

A Brain-Friendly Guide

Head First Mobile Web



Build
once, run
everywhere



Be more supportive
(of your users)



Find your
way with
geolocation



Put your
pages on a
small-screen diet



Shape-shift your
sites with Responsive
Web Design



O'REILLY®

Lyza Danger Gardner
& Jason Grigsby

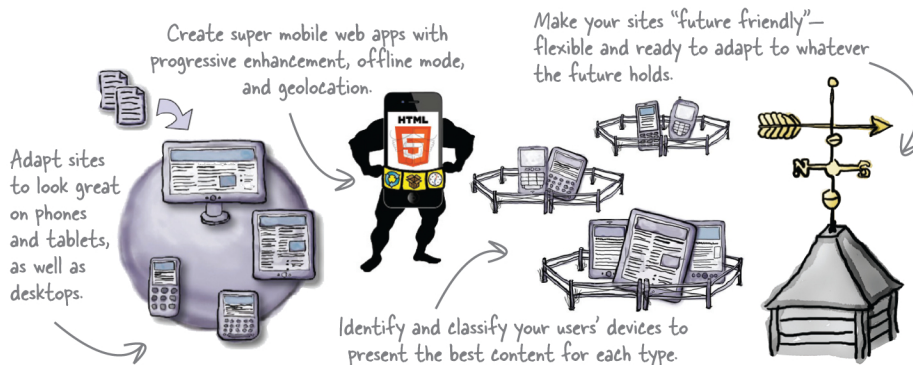
Head First Mobile Web

Mobile

What will you learn from this book?

Mobile web usage is exploding. Soon, more web browsing will take place on phones and tablets than on PCs. Your business needs a mobile strategy, but where do you start? *Head First Mobile Web* shows how to use the web technology you're already familiar with to make sites and apps that work on any device of any size. Put your HTML, JavaScript, and CSS skills to work, and then optimize your site to perform its best in the demanding mobile market. Along the way, you'll discover how to adapt your business strategy by targeting specific devices.

- Navigate the increasingly complex mobile landscape
- Take both technical and strategic approaches to mobile web design
- Use the latest development techniques, including Responsive Web Design and server-side device detection with WURFL
- Learn quickly through images, puzzles, stories, and quizzes



Why does this book look so different?

We think your time is too valuable to waste struggling with new concepts. Using the latest research in cognitive science and learning theory to craft a multisensory learning experience, *Head First Mobile Web* uses a visually rich format designed for the way your brain works, not a text-heavy approach that puts you to sleep.

US \$44.99

CAN \$47.99

ISBN: 978-1-449-30266-5



twitter.com/headfirstlabs
facebook.com/HeadFirst

O'REILLY®

oreilly.com
headfirstlabs.com

“Head First Mobile Web successfully presents practical techniques all readers can use immediately, while giving plenty of foundation and resources for more experienced developers to build upon. Lyza and Jason have created a pragmatic introduction to the chaotic world of mobile web development as it is today, with a glimpse of how we can and should approach it for tomorrow.”

—Stephen Hay,
Founder and principal
of Zero Interface

“An excellent introduction to a complex but exciting topic. Hands-on from the get-go, Head First Mobile Web provides an excellent introduction to the challenges and opportunities available when exploring the next chapter in web design.”

—Stephanie and
Bryan Reiger,
Designers at Yübu

Advance Praise for *Head First Mobile Web*

“If you have been considering buying a book about mobile development that is cross-browser and cross-vendor, you should stop right now and buy *Head First Mobile Web*. It’s written by amazingly smart people [who] have great experience on mobile and don’t stop at one platform, but work on all of them. Many developers spend days arguing [whether] they should go native or web. This book smoothly goes from introductory topics to advanced ones, giving you all the needed information to create exciting content for mobile.”

— **Andrea Trasatti, leader of the DeviceAtlas project and cocreator of the WURFL repository of wireless device capability information**

“A pragmatic introduction to the chaotic world of mobile web development as it is today, with a glimpse of how we can and should approach it for tomorrow. *Head First Mobile Web* successfully presents practical techniques all readers can use immediately, while giving plenty of foundation and resources for more experienced developers to build upon.”

— **Stephen Hay, web designer, developer, speaker, and cofounder of the Mobilism conference**

“Hands-on from the get-go, *Head First Mobile Web* provides an excellent introduction to the challenges and opportunities available when exploring the next chapter in web design.”

— **Bryan and Stephanie Rieger, founders of yiibu.com**

Praise for other *Head First* books

“*Head First Object-Oriented Analysis and Design* is a refreshing look at subject of OOAD. What sets this book apart is its focus on learning. The authors have made the content of OOAD accessible [and] usable for the practitioner.”

— **Ivar Jacobson, Ivar Jacobson Consulting**

“I just finished reading HF OOA&D, and I loved it! The thing I liked most about this book was its focus on why we do OOA&D—to write great software!”

— **Kyle Brown, Distinguished Engineer, IBM**

“Hidden behind the funny pictures and crazy fonts is a serious, intelligent, extremely well-crafted presentation of OO analysis and design. As I read the book, I felt like I was looking over the shoulder of an expert designer who was explaining to me what issues were important at each step, and why.”

— **Edward Sciore, Associate Professor, Computer Science Department, Boston College**

“All in all, *Head First Software Development* is a great resource for anyone wanting to formalize their programming skills in a way that constantly engages the reader on many different levels.”

— **Andy Hudson, Linux Format**

“If you’re a new software developer, *Head First Software Development* will get you started off on the right foot. And if you’re an experienced (read: long-time) developer, don’t be so quick to dismiss this...”

— **Thomas Duff, Duffbert’s Random Musings**

“There’s something in *Head First Java* for everyone. Visual learners, kinesthetic learners, everyone can learn from this book. Visual aids make things easier to remember, and the book is written in a very accessible style—very different from most Java manuals.... *Head First Java* is a valuable book. I can see the *Head First* books used in the classroom, whether in high schools or adult ed classes. And I will definitely be referring back to this book, and referring others to it as well.”

— **Warren Kelly, Blogcritics.org, March 2006**

“Rather than textbook-style learning, *Head First iPhone and iPad Development* brings a humorous, engaging, and even enjoyable approach to learning iOS development. With coverage of key technologies including core data, and even crucial aspects such as interface design, the content is aptly chosen and top-notch. Where else could you witness a fireside chat between a UIWebView and UITextField!”

— **Sean Murphy, iOS designer and developer**

Praise for other *Head First* books

“Another nice thing about *Head First Java*, Second Edition, is that it whets the appetite for more. With later coverage of more advanced topics such as Swing and RMI, you just can’t wait to dive into those APIs and code that flawless, 100,000-line program on java.net that will bring you fame and venture-capital fortune. There’s also a great deal of material, and even some best practices, on networking and threads—my own weak spot. In this case, I couldn’t help but crack up a little when the authors use a 1950s telephone operator—yeah, you got it, that lady with a beehive hairdo that manually hooks in patch lines—as an analogy for TCP/IP ports...you really should go to the bookstore and thumb through *Head First Java*, Second Edition. Even if you already know Java, you may pick up a thing or two. And if not, just thumbing through the pages is a great deal of fun.”

— **Robert Eckstein, Java.sun.com**

“Of course it’s not the range of material that makes *Head First Java* stand out, it’s the style and approach. This book is about as far removed from a computer science textbook or technical manual as you can get. The use of cartoons, quizzes, fridge magnets (yep, fridge magnets...). And, in place of the usual kind of reader exercises, you are asked to pretend to be the compiler and compile the code, or perhaps to piece some code together by filling in the blanks or...you get the picture.... The first edition of this book was one of our recommended titles for those new to Java and objects. This new edition doesn’t disappoint and rightfully steps into the shoes of its predecessor. If you are one of those people who falls asleep with a traditional computer book, then this one is likely to keep you awake and learning.”

— **TechBookReport.com**

“*Head First Web Design* is your ticket to mastering all of these complex topics, and understanding what’s really going on in the world of web design.... If you have not been baptized by fire in using something as involved as Dreamweaver, then this book will be a great way to learn good web design.”

— **Robert Pritchett, MacCompanion**

“Is it possible to learn real web design from a book format? *Head First Web Design* is the key to designing user-friendly sites, from customer requirements to hand-drawn storyboards to online sites that work well. What sets this apart from other ‘how to build a website’ books is that it uses the latest research in cognitive science and learning to provide a visual learning experience rich in images and designed for how the brain works and learns best. The result is a powerful tribute to web design basics that any general-interest computer library will find an important key to success.”

— **Diane C. Donovan, [California Bookwatch: The Computer Shelf](http://CaliforniaBookwatch.com)**

“I definitely recommend *Head First Web Design* to all of my fellow programmers who want to get a grip on the more artistic side of the business.”

— **Claron Twitchell, UJUG**

Other related books from O'Reilly

jQuery Cookbook

jQuery Pocket Reference

jQuery Mobile

JavaScript and jQuery: The Missing Manual

Other books in O'Reilly's *Head First* series

Head First C#

Head First Java

Head First Object-Oriented Analysis and Design (OOA&D)

Head First HTML with CSS and XHTML

Head First Design Patterns

Head First Servlets and JSP

Head First EJB

Head First SQL

Head First Software Development

Head First JavaScript

Head First Physics

Head First Statistics

Head First Ajax

Head First Rails

Head First Algebra

Head First PHP & MySQL

Head First PMP

Head First Web Design

Head First Networking

Head First iPhone and iPad Development

Head First jQuery

Head First HTML5 Programming

Head First Mobile Web

Wouldn't it be dreamy if there were a book to help me learn how to build mobile websites that was more fun than going to the dentist? It's probably nothing but a fantasy...



Lyza Danger Gardner
Jason Grigsby

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Head First Mobile Web

by Lyza Danger Gardner and Jason Grigsby

Copyright © 2012 Cloud Four, Inc. All rights reserved.

