DESIGNING FOR EVERY DEVICE

Responsive MOBILE DESIGN

Phil **DUTSON**

Praise for *Responsive Mobile Design*

"Whether you're building or refining your skill set, *Responsive Mobile Design* is the quintessential guide to getting up to speed with modern web practices. Phil's unique background and expertise grant him insights that both the hardcore programmer and pixel perfect designer will find invaluable."

—**Jacob R. Stuart**, Web/UI Designer

"It's impossible to build for the web today without taking various screen sizes and form factors into account; you never know if your user will be on a phone, tablet, or desktop. This book helps lay the groundwork for building responsive web designs. It's really a must-read."

—**Cameron Banga**, Co-founder, 9magnets, LLC

"Anyone looking for a comprehensive book on responsive design tactics would do well to pick up a copy of *Responsive Mobile Design*. Phil does a stellar job of breaking down the how and why of RWD in this practical guide to designing for a wide spectrum of screen sizes and devices."

—**Dennis Kardys**, Design Director, WSOL

"While the three initial technical ingredients of RWD (fluid grids, flexible images, and media queries) still stand true, building a site today requires much more thought and know-how than it used to. This book will take you beyond the basics and teach you the ins and outs of modern web development."

—**Erik Runyon**, Director of Web Communications, University of Notre Dame

"Phil Dutson unveils a dummy-proof treasure trove of essential mobile design advice, resources, and examples bound to enlighten designers and developers alike. This book belongs on every web designer's shelf—a comprehensive guide to return to time and time again."

—**Kaylee White**, Web Designer, SEO.com

This page intentionally left blank

Responsive Mobile Design

Responsive Mobile Design

Designing for Every Device

Phil Dutson

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Control Number: 2014944246

ISBN-13: 978-0-133-88821-8

ISBN-10: 0-133-88821-5

Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.

First printing: September 2014

Editor-in-Chief
Mark Taub

Executive Editor
Laura Lewin

Development Editor
Sheri Cain

Managing Editor
Kristy Hart

Senior Project Editor
Lori Lyons

Copy Editor
Krista Hansing Editorial Services

Indexer
Tim Wright

Proofreader
Debbie Williams

Technical Reviewers
Cameron Banga
Dennis Kardys
Jacob Stuart

Editorial Assistant
Olivia Basegio

Cover Designer
Chuti Prasertsith

Compositor
Nonie Ratcliff

Manufacturing Buyer
Dan Uhrig

To my friends and family, who remind me to look at everything through the eyes of an inquisitive 5-year-old without fear of mashing buttons until everything is working properly.

—Phil

This page intentionally left blank

Contents-at-a-Glance

Contents

This page intentionally left blank

PREFACE

The phrase "Responsive Mobile Design" doesn't really roll off the tongue, and even when placed under a microscope it tends to shift and blur, making it difficult to gain a full appreciation for what it is.

When you boil it down, it comes down to a paradigm shift. In breadth of design, this isn't really a new concept. It is more like the time when you first realized you could draw things in a third-dimensional perspective, and suddenly a new fascination with cubes, spheres, focal-points, and shadows started to overtake most of your sketches.

Being able to step back and realize that people want information as soon as possible, and having it fit on the device they happen to have at the moment, you can gain an appreciation for making sure that they get what they want in the most aesthetically pleasing way possible.

That is Responsive Mobile Design: the fusing of content, structure, and beauty to deliver experiences that users will continue to keep coming back to.

This book is full of my experiences with mobile devices, design, and even a smidgen of code that can help get your creation into the hands of millions of mobile users in the best way possible. Along the way, some topics will be lightly brushed over while others will have their intricacies beat upon like the soothing double-bass of your favorite Swedish-metal band.

To effectively use this book, you should have some experience with web design or development. That being said, this should also make an excellent resource for project and team managers who would like to learn current methodology and concepts they can use with their team.

Some topics are just not easily covered, or covered in proper detail, without an accompanying site with which to follow along or see some examples. I have created a website that you can leverage for various tools, tips, tutorials, and examples. Visit www.mobiledesignrecipes.com/ on your desktop or mobile device to find these resources.

You can also reach out to me on Google+ (+PhilDutson) or Twitter (@dutsonpa).

ABOUT THE AUTHOR

Phil Dutson is a Solution Architect over client-side and mobile implementation for ICON Health & Fitness. He is the author of *Sams Teach Yourself jQuery Mobile in 24 Hours*; *jQuery, jQuery UI, and jQuery Mobile Recipes and Examples*; and *The Android Developer's Cookbook, 2nd Edition*. He enjoys learning and writing about technology, and spreading enlightenment to the world one portal at a time with his sons playing Ingress.

