



Community Experience Distilled

Mobile First Bootstrap

Develop advanced websites optimized for mobile devices using the Mobile First feature of Bootstrap

Alexandre Magno

[PACKT] open source*
PUBLISHING community experience distilled

www.allitebooks.com

Mobile First Bootstrap

Develop advanced websites optimized for mobile devices using the Mobile First feature of Bootstrap

Alexandre Magno

[PACKT] open source 
PUBLISHING community experience distilled
BIRMINGHAM - MUMBAI

Mobile First Bootstrap

Copyright © 2013 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: December 2013

Production Reference: 1111213

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

ISBN 978-1-78328-579-2

www.packtpub.com

Cover Image by Asher Wishkerman (a.wishkerman@mpic.de)

Credits

Author

Alexandre Magno

Project Coordinator

Akash Poojary

Reviewers

Julien Renaux

Felipe Silva

Proofreader

Linda Morris

Acquisition Editors

Meeta Rajani

Owen Roberts

Indexer

Rekha Nair

Commissioning Editor

Amit Ghodake

Production Coordinator

Kirtee Shingan

Technical Editors

Monica John

Tarunveer Shetty

Cover Work

Kirtee Shingan

Copy Editors

Aditya Nair

Shambhavi Pai

Alfida Paiva

Laxmi Subramanian

About the Author

Alexandre Magno has worked for 10 years as a web developer, and is currently working as a software engineer at `globo.com`. He is a very active contributor in the open source community with plenty of projects and acts as a jQuery evangelist and a responsive design missionary. As a multidisciplinary developer, he has strong experience in a wide range of server-side frameworks and CMS such as Ruby on Rails, Django, WordPress, and exploring Node.js. He has developed many libraries that are widely used at `globo.com`, and was one of the first developers to develop mobile websites with a responsive design in the company he worked at. He is an active contributor of Twitter Bootstrap and the creator of one of its branches, the Globo Bootstrap, which is the first translation of Bootstrap to Portuguese, and also developed some components used in `globo.com`.

He is very passionate about web development, and he writes about it in his blog at `alexandremagno.net`. He has already contributed in the publishing of a web magazine about jQuery UI and has even made presentations in some technical events, such as the International Free Software Forum (FISL). Writing this book for him is a great step further, after these achievements.

Besides technology, he is a musician and song writer too. He likes to remember every moment of his life with a music lyric. At this moment, for example, the verse would be "We are the champions, my friend".

Acknowledgements

It was a long journey to get this project into reality, and I would like to thank the team behind this book's publication for their help.

But I would never forget to thank my mom, my family, and friends that support me all the time. I'm sure they're all very excited about this book. Although some of the people who are close to me have no idea what exactly this book is for (for them, I'm just the "nerd guy" or the "geek"), they're still interested in checking it out. It's pleasant to spread the technical knowledge for everyone with a strong demonstration through this book about how the Web is changing society and people's lives. This reason is far enough to make me proud to write this book.

It was very enjoyable to get involved in a new web paradigm and write my piece of contribution to make it a part of the whole new world of web technologies across devices. Mobile First and Bootstrap are powerful sources to make a developer, a designer, or a curious advanced developer get into the brand new mobile web. After this book, you will think about the Web in a different way, and I'm guiding you to meet the new era that will change our minds from fixed width to fluid. So, let's make your mind flexible to think about how to apply Bootstrap correctly with the Mobile First approach and make you capable of doing amazing mobile web projects.

About the Reviewers

Julien Renaux is a software engineer specialized in frontend development. Currently working in France, Julien worked in four continents for companies such as eBay, in which he used Bootstrap since the first release in 2011. He is web-passionate and a JavaScript-aholic, and he loves to learn new technologies and share his experience and enthusiasm on his blog: <http://julienrenaux.fr>.

Felipe Silva is originally from Rio de Janeiro, and now lives in Brooklyn. He has more than seven years' experience in frontend development. He is now a part of the video team at The New York Times as a senior JavaScript engineer. Previously, he worked at Huge, a digital agency based in Brooklyn, and during that time, he built several responsive products for companies such as Four Seasons, GE Capital, Target, and others. Back in Brazil, he used to work for globo.com, the Internet arm of Globo, which is the largest media conglomerate in Latin America.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

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.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across 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 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 nine entirely free books. Simply use your login credentials for immediate access.

Instant Updates on New Packt Books

Get notified! Find out when new books are published by following @PacktEnterprise on Twitter, or the *Packt Enterprise* Facebook page.

Table of Contents

Preface	1
Chapter 1: Bootstrap 3.0 is Mobile First	7
Bootstrap reviewed	8
Desktop to responsive	10
The new mindset – Mobile First	11
Practical example – The responsive dropdown	15
Now Bootstrap uses Bower and Jekyll	16
Running the docs	16
Version 3 in progress in the Github repository	16
Installing Jekyll	16
Bower	17
First step to responsiveness	18
Making changes in the Bootstrap source code	18
Running tests	19
Summary	19
Chapter 2: Designing Stylesheet in Bootstrap 3	21
The grid system	22
Semantic grids	22
The grid framework	24
Breakpoints and completely fluid layouts	25
Predefined classes to control responsive flow	25
Forms in different resolutions	27
The icon library	28
Responsive utilities	29
Responsive classes	29
Semantic grid variables and functions	30

Relative units	30
Navigation	31
Summary	33
Chapter 3: JavaScript, the Behavior in Mobile	
First Development	35
The carousel example	36
A touch of enhancement	37
Data attributes	41
Mobile First and progressive enhancements	42
Be semantic in your HTML markup	42
Unobtrusive JavaScript	44
Follow the Bootstrap tips about accessibility	44
Test a site in a lynx browser	45
Namespace events	46
JavaScript on the server	46
Summary	47
Chapter 4: Getting it All Together – a Simple Twitter App	49
Bootstrapping our application	50
Inserting a customizable version of Bootstrap	50
The project template	52
The Bootstrap modal component example	55
Geolocation	56
The Twitter API search	58
Make a search	60
Going from a tablet device to desktop screen resolution	61
The choice between a web app and mobile application	63
Summary	64
Chapter 5: Performance Matters	65
Responsive images	66
Load on demand	72
Optimizing icons	72
Summary	73
Index	75

Preface

We are living in a rapidly changing time because of the way we use the Web. To follow up on this currently changing paradigm, the development cycle finds its way to follow this new scenario. The frontend community is building amazing tools to make it possible to deal with so many changes. In the past, we had to deal with browsers that did not respect the standards. Now, we are dealing with new devices with a wide range of features. The Web is going mobile. With so many limitations on one side, and the power of flexibility on another side, the mobile and tablet just cannot be a substitute for the huge and wide desktop screens. The mobile needs are complementary to the desktop's standard nature. On the other hand, there's a brand new opportunity with its mobile use and simplicity that makes us see an ongoing challenge to expand our business. This way we have a presence in the ever-growing online market. The Web is not only limited to a desktop anymore. We have to think wider.

The most famous frontend frameworks also have to undergo a deep change to meet this demand. They are radical. In the new Bootstrap release, they decided to get their learning from the oldest version and redesign from scratch, but the focus is now on Mobile First as a user environment. Why do you have to focus on the mobile environment now?

Well, in the new paradigm, we will just get a Mobile First framework for simple development like we used to. We have to follow it up and see how powerful an online product cross device with no disturbing complexity can be, even though it sounds complicated.

We know that Bootstrap is not a silver bullet, but it's a great option to start your development cycle for Mobile First. It makes things simple, and we should know how it does it, because this Version 3 is based on a lot of contributions from Bootstrap developers. This is a strong reason for you to follow the same principles.

You may have already used Bootstrap or visited sites that use it. It's not hard to identify Bootstrap's basic template in a lot of websites around here, for example, the documentation APIs of open source projects. Research of the meanpath shows that Bootstrap is present in 1 percent of the 150-million websites worldwide, powering 1 percent of the Web (<http://blog.meanpath.com/twitter-bootstrap-now-powering-1-percent-of-the-web/>). So, we know the power of its use and the amazing things we can do, but now we can do even more, and you have a chance to explore its new capabilities.

In this book, we will not just look at the Bootstrap changes, but also cover the mindset that makes us think mobile and go through an efficient multidevice development. We will develop a sample short message app that will be called Cochichous, which users can check the nearest messages sent by other users using Twitter API. We will explore HTML5 and JavaScript capabilities, as well the grid and the Bootstrap plugins. Besides that, we will also discover what has changed with this new important release.

What this book covers

Chapter 1, Bootstrap 3.0 is Mobile First, introduces you to the Mobile First development, and makes you understand why Bootstrap was redesigned to this new approach and the reasons we should take care of. We will get a little vision about the responsive design and how Mobile First works with it. Then, we will get in contact with Bootstrap documents to start checking the new documentation and make tests to familiarize ourselves with a Mobile First website.

Chapter 2, Designing Stylesheet in Bootstrap 3, gets us started with the CSS structure and the main grid changes, as well as showing us an example on how to make decisions about breakpoints using Bootstrap grid classes. We will get an overview of how the navigation deals with Mobile First, and how to get forms optimized to different devices.

Chapter 3, JavaScript, the Behavior in Mobile First Development, will lay the foundation of stylesheets, and we will get in touch with JavaScript. In Bootstrap, this is represented by JavaScript plugins using jQuery that's almost intact in this version. It's a great opportunity to learn how to adapt the Bootstrap JavaScript plugins to work with a new device's capability and how to explore its pattern to build your own plugins. There's a clear explanation about progressive enhancement that we probably have already heard before, but now we are focused on applying Mobile First techniques.

Chapter 4, Getting it All Together – a Simple Twitter App, will get all the pieces already learned to make a real Mobile First sample and a simple product to allow us to have an overall idea of how to develop a Mobile First new product from scratch. We start to deal with the capability offered by the mobile browser to deliver a better experience on mobiles for our users. Then, we get all the points that surround the decision about a web app or a native app.

Chapter 5, Performance Matters, will cover what needs to be done to have our mobile experience optimized, how this optimization affects the accuracy still present in the desktop, and how the Mobile First development can be a powerful tool to make our website faster. We will learn three main techniques: optimizing images, loading components on demand, and the use of fonts to render icons.

What you will need for this book

You will just need a computer and an Internet connection. Use your favorite development tool that you are already using to develop your HTML/CSS/JavaScript layers from your application. It would be good to have a mobile phone to test your projects and feel how your site works on a mobile.

Who this book is for

This book assumes that you are already familiar with Bootstrap to understand the huge differences in this third version. But, if you're discovering Bootstrap right now, don't worry, just get into the documents and use the book as a guide to use Mobile First development with Bootstrap. If you are already familiar with mobile web app development, this book is helpful in allowing you to use this knowledge with Bootstrap as a frontend framework for your Mobile First needs. Bootstrap is for everyone, so if you are familiar with frontend technologies such as JavaScript, HTML, and CSS, you will have a chance to customize and use it as a strong frontend tool or even use Bootstrap as the style guide for your projects in your company. But, if you are not familiar with frontend, for example, if you are a designer, Bootstrap could be helpful, and you just need notions of HTML to use Bootstrap for your websites. This is possible just with the lessons learned from the book and the Bootstrap documentation.

Conventions

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

Code words in text are shown as follows: "The following lines of code apply a conditional statement to the `$(window)` object that checks its width against a value greater than 960 and then makes a loop in each image to change the attribute source accordingly with this condition."


A block of code is set as follows:


```
if($(window).width() > 960) {  
    $('img').each(function(){  
        $(this).attr('src', $(this).data('desktop'));  
    });  
}
```

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

```
grunt
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "In your account that you just created, go to your account menu and click on **Create a new application**".

 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 may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via 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 on 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 for all Packt books you have purchased from your account at <http://www.packtpub.com>. 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.

Web page for the book

There is a web page for the example chapter at <http://cochichous.herokuapp.com> with all the code used in *Chapter 4, Getting all together – a Simple Twitter App* so you can explore the whole example.

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 would 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 on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright 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

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Bootstrap 3.0 is Mobile First

Twitter Bootstrap framework's upcoming version is Mobile First. This milestone is not just technical, it's strategic. It follows the current paradigm of design for the Web. It's a design for the future.

But why Mobile First? Why did Bootstrap completely change its course from Desktop First to Mobile First to get into this new way to develop more suitable websites and web applications? Why did the most popular frontend framework embrace this change at a time when responsive web design is continuously growing with better suited and standard techniques such as media-queries, fluid layout, and JavaScript on demand?

Mobile browsers are increasing support for the brand new HTML5 and CSS3, with the philosophy to offer, for older browsers, a less stylized but fully functional component, and for capable browsers a rich and full experience that comes from mobiles to larger screens such as TVs.

For older browsers (such as IE 8 and IE 9), Bootstrap has functional support, but enhanced features such as rounded corners and a placeholder attribute for tips in input fields are not supported for these browsers. To see the full details on browser support, check the Bootstrap documentation from the **Getting started** section (<http://getbootstrap.com/getting-started/#browsers>).

We are living at a time when mobile use is increasing at a pace that will soon surpass desktop usage (<http://www.businessinsider.com/mobile-will-eclipse-desktop-by-2014-2012-6>). Apart from the statistics, one thing we can presume is that the web scenario is changing so fast that we have to embrace the certainty of devices getting better and smarter.

In this chapter, we will explore the main changes in Bootstrap 3. If you are already familiar with Bootstrap 2, check the migration guide (<http://getbootstrap.com/getting-started/#migration>) to have a practical overview about what has changed. If you're not familiar with Bootstrap, there's nothing that's too difficult for you to understand directly from this book about this new version. The only thing you need to have in mind is the Mobile First approach, which is covered well in this book.

You will be guided to design with Mobile First, discover why Mobile First is so important, and how to make Bootstrap a powerful frontend platform to make your site friendly for a wider range of devices.

We can take a step further and add to your previous Bootstrap knowledge by thinking of a concrete way to design processes as a continuous layer of capabilities and embrace the constraints and not fight with them. Mobile First with Bootstrap is an elegant solution for frontend development. Combined with server-side techniques, we get a full bag of solutions to get your product better suited to different users and needs in different platforms.

This chapter will cover the following topics:

- Bootstrap reviewed
- Desktop to responsive
- The new mindset – Mobile First!
- Now Bootstrap uses Bower and Jekyll
- Running the docs

Bootstrap reviewed

In the third era of Bootstrap that is coming, the developers have redesigned the whole framework with a different approach. Let's get started building interface components of small and simple screens, instead of adapting the existent UI components to fit in a constrained environment. From mobile, we will then go to desktop. However, we will not adapt the experience as we usually do with responsive design going from desktop to mobile. Now with Mobile First we will enhance accordingly as we increase the device screens.

Why should I do this if my target audience will be using desktops? Going to mobile indirectly benefits desktop users. But how? To better understand this, let's recap Bootstrap history for a while.

In 2011, Bootstrap was launched to serve as a live and agnostic style guide that was used by Twitter to create their products. It became an open source framework at that time. It was a time when we worked in pixel-perfect layouts and explored CSS3 animations, and we found in Bootstrap a well-documented and standardized set of features.

Bootstrap creates a new design for the browsers because you don't need to define basic interface elements from scratch, such as buttons. At the same time, you have utility elements like badges to cover the most common interface elements. Bootstrap does what a framework is supposed to do: Bootstrapping! The term means the act of taking off a new project; it's like saying, "give me the tools that I will need to start developing my application for different needs". Bootstrap is a toolkit belt with standard conventions from well-defined classes with clean and practical documentation to live code that is ready to use and be customized for your needs. It's not a magic solution to solve the interface element reuse issue, but it's a kick-start. It fits in so many scenarios that developers are increasing its use with their own tools.

"CSS moved beyond type, forms and grids. People get tired to create the same stuffs"

– Mark Otto, one of Bootstrap's creators, in the *Desktop First to Mobile First Bootstrap presentation* (<https://speakerdeck.com/mdo/desktop-first-to-with-bootstrap>)

A must-have from this breeding ground of possibilities is the Bootstrap extension font-awesome (<http://fontawesome.github.io/Font-Awesome/>). It uses font-face, which is widely supported and flexible, instead of sprites for icons. With a single CSS file and font resources used to render the custom fonts, you have a tool that can handle all your icons. This shows the flexibility of Bootstrap tools; for example, font-awesome is independent, works as a standalone project, and is a great fit with Bootstrap.

There are a lot of ways to use Bootstrap. You can customize and extend components, from editing the source code in LESS variables or customize via the Bootstrap download page (<http://getbootstrap.com/customize/>).

At the time of this writing, Bootstrap is the most popular project on Github, so it's just one more reason to consider its importance. There's now an official Bootstrap Expo (<http://expo.getbootstrap.com/>). This is one of the changes in this new version. Bootstrap Expo is the official directory for websites and web applications that are being developed using this framework.

A lot of developers get their first touch with the capabilities of HTML5 and CSS3 with this framework. Bootstrap has amazing capabilities such as offering a responsive grid, dozens of JavaScript components, and a customizer in a web interface or through the LESS variables, if you're an experienced developer. It's suitable for any level of developer and designers because it has solutions that suit both scenarios. This is the second of Bootstrap's main philosophies – it's made for everyone.

Desktop to responsive

With the rise of smart phones, there is a need for responsive content to cover the growing demand. It's possible to add an optional file with media queries and a bunch of CSS code and be adapted to mobile needs.

Media queries, a CSS3 module introduced in June 2012, is a basic structure that gives a namespace with a bunch of CSS rules and declarations according to the user resolution, density, and screen capabilities. So, with CSS files, it is possible to manage the ongoing rise of smartphones. It was possible with just one stylesheet file with good support to adapt according with the device and make a website mobile friendly.

In Bootstrap Version 2, we used to have an optional file (`responsive.less`) that used to have all the media queries necessary for Bootstrap to work well with mobiles.

Another good news is that we can adapt to tablets as a bonus. We have breakpoints for the most common mobile resolutions — this means we have a range of width (768 px to 979 px) that can represent tablet devices. A breakpoint is the extreme point (minimum and/or maximum) where you can define CSS rules specific to that range and change your layout. This could be achieved with a simple declaration of media queries in your CSS:

