SAMPLE CHAPTER

Apache Cordova IN ACTION

Raymond K. Camden

MANNING

Apache Cordova in Action

by Raymond K. Camden

Sample Chapter 5

brief contents

5 Mobile design and user experience

This chapter covers

- What works (and what doesn't) on mobile devices
- How to use Bootstrap for responsive, mobile-optimized design
- An overview of Mobile UI frameworks

So far we've discussed how to install Cordova, how to generate native binaries from HTML, and how to make use of fancy device features with plugins. For the most part, what we've discussed has been fairly straightforward. Install an SDK, install the command-line tool, write some HTML, and whammo!, see it on your device.

5.1 Congratulations—you're a (horrible) mobile developer!

Okay, that may be just a tiny bit over the top, but most likely there's a bit of truth to it as well. What we haven't yet discussed is how to create a *good* mobile application. Taste is subjective. While it's difficult to precisely describe what makes a good mobile application, there are definitely guidelines that help define what a successful mobile application looks like. And notice I'm not saying a successful *hybrid* mobile

application. Your users don't care what you used to build your application. They only care about the end result. Therefore, the guidelines for a good hybrid mobile application are going to be the same as a good 100% native-built mobile application.

A good mobile application is readable on a variety of form factors. Whether opened in an iPhone 5 or some enormous Android phablet, text should be readable and buttons easy to click with fat fingers. A good mobile application demonstrates these features:

- It has a simple, easily understandable UI. By using common design idioms (a shopping cart icon, for example) users have a better idea of what to expect when using your application.
- It performs well with little to no noticeable lag.
- It works in a variety of network conditions (offline and online).

5.1.1 A good example of a bad UI

Imagine the most simple application possible—an application that prompts for your name and then tells you hello. Figure 5.1 is a mockup of the UI for the application, both the initial view and what's displayed after entering a name.

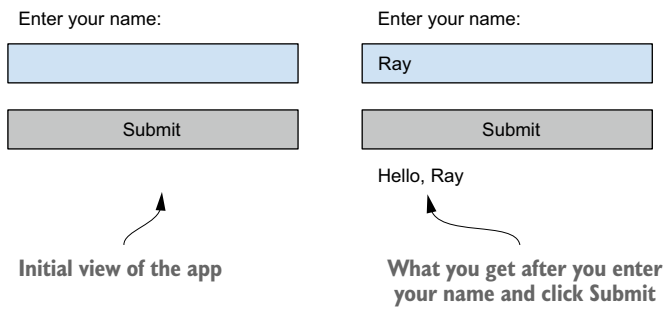

Figure 5.1 A simple little application. Looks pretty now, right?

Building this application would be rather trivial. You should create a new Cordova application if you want to test this. The source code is available in the zip downloaded from the book's website. You'll find it in the c5/simple folder. The following listing represents the HTML used for the application.

Listing 5.1 Simple application HTML (simple/www/index.html)

```
<html>
    <head>
        <title>Simple App</title>
    </head>
    <body>

        <form id="nameForm">
            <label for="name">Enter your name</label>
            <input type="text" id="name" name="name">
            <input type="submit" value="Submit">
        </form>
```

Prompts for your name

User enters name

Submits form

```
            <div id="result"></div>

            <script src="cordova.js"></script>
            <script src="js/app.js"></script>
    </body>
</html>
```

Empty div that will be filled with user's name

There isn't anything particularly interesting about this code, but note the lack of any styling via an embedded or included CSS file. That's totally okay—you *don't* have to style anything, but as you can probably guess, this is going to bite us in the rear in a few moments. Now look at the JavaScript in the next listing.

