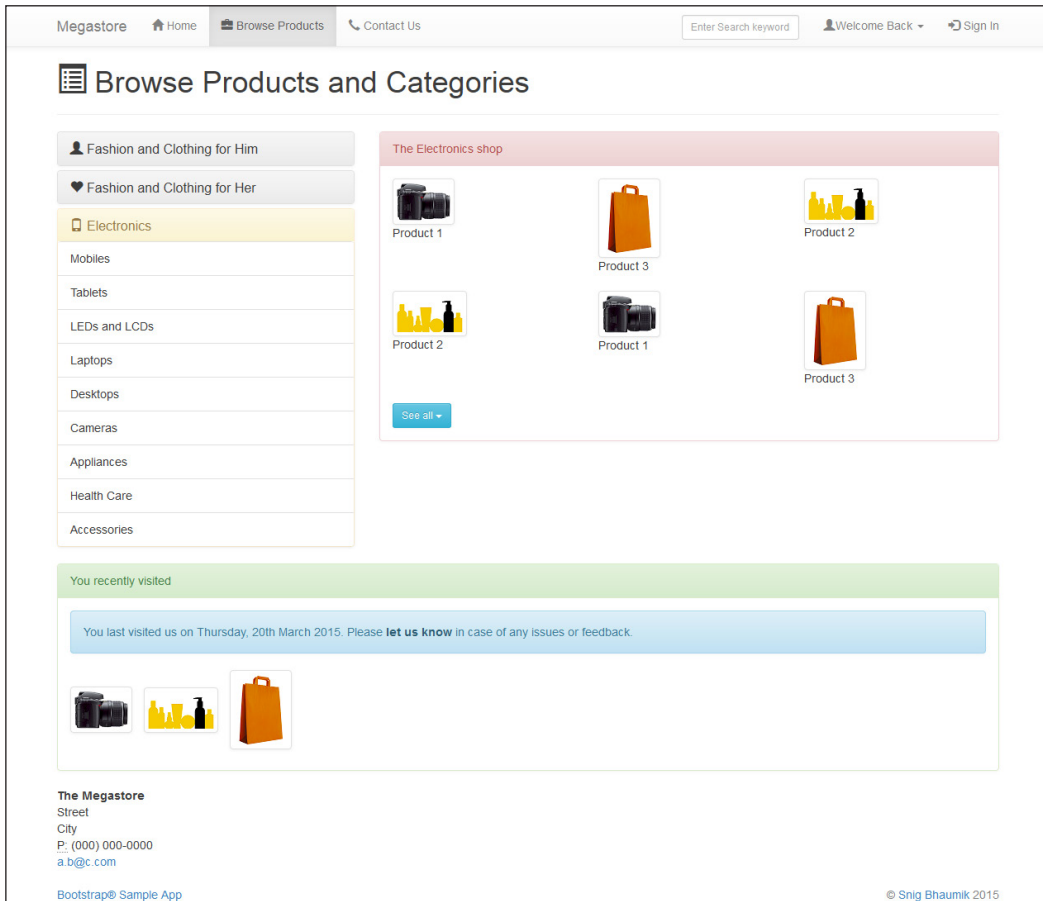
The complete code for the two pages can be found in the downloadable code available with this book. The screenshots of the final version of the two main pages – the **Home** page and **Browse Products** page are as shown:



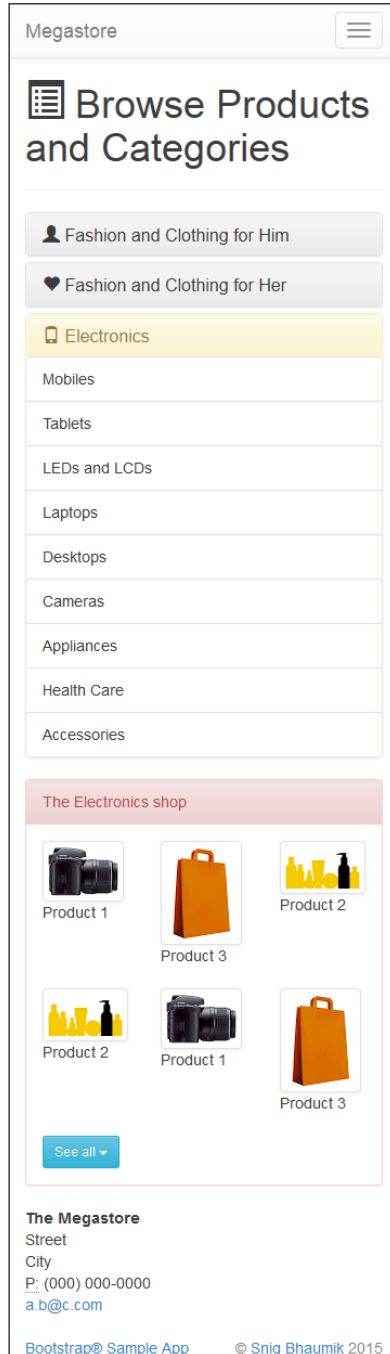
In a small mobile device, it will look like the following screenshot:



Here is the desktop view of the **Browse Products** page:



Finally, the mobile version of the page is as shown:



Summary

In this chapter, we completed exploring almost all of the features and functionalities of Bootstrap. We saw the CSS classes, the available components, and finally the JavaScript add-ons. To reiterate the same, some important CSS classes are the grid system, responsive classes, data entry forms tables, and images. Similarly, some important components are panels and wells, toolbars and button groups, breadcrumbs and navigation controls, and Glyphicons. Finally, in this chapter, we explored the JS add-ons: modal windows, tooltips and popovers, carousels, accordions, and alerts

In the next chapter, we will start creating our development environment of Bootstrap and will customize default Bootstrap options.

6

Compiling and Building Bootstrap

Until now, we have discussed the basics of Bootstrap, understood the architecture, and explored all the features and functionalities of Bootstrap.


We have also created a simple web application using various Bootstrap components. However, we have used the Bootstrap CSS and JS files as it is and haven't changed or updated anything in any of the files. Even though updating of JavaScript files is very rare and you won't find any normal reason to change anything in the JS files, overriding the CSS files are very common, and in most of the cases, very much essential when you are building a web application using Bootstrap as frontend framework for final production use. You of course, would need to change the colors, fonts, and much more to meet your client's brand guide.

Even though it seems to be very easy to update the `bootstrap.css` file and enter your own colors, font names, and so on, but you can imagine that it is not at all the suggested approach.

As we have briefly discussed earlier, Bootstrap uses **LESS** preprocessor quite extensively and the right approach for overriding the CSS class values of Bootstrap is to change those variables in LESS files. Compiling and building Bootstrap means actually compiling those LESS files, generating the distributable CSS files, and also generating the final JavaScript files.

In this chapter, we will look into how we can establish the compilation and build an environment for Bootstrap. In the next chapter, we will explore how to update the LESS variables in order to override the default look and feel of Bootstrap using this development environment. Here are the major points that we will cover here:

- The list of tools required to build Bootstrap, how to install and configure those tools
- How to install Bootstrap source code
- How to compile and build Bootstrap using these tools

 We will be using various tools for compiling and building Bootstrap here. However, we will not discuss those tools in depth, but will mainly focus on how to use those tools to build Bootstrap.

Required tools

You will need the following tools:

- Node.js
- Grunt

Optionally, you may also use Bower; however, in our case, we will discuss about the steps involving and using node.js.

Node.js

Node.js is an open source runtime environment for server-side and networking applications. In simple terms, node.js is a server-side JavaScript runtime environment.

The whole framework is based on the JavaScript language – it is written in JavaScript, it is executed within a JavaScript runtime environment; the applications developed on node.js are also based on JavaScript. It uses Google V8 JavaScript engine to execute the end application code.

Along with node.js, an important component that comes in the bundle is the **Node Package Manager (npm)**. This component finds, extracts, and installs all the dependencies of a JavaScript application. In other words, it manages the whole dependency installation processes so that you need not have to find out which other elements we need to run and execute a particular software package.

Thus, we will use the npm engine in order to get the source code and all other related software of Bootstrap. We will first install node.js, and the npm engine will come along with that.



The npm has a few advantages over Bower. For example, npm manages to avoid dependency conflicts by using nested dependencies. While, on the other hand, Bower is optimized only for frontends.

Installing node.js

If you are using 64-bit Windows, you can get the latest node.js installer from <http://nodejs.org/dist/v0.12.1/x64/node-v0.12.1-x64.msi>.

However, for other distributions, visit <https://nodejs.org/download/>. There you will find various Windows, Linux, and Mac installer packages. After downloading this, install node.js in your machine. When installation is finished, make sure that you have the node.js installation folder included in your windows executable path. This is because we will use the npm **Command-line Interface (CLI)** later on. Let's test our installation:

1. Open windows command prompt.
2. Write `npm`, and press *Enter*. If you get the following screen, everything is in order:

```

C:\WINDOWS\system32\cmd.exe

D:\>npm
Usage: npm <command>

where <command> is one of:
  add-user, adduser, apihelp, author, bin, bugs, c, cache,
  completion, config, ddp, dedupe, deprecate, docs, edit,
  explore, faq, find, find-dupes, get, help, help-search,
  home, i, info, init, install, isntall, issues, la, link,
  list, ll, ln, login, ls, outdated, owner, pack, prefix,
  prune, publish, r, rb, rebuild, remove, repo, restart, rm,
  root, run-script, s, se, search, set, show, shrinkwrap,
  star, stars, start, stop, submodule, t, tag, test, tst, un,
  uninstall, unlink, unpublish, unstar, up, update, v,
  version, view, whoami

npm <cmd> -h      quick help on <cmd>
npm -l           display full usage info
npm faq          commonly asked questions
npm help <term>  search for help on <term>
npm help npm     involved overview

Specify configs in the ini-formatted file:
  C:\Users\Sng\.\npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@1.4.28 C:\Work\node.js\node_modules\npm
D:\>

```

Grunt

As their website suggests, Grunt is the JavaScript task runner. Almost like Apache Ant, where you put a long list of tasks to do, and the engine executes each task one by one. In case of JavaScript-based frameworks and applications, we use Grunt.


Like in case of Ant, we have a `build.xml` file; in case of Grunt, we have a `Gruntfile.js` file—where all the tasks that are to be executed are listed, which means this is the file where the Grunt project is configured.

In our current scope, we do not need to understand the full architecture of Grunt and Grunt file, we will just use this engine to compile and build Bootstrap.

However, first we have to install Grunt. For our purpose of compiling and building Bootstrap, we only need the `grunt-cli` module, not the full Grunt task runner. This is the CLI module of Grunt.

Installing Grunt-cli

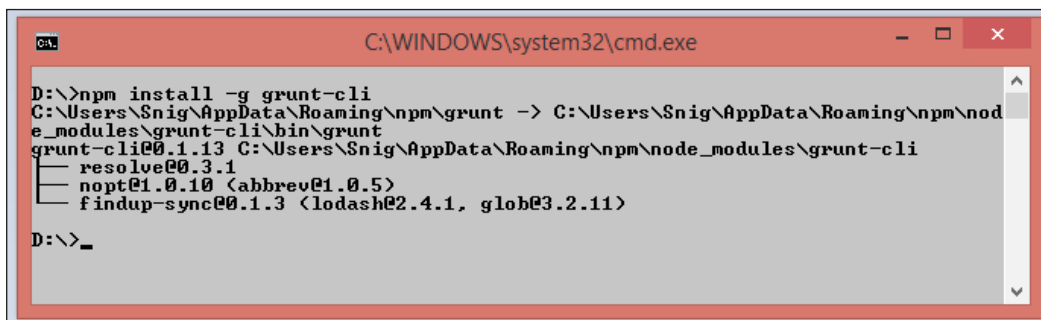
The good part is that nothing needs to be manually downloaded for this. We have downloaded and installed npm in an earlier section, and now the rest of the installations will be handled and managed by npm only. Open the windows command prompt with administrative privileges.

