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
# Table of Contents (Summary)

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

# 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?

<table>
<thead>
<tr>
<th>Who is this book for?</th>
<th>xxii</th>
</tr>
</thead>
<tbody>
<tr>
<td>We know what you’re thinking</td>
<td>xxiii</td>
</tr>
<tr>
<td>And we know what your brain is thinking</td>
<td>xxiii</td>
</tr>
<tr>
<td>Metacognition: thinking about thinking</td>
<td>xxv</td>
</tr>
<tr>
<td>The technical review team</td>
<td>xxx</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>xxxi</td>
</tr>
</tbody>
</table>
getting started on the mobile web

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.
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.
a separate mobile website

**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 crufty older phones, and building a separate mobile site.

Creature Comforts has agents in the field

How can agents get and share the info they need?

Send mobile users to a mobile-optimized website

Sniff out mobile users

Getting to know user agents

User agents; spawn of Satan?

Straight talk: Most major sites have a separate mobile website

When what you really want to do is (re-)direct

Take a peek at the script

How does the script work?

Make a mobile mockup

Special delivery...of complicating factors

Not all phones are smartphones...not by a sight

Let's keep it basic: Meet XHTML-MP

Why would we want to use that old thing?

Keep your nose clean with XHTML-MP

By the way, scrolling sucks

One last curveball

Access keys in action

What went wrong?

Fix the errors

Mobile-savvy CSS

Hmmm...something is missing

The button look is sorely missed!

Great success!
deciding whom to support

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 don’t support vs. those you can’t 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
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  
Mobile device data sources to the rescue  
Meet WURFL  
WURFL and its capabilities  
WURFL: Clever API code  
We can build an explore page, too  
An explore page: Setting up our environment  
A quick one-two punch to improve our explore page  
Put capabilities to work  
Use WURFL to help differentiate content  
Initialize the device and get the info ready  
Is this thing mobile?  
Make the page a bit smarter with WURFL  
The panic button: For phones only  
Device classes  
Expanding a lucrative part of Accelelet’s business  
Evaluate the home page wearing mobile-tinted glasses  
Group requirements into multiple mobile flavors  
Rounding out our device classes  
Get acquainted with the matching function  
What’s going on in that switch statement?  
Use the matching function to test capabilities  
Fill in the gaps in the device class tests  
We need a bigger safety net  
A stitch in time
build a mobile web app using a framework

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!

HTML5...app...what do these words even mean? 219
How "traditional" websites typically behave 220
How app-like 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
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!
build hybrid mobile apps with PhoneGap

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

Tartan Hunt

FIND THE TARTANS & WIN A TRIP TO EDINBURGH!
how to be future friendly

Make (some) sense of the chaos

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