```
@media (min-width: 768px) and (max-width: 979px) { ... }
```

But sometimes it's indispensable to rethink some elements — some of those already developed only for desktops — in a pixel-perfect scenario. There's no flexibility in a pixel-width accommodation. No matter how much the screen is different, the website will behave like you were using a desktop when we work with fixed units. This is when we can use a bunch of media queries to get more flexible. Even with this solution, redefining dimensions and CSS rules according to the device using media queries will solve screen flexibility issues but not solve performance issues on mobiles.

Performance is one of the main concerns when we go mobile. We have to consider scenarios where the Internet connection is slow and it is a recurrent issue. You will have to perform reverse engineering to make your JavaScript optimize loading, and combine it with server-side solutions. A worse solution would be to just hide content after considering what could be painful for your page load; for example, images have a deep impact on the final performance. Lower page response time is equivalent to more money spent, as we can see in this article about page loading versus user patience (<http://blog.kissmetrics.com/loading-time/>). One of the curious things this research points to is that mobile Internet users expect their browsing experience in phones to be comparable to what they get on their desktops.

We are living at a time when the Web is filled with rich content and we have faster Internet connections. We have to be prepared to offer the closest thing to a fast and optimized loading, at least for our most important content.

This does not involve just the use of CSS to hide content and show content depending on the device, as we can do using media queries. It's all about keeping the concepts simple and focused and developing each interface component thoughtfully from scratch – the primary use, with the constraints and its enhanced capabilities. It's not just about adapting, it's exploring the device's capabilities and delivering the best user experience across platforms.

Sounds familiar? Yes, for sure, the same concept as progressive enhancement, you might think. You're not wrong. Progressive enhancement was a term widely used at a time when we talked about HTML page dependency on JavaScript to be functional. Progressive enhancement is a strategy for web design that relies on semantic markup and technologies such as JavaScript. Nowadays, progressive enhancement is a longer term for Mobile First because it's not just about JavaScript disabled, as it was vastly talked before. A hundred of articles tried to show its benefits in a no JavaScript environment scenarios. Now progressive enhancement is about to be faster (<http://coding.smashingmagazine.com/2013/09/03/progressive-enhancement-is-faster/>).

Progressive enhancement is one of the three keys of Mobile First, together with responsive design and giving priority to content over navigation. So, these three rationales are at the background of all the details of Bootstrap 3, from your CSS components to your grid structure.

The new mindset – Mobile First

There's no way to be fully usable for all the possibilities of human interactions in one place, as well all the possible scenarios that come as NUI (Natural User Interface), such as touch, gestures, and voice and other innovative initiatives to interact.

But we need to adapt to the unknown. The desktop of today will be the smartphone of tomorrow. It doesn't make sense to think of desktops as the key platform anymore. Imagine a technology that has its own limitations but is flexible and can be relied on for different devices.

When we create a new project, we have to think mobile, not just deploy our sites as we usually do and simply ignore the mobile experience. If we decide to ignore this anyway and start to develop a web application for desktops, we will soon discover that mobile users grow considerably, and maybe it's too late to adapt then because our project for desktop becomes complex and any initiative to go mobile is too complex. Maybe it's not too late if we think positive, but it can become unviable to have an existing site going mobile in an efficient and fast way. *Luke Wroblewski*, the co-founder of Input Factory, said in his book *Mobile First*, from the *A List Apart series* (<http://alistapart.com/>), published by *A Book Apart*:

"...if you design for mobile first, you can create agreement up front on what matters most"

Let's presume we want to simply ignore mobile users for a while, and have our **Minimal Viable Product (MVP)** mobile friendly, thinking that mobiles are too expensive. On the other hand, consider that Mobile First is not only a great benefit for the mobile experience, but, it also enhances user experience for desktops. When you design thinking in the lines of constraints, and gradually increase experience as you progress with the philosophy of keeping only those elements that are of essential value, you get a thin, usable, fast, focused way to do simple tasks easily. With a mobile friendly mentality, you can keep things simple and throw away the unnecessary components.

With a lot of navigation options, it takes longer to decide which option to choose to complete a task, and this fact makes users angry and they would probably leave the website. The user's needs for each device is different.

Users are generally distracted while using their phones. They are not as focused when using mobiles as they are when working on a desktop. These distractions are present even at home; for example, talking with someone while checking some stuff on their phones. A faster feedback is necessary.

The last thing they need is a bad and overloaded desktop experience that requires concentration to choose between a whole bunch of navigation options. Remember that the mobile user taps – they don't have a mouse with a pointer. It makes them less precise. We have to consider that the user needs space and proper margins in each device to have better contact with the website and have complete control with no scope for mistakes and frustration. Everything has to be clear with the responsive site flow.

The Web is blurry. So many functions, so many confusions, and navigations and websites getting heavy. The necessity of a lot of information goes against the need for getting information easily. Usability and accessibility become a worry – a key point of successful projects, but still not obligatory. With Mobile First, progressive enhancement, performance, and interaction capabilities to achieve the final result are a priority from scratch.

This is just one of the reasons for unexpected result if your site was developed expecting the best higher-level support and dependence. The foundation is still HTML and CSS as a markup and presentation layer. The JavaScript came a layer above, handling the behavior, and CSS is responsible to control the behavior of animations and transitions. If your user couldn't have a good experience of your site on mobile and mobile was his first contact with your site, they will probably not access from their desktop to check if it's better.

Another point worth mentioning is about the use of screen readers. Its use is increasing. Read a complete report in this WebAIM survey: <http://webaim.org/projects/screenreadersurvey4/>. The usability improvements are in little changes such as specifying a language attribute, being semantic, and describing images. These little efforts are closely coupled with Mobile First.

A good example of how we can get a solution that has great benefits in a wide range of scenarios is a simple anchor link. With a simple anchor link wisely used to skip to a particular bit of content, it can be very exciting for screen reader users looking for that bit of content. This can be very precious for mobile users with little space in a very content-heavy site; for example, he does not want to see menu options but wants to go directly to the content. A simple solution like that can be golden because it's simple, a valid markup, and you can style with CSS.

Thinking Mobile First, we can ride in the growing opportunities generated by the exponential rise in the use of new devices, from little phones to TV browsers. We have a great possibility to also enhance user experience, from environments such as desktops on desks to smartphones in hands. Understand that your user's needs are beyond desktops. The key is to know that thinking Mobile First improves the overall experience for your customers.

Simplifying need be expensive or time consuming. It's all about building all the possible user needs in one platform and not writing the same application twice to get full support. We all remember that Apple iPhone was the most used phone and now we have the Android operational system for mobile phones as a strong competitor (<http://techland.time.com/2013/04/16/ios-vs-android/>). Native apps have their own benefits, but the Web is, even with the troublesome differences between responsive browsers, the most common platform. So, the big networking services such as Facebook and Twitter use the specificity of the native apps and the flexibility of the Web.

Even if you want to explore native capabilities, you can have a browser making part of its strategy. Links from applications point to browsers and applications are fully featured with links that point to some information. If you want to have your place in the market to go mobile with an app, you have to keep track on the Web too. These are not competitors, they coexist together and one solution can make the other stronger as a communication channel.

While native apps have great features and great advantages in performance, mobile web experience has its benefits too.

One of the benefits is its easy access. All smartphones have a decent web browser, even Opera Mini for simple phones. If you do it right, the user gets what he /she wants.

Here is my own personal experience. Once I lost my iPhone (the story of how I lost it deserves another book). I had an old phone and had to use it for a while. I became, for a short while, a potential Opera Mini user, but the only thing I could check were my tweets, and it worked perfectly. Well, I could tweet (that's what a tweeter is supposed to do). You should be able to tweet anywhere. And they made my experience in that phone the best possible – I could log in and then have access to tweets and even tweet about traffic in that little and limited phone, probably faster than my lost smart phone. Yeah, I was a minority and it was temporary, but this is just an example of the big picture: thinking mobiles satisfy users about the service as a whole, not just half-satisfied or satisfied on a specific device. I love Twitter. They are following me in my tough situation. I lost a phone, but I could communicate with an old one. Thanks to Twitter.

Another advantage for mobile browsers is that it is a single and flexible system instead of a specific system for each device. Developing a browser for the iOS and Android is hard because each one has its own language and tomorrow there will be more languages and possibilities. The Web is growing too in a different way alexmagno December 9, 2013 10:58 AM from native applications; the ongoing evolution uses well-known technologies such as HTML, CSS, and JavaScript.

Practical example – The responsive dropdown

The Bootstrap Navbar Component (<http://getbootstrap.com/components/#navbar>) is combined with drop-down menus and is widely used to offer a quick and easy way to give the user an option to navigate to other pages and executing actions.

As shown in the following screenshot, there's a menu to the left that branches into submenus to the right. There's a second option of changing this to a "flip" version, where you navigate through an item to another as if you are jumping between pages. The third solution works for both desktop and mobiles, with collapsed submenus always visible. Thus, we need to think simple in order to avoid creating hacks. It is best to have the optimal solution in both scenarios before even coding and avoid unnecessary workarounds.



Now Bootstrap uses Bower and Jekyll

The big change in this Bootstrap release is Mobile First as the design process. But, there are other relevant changes. From the same place that Bootstrap was born (Twitter), other son came to the world, and its name is Bower (<https://github.com/bower/bower>). The template engine used before, Hogan.js (forked from Mustache) is no more used. Now, in Version 3, Jekyll (<https://github.com/mojombo/jekyll>) is adopted as the template engine. Jekyll is a template language written in Ruby with native support by Github pages, and it makes it a lot easier to get everything up and running. With Jekyll, is not necessary to compile the `mustache` templates to HTML, as we had to in the earlier version of Bootstrap.

Running the docs

Now, it's time to see Bootstrap V3 up and running and feel the changes for real. Explore the docs, as it is one of the main features of Bootstrap. You get all frontend snippets ready for use in your applications. The official documentation and release is available at <http://getbootstrap.com/>. But in this section, you will locally run your checkout version from Github repository.

Version 3 in progress in the Github repository

Execute a Git checkout (for a full Git reference, visit <http://git-scm.com/>) in your console to get a clone of Bootstrap in your machine, as follows:

```
git clone git://github.com/twbs/bootstrap.git
```

Now, you can install Jekyll and run the server to get the documentation live in your browser.

Installing Jekyll

Jekyll is a template framework that is very well accepted and widely used for blogs, for users that prefer code with known markdown languages. It is a way to create simple and static pages without the use of databases, just Ruby!

Jekyll has configuration options and a set of nice features for templates, but its use in Bootstrap is very simple; just reuse snippets of HTML, like layout templates. The main feature of Jekyll is that it is recognized as a template language for Github pages. By using Jekyll we don't need to compile to HTML before generating pages, because the Github hosting pages does this for us. So, if you use Jekyll and Github in a certain way, you have a hosted free website.

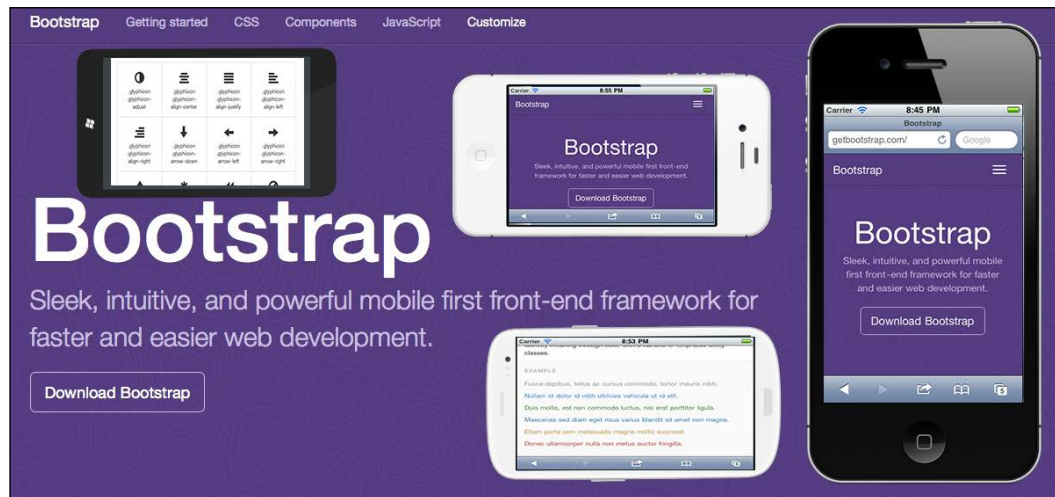
To install Jekyll, you must have Ruby installed. To get a full explanation, visit <https://github.com/mojombo/jekyll>.

After Jekyll is installed, you must execute:

```
jekyll serve
```

That's it. You now probably have your docs running at <http://localhost:9001>.

You should see the following screenshot on your desktop or other devices. Enjoy the links, view it in any device, and explore the documentation as you go:



Bower

Bower (<https://github.com/bower/bower>) is a package manager for frontend frameworks that is developed in Node.js to provide good management with client-side libraries. It's a complete API and works in a way that is very similar to NPM, the package manager for Node.js modules.

To install Bower, you have to execute the following command:

```
npm install bower
```

You can install Bower globally too, with the following command:

```
npm install bower -g
```

Basically, you should install NPM globally only if necessary, but not if it is not needed. Full details are available at NPM's website (<https://npmjs.org/>). Bower is a frontend package manager, so it makes sense for it to be global. To check the differences, visit the NPM blog (<http://blog.nodejs.org/2011/03/23/npm-1-0-global-vs-local-installation/>).

With Bower installed, you can install Bootstrap using the following:

```
bower install bootstrap
```

First step to responsiveness

There's a meta tag that you should include in the head of your document that tells the browser to give a responsive experience using your media queries to rule the layout:

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

This meta tag sets the content to the device width. It makes it mobile friendly, not a desktop site on a mobile. But, you have to define the right media queries. The width parameter inside the content attribute means that the viewport will be adapted and optimized for your device, and the initial scale is the applied zoom. But, there's an interesting discussion about its use at <http://stackoverflow.com/questions/14775195/is-the-viewport-meta-tag-really-necessary>.

So, we use this declaration:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=5.0">
```

It makes the user able to zoom in five times the initial scale.



