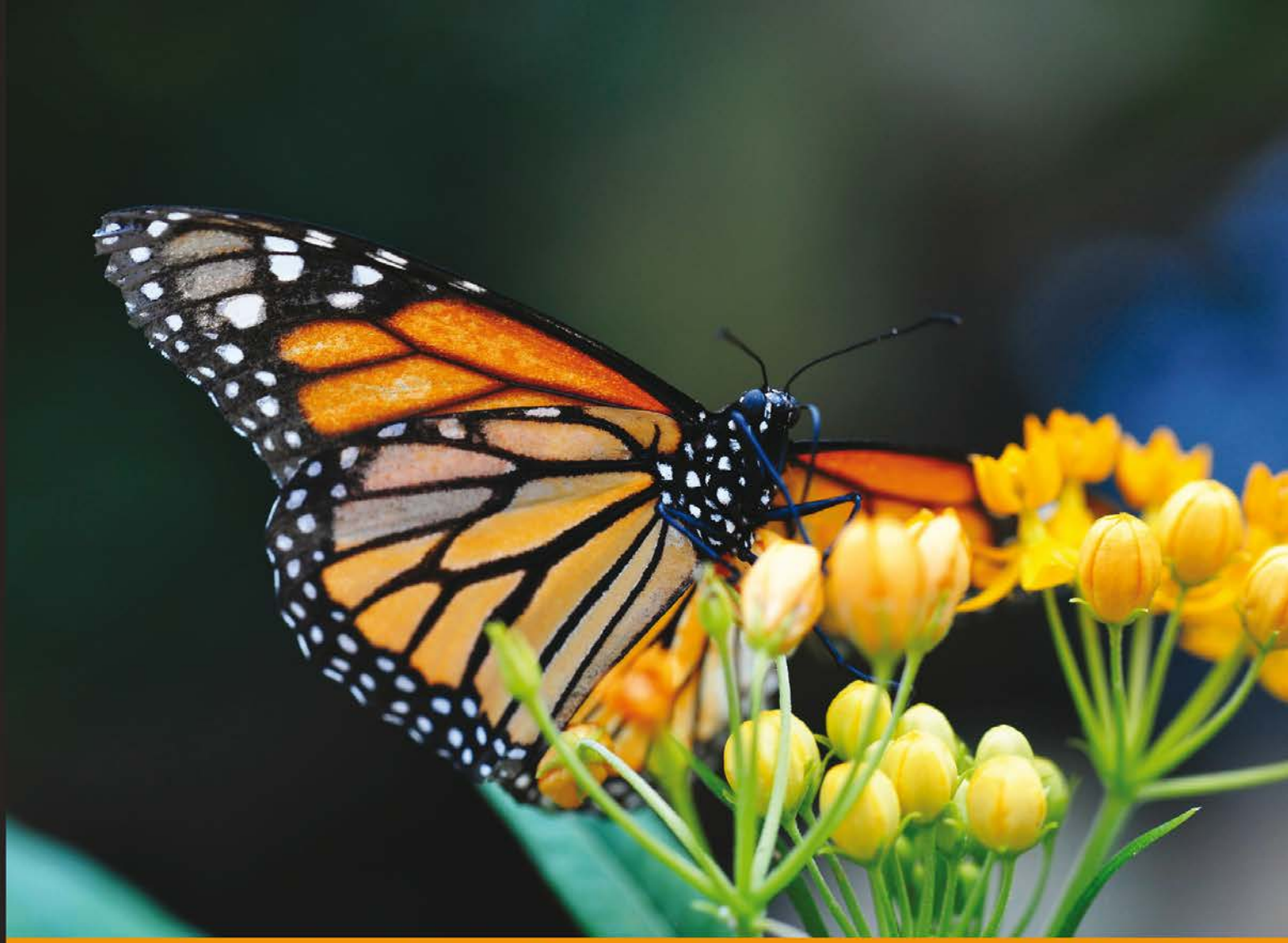


Community Experience Distilled

Responsive Web Design with HTML5 and CSS3

Second Edition

Build responsive and future-proof websites to meet the demands of modern web users