Printed in the United States of America.

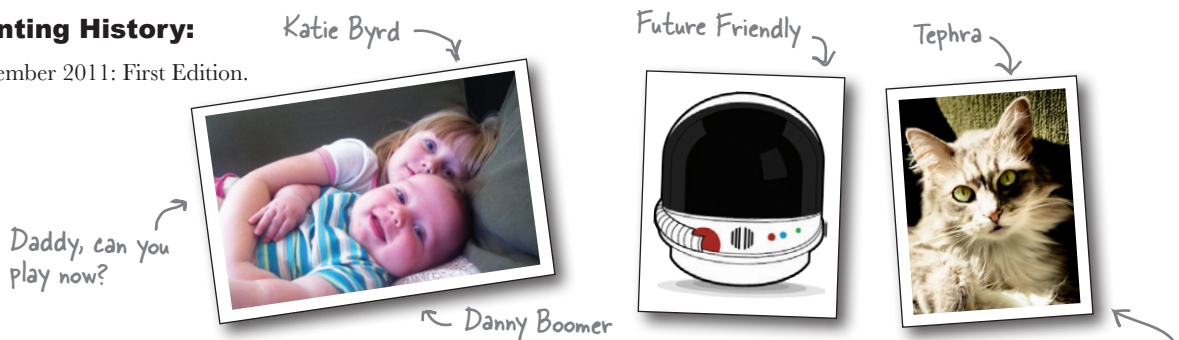
Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Series Creators:	Kathy Sierra, Bert Bates
Editor:	Courtney Nash
Design Editor:	Louise Barr
Cover Designer:	Karen Montgomery
Production Editor:	Kristen Borg
Production Services:	Rachel Monaghan
Indexer:	Ginny Munroe
Page Viewers:	Katie Byrd, Danny Boomer, the Future-Friendly Helmet, and Tephra

Printing History:

December 2011: First Edition.



The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. The *Head First* series designations, *Head First Mobile Web*, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and the authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

No feature phones were harmed in the making of this book.

ISBN: 978-1-449-30266-5

[M]

[2012-02-03]

To the phenomenal women in my family: my sister, Maggie; Momula, Fran; Aunt Catherine; stepmother, Christie; and above all, to the memory of my grandmother, Pearl, whose fierce and literate independence inspired generations.

—**Lyza**

To my parents for buying that Commodore 64 so many years ago; to my lovely wife, Dana, without whose support and understanding this book wouldn't have happened; and to Katie and Danny—yes, Daddy can play now.

—**Jason**



Lyza Danger Gardner (@lyzadanger) is a dev. She has built, broken, and hacked web things since 1996. Curiously, Lyza was actually born and raised in Portland, Oregon, the town where everyone wants to be but no one seems to be *from*.

Lyza started college early and cobbled together a motley education: a BA in Arts and Letters from Portland State University, followed by a master's program in computer science at the University of Birmingham (UK).

Lyza has written a lot of web applications (server-side devs, represent!), defeated wily content management systems, optimized mobile websites, pounded on various APIs, and worried a lot about databases. Fascinated by the way mobile technology has changed things, she now spends a lot of time thinking about the future of the Web, mobile and otherwise.

Since cofounding Cloud Four, a Portland-based mobile web agency, in 2007, Lyza has voyaged further into the deep, untrammelled reaches of Device Land, exploring the foibles and chaos of mobile browsers and the mobile web. She has an odd set of anachronistic hobbies, and it has been said that she takes a fair number of photographs. She owns a four-letter .com domain. We'll bet you can guess what it is and go visit her there.



Jason

In 2000, **Jason Grigsby** got his first mobile phone. He became obsessed with how the world could be a better place if everyone had access to the world's information in their pockets. When his wife, Dana, met him, he had covered the walls of his apartment with crazy mobile dreams. To this day, he remains baffled that she married him.

Those mobile dreams hit the hard wall of reality—WAP was crap. So Jason went to work on the Web until 2007, when the iPhone made it clear the time was right. He joined forces with the three smartest people he knew and started Cloud Four.

Since cofounding Cloud Four, he has had the good fortune to work on many fantastic projects, including the Obama iPhone App. He is founder and president of Mobile Portland, a local nonprofit dedicated to promoting the mobile community in Portland, Oregon.

Jason is a sought-after speaker and consultant on mobile. If anything, he is more mobile obsessed now than he was in 2000 (sorry, sweetheart!).

You can find him blogging at <http://cloudfour.com>; on his personal site, <http://userfirstweb.com>; and on Twitter as @grigs. Please say hello!

Table of Contents (Summary)

	Intro	xxi
1	Getting Started on the Mobile Web: <i>Responsive Web Design</i>	1
2	Responsible Responsiveness: <i>Mobile-first Responsive Web Design</i>	43
3	A Separate Mobile Website: <i>Facing less-than-awesome circumstances</i>	91
4	Deciding Whom to Support: <i>What devices should we support?</i>	137
5	Device Databases and Classes: <i>Get with the group</i>	151
6	Build a Mobile Web App Using a Framework: <i>The Tartanator</i>	217
7	Mobile Web Apps in the Real World: <i>Super mobile web apps</i>	267
8	Build Hybrid Mobile Apps with PhoneGap: <i>Tartan Hunt: Going native</i>	313
9	How to Be Future Friendly: <i>Make (some) sense of the chaos</i>	357
i	Leftovers: <i>The top six things (we didn't cover)</i>	373
ii	Set Up Your Web Server Environment: <i>Gotta start somewhere</i>	387
iii	Install WURFL: <i>Sniffing out devices</i>	397
iv	Install the Android SDK and Tools: <i>Take care of the environment</i>	403
	Index	417

Table of Contents (the real thing)

Intro

Your brain on mobile web. Here you are trying to learn something, while here your brain is, doing you a favor by making sure the learning doesn't stick. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether setting this BlackBerry Bold on fire is going to activate the sprinkler system." So how do you trick your brain into thinking that your life depends on knowing mobile web?

Who is this book for?	xxii
We know what you're thinking	xxiii
And we know what your brain is thinking	xxiii
Metacognition: thinking about thinking	xxv
The technical review team	xxx
Acknowledgments	xxxi

getting started on the mobile web

1

Responsive Web Design

Hey there! Are you ready to jump into mobile?

Mobile web development is a wildly exciting way of life. There's glamour and excitement, and plenty of *Eureka!* moments. But there is also mystery and confusion. Mobile technology is evolving at bewildering speed, and there's so much to know! Hang tight. We'll start our journey by showing you a way of making websites called **Responsive Web Design (RWD)**. You'll be able to adapt websites to look great on a whole lot of mobile devices by building on the web skills you already have.

index.html



styles.css



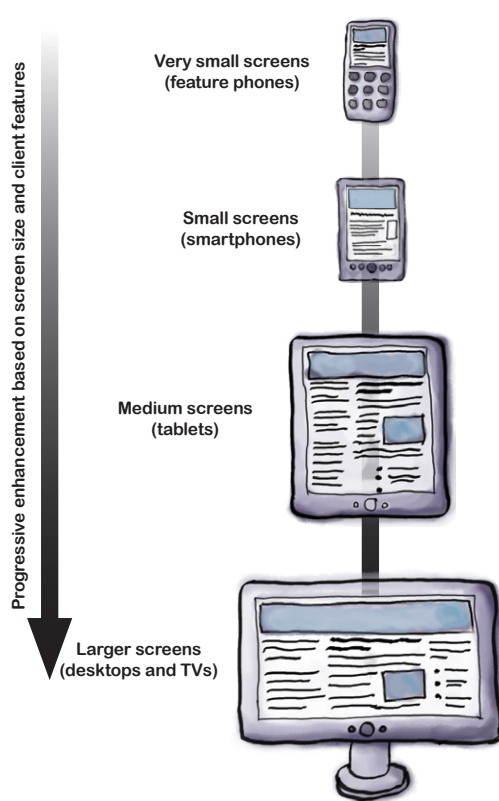
Get on the mobile bandwagon	2
Something odd happened on the way to the pub	4
If mobile phone web browsers are so great...	5
What's so different about the mobile web?	6
Responsive Web Design	10
Different CSS in different places	12
CSS media queries	13
The current structure of the Splendid Walrus site	15
Analyze the current CSS	16
What needs to change?	17
Identify the CSS that needs to change	18
Steps to creating the mobile-specific CSS	19
What's wrong with a fixed-width layout, anyway?	26
How is fluid better?	27
The fluid formula	28
Continue your fluid conversion	29
Context switching	31
What's wrong with this picture?	32
Fluid images and media	33
Remember to be responsible	36
That's a responsive site!	40
Responsive design is also a state of mind	41

responsible responsiveness

2

Mobile-first Responsive Web Design**That's a beautiful mobile site. But beauty is only skin deep.**

Under the covers, it's a different thing entirely. It may look like a mobile site, but it's still a desktop site in mobile clothing. If we want this site to be greased lightning on mobile, we need to start with **mobile first**. We'll begin by dissecting the current site to find the desktop bones hiding in its mobile closet. We'll clean house and start fresh with **progressive enhancement**, building from the basic content all the way to a desktop view. When we're done, you'll have a page that is optimized regardless of the screen size.