You can see this meta tag in an example that you could use to start a sample template from Bootstrap (<http://getbootstrap.com/examples/starter-template/>).

Making changes in the Bootstrap source code

Now, the command line common tasks executed in Bootstrap with Make are now working with Grunt (<http://gruntjs.com/>). You have to just use the same commands that you did with Makefile and do it in Grunt. Makefile is a script from Unix platforms that automatically compiles source files and it was used to execute Bootstrap commands in Version 2. The problem comes in the fact that it is only Unix based. With Grunt, which is JavaScript based, with Node installed, you can have the common commands executed in other platforms such as Windows.

Before starting, we can install the necessary dependencies if needed:

- Install Node (<http://nodejs.org/>)
- Install all the dependencies (available in `package.json`, so you have to be in the root `bootstrap` directory)

```
npm install
```

- The following is a command to generate a build via Grunt:

```
grunt
```

This command checks if the following aspects from your Bootstrap work copy are fully functional:

- Running JSHint on JavaScript
- Compiling LESS with recess
- Compiling template
- Compiling and minifying JavaScript

Running tests

If you are a JavaScript expert, you can develop new JavaScript plugins with jQuery, as well as fix JavaScript bugs. For this purpose, you need a test to reproduce the bug to send a pull request for Bootstrap. To check if your JavaScript tests break the framework, you need to execute the following command to run the JavaScript tests:

```
grunt test
```

Summary

This chapter makes you rethink and understand better the concepts behind Mobile First and why Bootstrap was motivated to be developed with this new paradigm. The Mobile First concepts change the design, code, and solutions. It's a new challenge for us. You learned that Bootstrap makes this challenge easier for us by adapting all the components and the philosophy for mobiles. Thinking mobile is very important to make us better use Bootstrap 3.

You already know some basics of Bootstrap's new structure. Jekyll and Bower are new technologies introduced with Version 3. We need to be at least a little familiar with them to go ahead.

You got in this chapter a simple and running tutorial to open the Bootstrap 3 docs, which is a valuable reference kit. One of the key aspects of Bootstrap is live documentation. It's a live style guide and all design elements are meta-documented. With this knowledge, you will be able to plan a website and explore the powerful features, starting from mobiles and finishing with any device that can browse the Internet.

Now, we are ready to get the CSS components redesigned for Mobile First and learn how they work with the new grid system, forms, and units and the issues that surround stylesheet development for mobiles.

2

Designing Stylesheet in Bootstrap 3

What's new in CSS implementation of this new Bootstrap version? One direct answer: responsiveness is not optional anymore. Responsiveness is now included in a single Bootstrap CSS and aims to be mobile friendly from the start.

In this chapter, we will discover the new features in the Bootstrap stylesheet that bring us a powerful choice to stylize the content in a completely reliable way for the device. With the new grid system, it's possible to define how the resolution will behave at different breakpoints and allows us to think about Mobile First from the scratch. We will discover flexible ways to work with semantic grids to explore the forms attached to the grid system, and make different form layouts adapt easily to take form usability a step further. Also, we will discover some tricks about the issue with relative units and how to use navigation components for friendly mobile navigation.

This chapter will cover the following topics:

- The grid system
- Forms in different resolutions
- The icon library
- Responsive utilities
- Relative units
- Navigation

The grid system

The Bootstrap Mobile First grid system meets all of the developers demands for flexibility. It's not just a simple grid system, as we knew in Bootstrap 2. It's more flexible, adapted to the mobile mindset with full control of all the responsive flows. This means Bootstrap 3 is an enhancement from the best of the existing grid framework, with full new support to the semantic grid system.

Semantic grids

One of the main drawbacks in Bootstrap 2 is the fact that there's no nice way to make the grid become semantic efficiently. This means that you have to stylize the grid classes into HTML in the following manner:

```
<div class="grid_x"></div>
```

In a semantic grid, you can use declarative markups to transform the grid system as we want, using the LESS mixins to define rows, columns, and grid elements, as shown in the following code:

```
footer { .column(12); }
```

LESS is a preprocessed CSS language stylesheet that makes use of variables and mixins for developing dynamic CSS. Mixins are a bunch of CSS declarations that can be used as stylesheet functions.

Before getting into the semantic grid system in Bootstrap 3, you should know a little about LESS and the power of mixins. In the following example, we use a basic mixin to make a `border-radius` property work in Firefox and Chrome for you to get familiar with a real mixin (first part is the LESS mixin, the second part is the LESS CSS, and the third part is vanilla CSS):

```
//mixin
.rounded {
  border-radius: 5px;
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px;
}
//CSS
#menu {
  color: gray;
  .rounded;
}
```

```
//Output
.rounded {
  border-radius: 5px;
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px;
}
```

Now, with the LESS mixins in mind, we can imagine a grid system generated by mixins. A great example of semantic grid, is Semantic.gs (<http://semantic.gs/>). You can achieve a semantic grid system from other frameworks too, such as Susy (<http://susy.oddbird.net/>).

With the following HTML (with Semantic.gs):

```
<header>...</header>
<article>...</article>
<aside>...</aside>
```

We can have the related CSS (with LESS):

```
//variables
@column-width: 60;
@gutter-width: 20;
@columns: 12;
//Including mixins
header { .column(12); }
article { .column(9); }
aside { .column(3); }
```

The preceding CSS gives us the freedom to control the responsive flow using the LESS mixins to construct the grid structure. All grid placements, are CSS responsibility. Because we use CSS to define grid placements and not a class in HTML to tell how many spaces the column will take, we put everything in its place without inserting extra markups and grid styles into CSS. There's a great advantage in using this for complex grids and not becoming a slave to the markup full of grid classes, because you're not limited to declaring grid classes that tell the dimension. Using a semantic grid, we have decoupled the layout from HTML and moved to CSS.

The semantic Bootstrap version from the same grid could be rewritten as follows:

```
.wrapper {
  .make-row();
}
.header {
  .make-sx-column(12);
}
```

```
.article {  
  .make-sx-column(9);  
}  
.aside {  
  .make-sx-column(3);  
}
```

The grid framework

In contrast to the semantic grid system, the grid framework is the common grid we have already known in Bootstrap that use classes in HTML to determine grid spaces.

The semantic grid is an extracted mixin of the grid framework. To simplify this, we could use:

```
grid_2 {  
  .make-column(2);  
}
```

Downloading the example code



You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. 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 can do this repeatedly to create an entire custom grid system. In fact, the semantic grid mixin from the Bootstrap core builds the Bootstrap grid framework.

Even with this flexibility, the Bootstrap is a symmetric grid. This means, it follows the `@column-width`, `@gutter-width` and `@columns` variables for building the whole grid with an equal distribution based on these variables. There are other unlimited ratios that are not symmetrical. These are called asymmetric grids. An asymmetric grid is useful for more complex needs.

We have unlimited grid tools; one of these is Singularity, which explores the power of an asymmetric grid (<http://scottkellum.com/2013/04/05/singularity-a-modern-grid-framework.html>). Singularity is a grid system rather than a grid framework. It has more robustness for exploring different ratios in custom grids. Bootstrap 3 is not just a grid system because the grid component is just one part of an entire framework. To see how it works, check the semantic grid examples in the Bootstrap documentation (<http://getbootstrap.com/css/#grid-less>).

The following table shows the main differences between the grid options in Bootstrap 3:

	Grid frameworks	Semantic grid	Asymmetric grids
Pros	<ul style="list-style-type: none"> • Simple to use • Predefined grid width as per CSS Classes 	<ul style="list-style-type: none"> • No extra markup • Grid dimensions decoupled from the markup 	<ul style="list-style-type: none"> • Organic grid organization • Does not follow fixed-ratio grid dimensions
Cons	<ul style="list-style-type: none"> • Has extra markup • Row width defined in HTML 	<ul style="list-style-type: none"> • Defining the width of every grid component is mandatory 	<ul style="list-style-type: none"> • Complex to use, more LESS variables

Breakpoints and completely fluid layouts

Breakpoints set a range of common resolution widths used for making significant changes in the layout. Bootstrap has a naming convention for this: phone, tablet, and desktop.

Sometimes, there are more breakpoint cases and matching criteria than the three main ones. So we have two scenarios: a pixel layout using breakpoints or a completely fluid layout.

Now, Bootstrap has a flat-grid framework that allows us to create more complex mobile grids; you can make use of the grid offsets, as we did for desktop in Bootstrap's previous versions.

Predefined classes to control responsive flow

The grid in Bootstrap 3 now has a unique class for defining a grid. There are no `row-fluid` classes anymore to say that the grid should behave as fluid in our stylesheet. We can use declarative classes to define the orientation for many breakpoints.

In Bootstrap 2, we would do the following for enabling a fluid layout:

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

In Bootstrap 3, we would do the following:

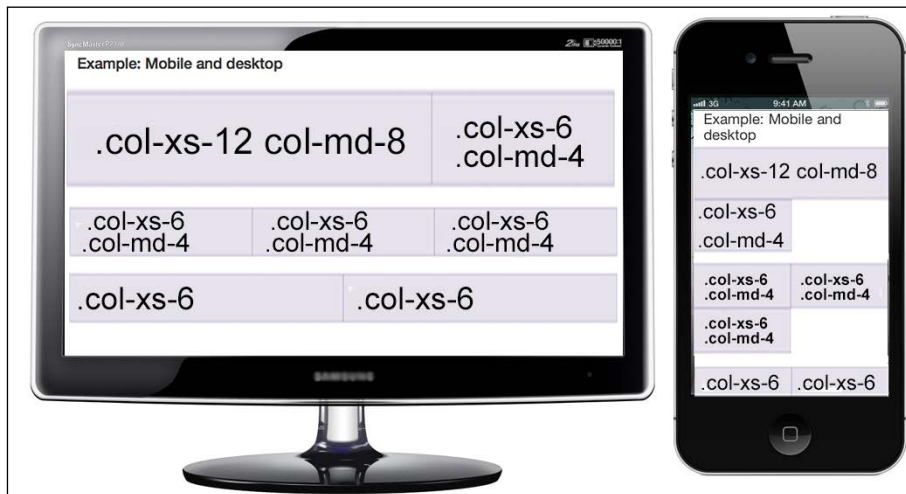
```
<div class="container">
  <div class="row">
    <div class="col-md-4"> ... </div>
  </div>
</div>
```

Now, in Version 3, there are grid classes that describe the screen resolution's ranges. This is an intuitive way for enabling us to differentiate between device width and its respective breakpoints.

In the mobile-stacked, one-column version, we already define how Mobile First will behave when a new breakpoint is reached, for example, from a 320-pixel phone to a 768-pixel tablet.

The default column class is `.col-xs-n`, where `n` is a number between 1 and 12, for extra small screens (phones below 768 px). It has 12 columns for mobile, and is prefixed with `lg`, for instance `.col-lg-n` for larger devices, where `n` is a number between 1 and 12, and we have `.col-sm-x` for tablets that work following the same rationale.

We have the setup, as shown in the following figure. We can compare the grid size of two devices and check out the behavior in wide and small resolutions:



As we can see in the previous figure, `.col-xs-12 .col-md-8` classes indicate 12 columns in mobile, and 8 columns in medium-screen devices such as desktops.

Then, `.col-xs-6 .col-md-4` means that extra small screens (xs) are 6-columns wide and medium screens, such as desktops, are 4-columns wide.

The last column is divided into two columns with the same width: 6 for right, 6 for left, both for mobiles as well as desktops. The extra small screen is the default one, so it will behave in the same way for desktops.

We can have the following variations:

Classes (* is a number between 1 and 12)	Devices
<code>.col-xs-*</code>	Below 768 px
<code>.col-sm-*</code>	Between 768 px and 992 px
<code>.col-md-*</code>	Between 992 px and 1200 px
<code>.col-lg-*</code>	Up to 1200 px

You can refer to the Bootstrap *Grid options* section in the documentation at <http://getbootstrap.com/css/#grid-options> for more details.

The Flex grid, a new draft specification for CSS level 3, works in a similar manner, because it uses CSS rules to define breakpoint changes (<http://www.w3.org/TR/css3-flexbox/>). This new box model can optimize the user interface design experience and make it possible for us to really build layouts with CSS in two dimensions. We define how the flex orientation should be, as the Bootstrap 3 does using the preceding classes.

Forms in different resolutions

The form layout still resides in the same grid naming conventions. With the power of Mobile First, you can have forms optimized to match the user's best experience of filling forms.

In the old Bootstrap versions, you use the grid classes for full control on form placement. However, the best practice is to not use classes in the form elements dedicated to make an input column work as a grid. You can wrap the input into a grid container and get its space. By default, the inputs keep your width as 100 percent.

In a form design, we do not need to specify the width in pixels. It seems easier to specify this at the beginning, but it's just a matter of time before you lose the flexibility and feel obligated to design each input size. There's a more convenient way to do this: relative and named sizes.

With named sizes, such as `small`, `medium`, and `large`, granulated names such as `big` and `bigger` are better for defining the input sizes.

This convention gives us the power to develop different form designs, without messy layouts in different resolutions, because it is already handled by Bootstrap to behave accordingly in different devices.

If we try to have a fixed width in inputs, we may get into trouble. As the window resizes, the input will keep that size. We may lose the power of the grid framework and also lose a great opportunity to follow the Bootstrap framework to have a flexible grid with little effort. It's really a good practice to use your inputs inside a grid container and leave the control and command with the grid.

We can't forget to mention that using forms semantically is a wise decision for avoiding unnecessary code. With the simple usage of the input in HTML 5 type attributes, we have an easier method for collecting the best user experience for each device. For example, a date-input format can make the date you select in a smart phone look the same as used in apps. Input with the `file` type lets you attach a picture easily and it even lets you use your smart phone camera to take a picture instantly and use for your website.

The icon library

Bootstrap 2 usually uses images to render a basic set of icons as sprites. However, it takes time to load and sometimes cannot be essential thinking in terms of performance for a website page. We always have to import an extra image and sometimes lose flexibility with it when we need optimized sizes in accordance with the device.

In different dimensions, we have to deal with different optimized image sizes, which are optimized for a given device; this results in more image files, extra storage space, more dependencies, and so on.