Ben Frain

[PACKT] PUBLISHING

Responsive Web Design with HTML5 and CSS3

Second Edition

Build responsive and future-proof websites to meet the demands of modern web users

Ben Frain

PACKT PUBLISHING

BIRMINGHAM - MUMBAI

Responsive Web Design with HTML5 and CSS3
Second Edition

First published: April 2012

Second edition: August 2015

Production reference: 2200815

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

ISBN 978-1-78439-893-4

www.packtpub.com

Credits

Author
Ben Frain

Reviewers
Esteban S. Abait
Christopher Scott Hernandez
Mauvis Ledford
Sophie Williams

Commissioning Editor
Edward Gordon

Acquisition Editors
Edward Gordon
Subho Gupta

Content Development Editor
Pooja Nair

Technical Editor
Ankita Thakur

Copy Editors
Rebecca Youé
Sonia Cheema

Project Coordinator
Bijal Patel

Proofreader
Safis Editing

Indexer
Mariammal Chettiyar

Production Coordinator
Nilesh R. Mohite

Cover Work
Nilesh R. Mohite

About the Author

Ben Frain has been a web designer/developer since 1996. He is currently employed as a Senior Front-end Developer at Bet365.

Before the web, he worked as an underrated (and modest) TV actor and technology journalist, having graduated from Salford University with a degree in Media and Performance.

He has written four equally underrated (his opinion) screenplays and still harbors the (fading) belief he might sell one. Outside of work, he enjoys simple pleasures. Playing indoor football while his body and wife still allow it, and wrestling with his two sons.

His other book, *Sass and Compass for Designers* is available now. Visit Ben online at `www.benfrain.com` and follow him on Twitter at `twitter.com/benfrain`.

I'd like to thank the technical reviewers of this book for giving up their free time to provide valuable input. Thanks to them, this is a better product.

I'd also like to thank the web community at large for their continued sharing of information. Without them, I wouldn't be able to enjoy my working days as a web developer.

Most importantly, a note of appreciation for my family. Many episodes of sub-standard TV (wife), cups of tea (parents), and piratical sword-fights (sons) were sacrificed for the writing of this book.

About the Reviewers

Esteban S. Abait is a senior software architect and former PhD student. He has experience devising the architecture of complex software products, and planning their development. He has worked both onsite and offshore for clients such as Cisco, Intuit, and Southwest. Throughout his career, he has worked with different technologies such as Java, PHP, Ruby, and Node.js among others. In recent years, his main interests have revolved around web, mobile and REST APIs. He has developed large, maintainable web applications using JavaScript. In addition, he has worked to assess clients on REST best practices. On the other hand, he has worked on high traffic websites, where topics such as replication, sharding, or distributed caches are key to scalability.

Esteban is currently working at Globant as a technical director. In this role, he works to ensure projects' delivery meet their deadlines with the best quality. He also designs software program training, and interviews software developers. In addition, he usually travels to clients to provide consultancy on web technologies.

Globant (`http://www.globant.com/`) is a new breed of technology service provider, focused on delivering innovative software solutions by leveraging emerging technologies and trends. Globant combines the engineering and technical rigor of IT service providers with the creative and cultural approach of digital agencies. Globant is the place where engineering, design, and innovation meet scale.

Christopher Scott Hernandez is a designer turned developer who has been working on the Web since 1996, when he built the Web's first boat upholstery site for his dad. He's since moved on to bring his expertise to companies small and large, having worked on some of the most visited sites in the world including eBay, LinkedIn, and Apple.

He was also a technical reviewer for *HTML5 Multimedia Development Cookbook*, *Packt Publishing*. Chris is an avid reader and lover of books. When he's not pushing pixels and writing code, he enjoys spending time with his wife and daughter exploring the parks and trails of the beautiful Austin, Texas.