	Just when you thought it was time to celebrate...	44
	Is there really a problem? How do we know?	45
	What to do when things aren't blazing fast	47
	Don't let its looks fool you, that's a BIG page	48
	There's gold in 'em HAR hills	49
	Find the drags on page speed	51
	Where did that Google Maps JavaScript come from?	53
	It looks mobile friendly, but it isn't	55
	Mobile-first Responsive Web Design	56
	What is progressive enhancement?	57
	Fix the content floats	60
	Mobile-first media queries	61
	Surprise! The page is broken in Internet Explorer	62
	One src to rule them all	68
	Zoom in on the viewport <meta> tag	72
	The right to zoom?	73
	Add the map back using JavaScript	74
	Build a pseudo-media query in JavaScript	76
	Add the JavaScript to the On Tap Now page	77
	These widgets aren't responsive	79
	Move iframe attributes to CSS equivalents	80
	Remove attributes from the JavaScript	81
	The map overlap is back	83
	Let the content be your guide	84
	Breakpoints to the rescue	87

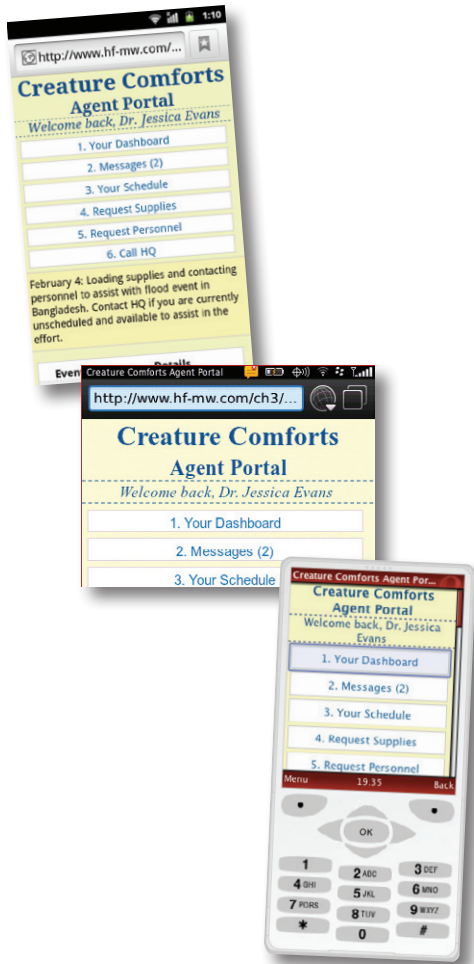
a separate mobile website

3

Facing less-than-awesome circumstances

The vision of a single, responsive Web is a beautiful one...

in which every site has one layout to rule them all, made lovingly with a mobile-first approach. Mmm...tasty. But what happens when a stinky dose of reality sets in? Like legacy systems, older devices, or customer budget constraints? What if, sometimes, instead of mixing desktop and mobile support into one lovely soup, you need to keep 'em separated? In this chapter, we look at the nitty-gritty bits of **detecting mobile users, supporting those cruffy older phones, and building a separate mobile site.**



Creature Comforts has agents in the field	92
How can agents get and share the info they need?	93
Send mobile users to a mobile-optimized website	96
Sniff out mobile users	97
Getting to know user agents	98
User agents: spawn of Satan?	101
Straight talk: Most major sites have a separate mobile website	104
When what you really want to do is (re-)direct	105
Take a peek at the script	106
How does the script work?	107
Make a mobile mockup	108
Special delivery...of complicating factors	110
Not all phones are smartphones...not by a sight	113
Let's keep it basic: Meet XHTML-MP	114
Why would we want to use that old thing?	115
Keep your nose clean with XHTML-MP	116
By the way, scrolling sucks	119
One last curveball	119
Access keys in action	123
What went wrong?	124
Fix the errors	125
Mobile-savvy CSS	127
Hmmm...something is missing	132
The button look is sorely missed!	133
Great success!	134

deciding whom to support

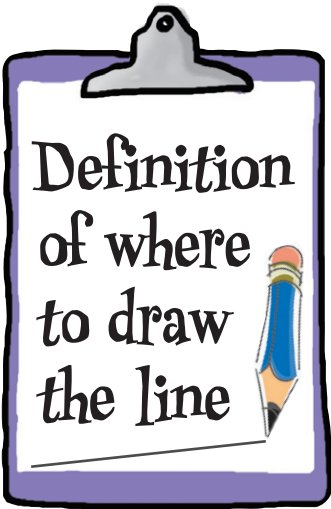
4

What devices should we support?

There aren't enough hours in the day to test on every device.

You have to draw the line somewhere on what you can support. **But how do you decide?** What about people using devices you can't test on—are they left out in the cold? Or is it possible to build your web pages in a way that will reach people on devices you've never heard of? In this chapter, we're going to mix a magic concoction of **project requirements** and **audience usage** to help us figure out **what devices we support** and **what to do about those we don't**.

How do you know where to draw the line?	138
Step away from the keyboard for a second	139
Things you <i>don't</i> support vs. those you <i>can't</i> support	140
Ask questions about your project	142
Ingredients for your magic mobile potion	144
Draw from your cupboard of tools and data	145
How do I know my customers have the right stuff?	150

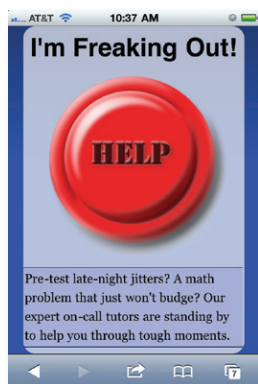
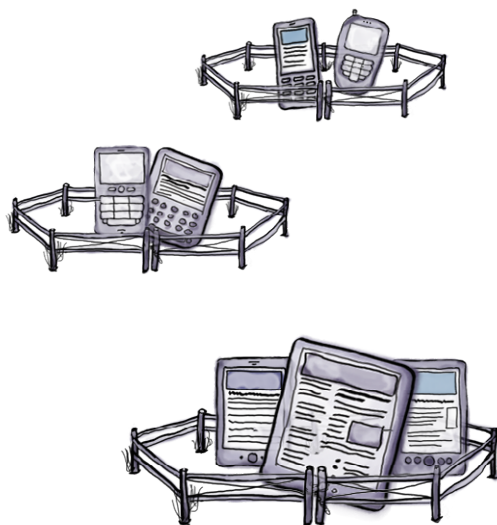


device databases and classes

5

Get with the group**Setting the bar for the devices we support doesn't take care of a few nagging issues.**

How do we find out enough stuff about our users' mobile browsers to know if they measure up before we deliver content to them? How do we avoid only building (lame) content for the lowest common denominator? And how do we organize all of this stuff so that we don't lose our minds? In this chapter, we'll enter the realm of **device capabilities**, learn to access them with a **device database**, and, finally, discover how to group them into **device classes** so that we can keep our sanity.



A panic button for freaked-out students	152
Mobile device data sources to the rescue	154
Meet WURFL	155
WURFL and its capabilities	156
WURFL: Clever API code	159
We can build an explore page, too	160
An explore page: Setting up our environment	161
A quick one-two punch to improve our explore page	168
Put capabilities to work	170
Use WURFL to help differentiate content	170
Initialize the device and get the info ready	172
Is this thing mobile?	172
Make the page a bit smarter with WURFL	176
The panic button: For phones only	177
Device classes	181
Expanding a lucrative part of AcedIt!'s business	182
Evaluate the home page wearing mobile-tinted glasses	183
Group requirements into multiple mobile flavors	184
Rounding out our device classes	185
Get acquainted with the matching function	191
What's going on in that switch statement?	192
Use the matching function to test capabilities	193
Fill in the gaps in the device class tests	200
We need a bigger safety net	211
A stitch in time	212

build a mobile web app using a framework

6 The Tartanator

“We want an app!” Just a year or two ago, that hallmark cry generally meant one thing: native code development and deployment for each platform you wanted to support. But native isn’t the only game in town. These days, web-based apps for mobile browsers have some street cred—especially now that hip cat **HTML5** and his sidekicks, **CSS3** and **JavaScript**, are in the house. Let’s dip our toes into the mobile web app world by taking a **mobile framework**—code tools designed to help you get your job done quickly—for a spin!

Hmmm...it's...nice,
but can you make
it feel more...like a
native app?



HTML5...app...what do these words even mean?	219
How “traditional” websites typically behave	220
How applike websites often behave	221
The master plan for phase 1 of the Tartanator	224
Why use mobile web app frameworks?	225
Our choice for the Tartanator: jQuery Mobile	226
Build a basic page with jQuery Mobile	228
Mark up the rest of the page	229
The HTML5 data-* attribute	231
Link to multiple pages with jQuery Mobile	234
Take the list from blah to better	241
Drop in the rest of the tartans	243
Filter and organize a list	244
Add a footer toolbar	249
Make the toolbar snazzy	250
Finalize the structure	251
Time to make that tartan-building form	253
Translate tartan patterns to a form	255
Build an HTML5 form	256
It’s time to add some basic fields	257
Lists within lists let the users add colors	258
Color-size ingredient pairs: The color select field	259
Color-size field pairs: The size field	260
Link to the form	262

mobile web apps in the real world

7

Super mobile web apps

The mobile web feels like that gifted kid in the class.

You know, kind of fascinating, capable of amazing things, but also a mysterious, unpredictable troublemaker. We've tried to keep its hyperactive genius in check by being mindful of constraints and establishing boundaries, but now it's time to capitalize on some of the mobile web's natural talents. We can use **progressive enhancement** to spruce up the interface in more precocious browsers and transform erratic connectivity from a burden to a feature by crafting a thoughtful **offline mode**. And we can get at the essence of mobility by using **geolocation**. Let's go make this a super mobile web app!



It looks nice...	268
Mobile apps in the real world	270
Ready, set, enhance!	274
Make a better form	275
A widget to manage the list of colors and sizes	276
A peek under the hood	277
So, that's the frontend enhancement...	278
...and now for the backend	280
The two sides of generate.php	281
One last thing!	282
Offline is important	284
A basic recipe to create a cache manifest	285
Dev tools to the rescue	286
Serve the manifest as the correct content-type	287
Victory is (finally) ours	297
How geolocation works	298
How to ask W3C-compliant browsers where they are	299
Start in on the Find Events page: The baseline	301
Let's integrate geolocation	303
Nothing found	309

build hybrid mobile apps with PhoneGap

8

Tartan Hunt: Going native

Sometimes you've got to go native. It might be because you need access to something not available in mobile browsers (yet). Or maybe your client simply *must* have an app in the App Store. We look forward to that shiny future when we have access to everything we want in the browser, and mobile web apps share that sparkly allure native apps enjoy. Until then, we have the option of **hybrid development**—we continue writing our **code using web standards**, and use a **library to bridge the gaps** between our code and the device's native capabilities. **Cross-platform native apps built from web technologies?** Not such a bad compromise, eh?