Glyphicons (<http://glyphicons.getbootstrap.com/>) is an icon library for Bootstrap; part of its core responsibilities is to manage and give utility classes for defining an icon:

```
<i class="icon-camera"></i>
```

This library uses font-face to generate icons instead of sprites, so as an isolated icon library, Glyphicon is getting better and now adapts to flexibility. One of the greatest benefits of this, is that it now uses font-face to font rendering and lets us size the icon according to the font size as well as use colors, animations, and CSS effects, such as shadow. This is a great method to handle icons, but you can't forget that fonts as icons have limitations and we can use them only for monochromatic icons.

Responsive utilities

Bootstrap now has a full set of mixins used for helping you improve the responsive experience. It's time to explore semantic grid mixins and other responsive utilities.

Responsive classes

There are responsive utility classes for making mobile development easier and friendly. With visible and hidden classes, combined with large, small, and medium devices, we have a complete and powerful way to control our device experience to sometimes not display some information in each specific device, as shown in the following table:

Classes	Devices
<code>.visible-sm</code>	Small (up to 768 px) visible
<code>.visible-md</code>	Medium (768 px to 991 px) visible
<code>.visible-lg</code>	Larger (992 px and above) visible
<code>.hidden-sm</code>	Small (up to 768 px) hidden
<code>.hidden-md</code>	Medium (768 px to 991 px) hidden
<code>.hidden-lg</code>	Larger (992 px and above) hidden

Semantic grid variables and functions

Now, we can use a flexible way for defining our grid in CSS (with LESS):

```
.make-row(@gutter: @grid-gutter-width);  
  
.make-sm-column(@columns; @gutter: @grid-gutter-width);  
  
.make-sm-column-offset(@columns);  
  
.make-sm-column-push(@columns);
```

The same applies for the `md` and `xs` prefixes; so, basically, we have the following formula for the semantic grid functions:

```
.make-type-column(@columns; @gutter: @grid-gutter-width);  
  
.make-type-column-offset(@columns);  
  
.make-type-column-push(@columns);
```

The type can be `xs`, `sm`, `md`, and `lg` (extra small, small screen, medium, and large).

With these semantic grid mixins, you can create a grid structure. Use `.make-row()` to create a row to be nested with a composition of column layouts that have a single parameter for defining width. Use `.make-column` to create a column grid that sums up to 12. You can use `.make-type-offset` to give an offset, like we usually do in Bootstrap, as well as while pushing and pulling columns.

Relative units

Which unit of measure should we adapt for our mobile needs? Do we really not need the EM unit anymore? When IE6 did not support font sizing, EM was the only way for font sizing in IE without using JavaScript. For Mobile First, the EM unit has its own advantages.

We need a font unit that allows us to maintain the aspect ratio, as per the screen size. While considering the adequate font unit and while comparing it to the grid, the fonts should follow the grid in a certain ratio. Instead of defining a base font-size in each media-query breakpoint and for each element, redefine the font-sizes. It's better to have a proportional and relative unit ratio, and this can be achieved with an EM or percentage unit.

In this new version, the Bootstrap milestone has a lot of discussions about using the EM unit over pixels, but the limitations with its use don't justify the benefits. They still support IE8 with a pixel fallback, and this approach will generate duplicate code. Nesting the EM unit is historically a headache and a cross browser issue, and sometimes, we need extra math to compute the pixel values.

We can use a mixin to convert the font units, but that's not an ideal solution, and Bootstrap will probably change it in future releases.

Pixel values are still used for font sizing and are controlled by `@font-size-base`, `@font-size-large`, `@font-size-small` that hold fonts for the default size of large screens and small screens respectively.

Navigation

Now, with responsive navigation we don't have more optional CSS files to make your menu fully usable.

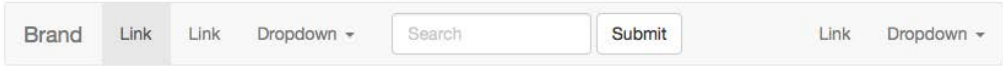
The menu now shrinks to a completely adapted menu for all user needs. The famous technique of transforming a full menu into a mobile menu experience is free of any setup. Just add the `navbar` and `navbar-default` classes, and you get a main navigation bar for your website.

The navigation bar uses the JavaScript Collapse plugin (<http://getbootstrap.com/javascript/#collapse>) to have a default responsive navigation bar using just links and lists. Please check out the Bootstrap documentation (<http://getbootstrap.com/components/#navbar>) to see more details about this component.

We can use simple classes such as `navbar` and `navbar-inverse` to have a fully clean experience in different contrasts, so we have `navbar` with a white background and the inverse class when we wish to have `navbar` with a black background.

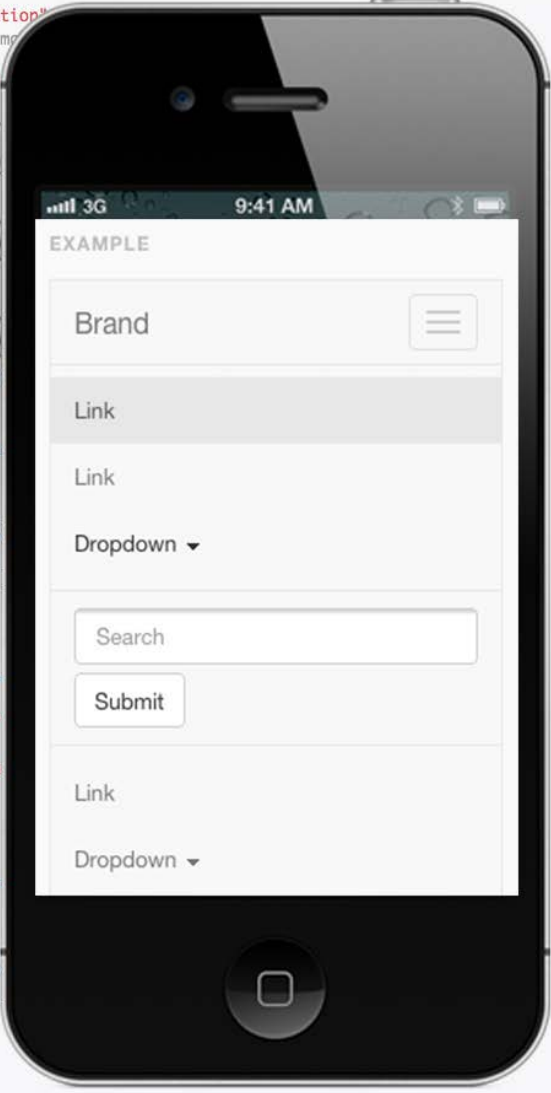
From a Mobile First point of view, we start with a menu that can be toggled and fully collapsed, which becomes a horizontal menu as the viewport increases. This is shown in the following figure:

EXAMPLE



```
<nav class="navbar navbar-default" role="navigation">
  <!-- Brand and toggle get grouped for better mobile support
  -->
  <div class="navbar-header">
    <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#navbar-collapse">
      <span class="sr-only">Toggle navigation</span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="#">Brand</a>
  </div>

  <!-- Collect the nav links, forms, and other
  -->
  <div class="collapse navbar-collapse">
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
      <li class="dropdown">
        <a href="#" class="dropdown-toggle" data-toggle="dropdown">
          <ul class="dropdown-menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li><a href="#">Separated link</a></li>
            <li><a href="#">One more separated link</a></li>
          </ul>
        </a>
      </li>
    </ul>
    <form class="navbar-form navbar-left" role="search">
      <div class="form-group">
        <input type="text" class="form-control">
      </div>
      <button type="submit" class="btn btn-default">Submit</button>
    </form>
    <ul class="nav navbar-nav navbar-right">
      <li><a href="#">Link</a></li>
      <li class="dropdown">
        <a href="#" class="dropdown-toggle" data-toggle="dropdown">
          <ul class="dropdown-menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li><a href="#">Separated link</a></li>
          </ul>
        </a>
      </li>
    </ul>
  </div><!-- /.navbar-collapse -->
</nav>
```



Summary

As we could see in this chapter, Bootstrap CSS got cleaner and more powerful in Version 3. We took advantage of Mobile First to get the most well-designed style. With Mobile First in mind, it becomes simple to progress and design our application using the Bootstrap grid and utility classes. Bootstrap was revisited and the Mobile First CSS gave us many approaches for different application needs for mobiles.

You learned different grid conventions adapted for Bootstrap, and got introduced to the semantic grid, and saw how it works with the use of this flexible tool to make Bootstrap grid more convenient for our mobile needs.

We saw how the grid can be a great auxiliary to form design and how powerful it is to keep our form semantic.

We got in touch with the issues with relative units. We saw that Bootstrap tries to explore the EM unit but still has a counterpart that overlaps its benefits, so Bootstrap still has to deal with pixels and use it to define font sizing.

We saw a fully responsive menu, widely used in desktop and mobile applications with a simple snippet of code, and the use of a thin JavaScript for offering simple cross-platform navigation without any tricks.

In the next chapter, we will explore the JavaScript in Mobile First as a behavior layer for the lessons learned from the Bootstrap CSS.

3

JavaScript, the Behavior in Mobile First Development

JavaScript in Mobile First Development explores the best of the APIs offered by the browser, but there are different browsers and so many devices. There are a lot of APIs to explore the device capability.

In Bootstrap 3, the JavaScript jQuery plugins for Bootstrap have fixed a lot of bugs. One of the biggest changes was the addition of namespace events to provide a no-conflict environment for Bootstrap JavaScript plugins.

In this chapter, you will learn how to enhance the behavior of your mobile-to-desktop experience. Get the best optimized JavaScript to achieve the right direction to your web application. Let's get started with Bootstrap JavaScript!

Bootstrap, as a frontend framework, takes JavaScript to the server using Node.js and Grunt, which is a powerful tool to manage common JavaScript and CSS tasks, such as running tests and minifying JavaScript files.

This chapter will cover the following topics:

- The carousel example
- Data attributes
- Mobile First and progressive enhancements
- Namespace events
- JavaScript on the server with Node.js tools



There's no deep change in the JavaScript structure in Bootstrap 3, so this chapter will cover the good practices to work with JavaScript applied on a Mobile First project with Bootstrap.

The carousel example

In all Bootstrap JavaScript plugins, the JS works as an enhancement layer and is often available with an equivalent CSS component improved by JavaScript. One good example is the dropdown CSS component (<http://getbootstrap.com/components/#dropdowns>) and a jQuery plugin (<http://getbootstrap.com/javascript/#dropdowns>).

Another example is about CSS transition. If one specific browser supports CSS transition, then make animation through CSS using transition as shown in the following example:

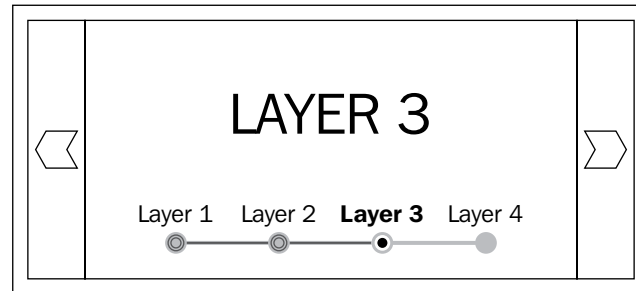
```
if($.support.transition){ that.$element.addClass('fade in') }
```

The preceding example is recurrent in the Bootstrap JavaScript plugin's code, and it checks if the browser supports transition. If the condition is true in the `if` statement, then add a CSS class that fades the element through animation of the `opacity` property:

```
.fade {  
  opacity: 0;  
  .transition(opacity .15s linear);  
  &.in {  
    opacity: 1;  
  }  
}
```

For better mobile experience with device events, we can use frameworks such as jQuery mobile (<http://jquerymobile.com/>), because mobile events is not a part of the Bootstrap core.

When the Bootstrap carousel plugin (<http://getbootstrap.com/javascript/#carousel>), was redesigned, the navigation became larger, and increased the contact area for touch devices. This is one of the important principles of design for touch-based devices.



As we can see, some JavaScript components were redesigned to better achieve mobile needs. Some of these improvements satisfy desktop users too. It's always the best choice to plan the overall experience at first before thinking in device-specific interactions.

A touch of enhancement

Now that we have a well-designed Carousel, we can give our enhancement. At first, we will spend all our efforts to provide a good overall experience. Then, we will start to think about every specific feature of a device and explore its compatibilities. Keep this flow in mind to make design decisions for your Mobile First project.

We will use jQuery mobile to enable touch events to make our carousel navigation available by touchable interactions. You can download this library from the jQuery mobile website, (<http://jquerymobile.com/download-builder/>), with a custom download (Alpha version) you will get the events and components that you need. In our case, it just enhances the carousel with swipe events; we should just use swipe events in the custom download options.

Include the reference for the current file that you downloaded and insert the following code in the head of your main HTML file:

```
<script src="jquery.mobile.custom.min.js"></script>
```

We can link it directly from jQuery CDN too. In this case, the file is hosted on jQuery servers, and you can take advantage of optimized loading if your visitor already has a downloaded copy of jQuery from the same CDN. In this case, there's no need to download the file again on the user computer:

```
<script src="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.js"></script>
```

Now, there are swipe events that can easily link the next and previous button methods of the Carousel plugin in the following code:

```
$(document).ready(function() {
  $('#layers').on('swiperight', function() {
    $(this).carousel('prev');
  }).on('swipeleft', function() {
    $(this).carousel('next');
  });
});
```

We attach the event in the `layers` div, located in the following Carousel markup (<http://getbootstrap.com/javascript/#carousel>):

```
<div id="layers" class="carousel slide">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#carousel-example-generic" data-slide-to="0"
class="active"></li>
    <li data-target="#carousel-example-generic" data-slide-to="1"></
li>
    <li data-target="#carousel-example-generic" data-slide-to="2"></
li>
  </ol>

  <!-- Wrapper for slides -->
  <div class="carousel-inner">
    <div class="item active">
      
      <div class="carousel-caption">
        ...
      </div>
```

```

    </div>
    ...
</div>

<!-- Controls -->
<a class="left carousel-control" href="#carousel-example-generic"
data-slide="prev">
  <span class="icon-prev"></span>
</a>
<a class="right carousel-control" href="#carousel-example-generic"
data-slide="next">
  <span class="icon-next"></span>
</a>
</div>

```

The carousel starts with a markup for indicators, and then proceeds to the content and the controls, which hold the markup for the right and next directions.

With jQuery mobile downloaded and placed in our HTML document, we could use the new events `swiperight` and `swipeleft` to call the associated next and previous methods of the carousel to slide for each direction.

With a simple JavaScript, we can enhance the experience on a mobile efficiently. Is it really distinct and optimized? Not yet. We can have a more pragmatic script that checks if the browser supports swipe events before attaching it to the DOM through JavaScript. The following code shows a simple optimized solution:

```

function giveTouch() {
  $('#layers').on('swiperight', function(){
    $(this).carousel('prev');
  }).on('swipeleft', function(){
    $(this).carousel('next');
  });
}

$(document).ready(function(){
  if(window.DocumentTouch && document instanceof DocumentTouch) {
    giveTouch();
  }
});

```

We are delegating enhancement in accordance with device support, wisely using the behavior through feature detection, instead of blurry and specific device detection, as shown in the following example:

```
If($.browser.msie) {  
    //dummy for IE  
}
```

A great library that gave me the chance to participate in its development is the Responsive Hub (<https://github.com/globocom/responsive-hub>). It is used in the following code to check if a touch event is supported. The Responsive Hub is an alternate and simple way to check for event support in mobile devices, as well as for control screen size changes in JavaScript change events. So, we could rewrite the code of the feature detection, as shown in the following code snippet:

```
$(document).ready(function() {  
    if($.responsiveHub("isTouch")) {  
        giveTouch();  
    }  
});
```

It was developed at Globo.com as an open source project to make Globo.tv responsive (<http://globo.tv>). With this library, you can have defined screen resolution as a alias (phone, tablet, web). Then we have events associated through layout properties as screen size and touch verification, as shown in the following code:

```
$.responsiveHub("ready", ["phone", "tablet", "web"], function(event)  
{  
    alert(event.layout); // Current layout  
    alert(event.touch); // supports touch events  
});
```

As we saw in the preceding section, the Responsive Hub acts as an event listener to a ready DOM, and then you can listen for layout changes through the layout object with dimensions and touch event's support.