ACKNOWLEDGMENTS

Creating a book is a monumental feat, but there are major players who help the process finish in a timely manner. First, I want to thank Laura Lewin for all her work and the dedication she put into this project. I would also like to thank Olivia Basegio for all of the last-minute emails and replies to random questions that I seem to send her. This book could not have happened without the dedication of the production team including Kristy Hart, Lori Lyons, Krista Hansing, Mark Taub, and all the other unsung heroes who make books like this polish into diamonds.

I have a special thank you for the extremely talented, remarkable, and absolutely brilliant resources who are Cameron Banga, Dennis Kardys, and Jacob Stuart. I appreciate the work that each of you put into being my technical editors. Thank you for all the insight, help, and comments you contributed to the flow, understanding, and clarity of each chapter. Thanks also to Sheri Cain, who as my development editor, put the reins on the project to make sure that my tangents made sense and that each chapter had clarity and focus from beginning to end.

I also want to thank my family for allowing me to disappear so many nights a week and still think I'm a pretty cool guy. Without the help of Ethan, Kile, Josie, Sam, and Anna, I wouldn't have anyone to use for the images that added a personal touch to the sample designs and tutorials in this book.

CHAPTER 8

RETROFITTING AN EXISTING SITE

It doesn’t matter whether you are working for yourself, as a freelancer, for a corporation, or even as part of a design studio—at some point, you will be asked to take an existing site and make it work on everything.

This might seem overwhelming but is by no means impossible. In this chapter, you learn about choosing the proper layout, working with site components, and keeping some important issues in mind when going mobile.

When starting the conversion process, I work through three basic areas. I start by creating a block-level layout of my current design, then I work on handling each component, and finally, I work on adding and fine-tuning features to make the mobile experience more enjoyable, easy to use, and more native.

You will need to determine your layout while you work through the existing design. A responsive design layout gives you a fluid and flexible site, whereas an adaptive approach helps you ease into the fluid process by giving you some elements of pixel-perfect layouts while exposing your design to media query–based control. You might also end up with a hybrid layout that uses elements of both responsive and adaptive design.

Choosing a Proper Layout for Mobile

Most sites that are in need of conversion to mobile devices have been designed to fit somewhere between 960px and 1140px in width. With the iPhone 5 currently having a maximum visible resolution of 320x568px when viewed in portrait orientation and 568x320px when viewed in landscape orientation, you need to make many decisions and choices. Note also that these dimensions are the ones visible by dividing the actual number of pixels by the pixel density ratio (Retina screens have a ratio of 2, so 640x1136 becomes 320x568).

Block-Level Layout

You have many ways to begin the process of creating a design: You can break out the sketchbook and pencil, the prototype stencils, or any number of drag-and-drop applications. However, the method I lean on when working on retrofitting is the standard block-level layout.

If you are unfamiliar with block-level layout, the easiest thing to do is to look for the seams in your site. As a starting point to help you identify the blocks, you can use the following list:

- Header
- Side navigation or bar
- Content area
- Footer

To see this in action, Figure 8.1 shows a site with an overlay of how I would break down a site into blocks.

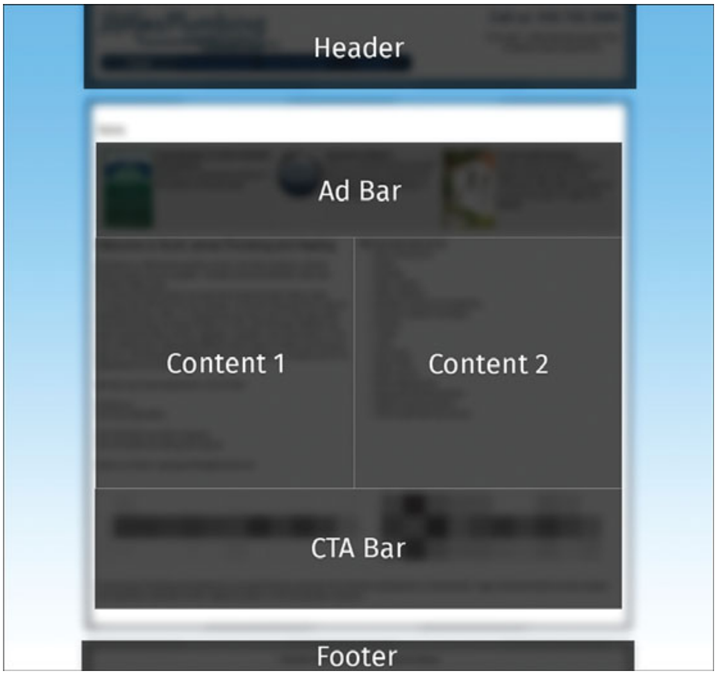


Figure 8.1 By grouping the site into content blocks, you can easily see the important areas of your site.