Opportunity knocks again	314
How do hybrid apps work?	317
Bridge the web-native gap with PhoneGap	318
Get acquainted with PhoneGap Build	321
How will the app work?	322
Keep track of discovered tartans	323
Anatomy of the Tartan Hunt project	324
Download your apps	328
Choose your adventure	329
Who's seen what? Store found tartans	334
What can localStorage do for us?	335
Check out what a browser supports	339
Use a function to show which tartans are found	340
The toggle and toggleClass methods	341
You found a tartan, eh? Prove it!	344
Rope in PhoneGap to take pictures	345
PhoneGap is almost ready for its close-up	347
Now we're ready for the mediaCapture API	348
How will we handle the success?	349
It always looks a bit different in real life	350
It's just a bit anonymous	351
One last thing!	353
We nailed it!	354

how to be future friendly

Make (some) sense of the chaos

9

Responsive Web Design. Device detection. Mobile web apps. PhoneGap. Wait...which one should we use?

There are an overwhelming number of ways to develop for the mobile web. Often, projects will involve **multiple techniques used in combination**. There is no single right answer. But don't worry. The key is to learn to go with the flow. **Embrace the uncertainty**. Adopt a **future-friendly mindset** and ride the wave, confident that you're flexible and ready to adapt to whatever the future holds.



Now what?	358
Time to dispel our collective illusions of control	361
A future-friendly manifesto	362
If you can't be future proof, be future friendly	364
App today, web page tomorrow	365
It's a long journey: Here are some guideposts	366
Mix up a batch of mobile goodness	369
Look toward the future	371



leftovers



The top six things (we didn't cover)

Ever feel like something's missing? We know what you mean... Just when you thought you were done, there's more.

We couldn't leave you without a few extra details, things we just couldn't fit into the rest of the book. At least, not if you want to be able to carry this book around without a metallic case and caster wheels on the bottom. So take a peek and see what you (still) might be missing out on.

#1. Testing on mobile devices	374
#2. Remote debugging	376
#3. Determine which browsers support what	382
#4. Device APIs	384
#5. Application stores and distribution	385
#6. RESS: REsponsive design + Server-Side components	386

set up your web server environment



Gotta start somewhere

You can't spell "mobile web" without the "web." There are no two ways about it. You're going to need a web server if you want to develop for the mobile web. That goes for more than just completing the exercises in this book. You need somewhere to put your web-hosted stuff, whether you use a third-party commercial web hosting service, an enterprise-class data center, or your own computer. In this appendix, we'll walk you through the steps of **setting up a local web server** on your computer and **getting PHP going** using free and open source software.

What we need from you	388
Only available locally	389
Windows and Linux: Install and configure XAMPP	390
Get going with XAMPP	391
Mac folks: It's MAMP time	392
Make sure you dock at the right port	393
Access your web server	394
phpInfo, please!	396



install WURFL

Sniffing out devices

The first step to solving device detection mysteries

is a bit of legwork. Any decent gumshoe knows we've got to gather our clues and interrogate our witnesses. First, we need to seek out the brains of the operation: the **WURFL PHP API**. Then we'll go track down the brawn: capability information for thousands of devices in a single **XML data file**. But it'll take a bit of coaxing to get the two to spill the whole story, so we'll tweak a bit of **configuration** and take some careful notes.

Who's got the brains?	398
And who's got the brawn?	399
Getting the two to work together	400
A bit of filesystem housekeeping	401
Take note!	402

install the Android SDK and tools



Take care of the environment

To be the master of testing native Android apps, you need

to be environmentally aware. You'll need to turn your computer into a nice little ecosystem where you can herd Android apps to and from virtual (emulated) or real devices. To make you the shepherd of your Android sheep, we'll show you how to download the **Android software development kit (SDK)**, how to install some **platform tools**, how to **create some virtual devices**, and how to **install and uninstall apps**.

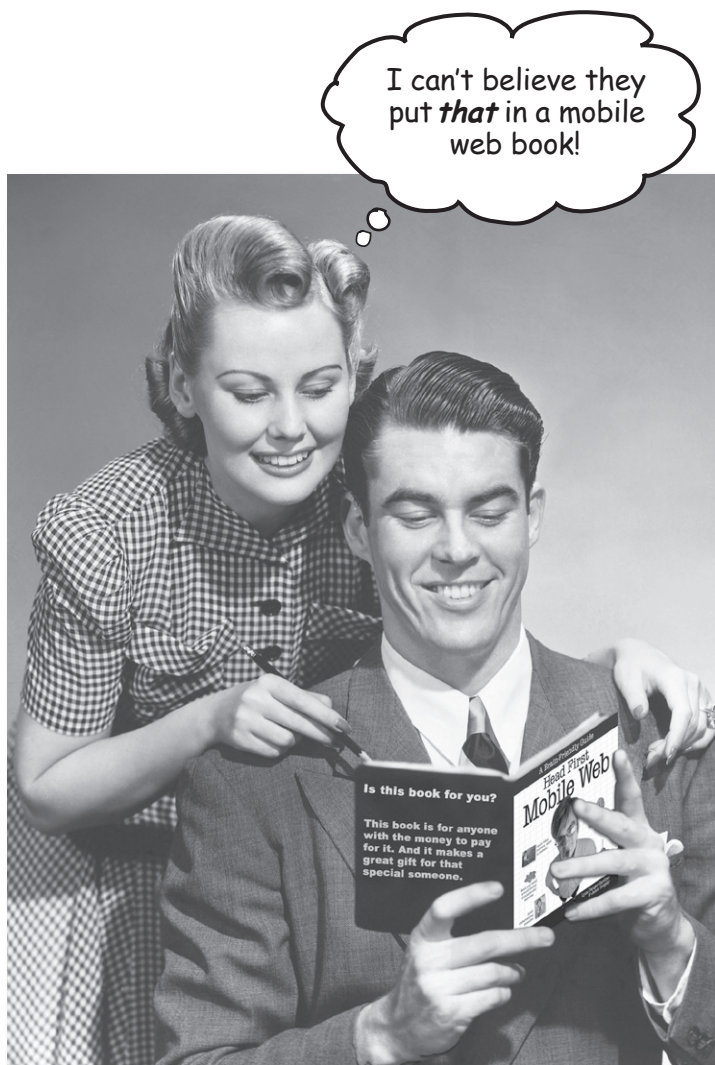
Let's download the Android SDK	404
Get the right tools for the job	405
Create a new virtual device	408
Find the right PATH	413

Index

417

how to use this book

Intro



In this section, we answer the burning question:
"So why DID they put that in a Mobile Web book?"

Who is this book for?

If you can answer “yes” to all of these:

- ① Do you have previous web design and development experience?
- ② Do you want to **learn, understand, remember**, and **apply** important mobile web concepts so that you can make your mobile web pages more interactive and exciting?
- ③ Do you prefer **stimulating dinner-party conversation** to **dry, dull, academic lectures**?

It definitely helps if you've already got some scripting chops, too. We're not talking rocket science, but you shouldn't feel visceral panic if you see a JavaScript snippet.

this book is for you.

Who should probably back away from this book?

If you can answer “yes” to any of these:

- ① Are you **completely new** to web development?
- ② Are you already developing mobile web apps or sites and looking for a **reference book** on mobile web?
- ③ Are you **afraid to try something different**? Would you rather have a root canal than endure the suggestion that there might be more than one true way to build for the Web? Do you believe that a technical book can't be serious if there's a walrus-themed pub and an app called the Tartanator in it?

this book is not for you.



[Note from marketing: this book is for anyone with a credit card. Or cash. Cash is nice, too. — Ed]

We know what you're thinking

"How can *this* be a serious mobile web development book?"

"What's with all the graphics?"

"Can I actually *learn* it this way?"

And we know what your *brain* is thinking

Your brain craves novelty. It's always searching, scanning, *waiting* for something unusual. It was built that way, and it helps you stay alive.

So what does your brain do with all the routine, ordinary, normal things you encounter? Everything it *can* to stop them from interfering with the brain's *real* job—recording things that *matter*. It doesn't bother saving the boring things; they never make it past the "this is obviously not important" filter.

How does your brain *know* what's important? Suppose you're out for a day hike and a tiger jumps in front of you. What happens inside your head and body?

Neurons fire. Emotions crank up. *Chemicals surge*.

And that's how your brain knows...

This must be important! Don't forget it!

But imagine you're at home, or in a library. It's a safe, warm, tiger-free zone. You're studying. Getting ready for an exam. Or trying to learn some tough technical topic your boss thinks will take a week, 10 days at the most.

Just one problem. Your brain's trying to do you a big favor. It's trying to make sure that this *obviously* nonimportant content doesn't clutter up scarce resources. Resources that are better spent storing the really *big* things. Like tigers. Like the danger of fire. Like how you should never again snowboard in shorts.

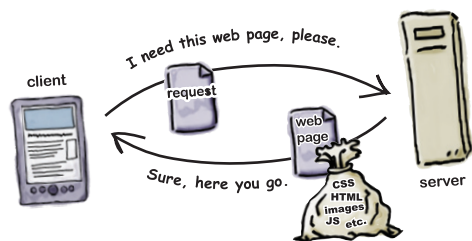
And there's no simple way to tell your brain, "Hey brain, thank you very much, but no matter how dull this book is, and how little I'm registering on the emotional Richter scale right now, I really *do* want you to keep this stuff around."