[ However, having the admin privilege is optional; if you can't manage that, go ahead with normal command prompt.]

Type this command:

```
npm install -g grunt-cli
```


You should see an output similar to the following screenshot:



```
C:\WINDOWS\system32\cmd.exe
D:\>npm install -g grunt-cli
C:\Users\Snig\AppData\Roaming\npm\grunt -> C:\Users\Snig\AppData\Roaming\npm\node_modules\grunt-cli\bin\grunt
grunt-cli@0.1.13 C:\Users\Snig\AppData\Roaming\npm\node_modules\grunt-cli
├─ resolve@0.3.1
├─ nopt@1.0.10 (abbrev@1.0.5)
├─ findup-sync@0.1.3 (lodash@2.4.1, glob@3.2.11)
D:\>_
```

As you can see, the `grunt-cli` module has been installed in my `C:\Users\Snig\AppData\Roaming\npm\node_modules` folder. The `grunt.cmd` command file is installed in `C:\Users\Snig\AppData\Roaming\npm` folder.

Check this output in your screen and mark the folder where `grunt-cli` has been installed. Check your windows `path` variable, if this folder has been added into the path or not. If not, you have to manually add this folder in your path.

 Opening the windows command prompt with admin privileges (mentioned previously) is required only to update this system path.


Another way of course is to install the `grunt-cli` in the local folder where you are planning to install the Bootstrap source code, compile, and build them, which means your Bootstrap development folder.

Let's say, this folder in our case is `D:\Bootstrap`. This will be our development folder where we will download and install Bootstrap and will compile it and build it. Later on, in the next chapter, we will customize Bootstrap in this folder only.

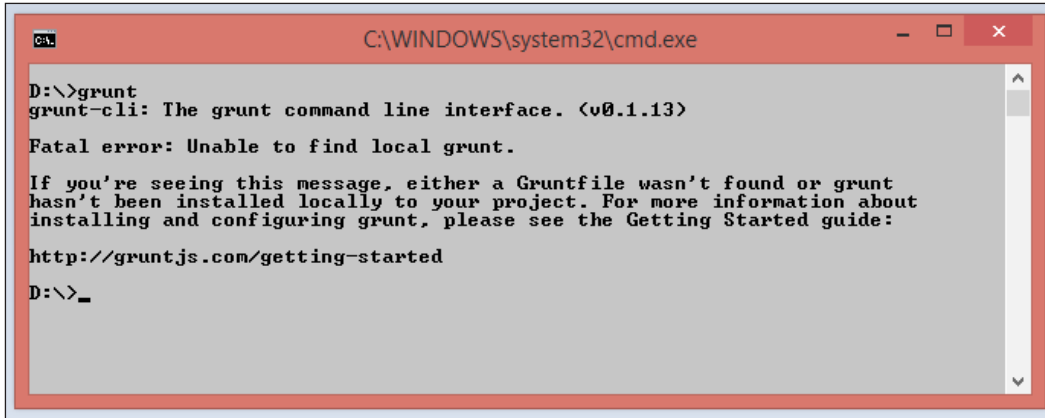
In order to install `grunt-cli` in this local folder, we run the following command:

```
npm install --prefix D:\Bootstrap grunt-cli
```

However, we will go ahead with installing Grunt as in the first method – with the `-g` option. This option ensures that the package is installed in global mode.

 For more options and information on installing packages using npm, please see <https://docs.npmjs.com/cli/install>.

Once everything is done, you can test the installation by executing command Grunt in the windows command prompt, as shown in the following screenshot:



```
CA C:\WINDOWS\system32\cmd.exe
D:\>grunt
grunt-cli: The grunt command line interface. (v0.1.13)
Fatal error: Unable to find local grunt.

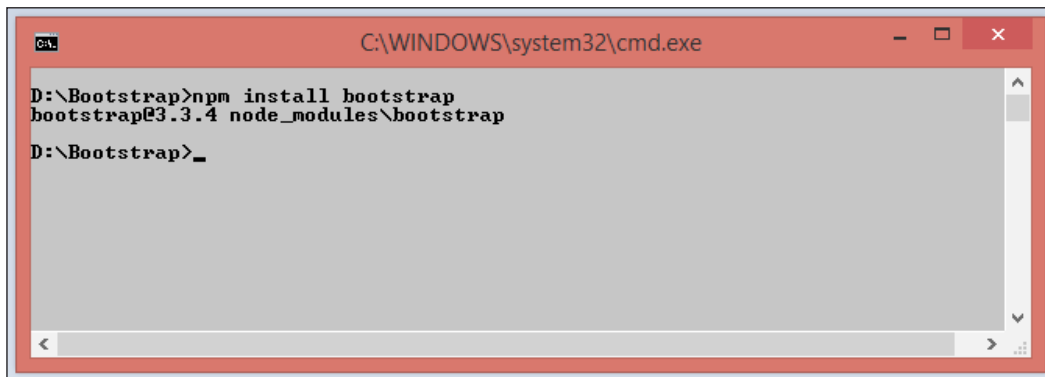
If you're seeing this message, either a Gruntfile wasn't found or grunt
hasn't been installed locally to your project. For more information about
installing and configuring grunt, please see the Getting Started guide:
http://gruntjs.com/getting-started
D:\>_
```

We are facing this error since we have executed grunt in a folder where no grunt project is configured, which means there is no `Gruntfile.js` file in this folder.

Installing Bootstrap

We will now install Bootstrap, get the Bootstrap source code, and prepare to create our development environment for Bootstrap. Follow these steps:

1. Open windows command prompt and navigate to the `D:\Bootstrap` folder.
2. Run the following command:
`npm install bootstrap`
3. You will get the following output:



```
CA C:\WINDOWS\system32\cmd.exe
D:\Bootstrap>npm install bootstrap
bootstrap@3.3.4 node_modules\bootstrap
D:\Bootstrap>_
```

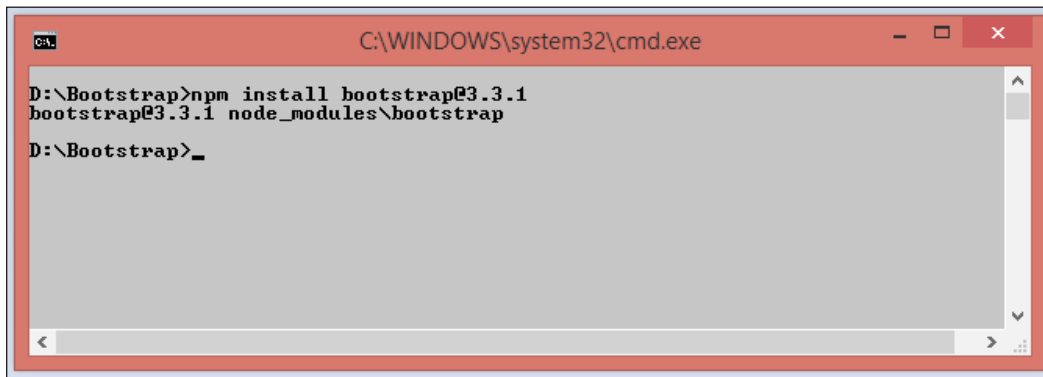
As you might have guessed, Bootstrap 3.3.4 version has been downloaded and installed. Now, let's have a look at our installation folder:

- You have got one new folder created named `node_modules`
- Inside the `node_modules` folder, another new folder has been created named `bootstrap`
- Inside this new folder, you have got the Bootstrap source code bundle

Note that the previous command downloads and installs the latest available version of Bootstrap. If you need any older versions, you execute this command (we are installing Bootstrap version 3.3.1 here in this command):

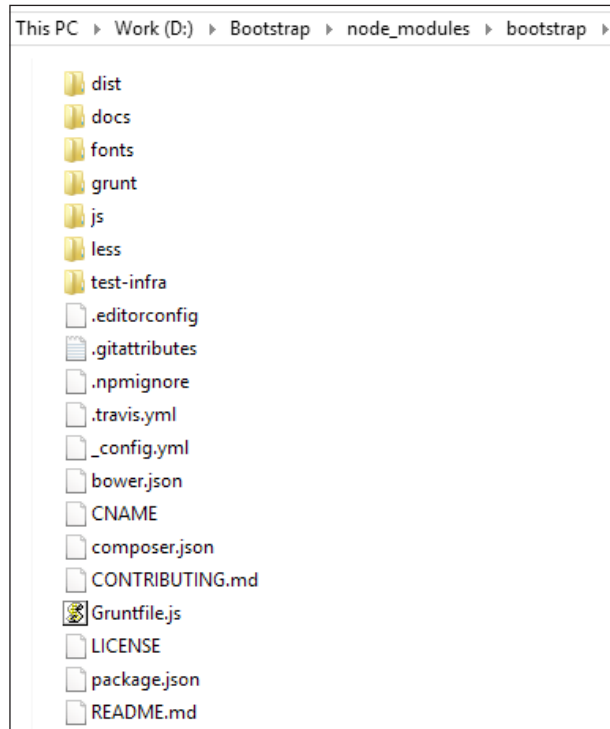
```
npm install bootstrap@3.3.1
```

You should get this:



```
ca. C:\WINDOWS\system32\cmd.exe
D:\Bootstrap>npm install bootstrap@3.3.1
bootstrap@3.3.1 node_modules\bootstrap
D:\Bootstrap>_
```

Now, if you check the contents of the newly created folders, you will see the downloaded source code bundle of Bootstrap:

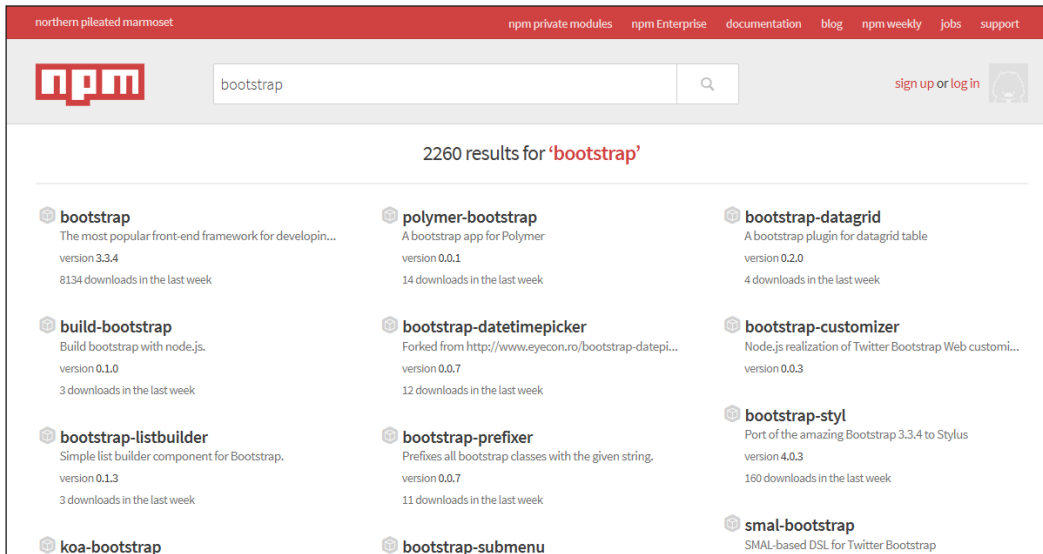


Interestingly, you can compare these files and folders with the source code bundle we had downloaded in *Chapter 2, Getting Started with Bootstrap*. There we had downloaded Bootstrap source code archive from GitHub using the URL <https://github.com/twbs/bootstrap/archive/v3.3.1.zip>. You can see both the downloaded versions are exactly the same. However, here we are creating our development environment for Bootstrap, not just downloading the source code.