Tip

It can be difficult to take pictures of a currently rendered site, but you can make use of several applications. On OS X, try using the Paparazzi! application (http://derailer.org/paparazzi/) to easily save images of entire site pages. As a bonus, Paparazzi includes Automator scripts that you can leverage to automate your workflow.

Windows users can use FireShot (http://getfireshot.com/) as a browser extension to capture site pages.

With the page broken down into blocks, you can get an idea of the content that each block contains, including components such as search areas, navigation, and widgets. This is helpful because it enables you to work on breaking down the site into smaller pieces and rearranging content to fit. Figure 8.2 demonstrates how the blocks are changed to show the page arranged to fit on a smaller screen.

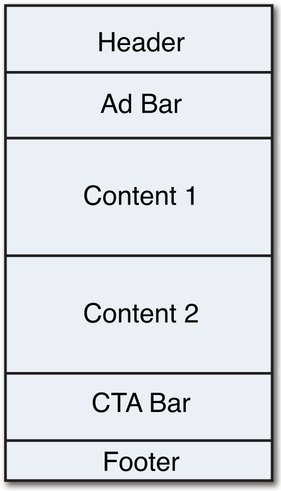


Figure 8.2 By rearranging the content blocks, you can visually see how the site will adapt for different screens. Note that some areas might change in size.

Even though all sections are visible, not all of them are actual size. You will need to change them based on the amount of content you have and how you decide to present them. It is also important to note that you will still need to work with the "fold." Mobile devices can make dealing with the fold complicated because you will have no way to reliably handle where the fold is exactly. If you have an analytics page on your site (Google Analytics, Adobe Omniture, or similar service), you should be able to get a list of device resolutions and construct a suitable landing experience for most of your users.

You can use various methods to create your layout. You can even get creative with bits of paper by cutting them out at the size you want and writing the name of the component in them. The point of using the block-level layout is to see how the page is going to flow and react based on the screen of the device viewing your site.

Now that you have a rough layout, it is time to decide whether you will be embracing a responsive layout or an adaptive one.

Responsive Layout

You know that using a responsive layout means that everything needs to be fluid and flowing and that it will use as much available space as it can get its pixel-loving virtual hands on.

This type of layout has little to no waste of screen real estate, but it also generally has enough whitespace to calm the mind and keep users from feeling forced into a cave so that they start breathing into a paper bag before claustrophobia sets in and they ultimately smash the Close Tab button on their browser and head for open spaces.

Choosing to roll with a responsive layout means that you now need to think about the following:

- Flexible percentage or `em`-based layout, with gutters that change based on screen width
- Text that might break in odd places
- Images that need to be swapped out or allowed to scale
- Acceptance of a design that is no longer pixel perfect

Tip

An `em` unit is the equivalent of the base unit of measurement on the body of your page. The general default is 16px. This can be helpful in doing quick layout changes without worrying about actual pixel values.

Figure 8.3 demonstrates how a site could transition from a small screen up to a larger one.