Listing 5.2 Simple application JavaScript (simple/www/js/app.js)

```
document.addEventListener("deviceready", init, false);
function init() {

    document.querySelector("#nameForm").addEventListener("submit",
            function(e) {
                    e.preventDefault();
                    var name = document.querySelector("#name").value;
                    var msg = "Hello, "+name;
                    document.querySelector("#result").innerText = msg;
    }, false);

}
```

So far so good. Because the application has one feature (get the name and display it), the code is trivial to the point of being pointless. To be clear, this is *not* something you want to use Cordova for, but it will be very useful in demonstrating the type of design issues you're going to be running into when building hybrid applications. Fire up the application and send it to an Android device to see how beautiful it looks—similar to figure 5.2.

While readable in this book, that text would be rather small on a real mobile device. The field where users need to enter their name is also somewhat small. Your users have rather large hands, and if there were anything else by that field it would be difficult to

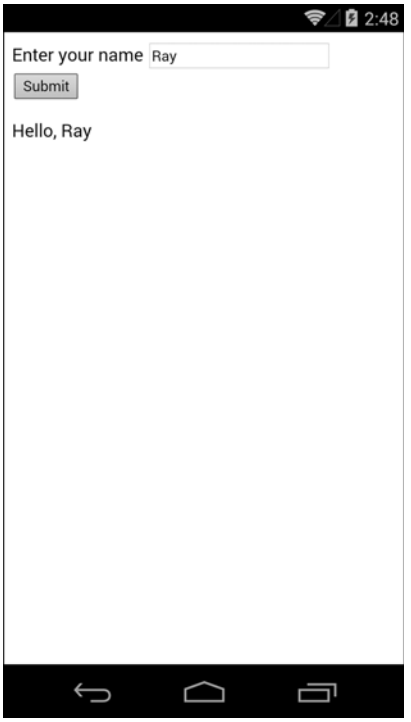

Figure 5.2 The simple application displayed on an Android device

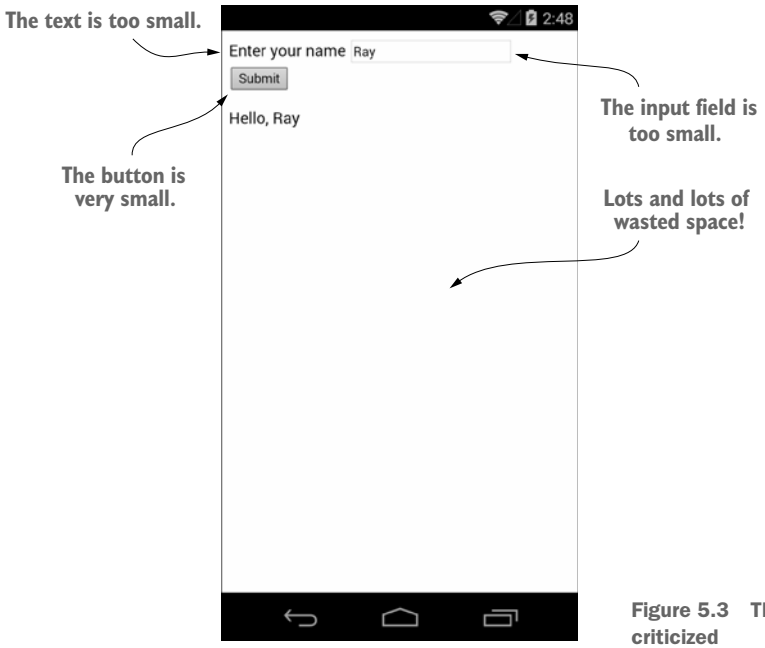

Figure 5.3 The application's UI criticized

not touch the wrong thing. Even worse, the button to submit the form is tiny. If your users need to carefully think before they interact with your application then you've probably got a problem. Figure 5.3 calls out these issues, and more.

This is a perfect example of a case where the code works but it isn't optimized for the mobile platform. This is definitely *not* a bug in Cordova. It simply reflects the fact that when building hybrid applications, you need to think differently than you would when building websites for the desktop. Let's fix it up.

5.1.2 Put some lipstick on the pig: improving the application with CSS

As mentioned earlier, one of the things missing from the application's HTML code was a CSS style sheet. You can improve the application by adding basic styling to your elements to make better use of the mobile form factor. Figure 5.4 demonstrates the improved application.