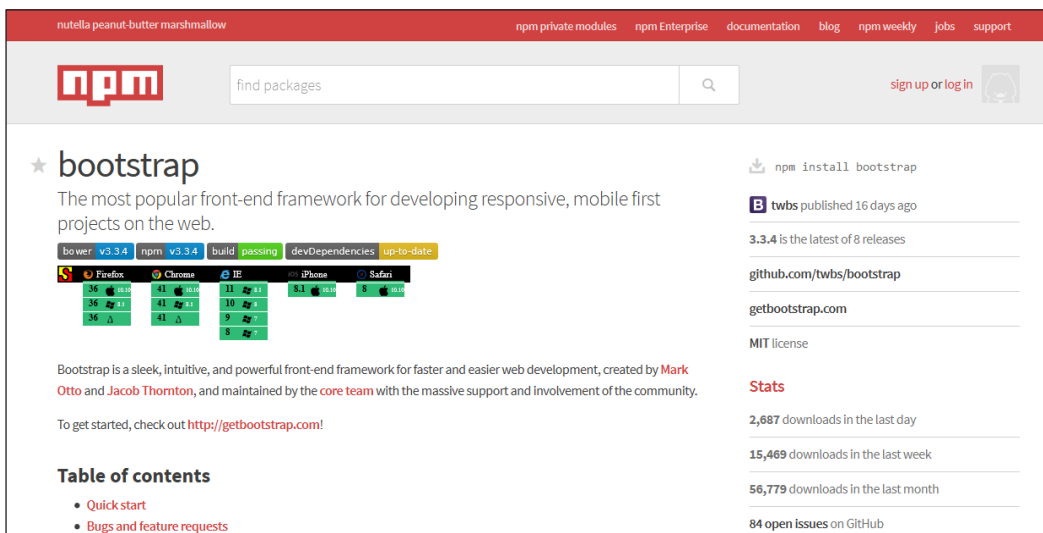
For understanding the background information on exactly what happened, npm is the repository and package manager for a number of JavaScript- and CSS-based tools and packages. Each of these packages are identified by a unique name.

In our case, earlier we have installed `grunt-cli` and now we have installed `bootstrap-grunt-cli` and here `bootstrap` is the name of the package which we are trying to install. For the list of packages available in npm, you can visit <https://www.npmjs.com/>.

If you search for package `bootstrap` there, you will get more than 2,000 results – this means that there are more than 2,000 packages available in npm by this name. Here is a screenshot:



If we click on the first one in this list, we will land into the main Bootstrap package page in npm (<https://www.npmjs.com/package/bootstrap>):



As of now, we have got the Bootstrap source code bundle with us, we are now ready to compile, build, and generate the final distributable files.



Note that, at the right-hand side menu, the installation command `npm install bootstrap` is mentioned. This is how you install Bootstrap via npm.

Compiling and building Bootstrap

If you explore the `bootstrap` folder, you can see a `dist` folder as well, which contains the final distributable CSS, JavaScript, and Font files. The target of building a project is to generate this `dist` folder so that it can be used in the actual application.

First thing we would do is to delete this `dist` folder, since our processes would regenerate this. Deleting this is however not essential, since in any case, the build process will create or override the existing files and folders.

Installing dependencies

To install the dependencies, follow these steps:

1. First, we need to install the other dependencies of building and running Bootstrap.
2. Go to the `D:\Bootstrap\node_modules\bootstrap` folder and run the command:

```
npm install
```

This will get the list of dependencies from the `package.json` file located in the current `bootstrap` folder and will install all of these. Taking a segment of the file, here are the list of dependencies as of Bootstrap 3.3.1:

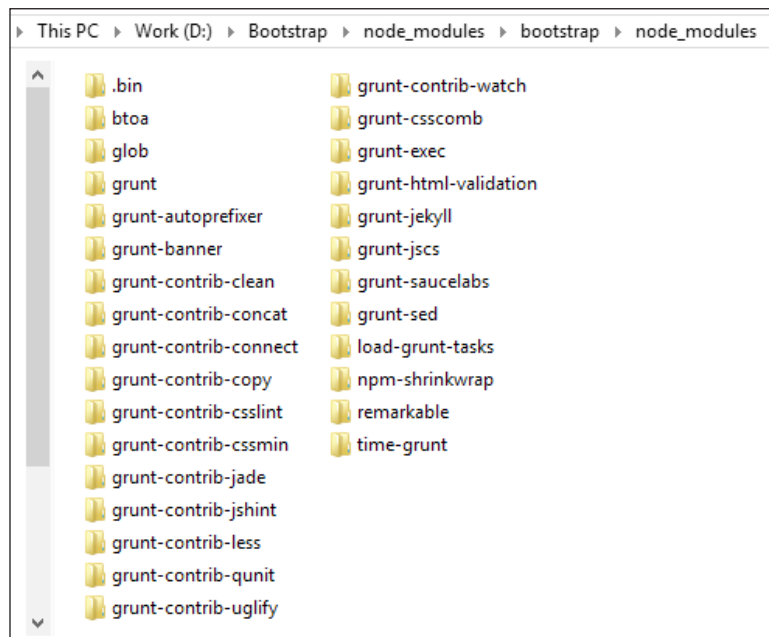
```
"devDependencies": {
  "btoa": "~1.1.2",
  "glob": "~4.0.6",
  "grunt": "~0.4.5",
  "grunt-autoprefixer": "~1.0.1",
  "grunt-banner": "~0.2.3",
  "grunt-contrib-clean": "~0.6.0",
  "grunt-contrib-concat": "~0.5.0",
  "grunt-contrib-connect": "~0.9.0",
  "grunt-contrib-copy": "~0.7.0",
  "grunt-contrib-csslint": "~0.3.1",
  "grunt-contrib-cssmin": "~0.10.0",
  "grunt-contrib-jade": "~0.13.0",
```

```

"grunt-contrib-jshint": "~0.10.0",
"grunt-contrib-less": "~0.12.0",
"grunt-contrib-qunit": "~0.5.2",
"grunt-contrib-uglify": "~0.6.0",
"grunt-contrib-watch": "~0.6.1",
"grunt-csscomb": "~3.0.0",
"grunt-exec": "~0.4.6",
"grunt-html-validation": "~0.1.18",
"grunt-jekyll": "~0.4.2",
"grunt-jscs": "~0.8.1",
"grunt-saucelabs": "~8.3.2",
"grunt-sed": "~0.1.1",
"load-grunt-tasks": "~1.0.0",
"npm-shrinkwrap": "~5.1.0",
"remarkable": "~1.4.0",
"time-grunt": "~1.0.0"
},

```

Thus, when the preceding command (`npm install`) is completed successfully (make sure that there are no errors in this step, otherwise the future steps may not work properly), you will get another `node_modules` folder beneath your `bootstrap` folder — this holds all the dependencies that are needed to build Bootstrap:



As you can see, all the dependencies have been installed as listed in the `package.json` file.

Building Bootstrap

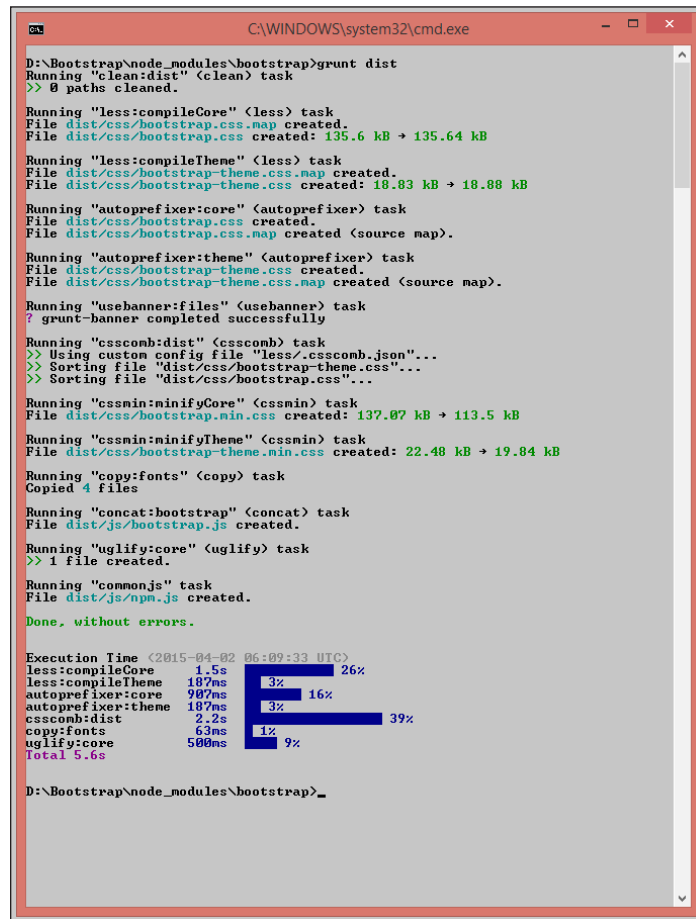
Finally, it is time to compile and build Bootstrap. We are essentially doing the following:

- Compiling here means actually compiling the LESS files and generating the actual and final CSS files
- Building means final packaging or bundling of all CSS, JavaScript, and Font files, which are ready to be consumed in the actual application.

We again, go to the `D:\Bootstrap\node_modules\bootstrap` folder and run this command:

```
grunt dist
```

If everything works fine, you would get this output:



```
D:\Bootstrap\node_modules\bootstrap>grunt dist
Running "clean:dist" (clean) task
>> 0 paths cleaned.

Running "less:compileCore" (less) task
File dist/css/bootstrap.css.map created.
File dist/css/bootstrap.css created: 135.6 kB → 135.64 kB

Running "less:compileTheme" (less) task
File dist/css/bootstrap-theme.css.map created.
File dist/css/bootstrap-theme.css created: 18.83 kB → 18.88 kB

Running "autoprefixer:core" (autoprefixer) task
File dist/css/bootstrap.css created.
File dist/css/bootstrap.css.map created (source map).

Running "autoprefixer:theme" (autoprefixer) task
File dist/css/bootstrap-theme.css created.
File dist/css/bootstrap-theme.css.map created (source map).

Running "usebanner:files" (usebanner) task
? grunt-banner completed successfully

Running "csscomb:dist" (csscomb) task
>> Using custom config file "less/.csscomb.json"...
>> Sorting file "dist/css/bootstrap-theme.css"...
>> Sorting file "dist/css/bootstrap.css"...

Running "cssmin:minifyCore" (cssmin) task
File dist/css/bootstrap.min.css created: 137.07 kB → 113.5 kB

Running "cssmin:minifyTheme" (cssmin) task
File dist/css/bootstrap-theme.min.css created: 22.48 kB → 19.84 kB

Running "copy:fonts" (copy) task
Copied 4 files

Running "concat:bootstrap" (concat) task
File dist/js/bootstrap.js created.

Running "uglify:core" (uglify) task
>> 1 file created.

Running "commonjs" task
File dist/js/npm.js created.

Done, without errors.

Execution Time (2015-04-02 06:09:33 UTC)
less:compileCore    1.5s    ██████████ 26%
less:compileTheme  187ms   ████████ 16%
autoprefixer:core   987ms   ██████████ 16%
autoprefixer:theme  187ms   ████████ 3%
csscomb:dist        2.2s    ██████████ 39%
copy:fonts          63ms    ████████ 1%
uglify:core         588ms   ██████████ 9%
Total 5.6s

D:\Bootstrap\node_modules\bootstrap>
```

Let's check the folder structure now. If you see the `bootstrap` folder, you will see the `dist` folder and files inside that has recreated. We will just have a brief look of what exactly happened behind the scenes. Grunt has executed at least 11 major tasks in this project:

Task	Description
<code>clean:dist</code>	Grunt has tried to delete the <code>dist</code> folder, which we had already removed earlier. Thus, you can see, you don't have to manually clean up the <code>dist</code> folder.
<code>less:compileCore</code>	In this step, Grunt has compiled the LESS files and code to generate the Bootstrap main CSS file.
<code>less:compileTheme</code>	In this step, Grunt has compiled the LESS files and code to generate the Bootstrap theme CSS file.
<code>autoprefixer:core</code>	Autoprefixer is a tool or method that parses CSS files and adds vendor-specific prefixes (for example, <code>-webkit-transition: -webkit-transform 1s;</code>). This task discovers all the places where you need this autoprefixer and adds the specific prefixes in the Bootstrap main CSS file.
<code>autoprefixer:theme</code>	This task discovers all the places where you need this autoprefixer and adds the specific prefixes in the Bootstrap theme CSS file.
<code>csscomb:dist</code>	Formatted and beautified the CSS files.
<code>cssmin:minifyCore</code>	Finally, when the CSS files has been generated, this task minifies the files. In other words in this task, the <code>bootstrap.min.css</code> file has been created.
<code>cssmin:minifyTheme</code>	Similar to the previous one, this task creates the <code>bootstrap-theme.min.css</code> file.
<code>copy:fonts</code>	As the name suggests, this task creates and copies all the Fonts files in the <code>dist</code> folder.
<code>concat:bootstrap</code>	Having handled the CSS files until now, this task concatenates and merges all the individual JavaScript files into one final file— <code>bootstrap.js</code> .
<code>uglify:core</code>	As the earlier tasks had minified the CSS files, this task also minifies the JavaScript file. In this step, the <code>bootstrap.min.js</code> file is created.