We think of a “Head First” reader as a learner.

So what does it take to *learn* something? First, you have to *get* it, and then make sure you don’t *forget* it. It’s not about pushing facts into your head. Based on the latest research in cognitive science, neurobiology, and educational psychology, *learning* takes a lot more than text on a page. We know what turns your brain on.

Some of the Head First learning principles:



Make it visual. Images are far more memorable than words alone, and make learning much more effective (up to 89% improvement in recall and transfer studies). It also makes things more understandable.

Put the words within or near the graphics they

Watch out, mobile web!
Here we come!

relate to, rather than on the bottom or on another page, and learners will be up to twice as likely to solve problems related to the content.

Use a conversational and personalized style. In recent studies, students performed up to 40% better on post-learning tests if the content spoke directly to the reader, using a first-person, conversational style rather than taking a formal tone. Tell stories instead of lecturing. Use casual language. Don’t take yourself too seriously. Which would you pay more attention to: a stimulating dinner-party companion, or a lecture?

Get the learner to think more deeply. In other words, unless you actively flex your neurons, nothing much happens in your head. A reader has to be motivated, engaged, curious, and inspired to solve problems, draw conclusions, and generate new knowledge. And for that, you need challenges, exercises, and thought-provoking questions, and activities that involve both sides of the brain and multiple senses.

Get—and keep—the reader’s attention. We’ve all had the

“I really want to learn this, but I can’t stay awake past page one” experience. Your brain pays attention to things that are out of the ordinary, interesting, strange, eye-catching, unexpected. Learning a new, tough, technical topic doesn’t have to be boring. Your brain will learn much more quickly if it’s not.

Touch their emotions. We now know that your ability to remember something is largely dependent on its emotional content. You remember what you care about. You remember when you *feel* something. No, we’re not talking heart-wrenching stories about a boy and his dog. We’re talking emotions like surprise, curiosity, fun, “what the...?”, and the feeling of “I rule!” that comes when you solve a puzzle, learn something everybody else thinks is hard, or realize you know something that “I’m more technical than thou” Bob from Engineering *doesn’t*.



Metacognition: thinking about thinking

If you really want to learn, and you want to learn more quickly and more deeply, pay attention to how you pay attention. Think about how you think. Learn how you learn.

Most of us did not take courses on metacognition or learning theory when we were growing up. We were *expected* to learn, but rarely *taught* to learn.

But we assume that if you're holding this book, you really want to learn about mobile web development. And you probably don't want to spend a lot of time. And since you're going to build more sites and apps in the future, you need to *remember* what you read. And for that, you've got to *understand* it. To get the most from this book, or *any* book or learning experience, take responsibility for your brain. Your brain on *this* content.

The trick is to get your brain to see the new material you're learning as Really Important. Crucial to your well-being. As important as a tiger. Otherwise, you're in for a constant battle, with your brain doing its best to keep the new content from sticking.

So just how do you get your brain to think that mobile web development is a hungry tiger?

There's the slow, tedious way, or the faster, more effective way. The slow way is about sheer repetition. You obviously know that you *are* able to learn and remember even the dullest of topics if you keep pounding the same thing into your brain. With enough repetition, your brain says, "This doesn't *feel* important to him, but he keeps looking at the same thing *over* and *over* and *over*, so I suppose it must be."

The faster way is to do **anything that increases brain activity**, especially different *types* of brain activity. The things on the previous page are a big part of the solution, and they're all things that have been proven to help your brain work in your favor. For example, studies show that putting words *within* the pictures they describe (as opposed to somewhere else in the page, like a caption or in the body text) causes your brain to try to make sense of how the words and picture relate, and this causes more neurons to fire. More neurons firing = more chances for your brain to *get* that this is something worth paying attention to, and possibly recording.

A conversational style helps because people tend to pay more attention when they perceive that they're in a conversation, since they're expected to follow along and hold up their end. The amazing thing is, your brain doesn't necessarily *care* that the "conversation" is between you and a book! On the other hand, if the writing style is formal and dry, your brain perceives it the same way you experience being lectured to while sitting in a roomful of passive attendees. No need to stay awake.

But pictures and conversational style are just the beginning.



Here's what WE did:

We used **pictures**, because your brain is tuned for visuals, not text. As far as your brain's concerned, a picture really *is* worth a thousand words. And when text and pictures work together, we embedded the text *in* the pictures because your brain works more effectively when the text is *within* the thing the text refers to, as opposed to in a caption or buried in the text somewhere.

We used **redundancy**, saying the same thing in *different* ways and with different media types, and *multiple senses*, to increase the chance that the content gets coded into more than one area of your brain.

We used concepts and pictures in **unexpected** ways because your brain is tuned for novelty, and we used pictures and ideas with at least *some emotional content*, because your brain is tuned to pay attention to the biochemistry of emotions. That which causes you to *feel* something is more likely to be remembered, even if that feeling is nothing more than a little **humor**, **surprise**, or **interest**.

We used a personalized, **conversational style**, because your brain is tuned to pay more attention when it believes you're in a conversation than if it thinks you're passively listening to a presentation. Your brain does this even when you're *reading*.

We included loads of **activities**, because your brain is tuned to learn and remember more when you **do** things than when you *read* about things. And we made the exercises challenging-yet-doable, because that's what most people prefer.

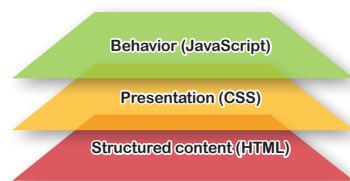
We used **multiple learning styles**, because *you* might prefer step-by-step procedures, while someone else wants to understand the big picture first, and someone else just wants to see an example. But regardless of your own learning preference, *everyone* benefits from seeing the same content represented in multiple ways.

We include content for **both sides of your brain**, because the more of your brain you engage, the more likely you are to learn and remember, and the longer you can stay focused. Since working one side of the brain often means giving the other side a chance to rest, you can be more productive at learning for a longer period of time.

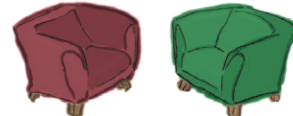
And we included **stories** and exercises that present **more than one point of view**, because your brain is tuned to learn more deeply when it's forced to make evaluations and judgments.

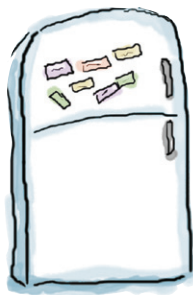
We included **challenges**, with exercises, and by asking **questions** that don't always have a straight answer, because your brain is tuned to learn and remember when it has to *work* at something. Think about it—you can't get your *body* in shape just by *watching* people at the gym. But we did our best to make sure that when you're working hard, it's on the *right* things. That **you're not spending one extra dendrite** processing a hard-to-understand example, or parsing difficult, jargon-laden, or overly terse text.

We used **people**. In stories, examples, pictures, etc., because, well, *you're* a person. And your brain pays more attention to *people* than it does to *things*.



Fireside Chats





Cut this out and stick it on your refrigerator.

Here's what YOU can do to bend your brain into submission

So, we did our part. The rest is up to you. These tips are a starting point; listen to your brain and figure out what works for you and what doesn't. Try new things.

1 Slow down. The more you understand, the less you have to memorize.

Don't just *read*. Stop and think. When the book asks you a question, don't just skip to the answer. Imagine that someone really *is* asking the question. The more deeply you force your brain to think, the better chance you have of learning and remembering.

2 Do the exercises. Write your own notes.

We put them in, but if we did them for you, that would be like having someone else do your workouts for you. And don't just *look* at the exercises. **Use a pencil.** There's plenty of evidence that physical activity *while* learning can increase the learning.

3 Read "There Are No Dumb Questions."

That means all of them. They're not optional sidebars—***they're part of the core content!*** Don't skip them.

4 Make this the last thing you read before bed. Or at least the last challenging thing.

Part of the learning (especially the transfer to long-term memory) happens *after* you put the book down. Your brain needs time on its own, to do more processing. If you put in something new during that processing time, some of what you just learned will be lost.

5 Drink water. Lots of it.

Your brain works best in a nice bath of fluid. Dehydration (which can happen before you ever feel thirsty) decreases cognitive function.

6 Talk about it. Out loud.

Speaking activates a different part of the brain. If you're trying to understand something, or increase your chance of remembering it later, say it out loud. Better still, try to explain it out loud to someone else. You'll learn more quickly, and you might uncover ideas you hadn't known were there when you were reading about it.

7 Listen to your brain.

Pay attention to whether your brain is getting overloaded. If you find yourself starting to skim the surface or forget what you just read, it's time for a break. Once you go past a certain point, you won't learn faster by trying to shove more in, and you might even hurt the process.

8 Feel something!

Your brain needs to know that this *matters*. Get involved with the stories. Make up your own captions for the photos. Groaning over a bad joke is *still* better than feeling nothing at all.

9 Create something!

Apply this to your daily work; use what you are learning to make decisions on your projects. Just do something to get some experience beyond the exercises and activities in this book. All you need is a pencil and a problem to solve...a problem that might benefit from using the tools and techniques you're studying for the exam.

Read me

This is a learning experience, not a reference book. We deliberately stripped out everything that might get in the way of learning whatever it is we're working on at that point in the book. And the first time through, you need to begin at the beginning, because the book makes assumptions about what you've already seen and learned.

We expect you to know HTML and CSS.

If you don't know HTML and CSS, pick up a copy of *Head First HTML with CSS & XHTML* before starting this book. We'll explain some of the more obscure CSS selectors or HTML elements, but don't expect to learn about that foundational stuff here.

We expect you to feel comfy around web scripting code.