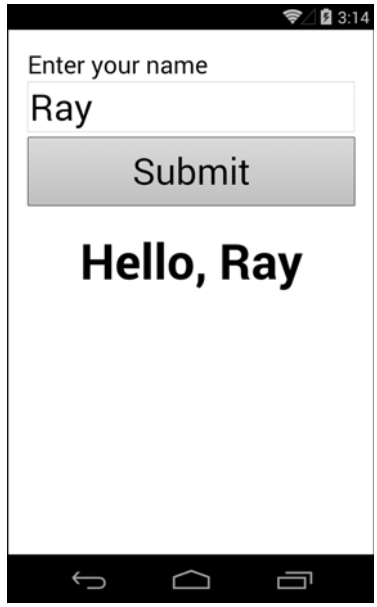

Figure 5.4 A (somewhat) prettier version of the application

Let's be honest. This "improved" application won't be winning any design contests. That's not your goal. What you wanted to do was make this application easier to use and more readable. While there's definitely *more* that could be done, figure 5.4 demonstrates how you can address the concerns identified in figure 5.3. The text is larger. Both the input field and the Submit button are much larger, which will be easier for users to tap with their fingers. The result area (where the user's name is displayed) takes up more space and you don't have a lot of empty whitespace being wasted. Listing 5.3 shows the CSS code used to make this design change. You can find the complete application in the c5/simple2 folder. Also note that the CSS file was included in the HTML via a simple `<link>` tag: `<link rel="stylesheet" type="text/css" href="css/app.css">`. You don't need to create a new Cordova application to test this—you can simply modify the existing one.

Listing 5.3 New CSS file for the improved application (simple2/www/css/app.css)

```
body {
        margin: 20px;
        font-size: 1.5em;
}

input[type=text] {
        width: 100%;
        font-size: 1.5em;
}

input[type=submit] {
     padding: 10px;
     width: 100%;
     font-size: 1.5em;
}

#result {
     font-size: 2em;
     font-weight: bold;
     text-align: center;
}
```

1 Modifies entire body of application

2 Modifies input field

3 Modifies Submit button

4 Modifies area where user's name displayed

There isn't a lot of code here and, as stated previously, more could be done, but this gets the job done. Each block in the CSS file in the listing modifies a different part of the application. The `body` block 1 updates the application as a whole. Remember that your application is a web page. You've added margins around all the edges and increased the base font size a bit. The next two blocks modify the `input` field 2 and Submit button 3. Changes to the font size specified at the body level don't apply here so they're specified too. The width is increased 4 to take the full width of a device minus the margins you specified in the body. You've made the `result` div use even bigger text, bold, and centered.

5.1.3 The meta viewport tag

If you've done any research in mobile web design, you've probably encountered the suggestion to add a `meta` tag to your page to specify a viewport setting, such as `<meta`

`name="viewport" content="width=device-width">`. At the simplest level, this tag tells the mobile device to consider the web page to have the same width as the device, something it will not do by default. So why wasn't it mentioned before?

Cordova applications automatically handle the viewport, and in fact, you have to modify a configuration setting (something we haven't touched on yet) to override this. This means you can, if you choose, skip setting this tag in your HTML. But because many people test in a browser before they work with native builds (something we'll cover in chapter 7), it makes sense to include the tag. If you look at the HTML for the updated simple application (c5/simple2/www/index.html), you'll see the tag is included even though it doesn't do anything for the application when run as a Cordova application. My suggestion? Include it!

5.2 Enhancing your Cordova UI with Bootstrap

While one solution would be to write all the CSS required for a well-designed app, maybe you don't want to, or may not have the skills to create a nice design. Many developers struggle with this—it's nothing to be ashamed of. Over the past few years, multiple libraries have been released that aim to make this task easier. They allow developers to include a style sheet (or multiple style sheets) to have a design automatically applied to their application. Some aren't necessarily "automatic" and may require you to add specific classes to your HTML to get the design. While not a magic bullet, these libraries can go a long way toward making your life easier as an application developer. They're especially useful when building proof-of-concept applications. You may be building something for a client who hasn't provided any design guidelines at all. By using one of these libraries you can easily add a professional-looking design to your application that can then be modified, or removed, later. For this chapter we'll be discussing one of the most popular of these libraries, Bootstrap.