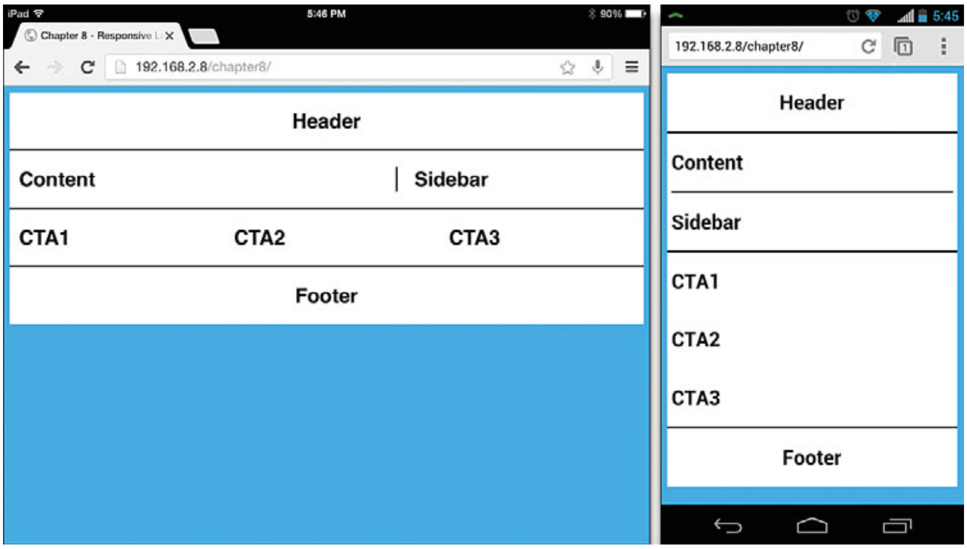


Figure 8.3 The content and image areas remain edge to edge as the transition is made to larger screens.

Embracing a fully responsive solution is difficult and requires serious planning and dedication for the design, user experience, and user interaction teams. Note that, in this design, CTA means *call to action*. These are areas of the design that draw users into clicking or tapping to see more information or lead them to a specific section of the site.

Adaptive Layout

Approaching your design with an adaptive layout can help others get used to the idea of using a design that changes depending on screen size and also gives some control back to your design. This is because of the locked-width flow of adaptive web design.

Whereas a responsive layout is a maximized experience, an adaptive layout gives you the capability to be pixel perfect. With each breakpoint, you set a maximum width for your content area and then use margins that grow until the next breakpoint is matched.

Figure 8.4 demonstrates how an adaptive layout transitions between sizes.

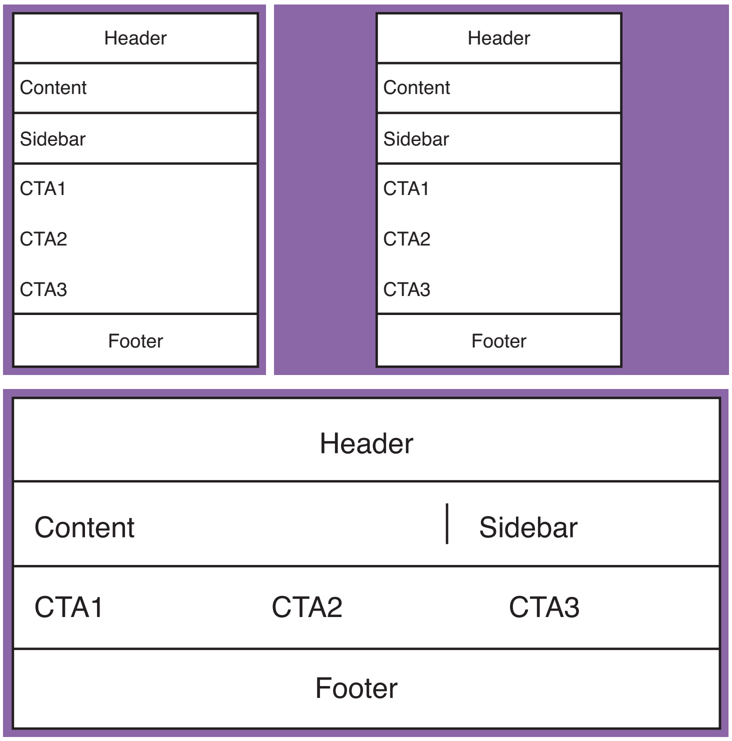


Figure 8.4 The design starts edge to edge (top left), but margins grow as the screen viewing the site increases in size (top right) until the next breakpoint is reached and the process starts over (bottom).

If you are a pixel-perfect designer, this method might work best because it will be more compatible with your existing flow and will feel like you are building mock-ups of the same site in different sizes.

No matter what layout you decide to use, you need to determine a method for handling all the components that are contained in each of the blocks of your site.

Working with Components

Everything in your site can be broken down into components. Sometimes these are simply elements. Other times, they are groups of elements. A search input, a navigation menu, and sliders are all examples of elements.

When creating the mobile or smaller versions of your site, you need to take into consideration what should happen to each of these elements.

Navigation

No matter how brilliant your current navigation system is, odds are, you will need to change it to make it fit on smaller devices.

You could always wrap the navigation to a new line; however, this often looks sloppy and comes across as lazy. Still, this will work if your navigation is text based and if you can align the words so that they are balanced and the line looks intentional instead of heavy on one side. If your navigation is reliant on hover states or mega menus, you will need to create a new system or method for handling all your links.