Mauvis Ledford is a full-stack founder and CTO specializing in the realm of the web, mobile web, and scaling applications on the cloud.

Mauvis has contributed to products at Disney Mobile, Skype, Netflix, and many start-ups in the San Francisco and New York City areas. He is currently CTO at Pathbrite, an EdTech start-up specializing in free, responsive, multimedia e-portfolios and digital resumes for everyone. Create your own at `http://www.pathbrite.com`.

Mauvis was also a technical reviewer for the first edition of *Responsive Web Design with HTML5 and CSS3*, *Packt Publishing* and *Building Hybrid Android Apps with Java and JavaScript*, *O'Reilly Media*.

Sophie Williams is a bit of a perfectionist and has a thing for typography. She has a degree in graphic design and is currently a web/UI designer at `www.bet365.com`. While she loves designing for the Web, she will always have a special place in her heart for letterpress and print. Outside of work, she makes mean cupcakes, experiments with arts and crafts, and loves to point out (to anyone who will listen) when anything in the real world is misaligned.

You can find Sophie at `www.sophiewill.com` or follow her on Twitter `@sophiewill13`.

www.PacktPub.com

Support files, eBooks, discount offers, and more

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

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

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

`https://www2.packtpub.com/books/subscription/packtlib`

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

Why subscribe?

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

Free access for Packt account holders

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

Table of Contents

Preface

A responsive web design provides a single solution that looks great on a phone, desktop, and everything in-between. It will effortlessly respond to the size of the user's screen, providing the best experience possible for both today's and tomorrow's devices.

This book covers every essential aspect of responsive web design. In addition, it extends the responsive design methodology by applying the latest and most useful techniques provided by HTML5 and CSS3, making designs leaner and more maintainable than ever before. It also explains common best practice methods of writing and delivering code, images, and files.

If you can understand HTML and CSS, you can build a responsive web design.

What this book covers

Chapter 1, *The Essentials of Responsive Web Design*, is a whistle-stop tour of the key ingredients in coding a responsive web design.

Chapter 2, *Media Queries – Supporting Differing Viewports*, covers everything you need to know about CSS media queries: their capabilities, their syntaxes, and the various ways you can wield them.

Chapter 3, *Fluid Layouts and Responsive Images*, shows you how to code proportional layouts and responsive images, and provides a thorough exploration of Flexbox layouts.

Chapter 4, *HTML5 for Responsive Web Designs*, covers all the semantic elements of HTML5, text-level semantics, and considerations of accessibility. We also cover how to insert video and audio into our pages with HTML5.

Chapter 5, *CSS3 – Selectors, Typography, Color Modes, and New Features*, gets to grips with the endless possibilities of CSS: selectors, HSLA and RGBA colors, web typography, viewport relative units, and a whole lot more.

Chapter 6, *Stunning Aesthetics with CSS3*, covers CSS filters, box shadows, linear and radial gradients, multiple backgrounds, and how to target background images to high-resolution devices.

Chapter 7, *Using SVGs for Resolution Independence*, explains everything we need to use SVGs inside documents and as background images, as well as how to interact with them using JavaScript.

Chapter 8, *Transitions, Transformations, and Animations*, our CSS gets moving as we explore how to make interactions and animations using CSS.

Chapter 9, *Conquer Forms with HTML5 and CSS3*, web forms have always been tough but the latest HTML5 and CSS3 features make them easier to deal with than ever before.

Chapter 10, *Approaching a Responsive Web Design*, explores the essential considerations before embarking on a responsive web design and also provides a few last minute nuggets of wisdom to aid you in your responsive quest.