All these steps (along with some others) are mentioned and configured in the `Gruntfile.js` file, which you can find in the `bootstrap` source code folder. Thus finally, you have got the distributable version of your own Bootstrap CSS and JS files.

Summary

In this chapter, we learned a few new tools and systems that are quite useful and popular nowadays while creating CSS and JavaScript applications – node.js and Grunt.

We saw how to install and configure these tools. Finally, we saw how to use these tools in order to compile Bootstrap LESS files and generate consumable CSS files. We learned how to package and build the Bootstrap files and generate the whole distributable bundle.

In the next chapter, we will see how to customize Bootstrap to meet your design and look-and-feel requirements.

7

Customizing Bootstrap

In the previous chapter, we have completed the setup of development and customization environment of Bootstrap.

In this chapter, we will see how to customize Bootstrap CSS in order to match your needs in terms of look and feel. We will use our Bootstrap development environment to customize the CSS.

There are essentially two methods for customizing the Bootstrap output:

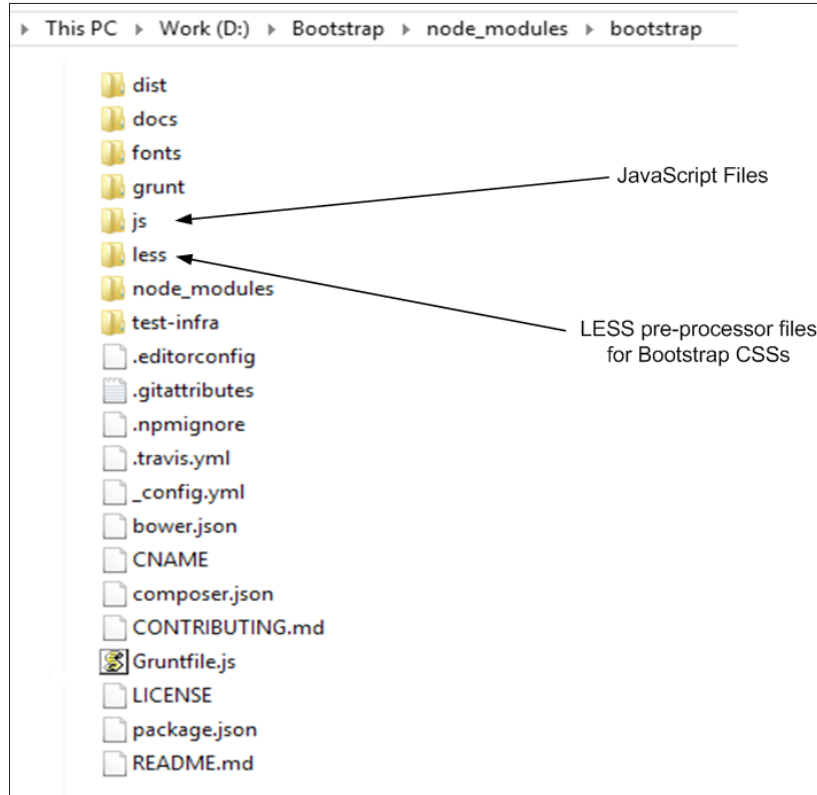
- By updating the Bootstrap source code using the build environment
- By using the Bootstrap web interface to generate customized code

We will explore both the methods in this chapter.

Customizing using the build environment

Just to recollect that this is the folder structure of our Bootstrap source code bundle we configured in the previous chapter. Please note that the list of files and folders can vary depending on the version of Bootstrap you are using.

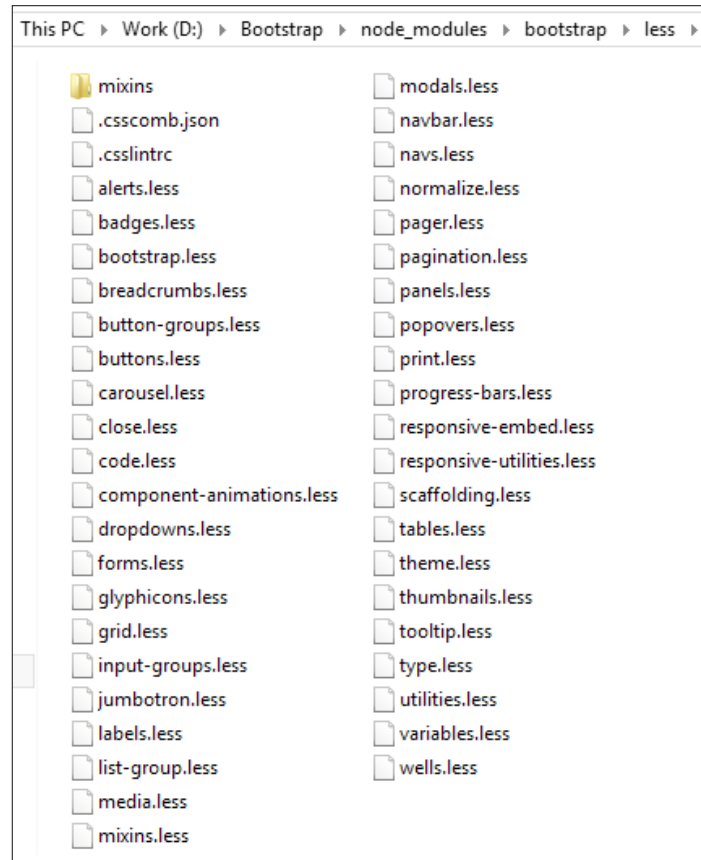
In our case, we are using Bootstrap 3.3.1:



As marked in the preceding screenshot, these are two important folders we will be dealing with while customizing the behavior of Bootstrap.

While it is very rare to change anything in the JavaScript code (in the `js` folder as marked in the screenshot), we will concentrate on the LESS files (in the `less` folder as marked in the preceding screenshot) to change the Bootstrap CSS.

Here are the contents of the `less` folder:



As you can see, there is a single and separate LESS file for each of the CSS components and objects available in Bootstrap, for example, buttons, panels, tables, thumbnails, wells, breadcrumbs, Glyphicons, and grids.

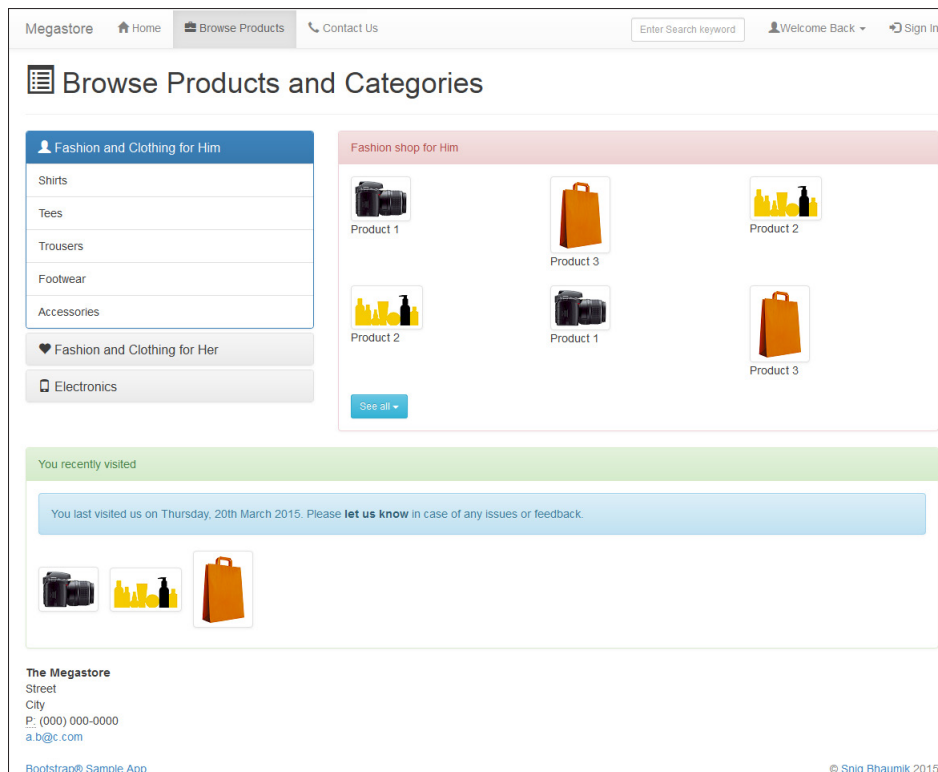
However, as well as the individual LESS files, we also have generic files such as:

- `bootstrap.less`
- `theme.less`
- `utilities.less`
- `scaffolding.less`
- `variables.less`

Each of these files can be explained as follows:

- The `bootstrap.less` file is the main collaborator file that encapsulates all the individual LESS files and generates the final big fat `bootstrap.css` file.
- The `theme.less` file is responsible for mainly generating the `bootstrap-theme.css` file.
- The `utilities.less` file creates the general utility classes, for example, some of the classes as in <http://getbootstrap.com/css/#helper-classes>.
- The `scaffolding.less` file is the main template of Bootstrap CSS file. It defines the page structure and template, and sets styles for the default HTML components (such as `html`, `body`, and `input`)
- The `variables.less` file is probably one of the most important files which we will deal with. This file encapsulates all the LESS variables that have been used at various locations in Bootstrap CSS.