You should consider two other methods when you are compressing your navigation. The first is to use a menu that drops in, and the second is to use an off-canvas solution.

Both solutions require the use of a menu button or icon that will take the place of your text.

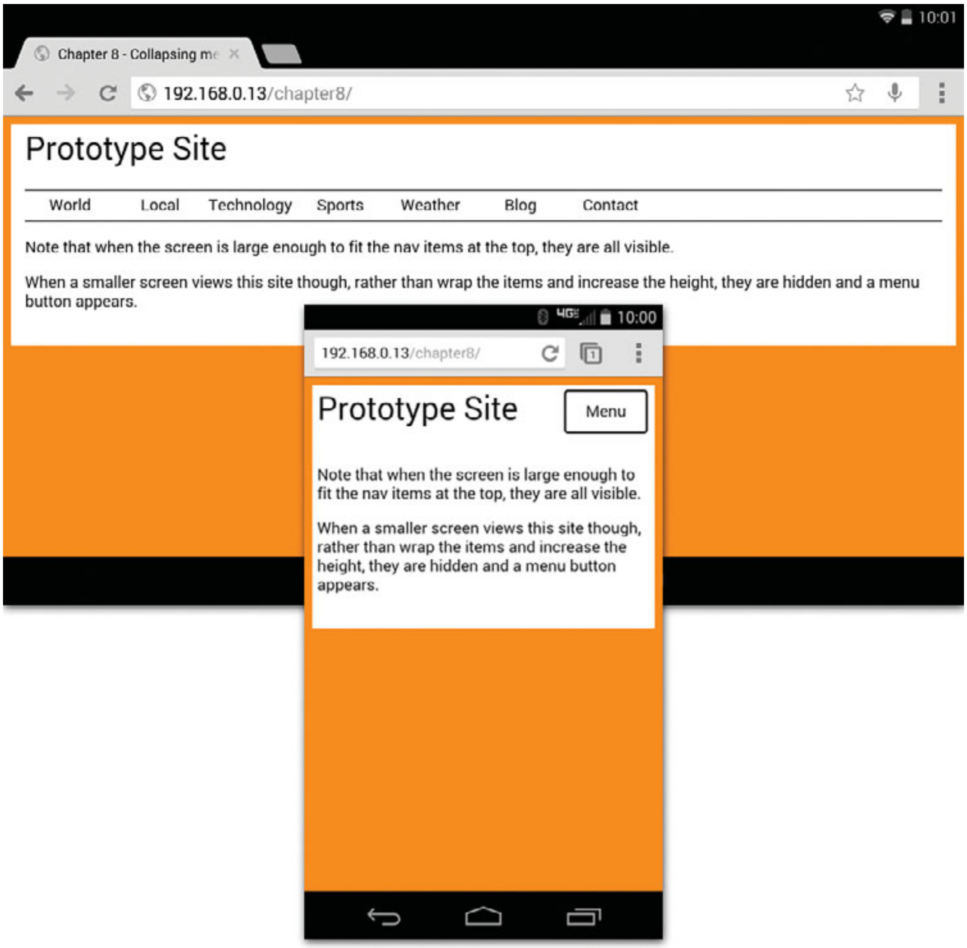
Tip

You might be thinking of immediately jumping on the “hamburger” icon for your menu. This might work for you, but consider a study that tested the hamburger icon, the word *Menu*, and the word *Menu* with a round border that made it look like a button (http://exisweb.net/mobile-menu-abtest). The results found that more users engaged with the word *Menu* when it appeared to be a button than with the other methods.

Applying a menu that will drop in requires either using multiple layers, injecting a block of code on a click or tap event, or using classes to change the height and visibility of the content area.

Using an off-screen navigation solution is similar, but it slides in the menu by using animation (either JavaScript or CSS3 transitions) to make the content appear. This should be familiar to you because it is the type of solution Facebook implements in its mobile application and is also used in many Google products as a way to access the menu. Google+ and Google Music use this type of navigation to give you access to settings, playlists, images, groups, and more.

Because every project is different, you will need to play with the breakpoints for when your site navigation changes from text to the menu change. Figure 8.5 demonstrates a site displayed at different sizes, with the navigation changing.



Note that when the screen is large enough to fit the nav items at the top, they are all visible.

When a smaller screen views this site though, rather than wrap the items and increase the height, they are hidden and a menu button appears.

Figure 8.5 As the site is viewed on smaller devices, the navigation changes so that it is hidden until activated by the Menu button.