Using Modernizr, we could have a nice elegant solution too:

```
Modernizr.load({  
    test: Modernizr.touch && Modernizr.csstransitions,  
    yep: 'carousel-swipe.js',  
    complete: function() {  
        buildTouch();  
    }  
});
```

In the preceding example, we test touch and transition that are easily verified by Modernizr methods and then by the Yep (<http://yepnopejs.com/>) library that is part of Modernizr. Using it we can load JavaScript files based on test conditions, as we illustrated in the preceding code.

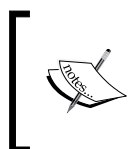
Data attributes

The data attribute is one of the most powerful tools for JavaScript plugins in Bootstrap. You don't need to write any lines of JavaScript to achieve common tasks, which could be in the future browsers.

Do you remember the time when a placeholder for an input element was possible only with JavaScript? Now it is well supported in HTML5. We don't need JavaScript to do this simple behavior anymore, but we can use JS as a fallback. This is what Bootstrap does in plugins with the power of data attributes.

The data attributes are fully supported in Bootstrap 3. You can use all the plugins to initialize them without using JavaScript.

But how is it possible? Well, with data attributes, you attach in your HTML attributes the custom interaction for your element. A good example is the **Bootstrap modal plugin**.



The modal plugin, due to Mobile First development, is now very well optimized for mobiles. It had a lot of issues in Version 2. It has now found a good way to centralize and be flexible accordingly to the device's screen size.

Let's get started with how to define the modal behavior in HTML:

```
<a data-toggle="modal" href="#myModal" class="btn btn-primary btn-large">Launch demo modal</a>
```

With the `data-toggle` attribute, you can specify the behavior of the link in an intuitive way, and use the `href` attribute to target the modal element using an ID. This is an unobtrusive way to build a plugin, for the following reasons:

- The `href` attribute is for complementary information on the page that makes a link work without JavaScript, because it acts as a simple anchor.
- JavaScript is not required to initialize jQuery plugins.
- The data attributes can be browser-native in the future.
- All modal setup can be achieved with HTML attributes.
- It's good for SEO; the crawlers will not miss any content and still link the information.

This is possible because all Bootstrap JavaScript plugins have a `data-api` pattern inside all of them. The following code is a small demonstration of a simple way to have your plugins fully customizable via `data-api`:

```
$(document).on('click.bs.carousel.data-api', '[data-slide], [data-slide-to]', function(e) {  
    // some javascript logic  
});
```

In the preceding example, there's an event being attached in a namespace called by the `click.bs.carousel.data-api` event, which is applicable to all `data-slide` attributes. You can either access variables inside plugins or do the internal implementation without affecting the event scope.

Check out the modal plugin's code to see how the Bootstrap pattern works. There's a Github project that I've created with the core of Bootstrap plugins (<https://github.com/alexanmtz/bootstrap-javascript-pattern>). This pattern is used in all the jQuery plugins from Bootstrap 3.

Mobile First and progressive enhancements

This section describes using JavaScript without having a JavaScript-enabled browser.

Progressive enhancement tries to take the best experience from limited devices to a full experience (that's why it meets the needs of Mobile First). On the other hand, there's another concept, the graceful degradation, which is focused on creating the best experience for an environment with less restrictions, exploring the rich web technologies as JavaScript at first, and then degrading for devices with more constraints.

We will follow some recommendations to achieve progressive enhancement and use them with this version of Bootstrap accurately.

Be semantic in your HTML markup

Always be semantic. We don't want to cheat the browser. We have to use the right element in the right place. It's just doing this to offer a JavaScript-rich experience as an enhancement to the basic HTML structure at first.

For example, consider a link and a button. In Bootstrap documentation, it is recommended that we should use buttons for actions and links to address resources. It seems simple, because that was the purpose, but with JavaScript, we can change the browser's default behavior and get unexpected results.

Check a working code of the following example at <http://jsfiddle.net/5xGGA/>.

The HTML code is as follows:

```
<form action="#" method="get">
  <input type="text" />
  <a class="link" href="#">submit</a>
  <input type="submit" value="submit" />
</form>
```

We could do the same in JavaScript (with jQuery), as follows:

```
$(function() {
  $('.link').on('click', function() {
    $('form').trigger('submit', ['called by link']);
    return false;
  });
  $('form').submit(function(e, whocalls) {
    if(whocalls) {
      alert(whocalls);
    }
    alert('form submit default');
  });
});
```

As we can see, if we hit *Enter* in the form, the event triggered will be the form submit. So, you can use an accessibility and primitive feature of the web, that sends a form with a simple hit on the *Enter* key. In this code, we pass data in the triggering event from the link to a form, and then when we click on the link, the data in the event is passed to the form's submission by taking one more unnecessary step. If we decide to change the submit button to a simple link and break the semantic of a form element and its components, we lose the total behavior of how a real form should work in your case. This is just an example of a lot of ways to confuse, what should be handled by JavaScript.

But how should we do that? With no line of JavaScript, we can do it in a natural way: use form to submit the process of user filled data!

So, in this example (see the fully working code at <http://jsfiddle.net/Uk2aS/1/>), we can see an analogy that keeps JavaScript away when it's possible and just uses it to enhance behavior.

Unobtrusive JavaScript

The unobtrusive JavaScript comes along with semantics and the Mobile First approach. With the unobtrusive JavaScript, we can handle a form submission via Ajax without the JavaScript obstructing the natural flow of form submission. That is what unobtrusive JavaScript is about.

With the same example of the previous HTML form (without the unnecessary link element), I will show how unobtrusive JavaScript works (<http://jsfiddle.net/5xGGA/>).

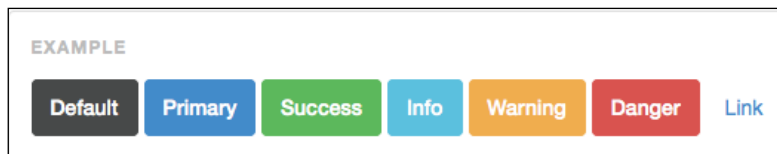
Using the following code, we will create an enhanced loaded form for users having JavaScript-enabled browsers:

```
$('#form').on('submit', function(e) {
    var action = e.target.action;
    var name = $('input[name]').val();
    var $self = $(this);
    $.post(action, {name: name}, function(data, status) {
        if(status=='success') {
            $self.append('<p> success </p>');
        }
    });
    return false;
});
```

At first, we get the form's action attribute from the HTML (captured by `e.target.action`), then the name value of the form's input, and a reference to the form element. With the name field, we could serialize many input fields with jQuery's `$('#form').serialize()`, send the information, and give feedback to the user. At the server side, we can handle the request and give the right response depending on the type of request. This is a simple way to use unobtrusive JavaScript at the application level. This JavaScript doesn't hurt our main form.

Follow the Bootstrap tips about accessibility

If we get the same example with link buttons and forms, we can see that Bootstrap recommends the right use of each element.



The following example shows different button states that represent a label. They're the same elements with a different class:

Well, Bootstrap buttons have predefined states that are well designed for different kinds of button behavior and links. When you want to define an action, we could use buttons, otherwise, if you want to point a resource to a defined URL, use links.

We can still make a button look like a link in the following way:

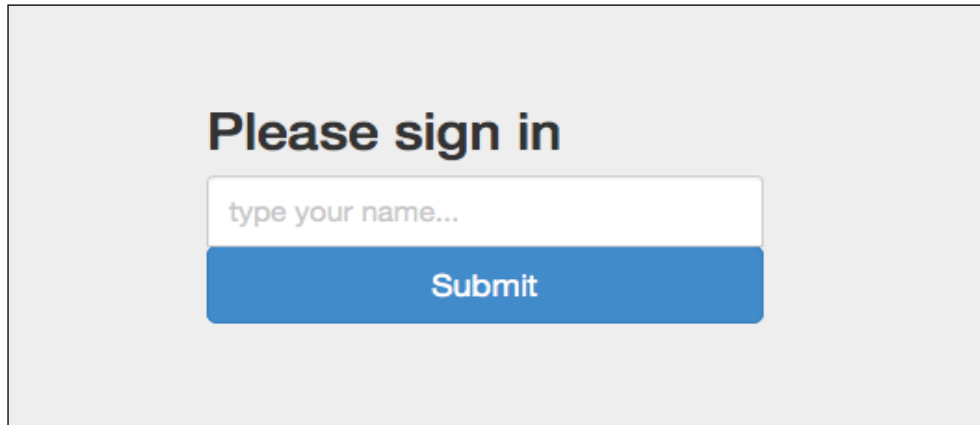
```
<button type="button" class="btn btn-link">Link</a>
```

We can similarly use a button to make a cancel action. It has to behave like one button, but it is a secondary action and could fit well when designed as a link too.

Test a site in a lynx browser

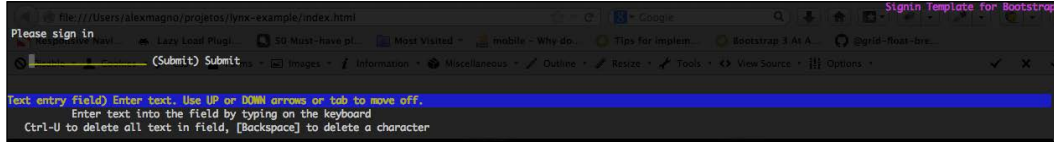
Lynx (<http://lynx.isc.org/>) is one of the first browsers, and it's how a Robot as a Google Crawler sees your site. It's the best tool to check real, progressive enhancement. However, it's the most primitive browser ever, so take care, because it's just a tool to help you make a more accessible website from the beginning.

I will show you a simple Bootstrap template from a simple sign-in form in a lynx browser.



This is the interface in Firefox 24.0

So, we have the site viewed in lynx:



Namespace events

One of the biggest changes in JavaScript, in the Bootstrap framework, is to allow namespace events to prevent JavaScript conflicts with other frameworks, and even have a better control of JavaScript behavior to attach the Bootstrap plugin events.

As we can see, we have an event with a default namespace model for Bootstrap plugins:

```
$.Event('close.bs.alert')
```

This is an example of an alert plugin. As we can see, this is a `close` event in Bootstrap's (`bs`) namespace of an alert plugin.

Another example of an event object used in Bootstrap plugins is as follows:

```
$.Event('show.bs.modal')
```

This is a `show` event in the `bs` namespace of the `modal` plugin.

JavaScript on the server

There are amazing tools to make Bootstrap better using JavaScript on the server. Now, it has the power of Grunt to execute JavaScript tasks. With Grunt, the tasks to build bootstrap packages are now written in pure JavaScript with Node.js. Grunt has a great way to extend with plugins, so you can create any kind of integration with powerful frontend tools such as Coffee Script, Less, Sass, JsHint, and many others.

Summary

As we saw in this chapter, JavaScript is a delicate part of Mobile First development. You have all CSS flexibility to adapt to devices that are up and running, and then JavaScript comes to enhance the web page behavior.

We reviewed some of the JavaScript little changes at Bootstrap 3 as namespace events and the power of data attributes. Then, we focus on the best practices to get a better idea to Bootstrap the JavaScript and have a manageable way to improve behavior accordingly with the device. We get a practical example to achieve this, making a carousel touchable in a simple way. We also explore the main JavaScript concepts of a Mobile First experience, terms used before those are reinforced now: unobtrusive JavaScript and progressive enhancement.

Now, we are prepared to get the little pieces together, HTML/CSS and JavaScript that now will be used to make a full web application working. We will combine the Mobile First design with the Bootstrap knowledge to get our project working cross device. In the next chapter, we will build a practical app to apply our knowledge in the real world about Mobile First with Bootstrap.

4

Getting it All Together – a Simple Twitter App

Now that we have a basement of all Bootstrap changes, let's see in practice how to develop a simple Mobile First web application. We will see from design to development, a lot of tips, tools, process, and code exploring; all the best that Bootstrap can provide. Let's get started with a little project for the Web, but once again in Bootstrap it's just a kick start. We will see a bit more about our project with techniques and discuss ways to make a better experience and use Bootstrap as a powerful framework for your mobile needs.

Here, we will learn to design a Mobile First full project from scratch with the little pieces of the previous chapter. We will deal with a new, really big challenge when we start to get into Mobile First in practice.

This is not a "hello world" example. The most famous example ever is already available in the *Getting started* section at *Bootstrap Docs*. And as we know, documentation is one of the key features of Bootstrap (<http://getbootstrap.com/getting-started/#template>).

We will follow the Mobile First development of Cochichous (<http://cochichous.herokuapp.com/>), a simple application to search tweets and see nearest ones (Cochicho is a Brazilian word that means the action when we talk in others, ears in a low voice, besides that, it's a bird species). It's a prototype from a sample web application to list the search results of tweets in your range.

We will cover the following topics in this chapter:

- Bootstrapping our application
- Bootstrap modal component example
- Geolocation
- Going from tablet device to desktop screen resolution
- The choice between web app and mobile application

Bootstrapping our application

As we know, Mobile First focuses on content. Let's imagine the overall experience and tasks in our sample short message application:

1. Give a search for keywords.
2. See search results from a location nearest to us from the Twitter network. Usually from his neighborhood, or even city. Let's define a location radius of 50 kilometers to search for.

That's it, two things, keeping it simple. We could imagine a lot of things, but to illustrate how the Mobile First process really works, let's keep it in two main functionalities. These two little features will be good as our first challenge. As we saw, Mobile First is content aware, this means that the content defines the layout flow across the devices. Let's focus on content. We will see how amazing it can be.

Inserting a customizable version of Bootstrap

As we saw in *Chapter 1, Bootstrap 3.0 is Mobile First*, if you have a cloned Bootstrap copy in your machine, you can change LESS files. You can generate a custom CSS from it to change common variables too. This includes font size, colors and spacing configuration as a general margin and padding. We will learn how to generate Bootstrap CSS from a LESS file, which is an option to get BT customized. Another option is the custom download page (<http://getbootstrap.com/customize/>).

You can always recompile and modify LESS files in a Bootstrap source file with the grunt command:

```
grunt
```


We are generating assets for Bootstrap to use in our application when we run the grunt command in the shell command line. As the final output, Bootstrap is just CSS and JavaScript generated from Grunt tasks.

Now, we can edit the variables file located at `variables.less` inside the LESS folder in Bootstrap's source root directory.

```
// Brand colors
// -----

@brand-primary:      #428bca;
@brand-success:     #5cb85c;
@brand-warning:     #f0ad4e;
@brand-danger:      #d9534f;
@brand-info:        #5bc0de;
```

This is an example of a portion of LESS variables inside `variables.less` where we can customize all the color pallets and see how well it is organized. We can customize font-sizes, measurements as overall padding and the margin, and finally component colors.

 The LESS variables are quite different in Bootstrap 3. They don't use color reference, as it was in Version 2 such as `main-blue`, `secondary-red`. Now, there are meaningful color names such as `primary`, `success`, `warning`, and others.