What you need for this book

- A text editor
- An evergreen browser
- A penchant for mediocre jokes

Who this book is for

Are you writing two websites: one for mobile and one for larger displays? Or perhaps you've already implemented your first 'RWD' but are struggling to bring it all together? If so, *Responsive Web Design with HTML5 and CSS3 Second Edition* gives you everything you need to take your websites to the next level.

You'll need some HTML and CSS knowledge to follow along, but everything you need to know about responsive design and making great websites is included in the book!

Conventions

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

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We can fix that prior problem easily by adding this snippet in the `<head>`."

A block of code is set as follows:

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

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "At its simplest, you pick a URL and click on **START TEST**."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

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

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

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

Customer support

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

Downloading the example code

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

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from: `https://www.packtpub.com/sites/default/files/downloads/8934OT_ColorImages.pdf`.

Errata

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

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

Piracy

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

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

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

Questions

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

1
The Essentials of Responsive Web Design

Only a few years ago, websites could be built at a fixed width, with the expectation that all end users would get a fairly consistent experience. This fixed width (typically 960px wide or thereabouts) wasn't too wide for laptop screens, and users with large resolution monitors merely had an abundance of margin either side.

But in 2007, Apple's iPhone ushered in the first truly usable phone browsing experience, and the way people access and interact with the Web changed forever.

In the first edition of this book, it was noted that:

> "*in the 12 months from July 2010 to July 2011, global mobile browser use had risen from 2.86 to 7.02 percent.*"

In mid-2015, the same statistics system (`gs.statcounter.com`) reported that this figure had risen to 33.47%. By way of comparison, North America's mobile figure is at 25.86%.

By any metric, mobile device usage is rising ever upwards, while at the other end of the scale, 27 and 30 inch displays are now also commonplace. There is now a greater difference between the smallest and the largest screens browsing the Web than ever before.

Thankfully, there is a solution to this ever-expanding browser and device landscape. A responsive web design, built with HTML5 and CSS3, allows a website to 'just work' across multiple devices and screens. It enables the layout and capabilities of a website to respond to their environment (screen size, input type, device/browser capabilities).

Furthermore, a responsive web design, built with HTML5 and CSS3, can be implemented without the need for server based/back-end solutions.

Beginning our quest

Whether you're new to responsive web design, HTML5, or CSS3, or already well versed, I'm hoping this first chapter will serve one of two purposes.

If you're already using HTML5 and CSS3 in your responsive web designs, this first chapter should serve as a quick and basic refresher. Alternatively, if you're a newcomer, think of it as a 'boot camp' of sorts, covering the essentials so we're all on the same page.

By the end of this first chapter, we will have covered everything you need to author a fully responsive web page.

You might be wondering why the other nine chapters are here. By the end of this chapter, that should be apparent too.

Here's what we will cover in this first chapter:

- Defining responsive web design
- How to set browser support levels
- A brief discussion on tooling and text editors
- Our first responsive example: a simple HTML5 page
- The importance of the viewport `meta` tag
- How to make images scale to their container
- Writing CSS3 media queries to create design breakpoints
- The shortfalls in our basic example
- Why our journey has only just begun

Defining responsive web design

The term, "responsive web design" was coined by Ethan Marcotte in 2010. In his seminal *A List Apart* article (`http://www.alistapart.com/articles/responsive-web-design/`), he consolidated three existing techniques (flexible grid layout, flexible images/media, and media queries) into one unified approach and named it responsive web design.

Responsive web design in a nutshell

Responsive web design is the presentation of web content in the most relevant format for the viewport and device accessing it.

In its infancy, it was typical for a responsive design to be built starting with the 'desktop', fixed-width design. Content was then reflowed, or removed so that the design worked on smaller screens. However, processes evolved and it became apparent that everything from design, to content and development, worked much better when working in the opposite direction; starting with smaller screens and working up.

Before we get into this, there are a couple of subjects I'd like to address before we continue; browser support and text editors/tooling.