Some plugins might help you with your off-canvas navigation:

- Foundation Zurb (http://foundation.zurb.com/docs/components/offcanvas.html)
- Twitter Bootstrap (http://getbootstrap.com/examples/offcanvas/)
- Pushy (www.christopheryee.ca/pushy/)

Search

Unless your site has an absolute absence of content, you probably have a search box. The good news is that this particular search input adapts very well to mobile layouts.

Depending on your developers, there is already a fantastic chance that you are using the proper input element. HTML5 introduced a special `input` element made just for searching. It looks like the following:

```
<input type="search" name="search" />
```

The benefit of using the search input for your search is that mobile browsers can change the keyboard that appears to use it, and they can even add an icon and show you previous search entries when the field is activated.

The drawback of using this input is that some browsers automatically style the input to match the styles of the OS. For example, iOS rounds the corners of the input to make it appear like the default search element of iOS. This is good because the user can visually understand that it is a search field, but it is also bad because it could throw off your design (rounded corners on a flat design or a second magnifying glass added to the one you have already put on the page).

You have several options for handling the display of your search bar. You can choose to put it just below the header (which contains your logo and menu buttons) of your site, or you can choose to place it in the drop-in or off-canvas navigation area.

Either option is fine, but you should do some A/B testing, use heat maps, or apply other testing methods to make sure the search remains accessible and easy to find. Mobile users can be a finicky lot and will leave your site if they can't find what they need as fast as possible.

Content Areas

You might not give much thought to the content of your site, but if you are running a comparison site, an eCommerce site, or an informational site, you will run into the problem of having tons of content and not enough screen to properly display it all.

You have three common solutions for handling this particular issue:

- An **accordion** (or drawer) shows a particular title or question. The content is hidden until the title is tapped or clicked, when the accordion opens and shows the hidden content. This is commonly used on FAQ pages and pages that want to hide content-rich areas until the user has activated them.

  Note that some accordions allow only one section open at a time and automatically close any other open sections. Pay close attention to your user testing to make sure you are not frustrating users who would like to have multiple sections open at the same time.
- **Tab systems** are useful to display short terms that then open content areas that the user can then go through and view. This particular system suffers from the same autoclosing feature of some accordions and also forces you to cram buttons and text into a small space. This system works best on medium-size screens, but you can also use it successfully on smaller screens if you pay proper attention to text or icon size to activate the tabs.
- With a **grid system**, you have the option of arranging your content into columns. As the screen size of the device viewing your site shrinks, these columns will start to compress until you decide that the content is no longer legible. At this point, you can "break" the columns so that they take up 100% of the available width instead of the preassigned 25%, 33%, or 50% of screen space.

  The downside to using columns to break apart displayed content is that your pages have the potential to get very long. I'm not talking about a couple hundred pixels long—more like a carpel tunnel–inducing, finger-sprinting marathon of a page of long swipes. You should stick to using this particular style of content display for smaller content areas.

Sliders

I'll skip the discourse on sliders because I know that, when it comes down to it, some people will tell you that they absolutely need one. In addition, when you are working on retrofitting a site, you will need to know how to handle them.

First, because speed should be one of your primary objectives, you need to remember that sliders are inherently slow. This is the result of having to load multiple images into an area and then get them restyled so that they can be displayed in order. DOM processing is also slowed, and mobile devices will have to work harder to download, inject, and then redraw all the images on and off canvas as needed. Another side effect that you might not be aware of is the tradeoff in battery life. As the mobile device works harder to display your page, CPU and memory are used more and battery life can be affected.

Second, sliders are problematic on mobile devices because of the "pause on hover" effect that no longer works with mobile design. With this effect, the slider on a desktop would normally continue to show one slide after the other, but it stops when the mouse cursor is hovering or stopped on one of the slides. Mobile users might also become frustrated with sliders when they do not respond to screen swipes and when the users have no way of pausing or stopping the sliders.

Many mobile users are comfortable with the concept of swiping to move content around, but when that behavior is unavailable or seems to act funny due to slide timing, this can be a source of aggravation.

To successfully use a slider on mobile devices, remember the following:

- Let users move the slider.
- Make your slider touch/swipe friendly.
- Minimize the amount of data or number of slides in the slider.
- Load content by use of a deferred method (such as lazy loading) so that the site does not appear broken or waiting to load content.

A couple sliders that work well with mobile devices are BXSlider (http://bxslider.com/) and Owl Carousel (http://owlgraphic.com/owlcarousel/).