We're not asking you to be a world-class JavaScript expert or to have done a graduate computer science project using PHP, but you'll see examples using both languages throughout the book. If the merest notion of a `for` loop makes you hyperventilate (or if you have no idea what we're talking about), you might consider tracking down a copy of *Head First PHP & MySQL* or *Head First JavaScript* and then heading on back here.

We expect you to know how to track things down.

We'll be blunt. The mobile web is an enormous topic, and mastering it involves expanding your *existing* web development skills. There are too many things to know about the Web for any one person to memorize, whether it's a detail of JavaScript syntax or the specifics of a browser's support for an HTML5 element attribute. Don't be too hard on yourself. Part of the toolset of a good web dev is keeping your Google chops sharp and knowing when and how to hit the Web to look up info about web topics. We bet you're good at that already.

We expect you to go beyond this book.

It's a big and beautiful mobile web world out there. We hope we can give you a shove to start you on your journey, but it's up to you to keep up your steam. Seek out the active mobile web community online, read blogs, join mailing lists that are up your alley, and attend related technical events in your area.

The activities are NOT optional.

The exercises and activities are not add-ons; they're part of the core content of the book. Some of them are to help with memory, some are for understanding, and some will help you apply what you've learned. ***Don't skip the exercises.*** They're good for giving your brain a chance to think about the ideas and terms you've been learning in a different context.

The redundancy is intentional and important.

One distinct difference in a Head First book is that we want you to *really* get it. And we want you to finish the book remembering what you've learned. Most reference books don't have retention and recall as a goal, but this book is about *learning*, so you'll see some of the same concepts come up more than once.

The Brain Power exercises don't have answers.

For some of them, there is no right answer, and for others, part of the learning experience of the exercise is for you to decide if and when your answers are right. In some of the Brain Power exercises, you will find hints to point you in the right direction.

Software requirements

As for developing any website, you need a text editor, a browser, a web server (it can be locally hosted on your personal computer), and the source code for the chapter examples.

The text editors we recommend for Windows are PSPad, TextPad, or EditPlus (but you can use Notepad if you have to). The text editors we recommend for Mac are TextWrangler (or its big brother, BBEdit) or TextMate. We also like Coda, a more web-focused tool.

If you're on a Linux system, you've got plenty of text editors built in, and we trust you don't need us to tell you about them.

If you are going to do web development, you need a web server. You'll need to go to Appendix ii, which details installing a web server with PHP. We recommend doing that now. No, seriously, head there now, follow the instructions, and come back to this page when you're done.

For Chapter 5, you'll need to install the WURFL (Wireless Universal Resource FiLe) API and data. And for Chapter 8, you'll need the Android SDK and some related tools. You guessed it: there are appendixes for those tasks, too.

You'll also need a browser—no, strike that—**as many browsers as possible** for testing. And the more **mobile devices with browsers** you have on hand, the better (don't panic; there are many emulators you can use if you don't have hardware).

For developing and testing on the desktop, we highly recommend Google's Chrome browser, which has versions for Mac, Windows, and Linux. Learning how to use the JavaScript console in Google's Chrome Dev Tools is well worth the time. This is homework you need to do on your own.

Last of all, you'll need to get the code and resources for the examples in the chapters. It's all available at <http://hf-mw.com>.

↑
The hf-mw.com site has the starting point of code for all the chapters. Head on over there and get downloading.

The code and resources for the examples in the chapters are all available at <http://hf-mw.com>.

The technical review team



Trevor Farlow is an amateur baker, recreational soccer player, and part-time animal shelter volunteer. When he's not walking dogs, scoring goals, or perfecting his New York-style cheesecake, he can be found learning the art of product ownership in a lean, mean, agile development team at Clearwater Analytics, LLC.

Brad Frost is a mobile web strategist and frontend developer at R/GA in New York City, where he works with large brands on mobile-related projects. He runs a resource site called Mobile Web Best Practices (<http://mobilewebbestpractices.com>) aimed at helping people create great mobile web experiences.

Stephen Hay has been building websites for more than 16 years. Aside from his client work, which focuses increasingly on multiplatform design and development, he speaks at industry events and has written for publications such as *A List Apart* and *.net Magazine*. He also co-organizes Mobilism, a highly respected mobile web design and development conference.

Ethan Marcotte is an independent designer/developer who is passionate about beautiful design, elegant code, and the intersection of the two. Over the years, his clientele has included *New York Magazine*, the Sundance Film Festival, the *Boston Globe*, and the W3C. Ethan coined the term *Responsive Web Design* to describe a new way of designing for the ever-changing Web and, if given the chance, will natter on excitedly about it—he even went so far as to write a book on the topic.

Bryan Rieger is a designer and reluctant developer with a background in theatre design and classical animation. Bryan has worked across various media including print, broadcast, web, and mobile; and with clients such as Apple, Microsoft, Nokia, and the Symbian Foundation. A passionate storyteller and incessant tinkerer, Bryan can be found crafting a diverse range of experiences at Yiibu—a wee design consultancy based in Edinburgh, Scotland.

Stephanie Rieger is a designer, writer, and closet anthropologist with a passion for the many ways people interact with technology. Stephanie has been designing for mobile since 2004 and now focuses primarily on web strategy, frontend design, and optimization for multiple screens and capabilities. A compulsive tester and researcher, Stephanie is always keen to discover and share insights on mobile usage, user behavior, and mobility trends from around the world.

Andrea Trasatti started creating WAP content in 1999 on the Nokia 7110, which in Europe was considered groundbreaking at the time. Andrea has led both WURFL and DeviceAtlas from their earliest days to success, and during those years built vast experience in device detection and content adaptation. You can find Andrea on Twitter as *@AndreaTrasatti*, regularly talking about mobile web and new trends in creating and managing content for mobile.

Acknowledgments

Our editor:

Thanks (and congratulations!) to **Courtney Nash**, who pushed us to create the best book we possibly could. She endured a huge raft of emails, questions, ramblings, and occasional crankiness. She stuck with us throughout this book and trusted us to trust our guts. And thanks to **Brian Sawyer** for stepping up at the end and taking us over the finish line.



Courtney Nash ↗

The O'Reilly team:

Thanks to **Lou Barr** for her unfathomably speedy and masterful design and layout magic. We're seriously blown away here. Thank you. Our gratitude goes to **Karen Shaner** and **Rachel Monaghan** for all the help juggling drafts, reviewers, and details!

Thanks to the rest of the O'Reilly folks who made us feel so welcomed: **Mike Hendrickson**, for suggesting this crazy idea in the first place; **Brady Forrest**, for introducing and championing us; **Tim O'Reilly**, for being the genuine, smart, and nice guy that he is; and **Sara Winge**, for her graciousness and overall awesomeness.

Our thanks:

Jason and Lyza work with the smartest people ever at Cloud Four. Our epic thanks to fellow cofounders **Aileen Jeffries** and **John Keith**, and the rest of the Cloud Four team: **Matt Gifford**, **Chris Higgins**, and **Megan Notarte**. This book is really a product of our collective mobile web obsession, and they, more than anyone, championed and endured this effort. Thanks a billion million zillion, you guys.

We'd also like to thank the mobile web community. In particular, we'd like to thank Josh Clark, Gail Rahn Frederick, Scott Jehl, Scott Jenson, Dave Johnson, Tim Kadlec, Jeremy Keith, Peter-Paul Koch, Brian LeRoux, James Pearce, Steve Souders, and Luke Wroblewski. We're proud and thankful to be part of this community.

Lyza's friends and family:

Thanks to **Bryan Christopher Fox** (Other Dev), without whose coding chops, insight, support, and all-around supergenius this book would not have been possible.

Huge shout-outs to my **friends** and **family**, who still seem to put up with me despite my long-term disappearance into Book Land. Thanks to Autumn and Amye, who showed stunning tenacity in the face of my constant unavailability. Thanks, Mike, always. And thanks to Dad, who always shows me how to find aesthetic and new adventure. Finally, thanks to Huw and Bethan of Plas-yn-Iâl, Llandegla, Wales, a fantastic, sheep-happy place where about a quarter of this book was written.

Jason's friends and family:

Thank you to my family for all of their support. Our parents, **Jan**, **Carol**, **Mark**, and **Doanne**, were a tremendous help in keeping our sanity as we juggled book writing, family, and moving.

Special thanks to my wife, **Dana Grigsby**, for making it possible for me to work on a book while we raised a baby and a preschooler and moved into a new house. I couldn't have done it without you.



↖ Lou Barr

Safari® Books Online



Safari® Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at <http://my.safaribooksonline.com>.

1 getting started on the mobile web

✧ *Responsive Web Design* ✧



Dashing, exciting, fascinating,
and oh-so-popular...but am I
ready to take the plunge?

Hey there! Are you ready to jump into mobile?

Mobile web development is a wildly exciting way of life. There's glamour and excitement, and plenty of ***Eureka!*** moments. But there is also mystery and confusion. Mobile technology is evolving at bewildering speed, and there's so much to know! Hang tight. We'll start our journey by showing you a way of making websites called ***Responsive Web Design (RWD)***. You'll be able to adapt websites to look great on a whole lot of mobile devices by building on the web skills you already have.

Get on the mobile bandwagon

There's a pretty good chance you own a mobile phone. We know that not simply because you bought this book (smart move, by the way!), but because it's hard to find someone who doesn't own a mobile phone.

It doesn't matter where you go in the world. Mobile phones are being used everywhere, from farmers in Nigeria using their mobiles to find which market has the best price for their crops, to half of Japan's top 10 best-selling novels being consumed and written—*yes, written*—on mobile phones.

At the beginning of 2011, there were 5.2 billion phones being used by the 6.9 billion people on Earth. **More people use mobile phones than have working toilets or toothbrushes.**

The time is now

So yeah, mobile is huge, but it's been big for years. Why should you get on the mobile bandwagon now?