5.2.1 Introducing Bootstrap

Bootstrap (figure 5.5) is used in multiple websites, from blogs to e-commerce sites, and is fully responsive. *Responsive web design* refers to a web page that can automatically

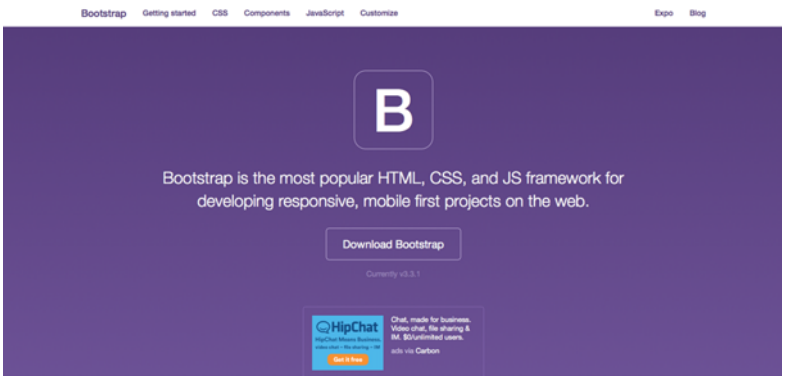

Figure 5.5 The Bootstrap website

adapt to different size viewports. This means a web page that looks good on a desktop machine will also look good on a mobile device. With Cordova, we aren't worried about the desktop, but mobile devices can also vary quite a bit in their sizes as well, ranging from smallish phones to mid- and large-size tablets.

Bootstrap is free and open source, which means you cannot only use it for free but you can also modify it as you see fit. For your needs here you'll be using it to improve the design of your applications. You can download Bootstrap from www.getbootstrap.com, but there's a copy in the zip file you downloaded from the book's website. The copy included in the zip may not be the most recent version, so I recommend downloading the latest bits from the website just to be sure. Figure 5.6 shows a "Bootstrapped" version of your simple application. It isn't radically different from the second version, but if the application were to continue to grow in scope, the benefits of using Bootstrap would become more and more apparent.

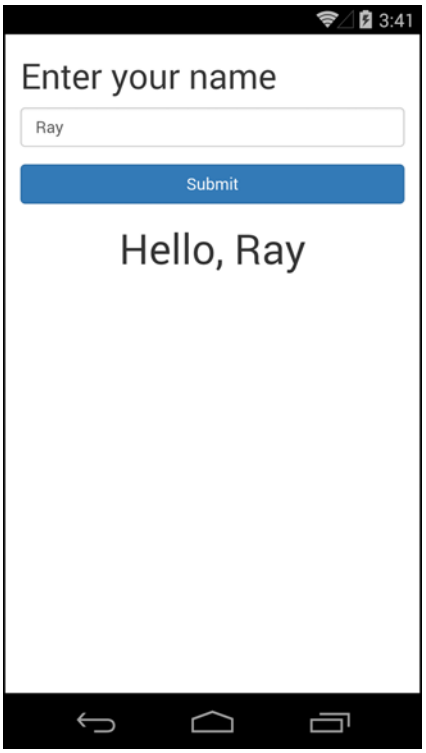

Figure 5.6 "Bootstrapped" version of your simple application

The following listing shows how you implement Bootstrap in the application. Your code base used the initial version of the sample application as a starting point, so we'll focus on the changes from that.