In order to change and update Bootstrap CSS, we will mostly be changing the values of the various variables of this file. Just to recollect that this is our **Browse** page as of now:



Now, let's change a few variables and values in the `variables.less` file to change the look and feel of this site:

1. Open the `variables.less` file.
2. The first section is `Colors`. The default code is:

```
@gray-base:           #000;
@gray-darker:         lighten(@gray-base, 13.5%); // #222
@gray-dark:          lighten(@gray-base, 20%);  // #333
@gray:               lighten(@gray-base, 33.5%); // #555
@gray-light:         lighten(@gray-base, 46.7%); // #777
@gray-lighter:       lighten(@gray-base, 93.5%); // #eee

@brand-primary:      darken(#428bca, 6.5%);
@brand-success:      #5cb85c;
@brand-info:         #5bc0de;
@brand-warning:      #f0ad4e;
@brand-danger:       #d9534f;
```

3. Let's change this as follows:

```
@gray-base:           #000;
@gray-darker:         #222;
@gray-dark:          #282828;
@gray:               #555;
@gray-light:         #888;
@gray-lighter:       #ADAFAE;

@brand-primary:      #2A9FD6;
@brand-success:      #77B300;
@brand-info:         #9933CC;
@brand-warning:      #FF8800;
@brand-danger:       #CC0000;
```

4. In the next `Scaffolding` section, change this code:

```
/** Background color for '<body>'.
@body-bg:           #fff;
/** Global text color on '<body>'.
@text-color:        @gray-dark;
```

into this:

```
/** Background color for '<body>'.
@body-bg:           #060606;
/** Global text color on '<body>'.
@text-color:        @gray-light;
```

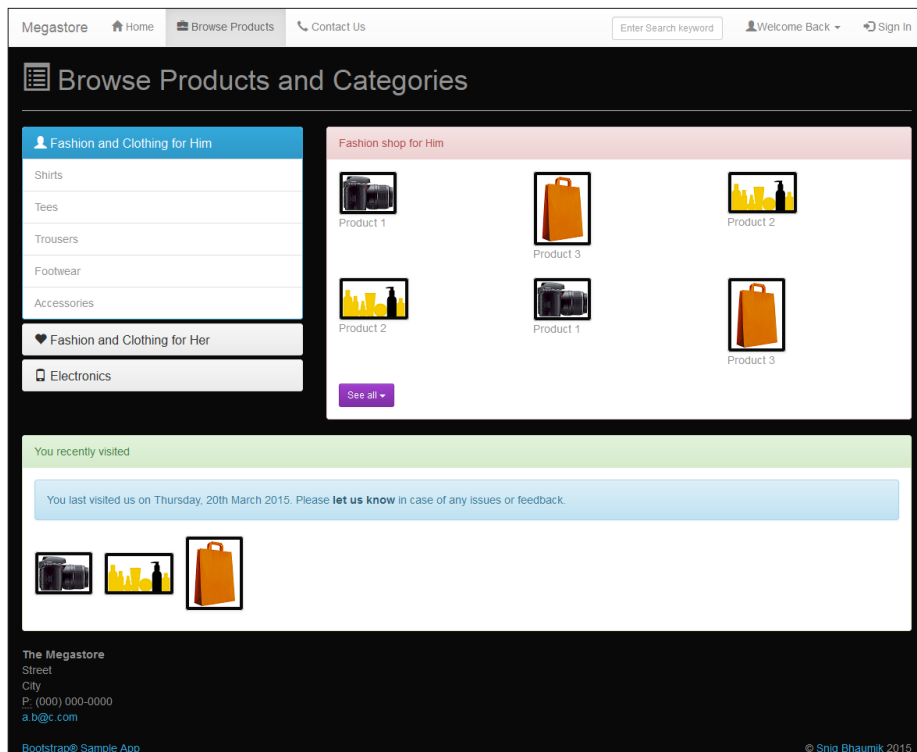
Now let's see what changes this brings to the page. To make this happen, we will build the code and deploy it to the web server:

1. Open the command prompt, go to the project folder using `D:\Bootstrap\node_modules\bootstrap`.
2. Build the project and run this command in the terminal: `grunt dist`.
3. New set of CSS and JavaScript files will be generated in the `dist` folder.
4. In order to deploy this updated code, we will copy the `css` folder (copying the `css` folder only since we haven't done any changes in JavaScript or Font) to the application `bs` folder (refer to *Chapter 2, Getting Started with Bootstrap* where we discussed the folder structure of the application we are building).



Please note that you do not need to copy the `css` folder and files manually each time you make any changes in those files, instead you can use suitable Grunt commands to automatically perform this job for you.

This is the new look of our page:



You can see that the background color of our page has been changed, as have the font colors. This has been done due to this line of code:

```
@body-bg:                #060606;
```

Here instead of having body color as white (#FFF), we have opted for #060606.

The @body-bg element is a LESS variable defined; this variable is used to set the background color of the body element in the `scaffolding.less` file, shown as follows:

```
body {  
  font-family: @font-family-base;  
  font-size: @font-size-base;  
  line-height: @line-height-base;  
  color: @text-color;  
  background-color: @body-bg;  
}
```

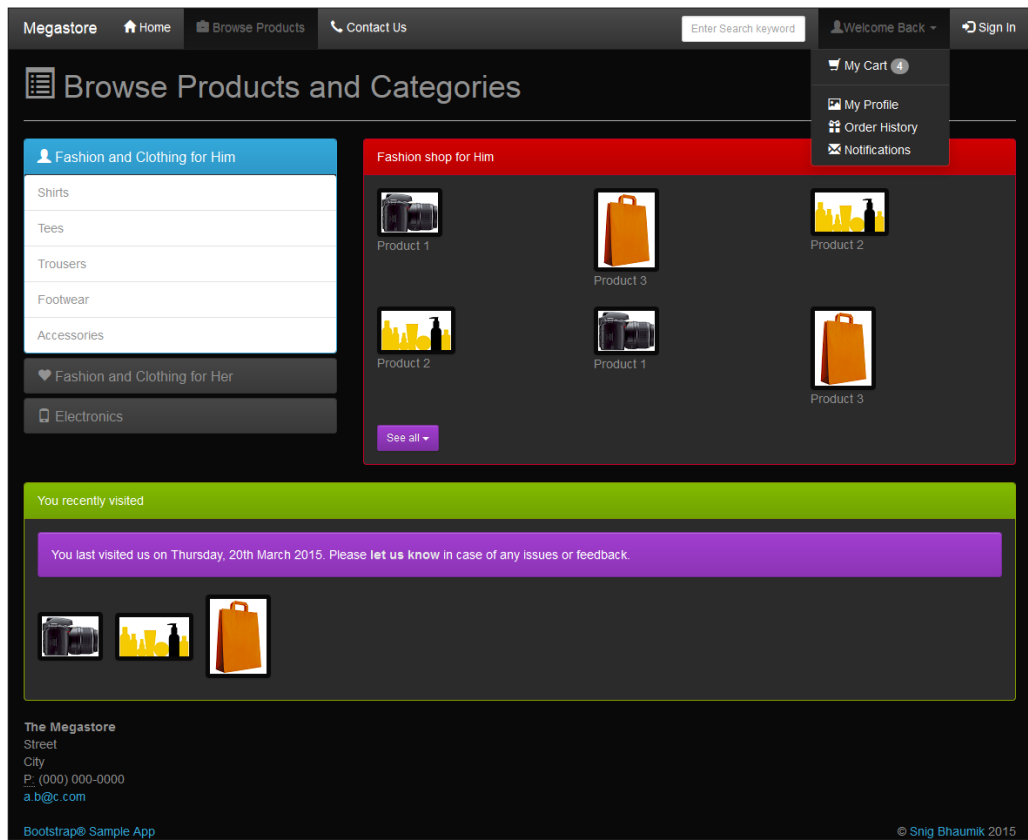
Similarly, the text-color, font-family, font-size, and other styles are also set from LESS variables.

All these LESS files contain lines of code and syntaxes as per LESS preprocessor language. We are not delving deep into LESS syntaxes here. However, it is advised that you get familiar with LESS before tweaking the Bootstrap LESS files.

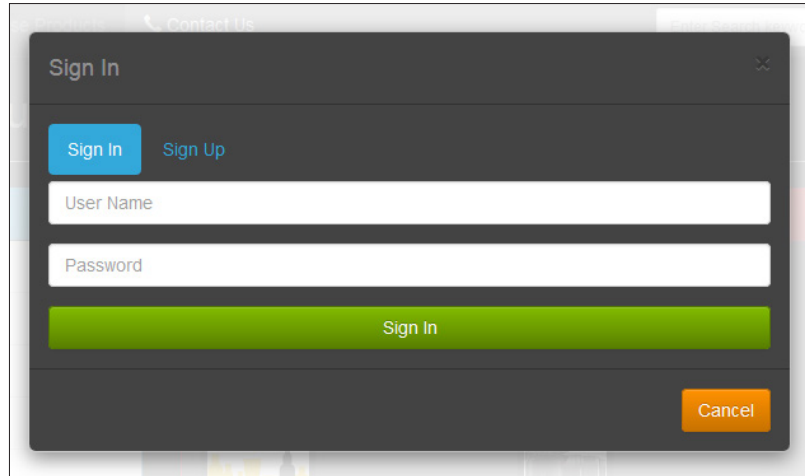


One good starting point to understand that LESS is their official website—<http://lesscss.org/>.

In this way, we keep on changing various LESS variables in the `variables.less` file. This is our final version of the look and feel of the **Browse Products** page:



Here is how the popup dialog box looks:



Some of the code we have changed in the `variables.less` file is as follows:

```
@dropdown-bg:                @gray-darker;
@dropdown-border:            rgba(255,255,255,0.1);
@dropdown-fallback-border:   #444;
@dropdown-divider-bg:        rgba(255,255,255,0.1);
@dropdown-link-color:        #FFF;

@navbar-default-link-color:   #FFF;
@navbar-default-color:        @text-color;
@navbar-default-bg:           @gray-dark;
```

```
@state-success-text:          #fff;
@state-success-bg:           @brand-success;
@state-success-border:      darken(spin(@state-success-bg, -10),
5%);

@state-info-text:           #fff;
@state-info-bg:            @brand-info;
@state-warning-text:       #fff;
@state-warning-bg:        @brand-warning;
@state-danger-text:       #fff;
@state-danger-bg:        @brand-danger;

@modal-content-bg:         lighten(@body-bg, 20%);
@modal-backdrop-bg:       #FFF;
@modal-backdrop-opacity:  .9;
@modal-header-border-color: @gray-dark;

@panel-bg:                 @gray-darker;
@panel-inner-border:       @gray-dark;
@panel-default-text:       @text-color;

@thumbnail-border:        transparent;
```

If you are wondering how these LESS files are transformed into proper and browser-consumable CSS files, the grunt task `less:compileCore` does the job. It invokes the LESS JavaScript compiler in the background, parses the LESS files, and finally generates the CSS files.

The LESS JavaScript compiler we are talking about here is `less.js`, located in the folder at `D:\Bootstrap\node_modules\bootstrap\node_modules\grunt-contrib-less\tasks`.

Thus, we have seen how we can update Bootstrap LESS files for customizing the default look and feel. In this way, you can update any other LESS files as well, if needed. The process of updating the JavaScript source files are also the same. All JS source files are located in the folder at `D:\Bootstrap\node_modules\bootstrap\js`. You can change any of these JS source files, build Bootstrap using grunt, and deploy them to the final application location for consumption.



Having said that, the preceding method does force you to update the actual source files of Bootstrap itself; thus, your customized code will be overwritten when you download an updated version of Bootstrap.

Hence, a better approach will be to create custom versions for each of the overridden files and put your customized code inside that. Finally, import and use these custom files. Thus, your custom code will not be affected when you download an updated version of Bootstrap.

Customizing using Bootstrap web interface

The second method to generate the customized CSS files is by using Bootstrap web interface—<http://getbootstrap.com/customize/>.

This interface enables you to pick and choose which CSS and JavaScript components you want in your project. For example, if you don't need the ScrollSpy JS component, then why should you deploy the corresponding JS code in your production server? The code will be loaded via the `bootstrap.min.js` file, and your HTML page size will be unnecessarily bigger. Thus, it is a good practice to avoid loading all those JavaScript components, which you actually are not going to use.

The following is the screenshot of this page:

Less files Toggle all

Choose which Less files to compile into your custom build of Bootstrap. Not sure which files to use? Read through the [CSS](#) and [Components](#) pages in the docs.