Setting browser support levels

The popularity and ubiquity of responsive web design makes it an easier sell to clients and stakeholders than ever before. Most people have some idea what responsive web design is about. The notion of a single codebase that will just work across all devices is a compelling offering.

One question that almost always comes up when starting a responsive design project is that of browser support. With so many browser and device variants, it's not always pragmatic to support every single browser permutation fully. Perhaps time is a limiting factor, perhaps money. Perhaps both.

Typically, the older the browser, the greater the work and code required to gain feature or aesthetic parity with modern browsers. Therefore, it may make more sense to have a leaner, and therefore faster, codebase by tiering the experience and only providing enhanced visuals and capabilities for more capable browsers.

In the previous edition of this book, some time was spent covering how to cater for very old desktop-only browsers. In this edition, we will not.

As I write this in mid-2015, Internet Explorer 6, 7, and 8 are all but gone. Even IE 9 only has a 2.45% worldwide share of the browser market (IE 10 is only 1.94% while IE 11 is rising nicely at 11.68%). If you have no alternative but to develop for Internet Explorer 8 and below, you have my sympathies and I'm afraid I must be upfront and advise you that there won't be a terrific amount you can use in this book.

For everyone else, you owe it to your client/paymaster to explain why developing for ailing browsers might be a mistake and investing development time and resource primarily for modern browsers and platforms makes good fiscal sense in every respect.

Ultimately however, the only statistics that really matter are yours. In all but extreme cases, the sites we build should at least be functional in every common browser. Beyond basic functionality, for any web project it makes sense to decide, in advance, what platforms you want to fully enhance the experience for, and which you are happy to concede visual/functional anomalies to.

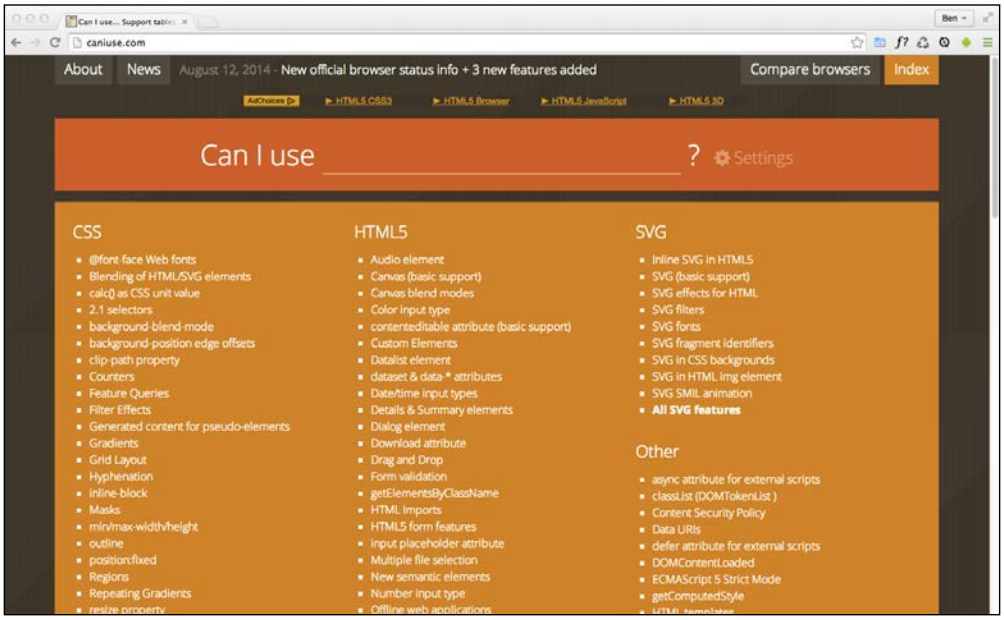
You'll also find that practically, starting with the simplest 'base level' experience and enhancing (an approach known as **progressive enhancement**) is easier than coming at the problem from the opposite direction—building the ultimate experience first then attempting to provide fall backs for less capable platforms (an approach known as **graceful degradation**).