If you take a look around (<http://bootstraphero.com/the-big-badass-list-of-twitter-bootstrap-resources>) for Bootstrap tools, you will be surprised. We can see a lot of great Bootstrap tools out there. So, always look for one of those before you get into action.

I discovered an amazing tool called **PaintStrap** (<http://paintstrap.com/>). This tool creates a CSS-customized version from a color palette. That's exactly what we need right now!

It has integration with **Adobe Kuler** (<https://kuler.adobe.com>) and **COLOURlovers** (<http://www.colourlovers.com>). It's a great tool to start defining an identity of your product and customize Bootstrap.

 Until this writing, the PaintStrap style used to generate files from your customized pallet for Bootstrap 2, but probably they will soon update to Bootstrap 3.

With LESS files modified and with the `grunt` command, it will generate a CSS file with two versions: a full (for development needs) and a minified (for production) version. All the files are generated in the `dist` folder located at Bootstrap's root directory. This output is our Bootstrap-customized version ready to be referenced in our template.

If you just need a simple Bootstrap download, you can download plugins and CSS components that you use in your app. Just visit the Customize section (<http://getbootstrap.com/customize/>), and there we can change LESS variables through the web interface, just filling out the parameters in a form. But, if you want to get in to the code to change the LESS variables, there's an option to download the framework from the Bootstrap source at GitHub (<https://github.com/twbs/bootstrap>).

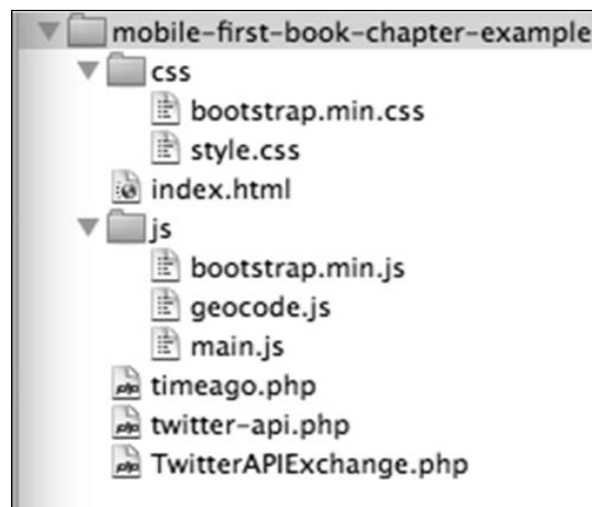
The project template

We have to start with the mobile template. Let's get a basic template to start with.

We should start with the mobile, right? Should we truly test in the mobile phone? You can start with a responsive design view; it's the simplest way. We can at least resize your browser window in development to simulate mobile dimensions. But, there's a tool that you can use to make mobile web development easier. It's just to activate the responsive mode in Firefox (https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_View) to have a preview to simulate your website in different resolutions, instead of resizing the window to simulate mobile dimensions.

Later, you can start to explore tools to simulate devices (<http://vanamco.com/ghostlab/>). With this tool you can test at real time by creating a profile for your devices. After the device profile is set up, you can use the environment created by the app available through a development URL from the Ghostlab server to access your project on the real mobile device. Using it, we can avoid configuring the settings in each phone in order to access the localhost.

This is the file structure of our sample application:



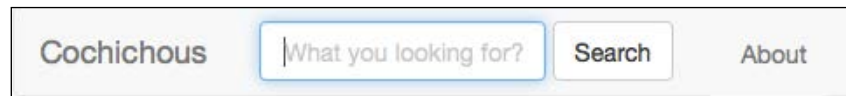
Let's define the grid and elements for the header in our `index.html` (the code for the HTML template can be found in `5792_04_01.html`).

Remember the `viewport` metatag, whose importance we already explained in *Chapter 1, Bootstrap 3.0 is Mobile First*. Also, let's reference our `bootstrap.min.css` that was customized before.

The preceding code shows the basic template of the header. We'll try to use all the Bootstrap elements, but sometimes we need to adjust some spacing (if possible, start to change spaces to relative units to keep proportions). We are using a `style.css` file to overwrite BT styles if needed.

In the template, we define 12 small device columns inside a row class and add a navbar component (<http://getbootstrap.com/components/#navbar>).

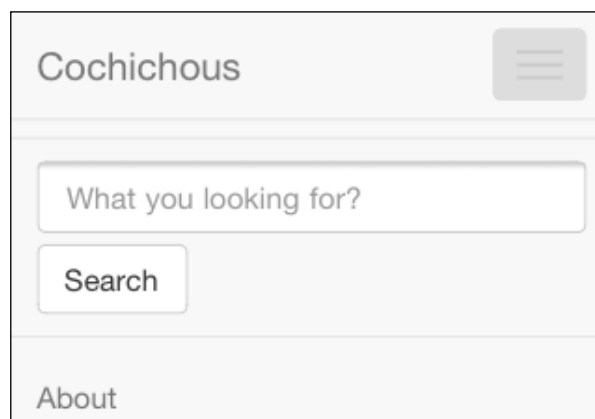
Here's an overview of our code result in a tablet device:



It's a brand and search form. There's a simple modal window to display the information about the Cochichous project. It's a section called **About** that will open the modal window.

The toggle menu (visible only in mobile devices) has a target attribute that points to the associated menu with the class `navbar-ex1-collapse`.

Another collapsed version of our header in mobile phones is shown in the following screenshot:



In the main page of our application there's a simple form with a **Search** button. Now, we will have panels that will accommodate the tweets. Panel is a new component in Bootstrap 3 (<http://getbootstrap.com/components/#panels>) that is very useful to show a simple collection of elements.

So, with these panels we could have the following search result making a Twitter search in the mobile and desktop devices:



The previous screenshot is an overall view of how the mobile search result would be and how it should be displayed on the desktop. The following is the code for a single search result that we will use to fill with results from the Twitter API. We now use a `col-md-12` class to target the desktop too (available in `5792_04_02.html` from the code bundle):

```
<div class="col-xs-12 col-md-12 single-message">
  <div class="panel">
    <p class="text-primary">This is a sample Tweet near from you
      ;-)</p>
    <p class="text-muted">
      6 minutes ago
    </p>
    <p class="text-muted">
      from <a
        href="http://www.twitter.com/alexanmtz">alexanmtz</a>
        , New York
      </p>
    </div>
  </div>
```

But at first, the user loads a screen where there's no result until the user makes a search and after the search the template will look like the following code (It can be found in 5792_04_03.html from the code bundle):

```
<div class="container">
  <div class="row">
    <div id="refresh">
      <div class="alert" id="no-search-message">
        <strong>Make a search</strong> above to find <strong>
          nearest Tweets.</strong>
      </div>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 col-md-12">
      <div class="alert alert-info">
        <strong>Your location:</strong>
        <p id="user-location"></p>
      </div>
    </div>
  </div>
</div>
```

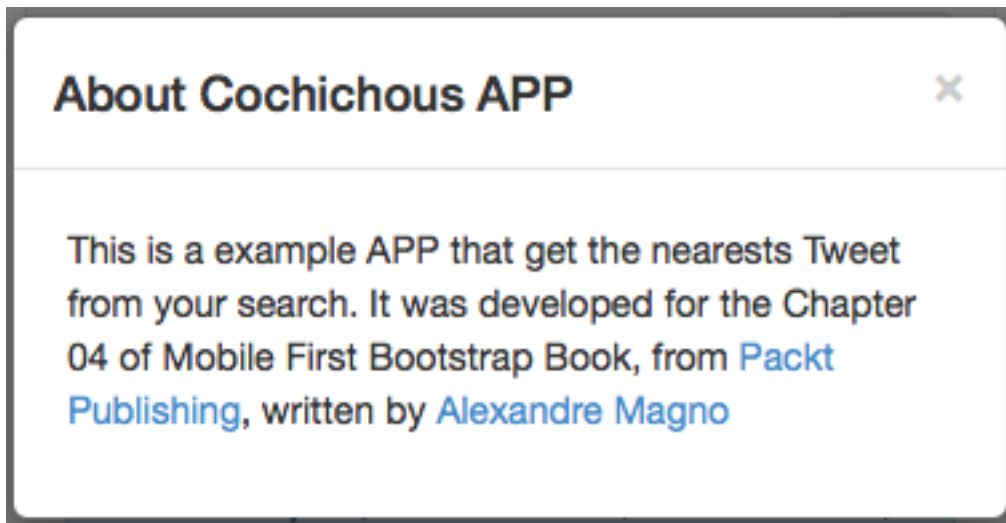
The Bootstrap modal component example

Then, we associate this modal in the same `index.html` below our previous example with one link called the **About** section:

```
<a data-toggle="modal" href="#myModal" >About</a>
```

We now have to define a basic modal markup (<http://getbootstrap.com/javascript/#modals>) that can be viewed in 5792_04_04.html from the code bundle. As we can check in this file, there's a header, text, and dismiss link. This link has the data-attribute `dismiss` value. Once this is enabled, we don't need to use JavaScript to initialize scripts. We can make the modal closing action just by defining an HTML attribute in the desired element.

In the preceding link to the About modal section, we get an href attribute with a value of a modal ID, in this case #myModal, and with data-toggle we are defining that this link is a toggle for a modal. When the modal receives the right data-api, it gets hidden and is activated when the associated button is linked. We have the following result when the user clicks on the **About** link:



Geolocation

Now that we have our mobile style layer, we will interact and use JavaScript. With Geolocation, we can explore the device capability to get the user location easily. Using a standard API, we will support cross device GPS and explore the mobile localization that the desktop can still offer its users.

We get the user information through a localization API from HTML5. The user will be prompted to give us permission to get his/her location. Then, with the user authorization we can have the current latitude and longitude. Using this information with the Twitter API we can obtain the nearest tweets through a search parameter. There's a technique through which we can obtain the address called reverse geocoding (<https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse>). It means from a given longitude and latitude, we can obtain the address.

We make a simple call to Google Maps API in the following way:

```
<script type="text/javascript"
  src="http://maps.googleapis.com/maps/api/js?sensor=false">
  </script>
```

So, the code starts defining the `geocoder` object (the code is located in the `geocode.js` file inside the `js` folder in the file structure of our first figure about our application tree and it can be found in `5792_04_01.js` from the code bundle).

The code in the file has a `geocoder` variable to be used later to create a new geocoder instance. So, we get the current position. There's a callback for success expressed as `successFunction`, that holds the localization and calls the `codeLatLng` function (it can be found in `5792_04_02.js`).

The result is an object that holds all the information about the user's localization.

We call a `getLocation` generic function that handles the `results` object in a friendly way to return the user to an appropriate local place. You can see the full code and how it works in `5792_04_02.js`.

```
$('#user-location').val(address_line);
$('#input[name="latlng"]').val(lat+', '+lng);
```

With the user locale data, we can display to the user his current place inside an alert component (<http://getbootstrap.com/components/#alerts>):

```
<div class="row">
  <div class="col-xs-12">
    <div class="alert alert-info">
      <strong>Your location:</strong>
      <p id="user-location"></p>
    </div>
  </div>
</div>
```

Here, we are using JavaScript to display location information to the user. And what if the users don't support the API? What about Internet Explorer 8? Well, in this example we would like to explore a powerful way to get the user's right location, and we target mobile devices as it was the main objective of the project. Without the user location, there's no reason for our application to exist. We can handle the data in the server to display alternative content, in this case it will be worldwide messages as a fallback. Even if we miss the information of user location, we can display messages and avoid stopping our application due to missing support from the browser.

We select the input through a jQuery selector with the name `latlng` and use the variables in comma-separated values, that is, it's like the Twitter API expects to send its parameters. We have the form with the following input hidden, that will be filled by the current user's latitude and longitude coordinates through JavaScript:

```
<input type="text" name="search" class="form-control"
  placeholder="What you looking for?">
```

There's a form action to make a request to a PHP script that returns the result block from the Twitter API. You can see the full form in `5792_04_07.html` from the code bundle.

With the information for the user location we can make an AJAX request to Twitter based on the user's location in the `main.js` file:

```
$(function(){
  $('#form[role="search"]').on('submit', function(e){
    $('#refresh').html('<div class="progress progress-striped
active">' +
  '<div class="progress-bar" role="progressbar"
  aria-valuenow="45" aria-valuemin="0"
  aria-valuemax="100" style="width: 100%">' +
  '<span class="sr-only"></span>' +
  ' </div>' +
  '</div>'
  );
  var search = $('#input[name="search"]').val();
  var latlang = $('#input[name="latng"]').val();
  var action_url = e.target.action;
  $('#refresh').load(action_url, {q: search, latlang: latlang});
  return false;
  });
});
```

In the preceding JavaScript code, we target a form with the role `search`. After it gets submitted, we can see the progress bar applied in the targeted content while the request is being made (<http://getbootstrap.com/components/#progress>).

We access the input and get its values. Then, we set the `search`, `latlang`, and `action_url` variables, which are necessary for our AJAX request to perform a search query.

You can check the full code in `5792_04_04.js` from your code bundle.

The Twitter API search

Now, we have our template for the application and let's get results from Twitter. First of all, we need to create a Twitter account to have an access to the developer features (<https://dev.twitter.com/>). With an account created, we get the benefits to use the whole API. In the account that you just created, go to your account menu and click on **Create a new application**; fill up the basic information about your app, such as name, description, and website. You are now ready to get your credentials to use in your own application.

Now, you will see this page that shows all the keys and setup of the application (the keys are like your password, so it's omitted here for you to use your own credentials):

OAuth settings

Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your ap

Access level	Read, write, and direct messages About the application permission model
Consumer key	oL1L80WgP0LanP000kLw
Consumer secret	gU4HwPQpabg5qsTEaLwY11yp02000Ww0MeCBE
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token
Callback URL	None
Sign in with Twitter	Yes

Your access token

Use the access token string as your "oauth_token" and the access token secret as your "oauth_token_secret" to sign r
Do not share your oauth_token_secret with anyone.

Access token	2278W104-g1xN857kqH0MB944B0opiaz0Q0F1uG4Q2AH
Access token secret	owInMMpp4lw90cFy3T1RPgBNDxvX8lu2Fv6w
Access level	Read-only

[Recreate my access token](#)

Take a note of these numbers that we will use to connect to the Twitter API with OAuth and make search requests. OAuth (<http://oauth.net/>) is a protocol standard for authentication for a given service, it is used by Facebook, Yahoo!, and many others too for authenticating through API. It's now required for the Twitter API v1.1.

So, take a note of your consumer key and consumer secret numbers.

Make a search

Now that we have our credentials to get access to the Twitter v1.1 API, we will use a PHP script to return a search result, based on a query and the information about latitude and longitude that we already have in our hidden field filled with JavaScript obtained values accessing the Geolocation API from HTML5.

To execute this PHP script you will need to run a web server such as Apache (<http://www.apache.org/>). If you have Linux and Mac OS, you already have Apache installed beforehand and you just need to create the files in a web server default folder and run in <http://localhost> in your browser (<http://unix.stackexchange.com/questions/47436/why-is-the-root-directory-on-a-web-server-put-by-default-in-var-www>). On Mac, there's a great tutorial for this (<http://georgebutler.com/blog/setting-up-local-web-server-on-os-x-snow-leopard-10-6/>) to set up a simple local server.

On Windows, you can use a **WampServer** (<http://www.wampserver.com/en/>) to get this script working. If you have a web host, you can just upload files to a host that supports PHP 5 and higher and test it live.

If you have a preferred language, you can make your own script, there are a lot of Twitter clients for Ruby, C#, Node.js, Python, and so on.