Common CSS	Components	JavaScript components
<input checked="" type="checkbox"/> Print media styles	<input checked="" type="checkbox"/> Glyphicons	<input checked="" type="checkbox"/> Component animations (for JS)
<input checked="" type="checkbox"/> Typography	<input checked="" type="checkbox"/> Button groups	<input checked="" type="checkbox"/> Dropdowns
<input checked="" type="checkbox"/> Code	<input checked="" type="checkbox"/> Input groups	<input checked="" type="checkbox"/> Tooltips
<input checked="" type="checkbox"/> Grid system	<input checked="" type="checkbox"/> Navs	<input checked="" type="checkbox"/> Popovers
<input checked="" type="checkbox"/> Tables	<input checked="" type="checkbox"/> Navbar	<input checked="" type="checkbox"/> Modals
<input checked="" type="checkbox"/> Forms	<input checked="" type="checkbox"/> Breadcrumbs	<input checked="" type="checkbox"/> Carousel
<input checked="" type="checkbox"/> Buttons	<input checked="" type="checkbox"/> Pagination	
<input checked="" type="checkbox"/> Responsive utilities	<input checked="" type="checkbox"/> Pager	
	<input checked="" type="checkbox"/> Labels	
	<input checked="" type="checkbox"/> Badges	
	<input checked="" type="checkbox"/> Jumbotron	
	<input checked="" type="checkbox"/> Thumbnails	
	<input checked="" type="checkbox"/> Alerts	
	<input checked="" type="checkbox"/> Progress bars	
	<input checked="" type="checkbox"/> Media items	
	<input checked="" type="checkbox"/> List groups	
	<input checked="" type="checkbox"/> Panels	
	<input checked="" type="checkbox"/> Responsive embed	
	<input checked="" type="checkbox"/> Wells	
	<input checked="" type="checkbox"/> Close icon	

jQuery plugins Toggle all

Choose which jQuery plugins should be included in your custom JavaScript files. Unsure what to include? Read the [JavaScript](#) page in the docs.

Linked to components	Magic
<input checked="" type="checkbox"/> Alert dismissal	<input checked="" type="checkbox"/> Affix
<input checked="" type="checkbox"/> Advanced buttons	<input checked="" type="checkbox"/> Collapse
<input checked="" type="checkbox"/> Carousel functionality	<input checked="" type="checkbox"/> Scrollspy
<input checked="" type="checkbox"/> Dropdowns	<input checked="" type="checkbox"/> Transitions (required for any kind of animation)
<input checked="" type="checkbox"/> Modals	
<input checked="" type="checkbox"/> Tooltips	
<input checked="" type="checkbox"/> Popovers (requires Tooltips)	
<input checked="" type="checkbox"/> Toggable tabs	

Another (and probably more important) requirement that you can accomplish via this web interface is to customize the Bootstrap CSS by changing the LESS variables (all those same variables that we customized in the previous section of this chapter):

Less variables Reset to defaults

Customize Less variables to define colors, sizes and more inside your custom CSS stylesheets.

Colors
Gray and brand colors for use across Bootstrap.

@gray-base #000	@gray-darker lighten(@gray-base, 13.5%)	@gray-dark lighten(@gray-base, 20%)
@gray lighten(@gray-base, 33.5%)	@gray-light lighten(@gray-base, 46.7%)	@gray-lighter lighten(@gray-base, 93.5%)
@brand-primary darken(#428bca, 6.5%)	@brand-success #5cb85c	@brand-info #5bc0de
@brand-warning #f0ad4e	@brand-danger #d9534f	

Scaffolding
Settings for some of the most global styles.

- Import
- Less components
- jQuery plugins
- Less variables**
- Colors
- Scaffolding
- Typography
- Iconography
- Components
- Tables
- Buttons
- Forms
- Dropdowns
- Media queries breakpoints
- Grid system
- Container sizes
- Navbar
- Navs
- Tabs
- Pills
- Pagination
- Pager
- Jumbotron
- Form states and alerts
- Tooltips
- Popovers
- Labels
- Modals
- Alerts
- Progress bars

What you only need to do is to update the values of these variables in this interface and finally press the **Compile** and **Download** button at the bottom of the page.

This will generate the Bootstrap CSS and JS files both in normal and minified forms. Of course, you will need to take extra care about the version of Bootstrap you are using since this interface will act only on the latest stable version. If you are using any specific (and older) version of Bootstrap, you should do the customization using the local build environment we have created here. That would also enable you to properly maintain your customizations in your local source code version control server such as SVN.

Summary

In the previous chapter, we learned how to establish the compilation and build environment of Bootstrap. In this chapter, we used this environment to generate our customized Bootstrap files.

We saw another way of doing this customization by using Bootstrap's official web interface.

As we have almost finished our journey exploring Bootstrap, in the next and final chapter, we will see how we can extend Bootstrap using a number of other third-party frameworks and components.

8

Extending Bootstrap

We now know how to use, build, and customize Bootstrap. We have explored most of the features and utilities of Bootstrap in various use cases and requirements.

However, on several occasions, you might feel that the styles and components offered by Bootstrap are not enough, and to create a high-end website with rich user experience, you might need some more features and built-in components that would help you design and generate responsive websites real quick and fast.

Fortunately, there are a lot of extensions on top of Bootstrap available in the open source community for various aspects and requirements – for example themes, UI components, jQuery plugins, notifications, tables, and navigations.

To understand the scenario better, if you are into the jQuery world, you have of course used the extensions of jQuery UI (<https://jqueryui.com/>) – these are components based on standard and default jQuery and are made to enable you with a lot of more features and controls that default jQuery does not offer. Similarly, we have a lot of community extensions written on top of default Bootstrap.



This is important to keep in mind that these extensions are developed by community members and any third-party vendors. Thanks to the community and persons who have developed these extensions and made available as open source license. However, the Bootstrap team does not take any responsibility on the performance and functionality of these extensions.


In this final chapter, we will explore a few selected extensions in order to make our Bootstrap-based application more feature-rich and powerful.

The following extensions will be discussed in this chapter:

- Extending the default Bootstrap theme
- Installing and using a tree-view control
- Installing and using a **What You See Is What You Get (WYSIWYG)** editor

Theme extension – Bootswatch


This is one of the largest free stores of themes developed on and using Bootstrap. Refer to their website <http://bootswatch.com/>, and you can see a number of free themes listed and available in this space.

 The free themes are distributed under the MIT license. However, it would be good if you read the usage terms and conditions before using the files.

In order to understand how to use them, we will download and use these themes. Let's say, we will go ahead with the *United* theme – <http://bootswatch.com/united/>. There are two ways you can use this theme (or any other):

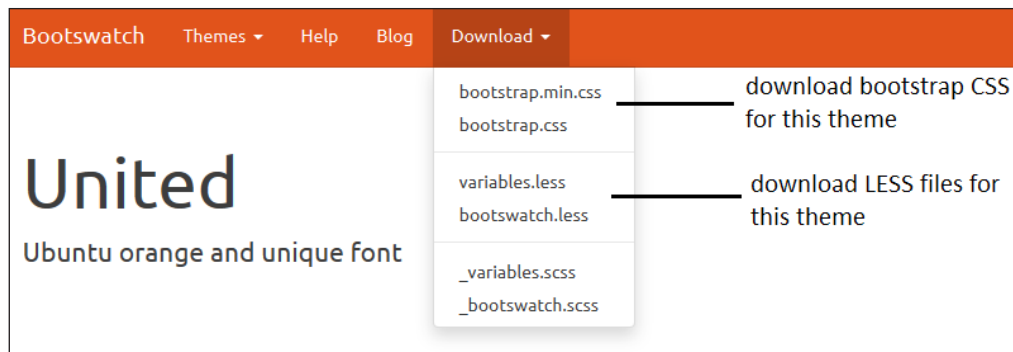
- By downloading the CSS files and directly using in your application.
- By downloading the LESS variables file and building and deploying it properly.

We will explore both the ways.

 Note that Bootstrap also supports SASS preprocessors in addition to LESS. If you are used to using SASS, you can adapt SASS also instead of LESS. However, we will discuss the LESS ways in this book.

Downloading the CSS files

To download the theme, you can go to the <http://bootswatch.com/united/> URL and download the **bootstrap.min.css** and/or **bootstrap.css** options from the top navigation bar:



Perform these steps:

1. Download the CSS files and go to the `css` folder of Bootstrap under the `bs` folder in our application (refer to *Chapter 2, Getting Started with Bootstrap* to see the application folder structure).
2. Copy the downloaded CSS files of *United* theme in this folder (you are essentially overwriting the existing CSS files here. So, it would be good if you take a backup of the original files).
3. Open the application in the browser, and you will see that the look and feel has been updated.

Using the LESS files

As you can imagine, in this method, we will download the LESS variables file and will use the build environment to generate the CSS files. Follow these steps:

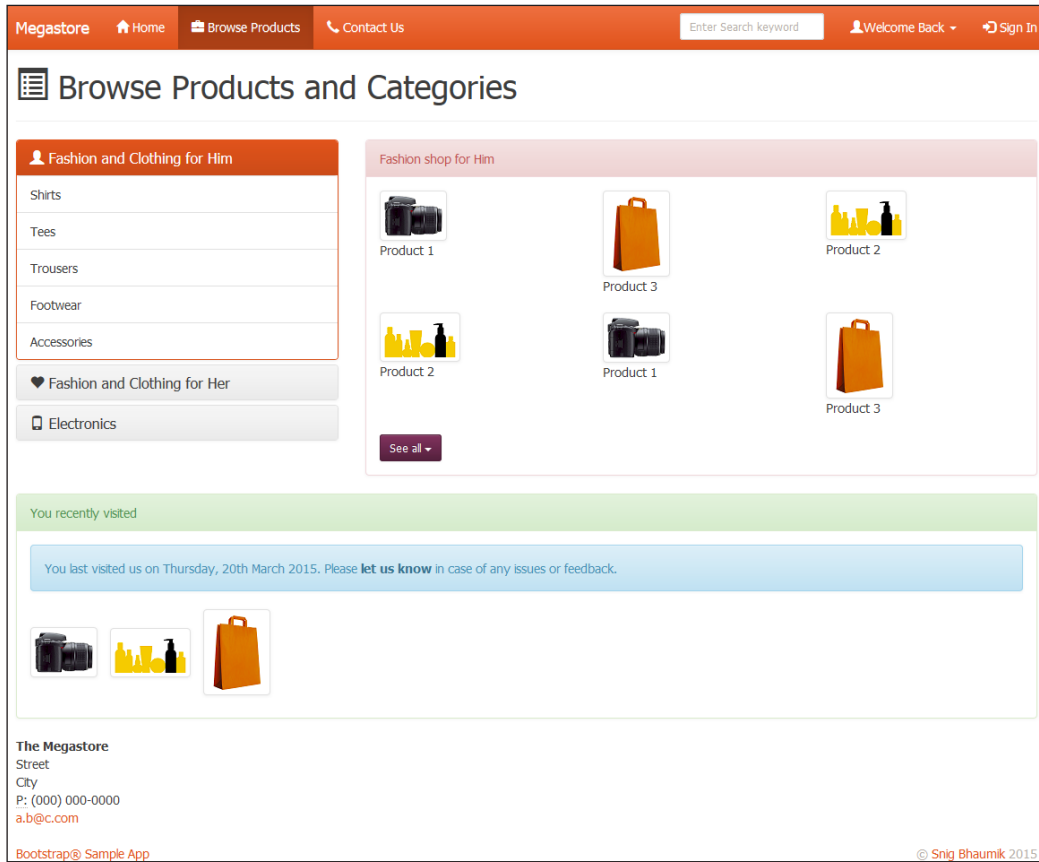
1. First, we download the LESS files (refer to the preceding screenshot) and copy them into our Bootstrap build environment, that is in the `D:\Bootstrap\node_modules\bootstrap\less` folder.



Here also, we are overwriting the existing `variables.less` file. So if you are not under any source code control server, it would be better to take backups before overwriting the existing files.

2. Next steps are pretty obvious, we compile and build the project (the command `grunt dist`). Copy the CSS files into the final application deployment folder.