Listing 5.4 Updated and "Bootstrapped" application (simple3/www/index.html)

```
<html>
  <head>
    <title>Simple App</title>
    <meta name="viewport"
          content="width=device-width, initial-scale=1">
    <link rel="stylesheet " type="text/css"
          href="lib/bootstrap/css/bootstrap.min.css" />
  </head>

  <body>

    <div class="container">

      <form id="nameForm">
      <h2>Enter your name</h2>
```

1 Includes Bootstrap CSS

2 Wraps entire content and adds a bit of padding

```
      <div class="form-group">
        <input type="text" id="name" name="name"
               class="form-control">
      </div>
      <div class="form-group">
        <input type="submit" value="Submit"
               class="btn btn-primary btn-block">
      </div>
      </form>

      <h1 id="result" class="text-center"></h1>

    </div>

    <script src="cordova.js"></script>
    <script src="js/app.js"></script>

  </body>

</html>
```

Wraps in a form-group block (3)

(4) Adds automatic formatting

(5) Adds styling to button

Handles centering text

First, you include the Bootstrap CSS file (1). Note that this replaces the custom CSS you created earlier. Notice all of Bootstrap is included in a subdirectory under www called lib/bootstrap. This is completely arbitrary. It makes sense to keep Bootstrap stuff within its own folder and lib (short for library) is commonly used as a place to store third-party libraries. This helps separate your code from the code you downloaded. Again, this is *not* a requirement, but it helps keep things organized.

Bootstrap works by a combination of automatic updates and creating a set of classes you can add to your code. `container` (2), is a class used to wrap all the content of a particular web page. (For a full look at all Bootstrap CSS classes, see the documentation at http://getbootstrap.com/css/.) This is then complemented by additional classes in your form fields ((3), (4)) that add formatting. As you can see in figure 5.6 the form fields are large and full screen.

This is even more apparent in the button change (5). Three classes—`btn`, `btn-primary`, and `btr-block`—are added to it. `btn` handles basic improvements to the button to give it a flat, stylish look. `btn-primary` adds a color (blue) to the button (there are other options). `btn-block` creates a full-screen block button. This combination was chosen purely for aesthetic reasons. You could decide that a different color, or no color, makes more sense. You could also choose multiple types of sizes. Consult the Bootstrap documentation for more about the types of things you can do with it. The point is with a relative little bit of typing you've got a nice, clean-looking design for your application.

5.2.2 Another example: the camera app

Let's look at another simple example. In chapter 4 you built an application that made use of the Camera plugin. Some very basic styling was done to the buttons used in the app as well as the image that was displayed. You can switch to using Bootstrap *only* to

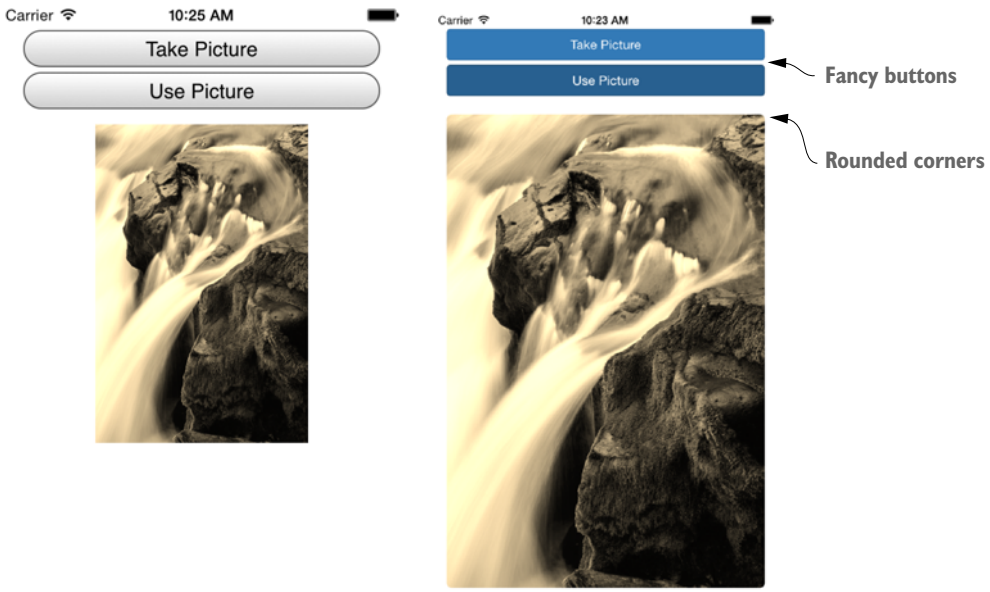

Figure 5.7 The old camera app (left) versus the Bootstrap version (right)