To exemplify why knowing this in advance matters, consider that if you were unlucky enough to have 25% of your website visitors using Internet Explorer 9 (for example), you'd need to consider what features that browser supports and tailor your solution accordingly. The same caution would be required if large amounts of your users are visiting with older mobile phone platforms such as Android 2. What you can consider a 'base' experience will vary depending upon the project.

If suitable data isn't available, I apply a simple and crude piece of logic to determine whether I should spend time developing a particular platform/browser version: if the cost of developing and supporting browser X is more than the revenue/benefit created by the users on browser X; don't develop specific solutions for browser X.

It's rarely a question of whether you could 'fix' an older platform/version. It's a question of whether you should.

When considering which platforms and browser versions support which features, if you aren't already, become familiar the `http://caniuse.com` website. It provides a simple interface for establishing what browser support there is for the features we will be looking at throughout.



A brief note on tooling and text editors

It makes no difference what text editor or IDE system you use to build your responsive web designs. If the simplest of text editors allows you to write your HTML, CSS, and JavaScript efficiently, that's absolutely fine. Similarly there are no requisite pieces of tooling that are essential to get a responsive web design out of the door. All you actually need is something that enables you to write HTML, CSS, and JavaScript. Whether your preference is Sublime Text, Vim, Coda, Visual Studio, or Notepad - it matters little. Just use what works best for you.

However, be aware that there are more tools available now (often free) to negate many of the manual and time-intensive tasks of building web sites than ever before. For example, CSS processors (Sass, LESS, Stylus, PostCSS) can help with code organization, variables, color manipulations, and arithmetic. Tools like PostCSS can also automate horrible and thankless jobs like CSS vendor prefixing. Furthermore, 'Linting' and validation tools can check your HTML, JavaScript, and CSS code against standards as you work, eliminating many time wasting typos or syntax errors.

New tools come out constantly and they are continually improving. Therefore, whilst some relevant and beneficial tools will be mentioned by name as we go, be aware that something better may be just around the corner. Hence we won't be relying on anything other than standards based HTML and CSS in our examples. You should however, use whatever tools you can to produce your front-end code as quickly and reliably as possible.

Our first responsive example

In the first paragraph I promised that by the end of this chapter you would know all you needed to build a fully responsive web page. So far I've just been talking around the issue at hand. It's time to walk the walk.

Code samples

You can download all the code samples from this book by visiting `rwd.education/download.zip` or via GitHub at `https://github.com/benfrain/rwd`. It's worth knowing that where individual examples are built up throughout a chapter, only the final version of the example is provided in the code download. For example, if you download the code samples for *Chapter 2*, *Media Queries – Supporting Differing Viewports*, the examples will be in the state they are at by the end of *Chapter 2*, *Media Queries – Supporting Differing Viewports*. No intermediate states are provided other than in the text.

Our basic HTML file

We will start with a simple HTML5 structure. Don't worry at this point what each of the lines do (especially the content of the `<head>`, we will cover that in detail in *Chapter 4*, *HTML5 for Responsive Web Designs*).

For now, simply concentrate on the elements inside the `<body>` tag. I'm pretty sure nothing there will look too unusual; a few div's, a graphic for a logo, an image (a tasty looking scone), a paragraph or two of text and a list of items.

Here's an abridged version of the code. For brevity I have removed the paragraphs of text in the code below as we only need to concern ourselves with the structure. However, you should know that it's a recipe and description of how to make scones; quintessentially British cakes.