Links

A major consideration when moving from a desktop-only site is the size of your links. You might have noticed while visiting various sites that buttons and links are much larger when viewed on mobile devices. This goes deeper than just the Web: The developer guidelines for Windows Phone, Android, and iOS also specify that touch targets, or areas where the user can tap, should be big enough for a finger to tap.

How big is that, exactly? Well, it varies, and various pixel density values can make it somewhat difficult to narrow down. However, the following sizing values will get you started in the right direction:

- Use a minimum size of 34px by 34px, but consider using at least 44px.
- The width of your target can be longer than 44px, but the height should be at least 34px.
- Be sure to provide adequate space between targets—use at least 8px, to minimize accidental tapping.

Learn more about designing for touch devices by visiting these sites:

- iOS guidelines on Layout (https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/LayoutandAppearance.html#//apple_ref/doc/uid/TP40006556-CH54-SW1)

- Windows Phone 8 Human Interface Guidelines (http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff967556(v=vs.105).aspx)
- Android Metrics and Grids (http://developer.android.com/design/style/metrics-grids.html)

Considerations When Going Mobile

Knowing how to handle layout and some of the components will help in the retrofitting process, but you need to be aware of some other surprises.

For example, using the `:hover` CSS pseudo class is generally not a good idea with mobile devices. Having a click-to-call button, dealing with modal windows, and even using input fields are all extra matters that need to be taken into consideration.

No More Hover

Mobile devices are currently in an interesting place. Some devices, such as certain Samsung devices, can actually detect hovering fingers or stylus pens, but most devices cannot. Many laptop manufacturers have also started to include touchscreens, making this a potentially larger problem for more than just mobile devices.

This has a tendency to "break" the `:hover` CSS pseudo class. What happens is that you tend to get a tap-to-activate action that triggers the hover and then forces you to tap again to make your selection or dismiss the hover. This can get confusing and frustrating, depending on the touch target areas of your site. This doesn't mean that you can no longer use hover, but it requires you to think ahead.

Think about it this way: Let's say that you have a category with several items underneath that appear in a drop-down list that is triggered by using `:hover`. Now, if you had clicked on the category name to go to the category page, all your mobile visitors would have to tap the name once to activate the drop-down and then tap the name again to go to the category page.

This leaves the mobile user wondering whether tapping the category name will close the drop-down or take them somewhere. To get around this, you need to add a link named View All or similar so that mobile users know that they have a safe place to tap to get where they want to go.

Click to Call

People love convenience, and mobile users thrive on it. This might be the reason you need to consider adding a Click to Call button. This is not new: Maximiliano Firtman talked about doing this in 2010 (www.mobilexweb.com/blog/click-to-call-links-mobile-browsers). It seems that many designers and developers overlooked it.

You are likely aware of the major benefit of talking to a person when making a purchase. By adding a Click to Call element to your design, you empower your users and your marketing teams to help make both parties happy.

A simple way to add Click to Call to your site is with an anchor element, like the following:

```
<a href="tel:+15555555555"></a>
```

You want to make sure that you have styled the element to `display: block` and have added `width` and `height` values to it as well. Finally, you should consider adding an icon to it to help visually convey to users that, by tapping the icon, they can instantly dial the number. Also, for users not on a smartphone, this will appear as a link that does nothing. Some operating systems are looking to solve this issue by incorporating features that, when clicked from a desktop, will dial through various Voice Over IP systems or even push the call directly to your phone.

Modal Windows

Not long after everyone agreed that pop-up windows were a terrible idea because of potential distraction and mistrust (thanks to malware and infested sites that added Close buttons that actually installed malware instead of closing the window), the modal window was born. This particular style of window allowed pages, images, videos, and more to be displayed within the main window.

Many different types of modal windows are available, but they all have one thing in common: They are terribly implemented on mobile devices. What worked on your desktop design is suddenly not an option on mobile.

To design around this particular problem, you can use the following solutions: using a new window modal and using a resizing modal.

New Window Modal

Use a modal that takes users to a new page. This is similar to the approach that would be used with a framework such as jQuery Mobile. The modal window becomes a transition that displays a new page with a Close or Back button that takes users back to the original page.

The disadvantage of this particular style is that you are jarring the user with two experiences, and some users might not realize that they are on a new page that they need to close to get back to where they were.

This is of particular note when using product image galleries because moving users to a new page could cause them to become distracted or irritated that they left the page they wanted.