improve the buttons while keeping the fancy sepia look. Figure 5.7 shows the original version of the application next to the updated Bootstrap edition.

Again, this isn't a radical change, but the Bootstrap version does look nicer and overall more professional. Let's look at the changes required to make this happen, shown in the following listing. If you want to try this version, you can either modify the older version (and remember to copy the Bootstrap library), or use the version included with the zip. It can be found at c5/camera_demo_bootstrap.

Listing 5.5 Updated camera index.html (c5/camera_demo_bootstrap/www/index.html)

```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="utf-8">
    <title>Basic Camera</title>
    <meta name="viewport" content="width=device-width">
    <link rel="stylesheet" type="text/css"
    href="lib/bootstrap/css/bootstrap.min.css" />
    <link rel="stylesheet" type="text/css" href="css/app.css" />
  </head>

  <body>

    <div class="container">
```

1 Bootstrap CSS

2 Application-specific CSS

```
      <button id="takePicture"
              class="btn btn-primary btn-block">Take Picture</button>
      <button id="usePicture"
              class="btn btn-primary btn-block">Use Picture</button>

      <img id="myImage" class="img-responsive img-rounded">

    </div>

    <script src="cordova.js"></script>
    <script src="js/app.js"></script>
  </body>
</html>
```

Updated button styling (3)

Updated image styling (4)

As before, to include Bootstrap you have to include the CSS file (1). But you can also include your own CSS (2). As you'll see in listing 5.6, the CSS is somewhat slimmed down now that Bootstrap is handling more for you. The buttons are updated (3) much like in the previous example. As a final step the Bootstrap image styling (4) is updated. Bootstrap has a responsive image class that handles making the image take up as much real estate as possible. This is important as you want the image to look nice on phones and tablets. The other class, `img-rounded`, adds rounded corners to the image. (I added this because. Yes, just because.)

Now let's look at the updated CSS.

Listing 5.6 Updated camera CSS file (c5/camera_demo_bootstrap/www/css/app.css)

```
body {
  margin-top: 20px;
}

img {
  margin-top: 20px;
  -webkit-filter: sepia(100%);
}
```

The CSS is now much slimmer as you're letting Bootstrap do more for you. Bootstrap doesn't add any padding to the top of the document as you've kept a margin on the body. You also want the image to be a bit below the buttons so a margin is used there as well. Finally, the sepia effect is maintained.

5.2.3 Bootstrap does more

We've only scratched the surface of what Bootstrap can add to an application. Along with nice styling for buttons it includes other components, some of which require a JavaScript file as well. I highly encourage you to look over the Bootstrap website to get an idea of what can be easily added to an application.

Bootstrap isn't meant to be a silver bullet. You cannot expect that it will be usable for *every* application. But it can be a great way to get started and a great way to focus your development time on other issues.

If you enjoy using Bootstrap, you can also make it look more unique. Many commercial and free themes exist that integrate well with Bootstrap. For examples, check out https://wrapbootstrap.com.

5.3 Mobile UI frameworks: an overview

Because so many developers need help when it comes to design, there are numerous mobile UI frameworks available. Many are much more than a simple UI framework. They may provide a user interaction (UX), such as pull to refresh, or other related services. This section looks at four such frameworks: Ionic, jQuery Mobile, Ratchet, and Kendo UI. We begin with Ionic, my favorite. My best advice is try each one yourself and find the one that best matches your development style. The best option is the one that makes you, and your app, successful.