If you want to see the full HTML file, you can download it from the `rwd.education` website.

```
<!doctype html>
<html class="no-js" lang="en">
    <head>
        <meta charset="utf-8">
        <title>Our first responsive web page with HTML5 and CSS3</
title>
        <meta name="description" content="A basic responsive web page
- an example from Chapter 1">
        <link rel="stylesheet" href="css/styles.css">
    </head>
    <body>
        <div class="Header">
            <a href="/" class="LogoWrapper"><img src="img/SOC-Logo.
png" alt="Scone O'Clock logo" /></a>
            <p class="Strap">Scones: the most resplendent of snacks</
p>
        </div>
        <div class="IntroWrapper">
            <p class="IntroText">Occasionally maligned and
misunderstood; the scone is a quintessentially British classic.</p>
            <div class="MoneyShot">
                <img class="MoneyShotImg" src="img/scones.jpg"
alt="Incredible scones" />
                <p class="ImageCaption">Incredible scones, picture
from Wikipedia</p>
            </div>
        </div>
        <p>Recipe and serving suggestions follow.</p>
        <div class="Ingredients">
```

```
            <h3 class="SubHeader">Ingredients</h3>
            <ul>

            </ul>
        </div>
        <div class="HowToMake">
            <h3 class="SubHeader">Method</h3>
            <ol class="MethodWrapper">

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

By default, web pages are flexible. If you were to open the example page, even as it is at this point (with no media queries present), and resize the browser window you'll see the text reflows as needed.

What about on different devices? With no CSS whatsoever, this is how that renders on an iPhone:

As you can see, it's rendering like a 'normal' web page would on an iPhone. The reason for that is that iOS renders web pages at 980px wide by default and shrinks them down into the viewport.

The viewable area of a browser is known technically as the **viewport**. The viewport is seldom equivalent to the screen size of a device, especially in instances where a user can resize a browser window.

Therefore, from now on, we will generally use this more accurate term when we are referring to the available space for our web page.

We can fix that prior problem easily by adding this snippet in the `<head>`:

```
<meta name="viewport" content="width=device-width">
```

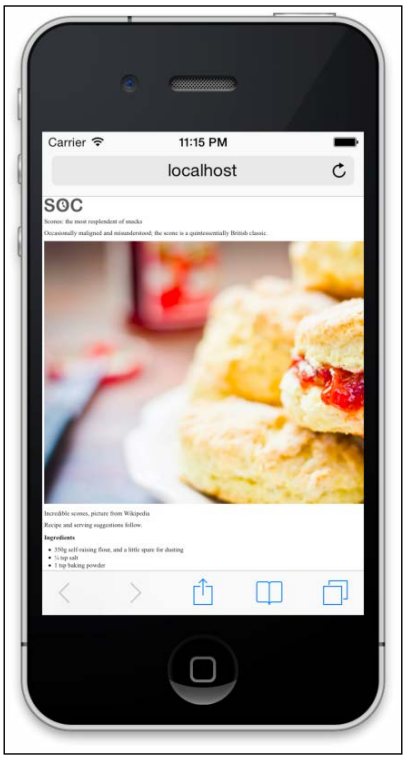
This viewport `meta` tag is a non-standard (but de facto standard) way of telling the browser how to render the page. In this case, our viewport `meta` tag is effectively saying "make the content render at the width of the device". In fact, it's probably easier to just show you the effect this line has on applicable devices:

Great! The text is now rendering and flowing at a more 'native' size. Let's move on.

We will cover the `meta` tag and its various settings and permutations (and the standards based version of the same functionality) in *Chapter 2, Media Queries – Supporting Differing Viewports*.

Taming images

They say a picture is worth a thousand words. All this writing about scones in our sample page and there's no image of the beauties. I'm going to add in an image of a scone near the top of the page; a sort of 'hero' image to entice users to read the page.



Oh! That nice big image (2000px wide) is forcing our page to render more than a little wonky. We need to fix that. We could add a fixed width to the image via CSS but the problem there is that we want the image to scale to different screen sizes.

For example, our example iPhone is 320px wide so we could set a width of 320px to that image but then what happens if a user rotates the screen? The 320px wide viewport is now 480px wide. Thankfully it's pretty easy to achieve fluid images that will scale to the available width of their container with a single line of CSS.