Another option would be to use Yahoo! Pipes (<http://pipes.yahoo.com/pipes/>). With the philosophy of rewiring the Web, this Yahoo! service acts like a proxy for your web services and APIs. It acts like an aggregator of services in one place. With YQL language, we can execute queries from different services, so we could make a query to get a Twitter search and use only JavaScript through JSONP. Unfortunately, until this writing it was not possible to use it because the new Twitter API v1.1 requires OAuth and Yahoo! Pipes, which is outdated with the new Twitter API.

We can take a look at `5792_04_01.php` from the code bundle that makes the request to return the search results.

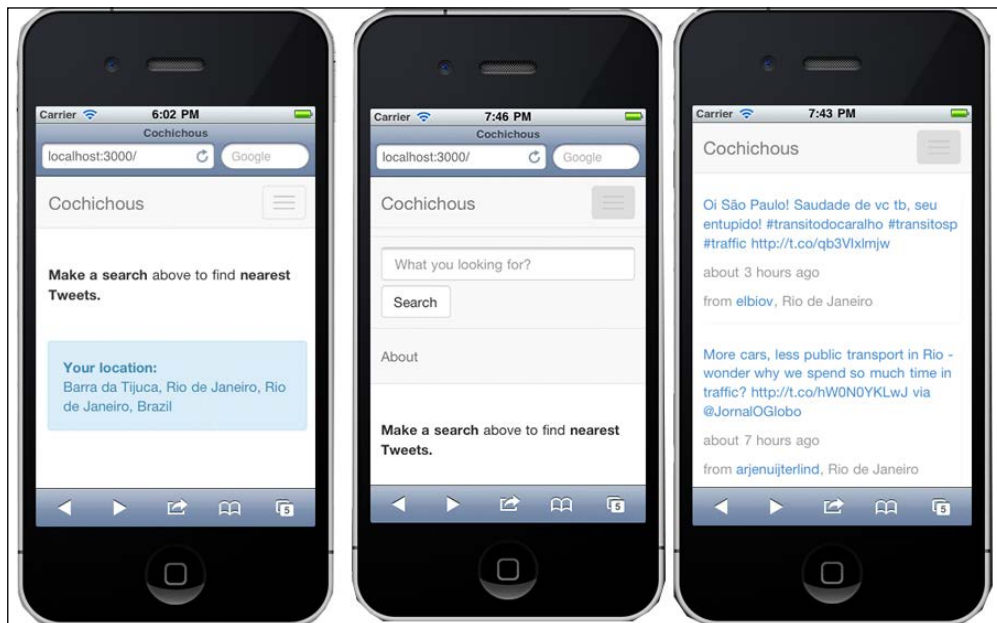
First, we include two libraries:

- That will make it easier to connect through Twitter
- Display the time of the tweet in a format such as *6 minutes ago* from the date

The first one included is `TwitterAPIExchange` (<https://github.com/J7mbo/twitter-api-php>). You can have it referenced in the `twitter-api.php` with `require_once` and include it in the same directory. The other one is `timeago` (<https://github.com/jimmiw/php-time-ago>), which is used to display the time in a user-friendly way about when the tweet was posted.

We have a settings variable that gets your credentials from the Twitter developers site to use with the **TwitterAPIExchange** to make a request to the Twitter REST API. We get the post's information from our form submission to build the URL and query string to perform in the Twitter API REST request. We then instantiate the **TwitterAPIExchange** class and get the tweets. So, we get this object of the tweet, extract the information we want (text, creation, name, and location), make a `foreach` loop in the given returned object, and generate a template with our HTML Bootstrap template, which we had already defined when we constructed the static files.

So, the following screenshot shows our fully working application on a mobile:



Going from a tablet device to desktop screen resolution

We have completed the first step, now it's time we move beyond. We are now going to the next breakpoint: tablet devices.

At its creation, the tablet was considered a "giant iPhone". When it was designed there were many things in common, but there are subtle differences that make the tablet our next desktop. With the ascension of tablets consolidated by Apple and now with Galaxy Tabs and other ones everywhere, people are getting lazy to use desktops. We now need the simplicity of a tablet device. But why do they differ? Besides the obvious, for sure.

The big difference is that in your phone you deal with one hand. The tablet use is similar to a desktop, but is still more practical. It can be as simple as it can get; a tablet experience is a mixed experience between mobile and desktop.

In our application example, we do not need to put extra classes in HTML to change breakpoints; it already has a good use in our iPad and desktop from the start. In Bootstrap 3, when you declare 12 columns, with `col-xs-12` class, it's a mobile 12 grid units by default, because if we targeted Mobile First, the mobile becomes the default instead of the desktop. Let's imagine that we want to divide our tablet into 3 columns, because we don't want to see very wide messages in this device; we want to see it as blocks, divided in four columns.

The only change to this layout responsiveness to the application will be the following:

```
<div class="col-xs-12 col-sm-4 single-message">
  <div class="panel">
    <p class="text-primary">Sample message</p>
    <p class="text-muted pull-right">
      2 minutes ago
    </p>
    <p class="text-muted pull-right">
      from Boston, Malden, USA %>
    </p>
  </div>
</div>
```

Note a class called `col-sm-4` that defines the breakpoint to a tablet. Our grid will fit in four columns in a tablet, and keep 12 columns in mobile phones.

If the desktop fits in four columns too, the same will work for `col-md-4`, but it's not necessary, because it will keep four columns for larger screen sizes until you define a class with different column size. We just need to define classes that will behave with a different grid setup over the screen sizes through devices.

Now, we are done with mobiles and tablets; let's go to desktops. This must have sounded strange earlier, but now we already have seen the big difference.

Now that we are dealing with the desktop, we start to think in mouse interactions and states that are consequences of it. But this Bootstrap version has already defined the hover states for the desktop.

On the desktop, we could keep the same setup as a tablet: three column wide, because there are just a few variations through the sizes, so we keep the fluid layout and no different breakpoints for this case is a good option.

The choice between a web app and mobile application

Yes, we could use native properties of the device to give a more optimized app experience using native apps over our Mobile First suggested solution. Every type of web app or native app has its advantages, but in recent times we do not consider mobile app in preference to web app. The mobile experience is not just to install apps, as we already know, but navigating through the Internet is being made more common than ever in mobile devices.

I'm not intending to say that a native app would be better. My answer would be, "a native app would be different, and that's it". This section is to show the difference between the two approaches to consider when we need to decide if Mobile First makes worth or a native application would be better, or even if the two solutions should coexist together.

As we have seen, the web app has a great flexibility. We can access it from everywhere by just having a browser and changing styles with the CSS media queries accordingly with the screen resolutions.

The native application that resides in the devices, is a friendly way to access your device services as applications. It has the great advantage to be managed and installed by the user in a download service, such as the Apple Store for the iPhone/iPad and Google Play for Android users. They can explore the enhancement of the hardware capability in the device with direct access to OS and hardware compatibility, using core features that in many cases browsers do not have access to such as the address book, SMS, and camera. The native app is closer to the device because it doesn't need a browser that intercepts it, so we can have, for example, a notification system and then execute the process in the background.

The native application has its own language for different platforms, so for Apple iPhone we have to use Objective-C and for Android we have to use Java. There's now a technology known as **PhoneGap** (<http://phonegap.com/>) that offers a native way to deliver the application using our frontend well-known languages. PhoneGap solves the big issue of using different languages but it falls into another key point: generating a native language with a framework is not comparable to a device implemented in its pure and native language.

But why both? Well, here we are not talking about the iPhone or Android, which are currently the most famous smartphones. Two years ago, Android did not have any relevance to be cited in a book, now it's dictating tendencies. We have to be scalable and try to reach the user anywhere, and get more potential satisfaction to them.

So, the answer now is Future Friendly (<http://futurefriendly.ly/>) and embrace the new opportunities that multiple devices can offer.

Summary

In this chapter, we put all our basic knowledge in Mobile First Bootstrap and work in a real situation.

You have followed the development of Cochichous, a sample application that finds the nearest tweets from Twitter API based on a search.

As we saw, there are lot of considerations and limitations, which we have to consider when developing Mobile First. So, we start to sketch and think cross device. We made a basic flow to make a search from Twitter REST API in a simple way in mobile devices, and with little changes we have an app that works in other devices too.

We explored Geolocation to show how to get the user location through an HTML5 web page. Then, we saw how to change the grid to adapt in other devices going from phone to tablet and from tablet to desktop. We made a simple use of the Bootstrap grid that gives us countless flexibility.

Then, we get into the unavoidable dilemma: it should be better if it was a native application? Analyzing the situations, and now we know how to use the power of both.

Until now, we did not consider a variable that affects it most when we deal with the mobile experience: performance. Let's start to worry with the key point in mobiles, that was partly achieved with our best practices here, but there are still some other important issues when we deal with the mobile experience to make it usable and fast. Now, we are ready to develop more reliable applications about performance in the next chapter.

5

Performance Matters

We now have enough knowledge to face the Mobile First challenges. We have already launched a production site. Along the time, we will see that we are wasting vital resources and discover that our site is not really usable because it is slow. Some aspects can be optimized and doing this we can reduce the page loading time significantly. For example, why load a high-resolution image in a small device?

In this chapter, we start to think about performance from the start, and how it impacts the usability and user commitment on our site. We will see how to load responsive images and the different options to do so. Besides that, you will discover how to load external files as needed and have the flexibility that font icons offer; that is, support for different sizes, density, and colors. In this chapter, we will see three main aspects that will make our app faster and scalable. These optimization methods will give you the additional benefit of having a more robust desktop website too. It is worth mentioning one more time that Mobile First is not just for mobile needs – it can even make your desktop faster.

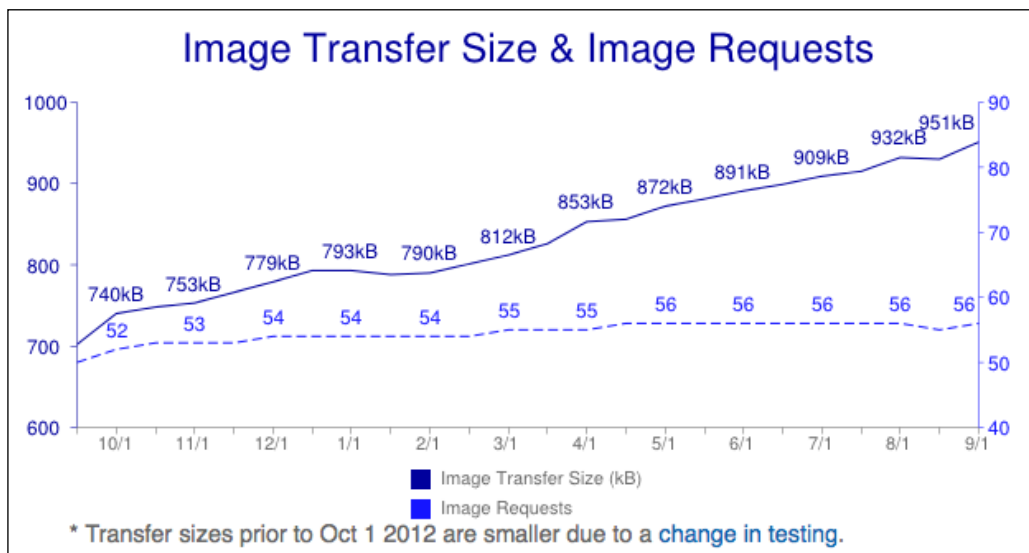
This chapter will cover the following topics:

- Responsive images
- Load on demand
- How to optimize icons

Responsive images

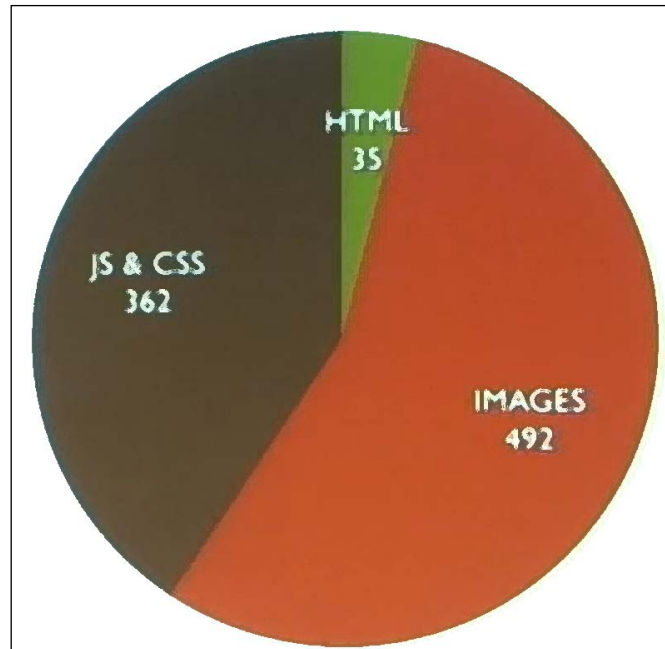
Images are a relevant point in web applications. More so, when you deal with user uploads because you have to handle all kinds of cases and image-processing methods. It is one of the biggest factors, responsible for around 80 percent of the frontend loading performance, together with icons, JavaScript, and CSS (<http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/>).

Websites are getting fatter. We are exploring new resources in an uncontrolled way because we have more availability, more memory and more facilities. As we can see in the following figure, the overall image length is increasing exponentially on the web:



We are not in evolution at this point. We are getting smaller, and using smaller processors and memories with more constraints. So, we have to be rational and focus on content. Every unnecessary byte loaded on your site can be a loss for audience. Research suggests that every 500 milliseconds of delay in loading time causes 20 percent reduction in traffic (<http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html>).

User patience is even shorter for mobiles (<http://www.developer-tech.com/blog-hub/2013/mar/15/mobile-users-becoming-more-demanding-less-patient/>). We should take care of that, and one of the heaviest aspects is about images. 60 percent of page weight is images (<http://httparchive.org/>).



The preceding diagram, taken from the Mobilism 2013 event presentation by John Clevely, shows how the BBC News builds its responsive news website (<http://vimeopro.com/mirabeaunl/mobilism-2013/video/68025331>).

To start with, we have to separate concepts. All we are talking about here is content images. The purpose of content images is to display a real picture or a photo. Even a diagram icon is not applicable in this case because it's monochromatic and we use a font method for rendering, allowing us to have more flexibility.

The other aspect is about image proportion. There's a recommended method, `max-width`, that takes the image proportion that is scalable to the device. We use pixel-defined images to make images fit in our body container. To remember this method, we just use a single CSS declaration:

```
img {  
    max-width: 100%;  
}
```

Bootstrap 3 has a wrapper that uses a simple `img-responsive` class (<http://getbootstrap.com/css/#overview-responsive-images>):

```

```

So, we now have a simple way of making our images flexible. Unfortunately, in this case, if an image has 1200 px width and 600 px height, the method will load all the 1200 px and 600 px and it will be resampled. This will waste a lot of KBs, which will decrease our page loading for slow connections, and even for average ones. However, the expectation from mobile users is almost instant loading; remember that user patience for mobiles is very limited.

The only thing we can do in the frontend to offer a solution for this, is to use **Scalable Vector Graphics (SVG)** to generate images, because SVG is vector scalable, so flexibility is in its core. However, SVG has its own application for drawings. Even if we can draw images in SVG and make them scalable for different sizes, we should not use these methods to substitute images because we would solve one problem to create another. SVG should be used when the final file size of the generated data is less than the equivalent image. But, SVG is usually also a good option for graphs because it is generated by XML instructions to draw the graph.