5.3.1 Ionic: UI, UX, and more

Easily my favorite, Ionic (http://ionicframe work.com/) provides a UI framework (figure 5.8), UX tool (like pull to refresh), and numerous other tools that dramatically enhance the power of hybrid mobile applications, and specifically Cordova. The only reason this book won't cover Ionic is that it requires some other prerequisites that would prevent some readers from being able to use it easily. Ionic relies on AngularJS, a popular JavaScript framework, but also one that's a bit complex for the uninitiated.

Ionic provides a command-line wrapper around Cordova itself. It adds logging and other productivity enhancements on top of what Cordova provides. In my personal opinion, Ionic is the best thing to happen to Cordova. You can read more in *Ionic in Action* (Manning Publications, 2014) by Jeremy Wilkin (www.manning.com/wilken/).

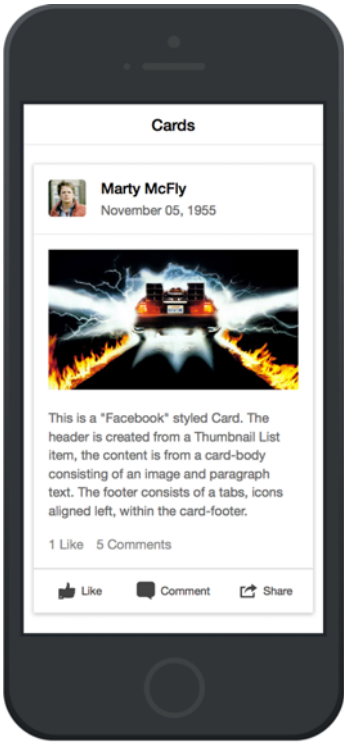

Figure 5.8 An example of one of Ionic's UI controls, a Card

5.3.2 jQuery Mobile: powerful and simple

One of the older options, jQuery Mobile (http://jquerymobile.com/), is one of the easiest UI frameworks to use, and like other frameworks, it provides other features for mobile development. Because of its ease of use (despite the name, you can do a heck of lot without writing any jQuery-related code), jQuery Mobile will be discussed in depth later in the book.

jQuery Mobile does most of its work within HTML, providing simple attributes to enhance controls into mobile-friendly widgets, like those shown in figure 5.9. For example, creating tabs is done by adding `data-role="tabs"` to a `div` block.

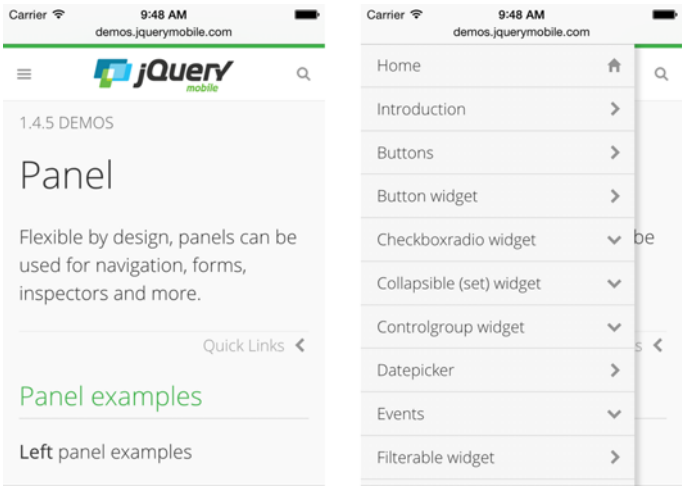

Figure 5.9 Two jQuery Mobile UI examples

5.3.3 Ratchet: Android and iOS friendly

Ratchet (http://goratchet.com/) works a bit like Bootstrap. You include a required CSS file (as well as JavaScript if need be) and then make use of particular classes in your elements to get the desired look. Like Bootstrap, Ratchet has multiple widgets you can use to create your application. Ratchet also includes support for different styling for both iOS and Android, as shown in figure 5.10. (jQuery Mobile, for example, will look the same on both devices.)