Because **the iPhone changed everything**. It sounds clichéd, but it is true. There were app stores, touchscreens, and web browsers on phones before the iPhone, but Apple was the first to put them together in a way that made it easy for people to understand and use.





Everyone has iPhones. And if they don't, are they really going to browse the Web?

The iPhone is fantastic, but people use a lot of different phones for a lot of different reasons. And the most popular phones are likely to change.

We have no way of knowing what the the leading phones will be when you read this book. Three years ago, Android was a mere blip on the radar. In 2011, it is a leading smartphone platform worldwide.

Mobile technology changes quickly, but there are a few things we feel confident about:

- 1 Every new phone has a web browser in it.**
You can probably find a new phone that doesn't have a web browser in it, but you have to look pretty hard. Even the most basic phones now come with decent browsers. Everyone wants the Web on their phone.
- 2 Mobile web usage will exceed desktop web usage.**
Soon the number of people accessing the Web via mobile phones will surpass those who use a computer. Already, many people say they use their phones more frequently than their PCs.
- 3 The Web is the only true cross-platform technology.**
iPhone, Android, BlackBerry, Windows Phone, WebOS, Symbian, Bada—there are more phone platforms than we can keep track of. Each one has its own specific programming hooks, meaning that if you want to write software for each, you have to start from scratch each time.

Mobile web has its own challenges, but there is no other technology that allows you to create content and apps that reach every platform.

So you're in the right spot at the right time. Mobile web is taking off, and you're ready to ride the rocketship. Let's get started!

Something odd happened on the way to the pub

Mike is the proprietor of The Splendid Walrus, a pub with a clever name and a cult-like following of local beer enthusiasts. Mike always has unusual beers on tap and highlights several of them on his website.

Before he realized his lifelong dream of pub ownership, Mike was a web developer. So he had no trouble putting together a respectable website for The Splendid Walrus himself.

The Splendid Walrus website is pretty sweet—I used to do this for a living, after all.



<http://www.splendidwalrus.com>

If mobile phone web browsers are so great...

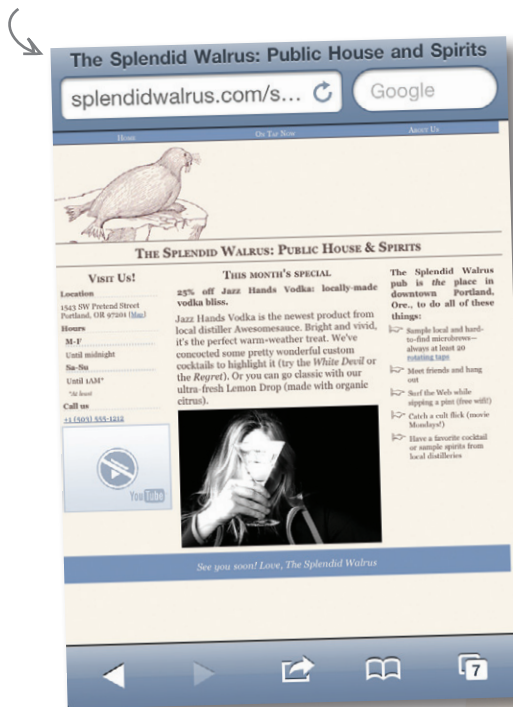
Mike built the Splendid Walrus website several years ago, when mobile browsing was still rudimentary and uncommon. It was made for—and tested in—desktop browsers like Firefox, Internet Explorer, and Safari.

Lots of newer mobile browsers have good reputations. They're increasingly sophisticated and powerful, and starting to feel like some of their desktop counterparts.

...shouldn't this just work?

Mike had a rude awakening when he looked at the Splendid Walrus site on his iPhone 4. It didn't look so hot on a friend's Android device, either.


Here's how the Splendid Walrus site looks on an iPhone 4...



...and here's how the site looks on a Motorola Backflip Android phone.



What's so different about the mobile web?



My iPhone has the Safari web browser on it. My site looks great in desktop Safari, so why does it look all messed up on my phone?

1 There are 86 billion different mobile web browsers.

OK, not quite that many. But when you're developing for the mobile web, sometimes it feels this way. Unlike the handful of leading desktop browsers, there are hundreds of different mobile browsers. Yikes.

And just when you think you're on top of all of them, a new one will pop up in, like, Thailand.

2 Support for web technologies varies wildly.

On older mobile browsers (or even recent ones on less powerful devices), you can pretty much forget about reliable CSS or JavaScript. Even the newest browsers lack support for some things, support them in bewilderingly different ways, or have weird bugs. It's the Wild West out here, folks!

3 Mobile devices are smaller and slower.

Yeah, we know. Newer mobile devices are state-of-the-art pocket computers. But they still pale in comparison to desktop (or laptop) computers in terms of processing power. Mobile networks can be flaky and downright poky, and data transfer is not necessarily free or unlimited. This means we'll need to think about putting our sweet but enormous, media-rich, complex sites on a performance-savvy diet.

4 Mobile interfaces require us to rethink our sites.

Just because a mobile browser can render a desktop website with few hiccups doesn't mean it necessarily *should*. Screens are smaller; interactions and expectations are different.

People with mobile devices use all sorts of input devices: fingers, stylus pens, the little nubbins they have on BlackBerry devices. Typing and filling out forms can be tedious at best. Squinting at type designed to fit a desktop browser window can give your users headaches and fury. You get the idea.



Exercise

Here's how Mike's iPhone 4 renders the Splendid Walrus website. It doesn't look so great. Can you spot the problem areas? Mark any problems you see.



1

.....

.....

.....

.....

2

.....

.....

.....

.....

3

.....

.....

.....

.....

4

.....

.....

.....

.....



Exercise Solution

Did you spot some of these problem areas?

1 The navigation links are all tiny and too small to read or click.

2 The embedded YouTube video doesn't work.

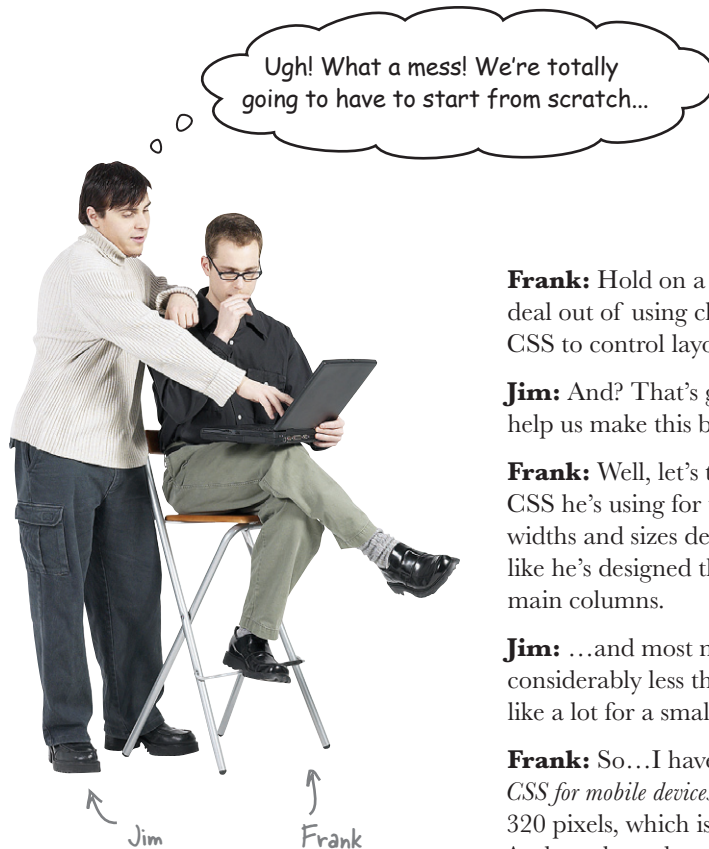


3 The three-column layout feels tight on this screen resolution, and the text is hard to read.

4 There is a weird gap on the right edge of the screen.

This is confusing and embarrassing. I want my customers with mobile devices to see a nice site. I'm out of my depth here. Can you help?





Frank: Hold on a minute. We know that Mike makes a big deal out of using clean, semantic HTML markup and uses CSS to control layout and styling as much as possible.

Jim: And? That's great and professional, but how does it help us make this better?

Frank: Well, let's think about this a bit. When I look at the CSS he's using for the Splendid Walrus site, I see a lot of widths and sizes defined to fit within a 960-pixel box. It looks like he's designed the site on a 960-pixel grid, with three main columns.

Jim: ...and most mobile devices have resolutions considerably less than 960 pixels. Also, three columns seems like a lot for a smaller screen.

Frank: So...I have to wonder...*what if we could use different CSS for mobile devices?* Say, maybe, CSS designed to lay out in 320 pixels, which is the width of a lot of smartphone screens? And maybe reduce the number of columns?

Jim: Nice idea, Frank. But I don't see how we could do that without a lot of server-side programming. I mean, how do we get mobile devices to use completely different CSS?

Frank: You know how Jill just got back from the Awesome Cool Mobile Web Camp conference and is all excited about that thing called *Responsive Web Design*?

Jim: How could I forget? It's all she's been talking about.

Frank: Well, she says it's getting a lot of attention from web developers and it sounds like it involves, at least in part, applying different CSS for different situations, without having to do heavy-duty programming. Apparently it's especially useful for developing mobile websites. I can't really remember the details, but maybe we should check it out.

Responsive Web Design

Responsive Web Design (RWD) is a set of techniques championed by web designer Ethan Marcotte. Sites designed with this approach adapt their layouts according to the environment of the user's browser, in large part by doing some nifty things with CSS.

Depending on the current value of certain browser conditions like window size, device orientation, or aspect ratio, we can apply different CSS in different circumstances. By rethinking the way we do page layouts, we can make formerly one-size-fits-all column and grid layouts flow more naturally across a continuum of browser window sizes.