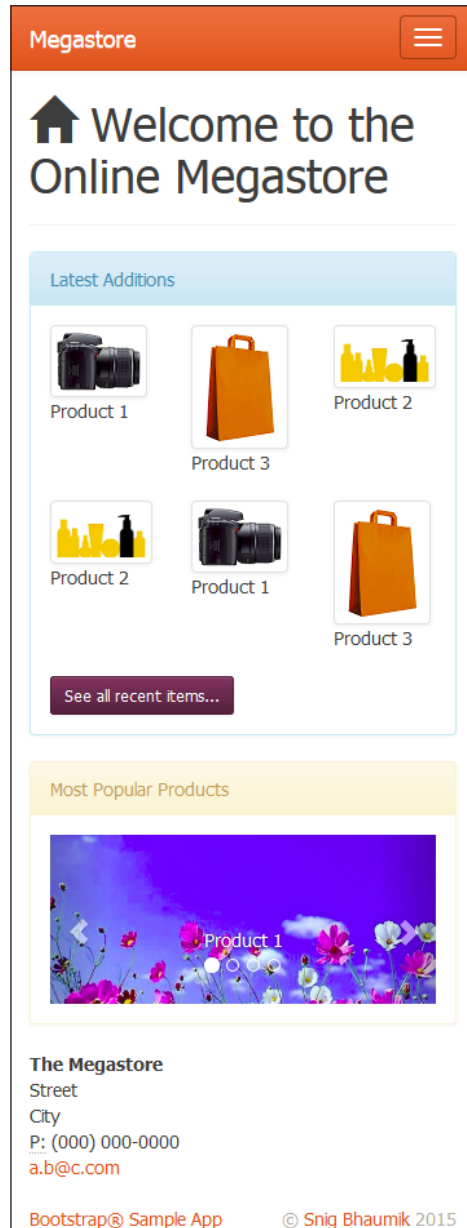
3. Open the application in the browser and see the new look and feel. It would look something similar to the following screenshot:



4. The new look of the popup dialog box is like this:

The image shows a 'Sign Up' popup dialog box. At the top left is the title 'Sign Up' and a close button (X). Below the title are two buttons: 'Sign In' and 'Sign Up'. The 'Sign Up' button is highlighted in orange. Below these buttons are five input fields: 'Your Full Name', 'Desired User Name', 'Your Email Id', 'Password', and 'Confirm Password'. At the bottom of the dialog box, there is a large green button labeled 'Sign Up' and a smaller yellow button labeled 'Cancel'.

5. Here is the mobile device view:



In a similar way, you can download and use any of these freely available themes. However, as you have rightly guessed, downloading and using the LESS files (instead of CSS files) is a better approach.

A tree-view control

A tree-view (sometimes called as Outline view) control in HTML is generally used in order to represent hierarchical information. Each item of the view can have multiple subitems (or children) and each of the subitems can further have multiple children. Also, this hierarchy can go on up to any level. Each item can be expanded to show the child items, and can be collapsed to hide them. One of the most commonly used tree view is the Windows Explorer, where all the folders and subfolders (and corresponding files) are displayed.

Here is an example of tree-view control built on Bootstrap. Such a control is not by default available in the Bootstrap distribution, and of course, you don't want to use just any other control in your web page, which might challenge your mobile device compatibility requirements.



In the community marketplace, a lot of such controls, components, and libraries are available. Should you require any such controls that are not available in the default Bootstrap distribution; it is always suggested to implement a control that is built on Bootstrap, rather than using any jQuery-based or any other JavaScript-based component. Even if you use such controls, make sure that they do not break the mobile compatibility.

This chapter shows some of these examples to guide you how to use such controls that are built on top of Bootstrap.

As an example, we have taken this component, which is popular in the Bootstrap community: <https://github.com/jonmiles/bootstrap-treeview>. Suppose, in the browser webpage, we are going to replace the panel-group view of the category list with a more appropriate rendering model—a tree view.

Here is the HTML code for using the control; this replaces the whole panel-group HTML section where the category listing were also embedded:

```
<div class="col-xs-12 col-sm-12 col-md-4 col-lg-4">
  <div id="categoryList"></div>
</div>
```

This simple line of HTML will do. Rest tricks are there is the JavaScript code.

```
var categories = [
  {
    text: "Fashion and Clothing for Him",
    icon: "glyphicon glyphicon-user",
    state: {
      expanded: true,
      selected: true
    }
  }
];
```

```
    },
    nodes: [
      {
        text: "Shirts",
      },
      {
        text: "Tees"
      },
      {
        text: "Trousers"
      },
      {
        text: "Footwear"
      },
      {
        text: "Accessories"
      }
    ]
  },
  ...

  ];
  $('#categoryList').treeview({
    data: categories,
    onNodeSelected: function(event, data) {
      $('#productHead').text(data.text);
    }
  });
```

As you can see, we have replaced the actual data embedded into the HTML elements into proper JSON data format in JavaScript. This method is of course better and suitable since you would want to invoke some AJAX or server-side call to pull the category list, which in turn returns the data in the JSON format.

We have also added a small event handler (**onNodeSelected**) in order to change the caption of the right-hand side section. Here is the output:

The screenshot shows a web application interface for 'Megastore'. The top navigation bar includes 'Home', 'Browse Products', and 'Contact Us', along with a search bar and user options like 'Welcome Back' and 'Sign In'. The main content area is titled 'Browse Products and Categories' and features a sidebar with a tree-view menu for 'Fashion and Clothing for Her' (including Shirts, Tees, Skirts, Footwear, Jewellery, Accessories) and 'Electronics'. The main product grid displays six items labeled 'Product 1' and 'Product 2' with corresponding icons. A 'See all' button is located at the bottom of the grid. Below the grid is a 'You recently visited' section with a message and three product icons. The footer contains contact information for 'The Megastore' and a copyright notice for 'Snig Bhaumik 2015'.

Installing the tree-view component

From the GitHub URL <https://github.com/jonmiles/bootstrap-treeview>, download the ZIP archive. In the `dist` folder of this archive, we have two files – a JS file and another CSS file. Copy these two files in our application folder and include them in the HTML page. Here are the files we are talking about:

Name	Size	Packed	Type	Modified	CRC32
.			File folder		
bootstrap-treeview.min.css	207	119	Cascading Style Sh...	30-03-2015 02:11	5CBFC60A
bootstrap-treeview.min.js	10,871	2,978	JavaScript File	30-03-2015 02:11	B24C13E2

We have included these files into our `browse.html` page:

```
<link href="bootstrap-treeview.min.css" rel="stylesheet">
<script type="text/javascript" src="bootstrap-treeview.min.js"></script>
```



As a normal convention, we include all additional JS files that are based on Bootstrap and extension of Bootstrap, after inclusion of the main Bootstrap JS file.

WYSIWYG editor and Font Awesome

WYSIWYG is a very common and popular component where you expect that the users of the web page to enter some HTML and formatted content; this control is particularly essential in a **content management system (CMS)**.

In this section, we will demonstrate an editor that is built on Bootstrap and is mobile-ready. The editor we will be using is `bootstrap-wysiwyg`; you can explore this on the <http://mindmup.github.io/bootstrap-wysiwyg/> page. We will replace the standard text area control in our **Contact Us** page with this editor in order to give a better user experience. Here is the HTML code:

```
<div class="btn-toolbar" data-role="editor-toolbar" data-target="#yourComments">
  <div class="btn-group">
    <a class="btn dropdown-toggle" data-toggle="dropdown"
      title="Font"><i class="icon-font"></i><b class="caret"></b></a>
    <ul class="dropdown-menu"></ul>
  </div>
  <div class="btn-group">
    <a class="btn dropdown-toggle" data-toggle="dropdown" title="Font
      Size"><i class="icon-text-height"></i>&nbsp; <b class="caret"></b></a>
    <ul class="dropdown-menu">
      <li><a data-edit="fontSize 5"><font size="5">Huge</font></a></li>
      <li><a data-edit="fontSize 3"><font size="3">Normal</font></a></li>
      <li><a data-edit="fontSize 1"><font size="1">Small</font></a></li>
    </ul>
  </div>
  <div class="btn-group">
    <a class="btn" data-edit="bold" title="Bold (Ctrl/Cmd+B)"><i
      class="icon-bold"></i></a>
```

```

    <a class="btn" data-edit="italic" title="Italic (Ctrl/Cmd+I)"><i
class="icon-italic"></i></a>
    <a class="btn" data-edit="strikethrough" title="Strikethrough"><i
class="icon-strikethrough"></i></a>
    <a class="btn" data-edit="underline" title="Underline (Ctrl/
Cmd+U)"><i class="icon-underline"></i></a>
</div>
<div class="btn-group hidden-xs hidden-sm">
    <a class="btn" data-edit="insertunorderedlist" title="Bullet
list"><i class="icon-list-ul"></i></a>
    <a class="btn" data-edit="insertorderedlist" title="Number
list"><i class="icon-list-ol"></i></a>
    <a class="btn" data-edit="outdent" title="Reduce indent
(Shift+Tab)"><i class="icon-indent-left"></i></a>
    <a class="btn" data-edit="indent" title="Indent (Tab)"><i
class="icon-indent-right"></i></a>
</div>
<div class="btn-group hidden-xs hidden-sm">
    <a class="btn" data-edit="justifyleft" title="Align Left (Ctrl/
Cmd+L)"><i class="icon-align-left"></i></a>
    <a class="btn" data-edit="justifycenter" title="Center (Ctrl/
Cmd+E)"><i class="icon-align-center"></i></a>
    <a class="btn" data-edit="justifyright" title="Align Right (Ctrl/
Cmd+R)"><i class="icon-align-right"></i></a>
    <a class="btn" data-edit="justifyfull" title="Justify (Ctrl/
Cmd+J)"><i class="icon-align-justify"></i></a>
</div>
<div class="btn-group hidden-xs hidden-sm">
    <a class="btn dropdown-toggle" data-toggle="dropdown"
title="Hyperlink"><i class="icon-link"></i></a>
    <div class="dropdown-menu input-append">
        <input class="span2" placeholder="URL" type="text" data-
edit="createLink"/>
        <button class="btn" type="button">Add</button>
    </div>
    <a class="btn" data-edit="unlink" title="Remove Hyperlink"><i
class="icon-cut"></i></a>
</div>
<div class="btn-group hidden-xs hidden-sm">
    <a class="btn" data-edit="undo" title="Undo (Ctrl/Cmd+Z)"><i
class="icon-undo"></i></a>
    <a class="btn" data-edit="redo" title="Redo (Ctrl/Cmd+Y)"><i
class="icon-repeat"></i></a>
</div>
</div>
<div id="yourComments">please tell us what you think about us.</div>

```

Here is the CSS code for our control:

```
#yourComments {
  background-color: white;
  border: 1px solid rgb(204, 204, 204);
  border-collapse: separate;
  border-radius: 3px;
  box-shadow: 0 1px 1px 0 rgba(0, 0, 0, 0.075) inset;
  box-sizing: content-box;
  max-height: 25em;
  min-height: 15em;
  outline: medium none;
  overflow: scroll;
  padding: 4px;
}
```

Finally, the JavaScript code is as follows:

```
function initEditorToolbar() {
  var fonts = ['Serif', 'Sans', 'Arial', 'Arial Black', 'Courier',
    'Courier New', 'Comic Sans MS', 'Helvetica', 'Impact', 'Lucida
Grande', 'Lucida Sans', 'Tahoma', 'Times',
    'Times New Roman', 'Verdana'],
    fontTarget = $('[title=Font]').siblings('.dropdown-menu');

  $.each(fonts, function (idx, fontName) {
    fontTarget.append($('- <a data-edit="fontName ' + fontName + '"
style="font-family:\'+ fontName + \' ">'+fontName + '</a></li>'));
  });
  $('[title]').tooltip({container:'body'});
  $('.dropdown-menu input')
    .click(function() {return false;})
    .change(function () {$(this).parent('.dropdown-menu').siblings('.
dropdown-toggle').dropdown('toggle');})
    .keydown('esc', function () {this.value='';$(this).change();});

  $('[data-role=magic-overlay]').each(function () {
    var overlay = $(this), target = $(overlay.data('target'));

```

```

        overlay.css('opacity', 0).css('position', 'absolute').
        offset(target.offset()).width(target.outerWidth()).height(target.
        outerHeight());
    });
};
initEditorToolbar();
$('#yourComments').wysiwyg();
$('#yourComments').cleanHtml();

```

In a desktop system, our new page now looks like this:

The screenshot shows a desktop browser window with a navigation bar at the top containing 'Megastore', 'Home', 'Browse Products', 'Contact Us', a search bar, 'Welcome Back', and 'Sign In'. The main content area features a contact form with the heading 'We would like to hear from you !!!'. The form has three main sections: 'Name' with a text input field containing 'Your name please'; 'Email address' with a text input field containing 'Your Email Id'; and 'Tell us' with a rich text editor. The rich text editor's toolbar is visible, and a font dropdown menu is open, listing fonts such as Serif, Sans, Arial, Arial Black, Courier, Courier New, Comic Sans MS, Helvetica, **Impact**, Lucida Grande, Lucida Sans, Tahoma, Times, Times New Roman, and Verdana. A 'Post It' button is located at the bottom right of the form. The footer contains contact information for 'The Megastore' and a copyright notice for 'Snig Bhaumik 2015'.