Figure 5.10 Two examples of Ratchet showing iOS versus Android views

5.3.4 Kendo UI: large and commercially supported

Kendo UI (www.telerik.com/kendo-ui1), which is part of a much larger set of free and commercial services, provides nearly 80 different UI widgets for use with mobile web pages and hybrid applications (figure 5.11). It can integrate with Bootstrap so an application could make use of both of them at once. Unlike Bootstrap, Kendo UI uses JavaScript to enhance controls as opposed to only adding classes to HTML. Kendo is the product of Telerik, which has an entire platform of services, including testing, server-side data persistence, and even web-based building services, that would be of interest to folks developing hybrid applications.

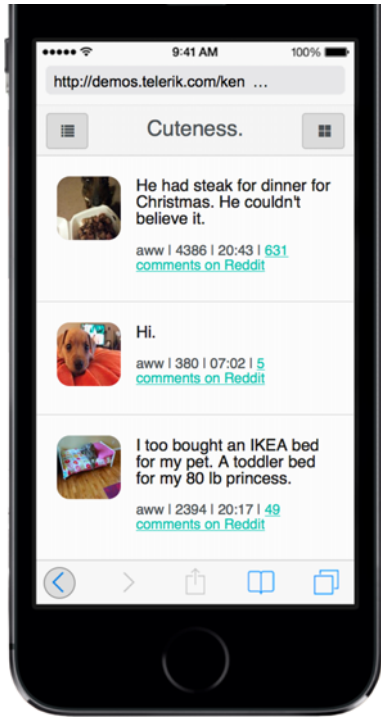

Figure 5.11 One of the Kendo UI demo applications (not my hand—honest)

5.4 Summary

Let's review the major topics covered in this chapter.

- Creating an application that works well on mobile devices requires planning ahead of time.
- UI frameworks like Bootstrap exist that make it easier to build mobile-friendly applications.
- Bootstrap can easily be added to an application to create a beautiful UI for your Cordova application.
- Because of the popularity of mobile (and hybrid) development, many different options exist for developers.

In the next chapter, we'll dig deeper into other concerns for building hybrid mobile applications. You'll learn about support for offline users and international users, and how to store data in the application.

MOBILE DEVELOPMENT

Apache Cordova IN ACTION

Raymond K. Camden

Developing a mobile app requires extensive knowledge of native programming techniques for multiple platforms. Apache Cordova lets you use your existing skills in web development (HTML, CSS, and JavaScript) to build powerful mobile apps. Your apps also get the power of integration with native device features like the camera and file system.

Apache Cordova in Action teaches you how to design, create, and launch hybrid mobile apps people will want to use. With the help of straightforward, real-world examples, you'll learn to build apps from the Cordova CLI and to make use of native device features like the camera and accelerometer. You'll learn testing techniques and discover the PhoneGap Build service and how to submit your apps to Google Play and the Apple App Store. Along the way, this helpful guide discusses mobile app design and shows you how to create effective, professional-quality UI and UX.

What's Inside

- Build mobile apps
- UI, UX, and testing techniques
- Deploy to Google Play and the Apple App Store
- Employ libraries like Bootstrap, jQuery Mobile, and Ionic

Readers should be familiar with HTML, CSS, and JavaScript. No experience with mobile app development needed.

Raymond Camden is a developer advocate for IBM. He is passionate about mobile development and has spoken at conferences worldwide.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/books/apache-cordova-in-action

"A great resource for beginners and experienced programmers alike."
—Ivo Štimac, KING ICT

"Covers Cordova from top to bottom."
—Jérôme Bâton, DRiMS

"Very thorough and well-written. Walks you through everything you need to write Apache Cordova-based applications."
—Becky Huett, Big Shovel Labs

"An easy-to-follow guide through a Cordova project."
—Gregory Murray, Volusion

ISBN 13: 978-1-63343-006-8
ISBN 10: 1-63343-006-5
53999
9 781633 430068

MANNING $39.99 / Can $45.99 [INCLUDING eBOOK]