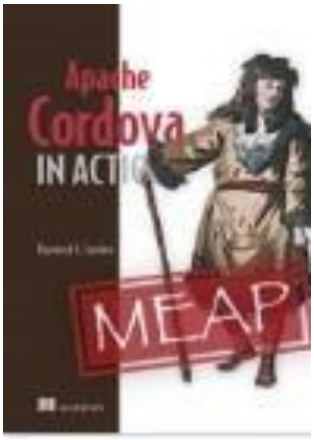
Congratulations - you're a (horrible) mobile developer!

By Raymond Camden, *Apache Cordova in Action*



You may know the fundamentals of building a mobile application, but that doesn't mean you know how to build a *good* one. Here's good example of bad UI.

Ok, that may be just a tiny bit over the top, but most likely there is a bit of truth to it as well. You know how to install Cordova, how to generate native binaries from HTML, and how to make use of fancy device features with plugins. That's all fairly straightforward: Install a SDK, install the command line tool, write some HTML, and whammo, see it on your device. But that doesn't mean you know how to create a *good* mobile application. Taste is subjective.

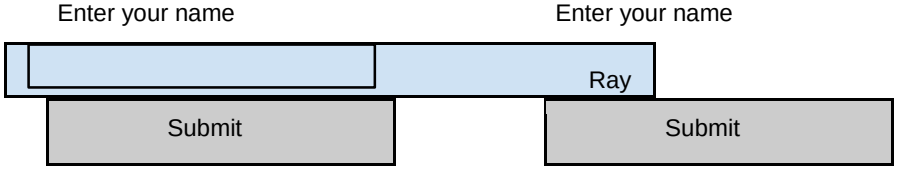
While it is difficult to precisely describe what makes a good mobile application, there are definitely a set of guidelines that help define what a successful mobile application looks like. And notice I'm not saying a successful *hybrid* mobile application. Your users don't care what you used to build your application. They only care about the end result. Therefore the guidelines for a good hybrid mobile application are going to be same as a good 100% native built mobile application as well.

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 user interface. 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 (both off and online).

A Good Example of Bad UI

Imagine the most simple application possible - an application that prompts you for your name and then simply tells you hello. Figure 1 is a mock-up of the UI for the application, both the initial view and what is displayed after entering a name.



For source code, sample chapters, the Online Author Forum, and other resources, go to
http://www.manning.com/camden/.

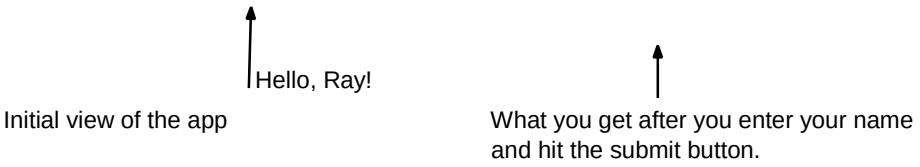


Figure 1 Our simple little application. Looks pretty now, right?

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 Manning web site. You will find it in the c5/simple folder. Listing 1 represents the HTML used for the application.

Listing 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>     #1
                   <input type="text" id="name" name="name">     #2
                   <input type="submit" value="Submit">          #3
            </form>

            <div id="result"></div>                               #4

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

#1 Label used to prompt for your name.
#2 Input field where the user enters their name.
#3 Button used to submit the form.
#4 Empty div that will be filled with the 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 OK - 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 listing 2.

Listing 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. Since our 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. Let's fire up the application and send it to an Android device to see how beautiful it looks.

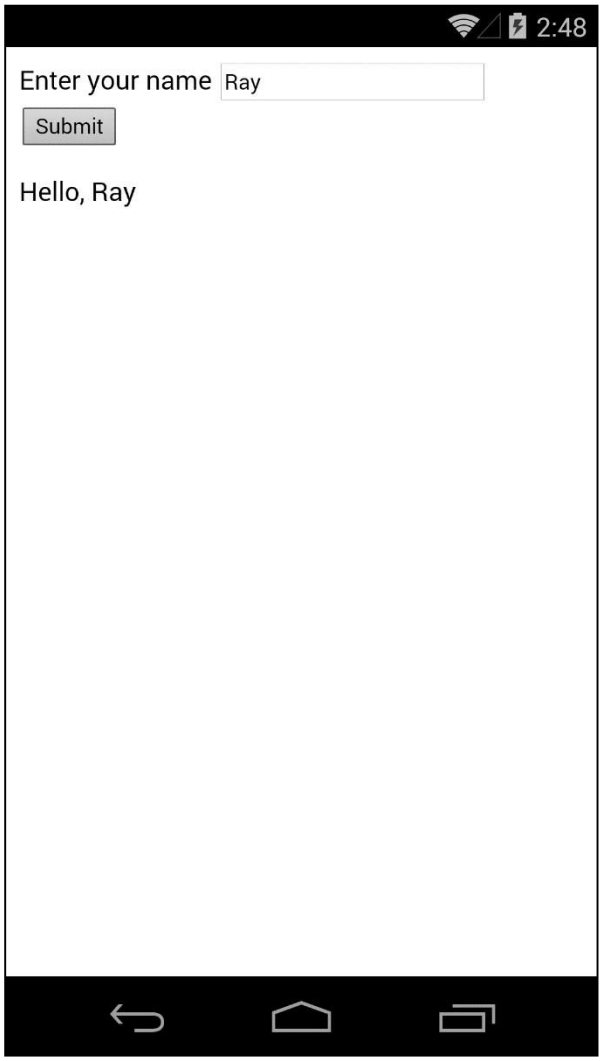


Figure 2 Our "simple" application displayed in Android.

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 author has rather large hands and if there was anything else by that field it would be difficult to not touch the wrong thing. Even worse, the button to submit the form is *tiny*. If your user needs to carefully think before they interact with your application than you've probably got a problem. Figure 3 calls out these issues, and more.

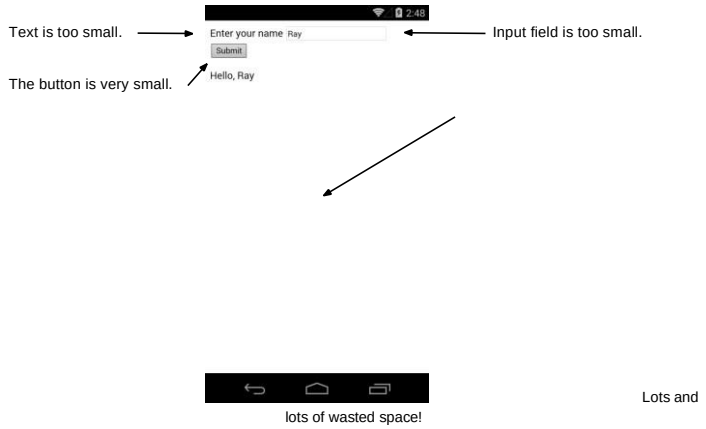


Figure 3 Our application's UI criticized.

This is a perfect example of a case where our code *works*, but 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 web sites for the desktop.