In a mobile device, this is the look of our new page:

The image shows a mobile-optimized contact form for 'Megastore'. At the top is an orange header with the brand name and a hamburger menu icon. Below the header, a large heading reads 'We would like to hear from you !!!' accompanied by a telephone icon. The form is organized into several sections: 'Name' with a text input field containing 'Your name please'; 'Email address' with a text input field containing 'Your Email Id'; and 'Tell us' which features a rich text editor toolbar with buttons for Bold (B), Italic (I), Underline (U), and Text Color (A), and a text area containing the placeholder text 'please tell us what you think about us.'. Below the text area is a checkbox labeled 'Subscribe Me !!!' and a prominent orange 'Post It' button with an envelope icon. At the bottom, contact information for 'The Megastore' is listed, including 'Street', 'City', 'P: (000) 000-0000', and 'a.b@c.com'. A footer at the very bottom reads 'Bootstrap® Sample App © Snig Bhaumik 2015'.

As you can see, the standard text area control to enter the comments has been replaced by a richer WYSIWYG editor. You should also note that, in case of mobile devices, we have made a number of toolbar buttons hidden; in this way, you can make some unimportant buttons unavailable in case of devices with lesser real estate.

Installing and using the WYSIWYG component

In order to install, configure, and use the WYSIWYG component, a few JS libraries need to be downloaded. The following are the prerequisites for the WYSIWYG component we are using.

The bootstrap-wysiwyg component

Download the bootstrap-wysiwyg component from the <https://github.com/mindmup/bootstrap-wysiwyg/> GitHub page. However, you only need the main JS file, thus you can download only this file from <https://raw.githubusercontent.com/mindmup/bootstrap-wysiwyg/master/bootstrap-wysiwyg.js>. Copy this file into the application folder and include in the HTML page:

```
<script type="text/javascript" src="bootstrap-wysiwyg.js"></script>
```

The jQuery hotkeys component

The WYSIWYG editor uses hotkeys (or shortcut keys) such as *Ctrl + B*, *Ctrl + U*, and *Ctrl + I* for standard operations. Thus, we need this library as well.

You can download this from the <https://github.com/jeresig/jquery.hotkeys> GitHub page. However, as for the earlier case, you only need the main JS file. Thus, you can download the <https://raw.githubusercontent.com/jeresig/jquery.hotkeys/master/jquery.hotkeys.js> file only.

Copy this file into the application folder and include in the HTML page:

```
<script type="text/javascript" src="jquery.hotkeys.js"></script>
```

Font Awesome

Font Awesome is the iconic font suite designed to be used in Bootstrap projects. You can extend your look, feel, and user experience to a great extent using these CSS classes, and of course these are all device friendly.

For more information about Font Awesome, refer to <http://fontawesome.github.io/Font-Awesome/>. Rather than downloading and installing this locally, we will use the file from CDN location. Thus, we write the following in our HTML page:

```
<link href="http://netdna.bootstrapcdn.com/font-awesome/3.0.2/css/font-awesome.css" rel="stylesheet">
```

Thus, when all the dependencies and libraries are included, the preceding HTML and JavaScript code will render the WYSIWYG editor in your page. You have already seen the HTML and JS code, and also how the editor is rendered in your page.

Here are some links from where you can explore more such components and libraries:

- <http://startbootstrap.com/bootstrap-resources/>
- <http://bootstraphero.com/the-big-badass-list-of-twitter-bootstrap-resources>



Summary

In this final chapter, we explored a few components and resources to extend the default behavior of Bootstrap. By using the community-enabled libraries, you can get almost all the components you need in order to implement any website requirements.

Thus, you can see, in conjunction with these third-party resources, that the power and flexibility of Bootstrap has been enhanced and extended a lot, and with very minimal coding and implementation efforts, you can develop a full-featured mobile-friendly website.

As an additional information, you can check the mobile-friendliness of your website by using Google Webmaster tools, for example, <https://www.google.com/webmasters/tools/mobile-friendly/>.

Index

A

accordions

- about 76, 77
- example 78-81

alerts 56, 85-87

application home page

- rewriting 47, 48

Autoprefixer tool 107

B

badges 56

Bootstrap

- about 7, 8
- application folder structure 25, 26
- building 104-107
- compiling 104
- compiling, tool requisites 96
- components 9, 10
- contents 8
- CSS 8
- customization 11
- customizing, with build
 - environment 109-118
- customizing, with web interface 119-121
- dependencies, installing 104, 105
- file structure 14
- HTML elements 30
- HTML structure 28
- installing 100-104
- JavaScript 10, 11
- obtaining 13, 14
- resources, URL 138

URL 102

using 22-24

utility CSS classes 46

web interface, URL 119

web interface, using for
customization 119-121

bootstrap-wysiwyg component

URL 137

Bootswatch

about 124

CSS files, downloading 124, 125

LESS files, using 125-128

URL 124

breadcrumbs 62, 63

button groups 57, 58

C

carousels 87-89

collapse component 76, 77

Command-line Interface (CLI) 97

content management system (CMS) 132

custom data attributes

about 66

URL 66

D

data entry forms

constructing 40

Contact Us page, finalizing 42-45

horizontal form, creating 41, 42

Don't Repeat Yourself (DRY) 17

dropdown 83-85

F

file structure, Bootstrap

- about 14
- CSS preprocessors 17-19
- mixins 20
- nesting 21
- operations 20
- precompiled bundle 14
- source code bundle 16
- variables 19

Font Awesome

- about 132-138
- URL 137

G

generic files

- bootstrap.less file 112
- scaffolding.less 112
- theme.less file 112
- utilities.less file 112
- variables.less file 112

Glyphicons

- about 50, 51
- URL 50, 51

grid system 36-40

Grunt

- about 98
- Grunt-cli, installing 98-100

H

HTML elements 30

HTML structure, Bootstrap

- about 28
- body section 29
- head section 28

I

images

- rendering 34, 35
- responsive images, displaying in sample application 35, 36

installation

- Bootstrap 100-104
- Grunt-cli 98
- Node.js 97

J

JavaScript add-ons

- accordions 76, 77
- alerts 85-87
- basic concepts 66
- carousels 87-89
- collapse 76, 77
- dropdown 83-85
- final preview, of application 89-92
- modal windows 67
- popovers 81-83
- tabs 72-76
- tooltips 81-83

JavaScript add-ons, basic concepts

- custom data attributes 66
- JavaScript APIs 67
- JavaScript events 67
- packaging add-ons 67

JavaScript APIs 67

JavaScript events 67

jQuery

- hotkeys component, URL 137
- URL 65

Jumbotron 61, 62

L

LESS preprocessor

- about 95
- URL 115

M

mobile-first philosophy 3, 4

modal windows

- about 67
- basic version 68, 69
- example 70-72

N

navigation bar 51-56

Node.js

about 96

installing 97

Node Package Manager (npm)

about 96

URL 103

P

packaged components

about 49

alerts 56

badges 56

breadcrumbs 62

button groups 57

Glyphicons 50

Jumbotron 61

navigation bar 51

page header 50

paginations 63

panels 58

toolbars 57

packaging add-ons 67

page header 50

paginations 63, 64

panels

about 58-60

wells 60, 61

pills 73

popovers 81-83

precompiled bundle

about 14

folder: css folder 15

folder: fonts folder 15

folder: js folder 15

R

responsive classes

about 31

display of elements, controlling 32, 33

lg marker 31

md marker 31

sm marker 31

xs marker 31

responsive design

basics 4

content, sizing to viewport 5

media queries, used for achieving
 responsiveness 6

navigation patterns 7

patterns 6

viewport, setting 5

responsive design, patterns

column drop 6

fluid design 6

layout shifter 6

S

source code bundle

about 16

folder: dist folder 16

folder: fonts folder 17

folder: grunt folder 17

folder: js folder 17

folder: less folder 17

Syntactically Awesome

Stylesheets (Sass) 19

T

tabs 72-76

toolbars 57, 58

tool requisites, for compiling Bootstrap

 Grunt 98

 Node.js 96

tooltips 81-83

tree-view control

about 129, 130

installing 131, 132

URL 129

U

United theme

URL 124

utility CSS classes 46

W

Web Accessibility standards

URL 46

What You See Is What

You Get (WYSIWYG)

about 124, 132-136

bootstrap-wysiwyg component 137

jQuery hotkeys component 137

URL 132

used, for installation 137

using 137



Thank you for buying Bootstrap Essentials

About Packt Publishing

Packt, pronounced 'packed', published its first book, *Mastering phpMyAdmin for Effective MySQL Management*, in April 2004, and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution-based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern yet unique publishing company that focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website at www.packtpub.com.

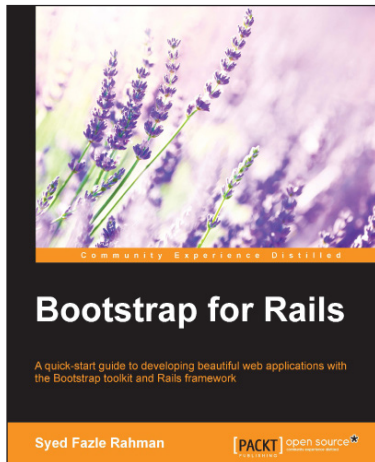
About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around open source licenses, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each open source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, then please contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



Bootstrap for Rails

ISBN: 978-1-78398-726-9 Paperback: 168 pages

A quick-start guide to developing beautiful web applications with the Bootstrap toolkit and Rails framework

1. Enhance your applications with Bootstrap modals and carousels.
2. Explore the usage of advanced Bootstrap components and plugins in Rails through various examples.
3. Use Bootstrap's Grid System to create beautiful responsive Rails applications with this hands-on guide.



Web Development with AngularJS and Bootstrap [Video]

ISBN: 978-1-78439-150-8 Duration: 01:48 hours

Use dynamic AngularJS code and Bootstrap styling to create effective web applications

1. Leverage the power of AngularJS and Bootstrap to develop a functioning web application.
2. Explore the world of open source AngularJS libraries.
3. Understand how to create your own custom components.

Please check www.PacktPub.com for information on our titles



Bootstrap Site Blueprints

ISBN: 978-1-78216-452-4 Paperback: 304 pages

Design mobile-first responsive websites with Bootstrap 3

1. Learn the inner workings of Bootstrap 3 and create web applications with ease.
2. Quickly customize your designs working directly with Bootstrap's LESS files.
3. Leverage Bootstrap's excellent JavaScript plugins.



Developing Mobile Web ArcGIS Applications

ISBN: 978-1-78439-579-7 Paperback: 156 pages

Learn to build your own engaging and immersive geographic applications with ArcGIS

1. Create multi-utility apps for mobiles using ArcGIS Server quickly and easily.
2. Start with the basics and move through to creating advanced mobile ArcGIS apps.
3. Plenty of development tips accompanying links to functional maps to help you as you learn.

Please check www.PacktPub.com for information on our titles