Resizing Modal

Many modal solutions currently employ a resizing technique to keep the contents of the window inside the available viewing space. These techniques work well for images (because most smartphone browsers can resize them to fit), but text content can be a major concern.

To handle the text elements, you need to either keep your content minimal or stick with using images. Regardless of the content, you need to make sure that close links are visible at all times so that users can exit the modal and return to the page they were on.

> **Tip**
>
> Test on the devices your users use. Unbelievable as it may seem, I created a modal window that worked perfectly for my devices (all had a minimum width of 360px), but I failed to test on an iPhone. My Close button was just barely offscreen and forced users to use the Back or Reload buttons on their browsers to get back to the page. This was a very costly mistake on my part. Don't learn this the hard way.

To visualize how a resizing modal works on multiple devices, see Figure 8.6, which shows the modal being used on an iPad and an Android phone.

Input Fields

The last consideration that needs your attention is the way your input forms work. You already know that search fields will change based on the input type; however, you might not have thought about how some of the built-in device features can sabotage your site.

You can leverage several HTML5 input types with properties to help get around these issues.

For email fields, use a type of `email` to add built-in browser validation:

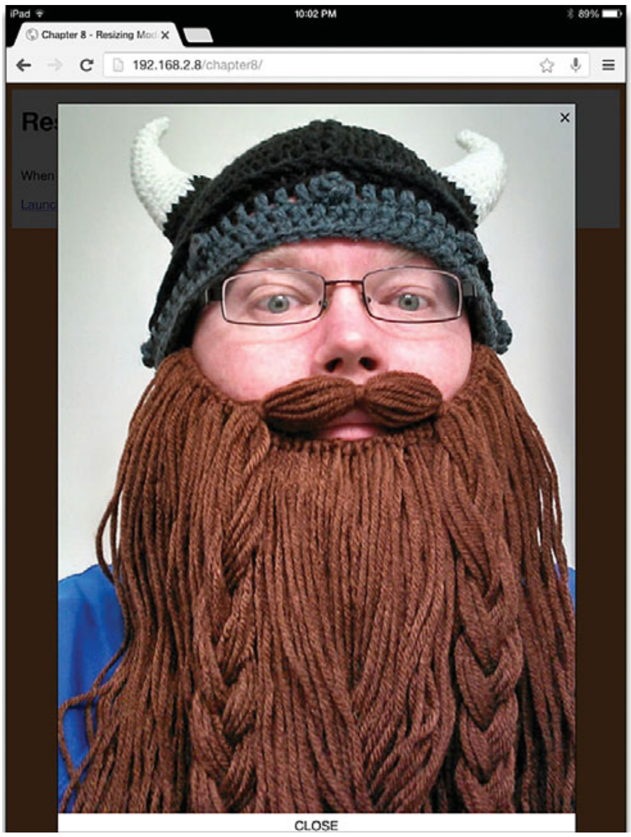
```
<input type="email" name="email />
```

Any iOS devices running iOS 5.0+ (which should be 100%) will, by default, disable the autocapitalization and autocorrect on this field. If you find that some users are still getting autocorrect or autocapitalization, you can add properties to the input like so:

```
<input type="email" name="email" autocorrect="off"
autocapitalize="off" spellcheck="false" />
```

This tells the browser that the field should not correct what the user has typed in. Note that these properties can also be used on text areas and text input elements.

It might seem like a small issue, and it doesn't play a direct role in the visual design process; however, as part of the user experience, paying attention to tiny interaction points is vital to a winning design, especially when it comes to mobile devices.



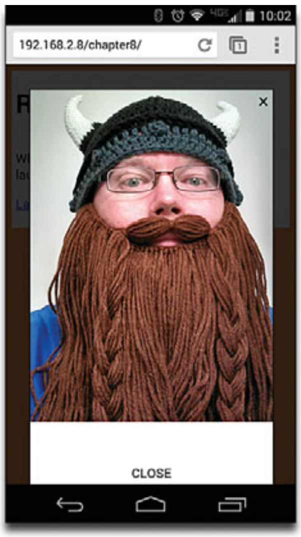


Figure 8.6 The image is clearly visible on both devices, while allowing access to close it.

Summary

In this chapter, you learned about the process of retrofitting a website. You learned about using a block-level strategy to isolate pieces of the site to work with, and then you moved on to working with the components that make up those blocks.

You also learned about many issues you need to be aware of when working a design to fit the needs of mobile users, including using sliders, hover states, search fields, text input fields, and modal windows.

This page intentionally left blank

INDEX