Read Ethan's original article for A List Apart about RWD at <http://bit.ly/nRePnj>.

RWD is one of the simplest and quickest ways to make a website work handsomely on a lot of devices—and you can use the web skills you already have.

The recipe for Responsive Web Design

There are three primary techniques for building a responsively designed website:

1 CSS3 media queries

Evaluating certain aspects of the current browser environment to determine which CSS to apply.

We can apply different CSS rules based on things like browser window width, aspect ratio, and orientation.

2 Fluid-grid layouts

Using relative CSS proportions instead of absolute sizes for page layout elements.

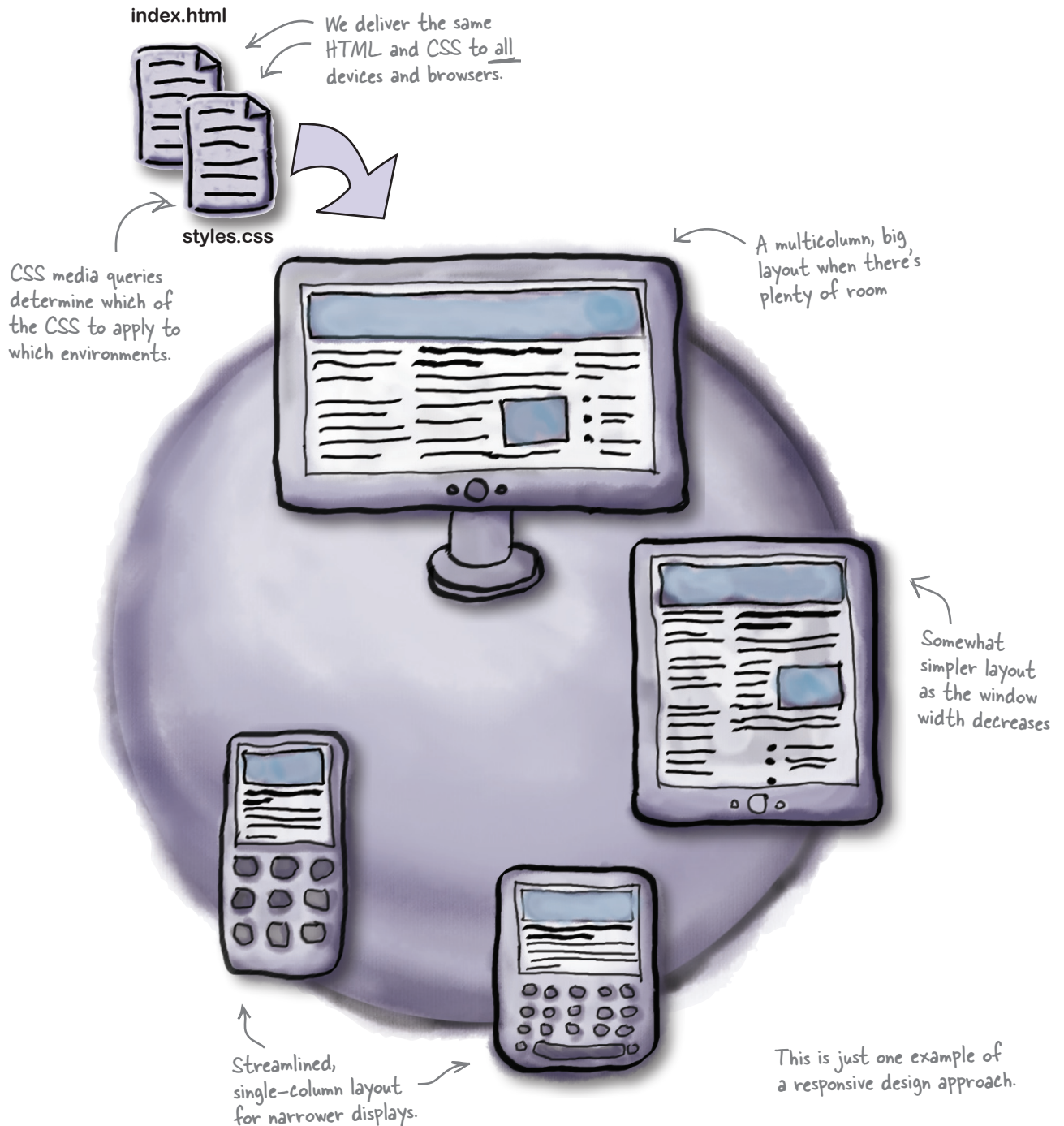
RWD uses percentages instead of pixels as units for columns and other layout elements.

3 Fluid images and media

Making our images and media scale to fit within the size constraints of their containers by using some CSS tricks.

Fluid images and media keep within the bounds of their parent elements, scaling proportionally with the rest of the layout.

An example of a responsively designed site



Different CSS in different places

If you've been doing web development for some time (and are CSS-savvy), you might be friends with **CSS media types** already.

We can use @media rules to apply CSS selectively.

CSS media type declarations inside of a CSS file look like this:

```
@media screen { /* CSS Rules for screens! */ }
```

"screen" is a media type.

The rules between the braces will only apply when the content is rendered on a screen.

Another way to use media types to apply CSS selectively is from within a <link> in your HTML document.

```
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

The rules in this external stylesheet will only be applied if the content is rendered on a print device (that is, a printer).

"print" is another media type.

Referencing the print media type like this is a common approach to creating print stylesheets—that is, CSS styles that only get applied when the content is printed.

Media types, meet media features

You have certain features—your age, your height—and so do media types. And just like The Splendid Walrus might want to establish a rule that requires the minimum age of patrons to be 21 before they apply alcohol, we might want to define certain CSS that we only apply to browser window widths within a certain range.

We're in luck! width, along with color and orientation, is one of the **media features** defined in CSS3 for all common media types. So, again, media **types** have media **features**.

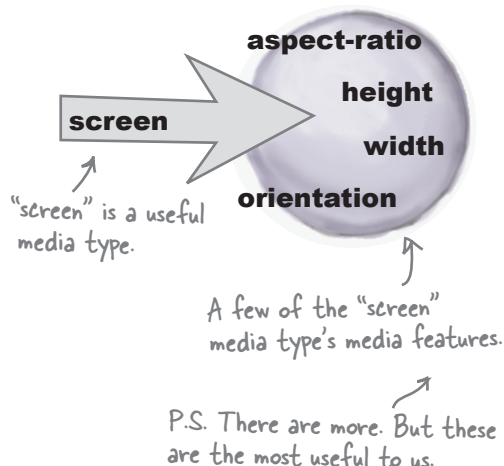
Media features on their own don't get us very far. We need a way to ask the browser about the states of the ones we care about and, well, do something about it. That's where **CSS3 media queries** come in.

Media Types Up Close



Common (and useful) media types include screen, print, and all. There are other, less common media types like aural, braille, and tv.

Curious? If you're the kind of person who reads technical specs for fun or to satisfy curiosity, you can see all of the media types defined in CSS2 on the W3C's site at www.w3.org/TR/CSS2/media.html.



CSS media queries

“screen” media type, we meet again!

“width” is a media feature we want to evaluate on the “screen” media type.

These CSS rules will only get applied if the media query evaluates to TRUE.

```
@media screen and (min-width:480px) { /* CSS Rules */ }
```

“min-” is a media query prefix. Rather intuitively, it means we want to query about a minimum width.

Unsurprisingly, there is also a “max-” prefix.

This means: **are we presently rendering content on a screen, AND is the window *currently* at least 480 pixels wide?**

Yes? OK! Apply these CSS rules.

Another example:

```
@media print, screen and (monochrome) { }
```

Logical “or” is represented by a comma. Yep, it’s a bit confusing.

“monochrome” is a media feature of the “screen” media type. It is either TRUE or FALSE.

Is this being rendered on a printer *OR* is it being rendered on a screen that is monochrome (black and white)?

Yes? Use these styles!

CSS3 media queries are logical expressions that evaluate the current values of media features in the user’s browser. If the media query expression evaluates as TRUE, the contained CSS is applied.



Exercise

Translating CSS media queries: You try it! Match the media query and its meaning.

```
@media all and (orientation: landscape) {}
```

```
<link rel="stylesheet" type="text/css"
href="my.css" media="screen and (color)" />
```

```
@media print and (monochrome) {}
```

```
@media screen and (color) { }
```

Apply the rules in this external stylesheet to color screens.

Apply these styles to black-and-white printers.

Apply these rules to color screens.

Apply these styles to all media types when in landscape orientation.



Exercise Solution

Were you able to decipher the media queries?

Media type of "all," you ask? Yep. This is what we use if we want to look at the same media feature across all media types

```
@media all and (orientation: landscape) {}
```

```
<link rel="stylesheet" type="text/css"
href="my.css" media="screen and (color)" />
```

```
@media print and (monochrome) {}
```

```
@media screen and (color) { }
```

Apply the rules in this external stylesheet to color screens.

Apply these styles to black-and-white printers.

Apply these rules to color screens.

Apply these styles to all media types when in landscape orientation.

OK. Now I can understand media queries and maybe even write my own. But what am I doing here? How do I write the CSS for mobile devices?



CSS: How different is different?

We have a tool that lets us apply different CSS to different situations. But now what?

Don't panic. We do need to write some mobile-friendly CSS, but we're not going to have to start from scratch. Nor are we going to have to have totally different CSS for our mobile devices—we can share a lot of what's already there.

To generate our mobile-friendly layout, we'll:

- ☐ Check out the current layout of splendidwalrus.com and analyze its structure.
- ☐ Identify layout pieces that need to change to work better on mobile browsers.
- ☐ Generate mobile-adapted CSS for those identified elements.
- ☐ Organize our CSS and selectively apply the mobile and desktop CSS using media queries.

We'll only write different CSS for those layout elements that need to be different for mobile.