There should be a better way to optimize images, right? Well, I can say that there's not a single good and definitive solution until now to hold optimized images in responsive designs that is scalable and well supported. We will see different options and their pros and cons.

To make things worse, Retina Display comes to the scene to make the issue even more complex. Because we need to have images with a higher pixel density in Retina Display, we need a double-sized image. For example, if we have a 600 px square image then we need an equivalent image of 1200 px as the pixel density in Retina Display has to be two times greater than the default display. In that case, the image in the retina is of 1200 px but we see it as 600 px with double-size pixel density (http://en.wikipedia.org/wiki/Retina_Display) support. As we can see, responsive images are a really hi-impact limitation to Mobile First at the moment.

There are different images for different breakpoints, and a lot of types of variables that we need to consider when rendering an image in your web application:

- Different images for different breakpoints
- Different images for different pixel densities
- Art direction
- Image formats

There is a specific solution for each variable, and there is another variant with a mix of each solution.

We can set different images for different breakpoints. It would work, but imagine this getting bigger and bigger. Duplicate images are not a good solution in the long run. As I said, there is now a need for different pixel densities with the use of Retina Display that doubles the pixel size, needing double-sized images to achieve the right resolution in supported devices.

Art direction is a smart method that cuts the image to the main content of the picture, but requires methods to process the image on the server. The end result is an image that is ready to be displayed with the focus on what matters in this photo. One of the great tools for that is Thumbor (<https://github.com/globocom/thumbor>). This is an open source project developed at Globo.com that uses Python and has smart cropping.

Data URI also needs to be mentioned. With data URI, we can *append* the image directly in the `src` attribute with data encoded in base 64 token format. This way we do not need to make a separate request to load an external resource. With data URI, it is possible to use the upload API in HTML5. This APIs embeds as data URIs in the image source attribute and give a preview in the browser before uploading to the server.

There is a new image format to help us compact image sizes while keeping the quality. Besides JPG, GIF, and PNG, which are the most common image formats, there's a WebP format that makes the final image size smaller without loss of quality. Google developed it. But, it is not yet well supported. However, it can become standard in the near future. The following table shows the strengths and weaknesses of various image formats:

Image format	Strengths	Weaknesses
GIF	Small graphic animations	Colors and alpha
PNG	Small, medium, and large images with alpha transparency	Larger images
JPG	Photos, selective quality, and progressive loading	Larger images
Data URI	Inline (no extra requests)	Larger images
WebP	Small	Chrome/Opera only

There's a solution using the `srcset` attribute that we can use to take advantage of an already known method that we mentioned before, `data-api`. The `srcset` attribute instructs the browser to set different size setups and change according to the resolution. This can be done with this simple JavaScript and HTML code:

```
<script>
  if($(window).width() > 960) {
    $('img').each(function() {
      $(this).attr('src', $(this).data('desktop'));
    });
  }
</script>

```

We have a simple solution to load the image according to the device, by just creating an attribute and check the resolution with JavaScript. So we can load different images accordingly with the `data` attribute that corresponds to the current screen resolution.

There's more accurate syntax in the following HTML:

```

```

In this syntax, we define a collection of image sources to different resolutions and the syntax as `2x` to define density. This is not well supported until now. This was introduced by WebKit in August 2013 (<http://mobile.smashingmagazine.com/2013/08/21/webkit-implements-srcset-and-why-its-a-good-thing/>). As of now, we need a polyfill to have better support for this, using libraries such as `Picturefill` (<http://scottjehl.github.io/picturefill/>).

But, if we think we have used the best solution for optimizing, just by loading the right image for the devices (because we are using JavaScript, and JavaScript rocks), we are completely wrong. If you check your browser, you will see that all images have been loaded and you can make the problem even worse.

The **World Wide Web Consortium (W3C)** came to the scene to offer some standards in this aspect. We will now see how it works.

The W3C proposed a tag `picture`, whose code is as follows:

```
<picture alt="Pizza">
  <source srcset="small-1.jpg 1x, small-2.jpg 2x">
```

```
<source media="(min-width: 18em)" srcset="med-1.jpg 1x, med-2.jpg 2x">
<source media="(min-width: 45em)" srcset="large-1.jpg 1x, large-2.jpg
2x">

</picture>
```

With this syntax, we define the image collections that the browser used to decide the image to be displayed.

It has great advantages, as follows:

- Audio/Video element syntax
- Media-query-based
- Supports non-pixel values
- Supports art direction
- Uses the 2x functionality of @srcset
- Fallbacks for unsupported types

On the other side, we have server-side technologies and a lot of solutions to take this responsibility out of the frontend and just deliver the image processed to be rendered.

Thumbor is an example and has already been mentioned. Sencha is another example (<http://src.sencha.io>). Adaptive images (<http://adaptive-images.com>) also do the same job. The drawback in all three is their availability on the server. We expect an external resource to generate the images in this case.

With Sencha, it is done in the following manner:

```
<img src='http://src.sencha.io/http://sencha.com/files/u.jpg'
alt='My smaller image' />
```

Dave Rupert (<http://twitter.com/davatron5000>) proposes a middle method called 1.5x hack. In this case we find a half way to balance up a size between the smaller (for mobile) and higher (for desktop). So we don't load all the images but an intermediary that will not affect one in preference to another (<http://daverupert.com/2013/06/ughck-images/>).

There's the clown car method too (<https://github.com/estelle/clowncar>), which counts with SVG. As I have already mentioned, and it's Bootstrap philosophy, we should work in an implementation that can be recommended by W3C in the future and be native. Until then, you use JavaScript to do the job. We can see once again that JavaScript programming is really being a superhero, doing for us what a browser would normally do.

Load on demand

We used `Modernizr` in our project in the previous chapter to detect browser compatibility. With `Modernizr`, we can do the following in our previous scenario:

```
Modernizr.load({
  test: Modernizr.geolocation,
  yep : 'geolocation.js',
  nope: 'find-by-ip.js'
});
```

Here, for example, we test for geolocation support.

If we find support, we load `geolocation.js`, which we have already explored, and it is all that GPS can offer. If we do not have support for this, we can still load a polyfill as a fallback if the supported feature fails to load.

There are other libraries, such as `Respond.js` (<https://github.com/scottjehl/Respond>), that can be loaded as a polyfill to support CSS3 media queries in browsers that don't support this (such as Internet Explorer 8). Well, it's an enhanced solution because we don't need to load an extra script in browsers that do this by default.

There are different loading methods using different libraries and approaches; for example, `lazy load` (<http://www.appelsiini.net/projects/lazyload>). This library delays the loading of images in long web pages and displays the images as the user scrolls to those particular images.

Optimizing icons

There is a golden rule to optimizing icons: use web fonts when applicable. You can use SVG for scalable icon graphs too.

One of the best tools that Bootstrap has is `font-awesome`. It has been mentioned before and is one of my favorite tools. We have a complete open source icon gallery that is generated by `font-icons`. `Glyphicons` is another great tool that is a part of the Bootstrap core. It's a great time to talk about `Glyphicons`, because they use `font-face` to render fonts. The use of sprite images to generate a collection of icons is no longer used in `Glyphicons`.

The nomenclature has very few changes—just load a CSS and use the following:

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

Now, we have an icon class.

With font-icons, we can use a character from a font and use it as an icon. With this solution, we can have all the flexibility that the font already has for type, such as applying shadow through CSS3, animations, and size. If you don't have to worry about icon sizes, it's one thing less to disturb you to get your application optimized. The font-face tool is loaded once and with a simple class, you can have an icon font without worries. There's a set of common icons included in this tool that support all basic needs of your web application.

If you need another set of fonts, or even want to create your own, use the open source project called Icosmith (<http://icosmith.com/>). It generates custom fonts from an SVG icon, contains classes already, and are ready for use.

Summary

In this chapter, we covered one more important layer in Mobile First Bootstrap with a final touch of enhancement in the time taken for site loading. The time taken for loading decides if our users are happy or angry. We have seen the three main aspects that affect page loading directly.

You now have the knowledge to make the best decisions in terms of using different options and tools to manage performance issues, handle images and icons, and load content on demand.

We saw that using the font-icons tool makes us improve the inclusion of icon families as font-families and discover how flexible it is and how simple it is to include in your project.

We discovered great tools that make the management of performance a real cost benefit for you. We also covered tools that provide better responsive images, keeping the quality and using smart methods, such as art direction, to deliver just the right content for the user. Now, we can think "mobile" at the highest level, because we know about Bootstrap Mobile First!

Index

Symbols

`.col-lg-*` class 27
`.col-md-*` class 27
`.col-sm-*` class 27
`.col-xs-*` class 27
`.hidden-lg` class 29
`.hidden-md` class 29
`.hidden-sm` class 29
`.visible-lg` class 29
`.visible-md` class 29
`.visible-sm` class 29

A

Adaptive images
URL 71
Adobe Kuler 51
alert plugin 46
application bootstrapping
about 50
customizable version, inserting 50, 51
project template 52-55
asymmetric grid
pros 25
URL 24

B

Bootstrap
documentation feature 49
download page 9
Bootstrap 3 35
Bootstrap carousel plugin
about 37
enhancement 37-40
Bootstrap documentation 24

URL 31
Bootstrap Expo
URL 9
Bootstrap Mobile First
about 8
reviewing 8, 9
Bootstrap modal component example 55, 56
Bootstrap modal plugin 41
Bootstrap navbar component 15
bootstrapping 9
Bootstrap source code
modifying 18, 19
tests, running 19
border-radius property 22
Bower
about 16
responsiveness 18
URL 17

C

close event 46
clown car method 71
Cochichous 49
COLOURlovers 51
CSS component 36

D

data attribute 41, 42
docs running
Git checkout, executing 16
Jekyll, installing 16
double-sized image 68

F

forms
 designing 28

G

geocoding 56
Geolocation 56, 57
GitHub
 URL 52
Glyphicons
 URL 28
grid framework
 about 24
 cons 25
 pros 25
grid system
 about 22
 breakpoints 25
 completely fluid layout 25
 grid framework 24
 icon library 28
 predefined classes 25, 26, 27
 predefined classes 27
 semantic grids 22, 23

I

icons
 optimizing 72, 73
Icosmith 73
image formats
 Data URI 69
 GIF 69
 JPG 69
 PNG 69
 WebP 69
Install Node 19

J

JavaScript
 using, on server 46
JavaScript Collapse plugin
 URL 31
Jekyll 16
jQuery plugin 36

L

lazy load
 URL 72
LESS variables 51
lynx
 Bootstrap template 45
 URL 45

M

max-width method 67
Minimal Viable Product (MVP) 12
mobile application
 choosing 63
Mobile First
 benefit 14
 new project, creating 12-14
Modernizr
 using 72

N

namespace events 46
NUI (Natural User Interface) 11

O

OAuth
 URL 59
opacity property 36

P

PaintStrap 51
PhoneGap 63
Picturefill 70
progressive enhancement 42
project template 52

R

relative units 30
Respond.js
 URL 72
responsive classes
 .hidden-lg 29
 .hidden-md 29
 .hidden-sm 29

- .visible-lg 29
- .visible-md 29
- .visible-sm 29
- responsive content**
 - need for 10, 11
- responsive dropdown 15**
- responsive images 66-71**
- responsive navigation 31**
- responsive utilities**
 - about 29
 - responsive classes 29
 - semantic grid variables 30

S

- Scalable Vector Graphics (SVG) 68**
- semantic grids**
 - about 22, 23
 - cons 25
- Semantic.gs**
 - URL 23
- Sencha**
 - URL 71
- sourceset attribute 70**
- Susy**
 - URL 23

T

- tablet devices**
 - moving to 61, 62
- Thumbor 69, 71**
- timeago library 60**

- TwitterAPIExchange class 61**
- TwitterAPIExchange library 60**
- Twitter API search**
 - about 58, 59
 - making 60, 61

U

- unobtrusive JavaScript 44**

W

- W3C**
 - about 70
 - advantages 71
- WampServer**
 - URL 60
- web app**
 - choosing 63
- World Wide Web Consortium. *See* W3C**

Y

- Yep**
 - URL 41



Thank you for buying Mobile First Bootstrap

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, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: 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 licences, 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, 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.

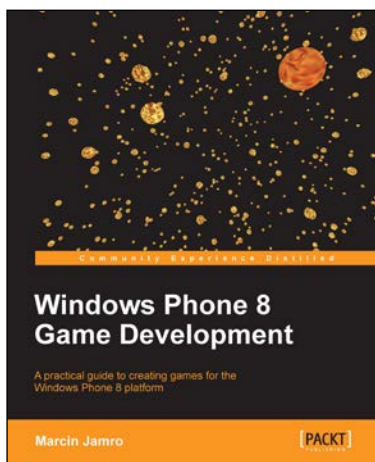


Twitter Bootstrap Web Development How-To

ISBN: 978-1-84951-882-6 Paperback: 68 pages

A hands-on introduction to building websites with Twitter Bootstrap's powerful front-end development framework

1. Learn something new in an Instant! A short, fast, focused guide delivering immediate results
2. Conquer responsive website layout with Bootstrap's flexible grid system
3. Leverage carefully-built CSS styles for typography, buttons, tables, forms, and more



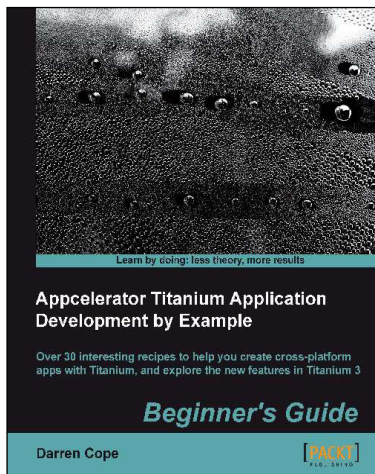
Windows Phone 8 Game Development

ISBN: 978-1-84969-680-7 Paperback: 394 pages

A practical guide to creating games for the Windows Phone 8 platform

1. Create a 3D game for the Windows Phone 8 platform
2. Combine native and managed development approaches
3. Discover how to use a range of inputs, including sensors
4. Learn how to implement geolocation and augmented reality features

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

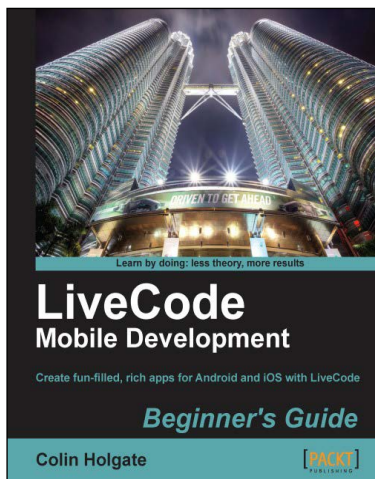


Appcelerator Titanium Application Development by Example Beginner's Guide

ISBN: 978-1-84969-500-8 Paperback: 334 pages

Over 30 interesting recipes to help you create cross-platform apps with Titanium, and explore the new features in Titanium 3

1. Covers iOS, Android, and Windows 8
2. Includes Alloy, the latest in Titanium design
3. Includes examples of Cloud Services, augmented reality, and tablet design



LiveCode Mobile Development Beginner's Guide

ISBN: 978-1-84969-248-9 Paperback: 246 pages

Create fun-filled, rich apps for Android and iOS with LiveCode

1. Create fun, interactive apps with rich media features of LiveCode
2. Step-by-step instructions for creating apps and interfaces
3. Dive headfirst into mobile application development using LiveCode backed with clear explanations enriched with ample screenshots

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