Sencha Touch in Action

by Jesus Garcia
Anthony De Moss
Mitchell Simoens

Chapter 1

Copyright 2013 Manning Publications
# brief contents

## PART 1 INTRODUCTION TO SENCHA TOUCH ...........................................1

1 ■ Introducing Sencha Touch  3  
2 ■ Using Sencha Touch for the first time  23  
3 ■ Sencha Touch foundations  40  

## PART 2 BUILDING MOBILE USER INTERFACES ...............................53

4 ■ Mastering the building blocks  55  
5 ■ Toolbars, buttons, and docked items  85  
6 ■ Getting the user’s attention  104  
7 ■ Data stores and views  123  
8 ■ Working with forms  149  
9 ■ Maps and media  177  

## PART 3 CONSTRUCTING AN APPLICATION ....................................195

10 ■ Class system foundations  197  
11 ■ Building Sencha Touch applications  231  

---

vii
You’re on the hook to build a mobile application. Perhaps you’ve been tasked with a project, or you have a great idea and want to make it a reality. Either way, to build your application you’re going to have to learn at least Objective C for iOS or Java for Android. It should be no surprise that if you want to support both types of devices you’ll have to learn and master both languages, unless you choose a third-party native framework like Sencha Touch to bridge the gap between the devices.

Chances are you have experience in HTML, CSS, and JavaScript and want to use what you already know to build your mobile application. The ability to tie in your prior experience is part of what makes Sencha Touch a good choice for folks like you and me, because it offers a wide range of UI widgets to choose from, as well as robust data, layout, and component models.

In this chapter you’ll begin your journey into the world of Sencha Touch, where you’ll learn what Sencha Touch is and the problems it aims to solve, such as enabling...
development of cross-platform user interfaces with HTML5. Then we’ll look at the widgets that the framework provides. Lastly we’ll discuss some of the ways you should think about developing your mobile application to avoid future performance issues.

What you’ll learn along the way is that developing mobile applications with this framework isn’t as difficult as with other technologies such as Objective C or Java.

1.1 What is Sencha Touch?

Sencha Touch was born out of the culture and many of the ideas from the venerable Ext JS framework and is the first mobile HTML5 JavaScript framework.

Sencha Touch solves cross-platform mobile app development problems by giving developers the tools necessary to build cross-platform applications that mimic natively compiled applications, while making full use of HTML5 and CSS3. It also allows developers who have years of experience on the web to develop cross-platform mobile apps that can exist solely on the web, or be deployed in an app store with either the Sencha native packager or tools like PhoneGap.

As of this writing, Sencha Touch runs on mobile WebKit-based browsers in iOS (iPhone, iPad) devices as well as on Android phones and tablets.
What is Sencha Touch?

An excellent example of a Sencha Touch application is Checkout, by Steffen Hiller, shown in figure 1.1 running on an iPad. Here you can see an application that makes use of Sencha Touch providing a rich UI with HTML5.

To see other Sencha Touch applications you can point your browser to http://sencha.com/apps and view the Sencha Touch App Gallery (figure 1.2). Here you can preview apps via images and even see them work live via embedded links.

Much like Ext JS, Sencha Touch creates the feel of a native application by means of a clever blend of HTML5, CSS3, and JavaScript, all optimized for the best possible mobile experience given the constraints of mobile devices today, such as limited CPU and memory for your applications.

Read about HTML5

HTML5 is a collection of technologies that includes enhancements to HTML itself, CSS3, and even JavaScript. It’s changing the way we develop web applications by providing JavaScript API access to do things like talk directly to a graphics card (WebGL), manipulate sound, and even provide offline storage. Though it’s not completely necessary to know everything about HTML5 to use Sencha Touch, it’s a good idea to get the basics down. A great site for learning about HTML5 is www.html5rocks.com.
Also like its big brother Ext JS, Sencha Touch is designed to be extensible and modifiable out of the proverbial box.

1.1.1 What Sencha Touch is not

Although Sencha Touch works on desktop WebKit browsers, like Safari and Chrome (to a limited extent), it isn’t designed for desktop rich internet applications. Upon its release lots of developers balked at the idea of this framework not functioning in Firefox or Internet Explorer.

The fact is that Sencha Touch is aimed at the development of mobile applications only. This means that if you’ve only developed applications with Firefox, IE, and their respective debugging toolkits, you’re going to have to leave your comfort zone.

``Sencha Touch” !== “Ext JS”

If you’re a veteran Ext JS developer you’ll feel right at home when learning Sencha Touch. It’s important to know that some significant differences exist between the two libraries. Throughout this book we’ll point out some of the differences, but we can’t cover every possible point. If you have doubts always check the API documentation.

If you’re unfamiliar with Ext JS and need to develop applications for the desktop web check out Ext JS in Action (Manning, 2010).

1.1.2 Lots of wiring under the hood

To make use of mobile device interactions Sencha Touch comes with a gesture library, which allows you to easily hook into gesture-based events, such as tap, pinch, and swipe. One way Sencha Touch comes close to the feel of native applications is by means of a custom physics-based Scroller class, which uses hardware-accelerated CSS3 transitions and includes key variables like slide friction and spring effects.

1.1.3 Hardware compatibility

Many mobile touch-screen smart devices are entering the marketplace today, which is driving the increase in demand for mobile applications. Though Sencha Touch aims at 100% compatibility across all mobile devices the best user experience is on iOS and high-powered Android devices.

Why the difference in user experience?

The main reason for the difference in user experience between iOS and Android has to do with the physical computing power of each device and how each device manufacturer compiles mobile WebKit for their device. Apple devices include a GPU and compile mobile Safari with GPU acceleration enabled. Most Android devices don’t have dedicated GPUs. And even for the ones that do, manufacturers typically don’t compile mobile WebKit to enable GPU acceleration.
Sencha Touch applications do such a great job of mimicking how native applications look and feel that it’s often easy to get lost in the fact that you’re using a web-based application. This especially holds true when the mobile WebKit toolbars are hidden from view.

1.1.4 Full-screen goodness

Figure 1.3 illustrates how a Sencha Touch application looks when accessing the application via mobile Safari (left) compared to accessing the application via a shortcut on your home screen.

After you look at figure 1.3 it should be clear that a full-screen view of a Sencha Touch application closely resembles a native application. Also having your app in full screen means that there’s more much-needed screen real estate available in your applications.

Sencha Touch offers a lot when it comes to UI widgets, but it’s certainly just the surface of this framework. If you’re like us you probably want to just skip ahead and dive into code. But before you get your hands dirty let’s browse through the library and discuss some of its features.

1.2 A 10,000-foot view

If you glanced at the Sencha Touch API documentation the sheer number of classes in the library might have overwhelmed you. To make sense of it all you must understand that these classes can be broken down into a few major groups. Table 1.1 describes the major groups into which Sencha Touch is broken down.
CHAPTER 1  Introducing Sencha Touch

Table 1.1  Describing the various sections of the Sencha Touch framework

<table>
<thead>
<tr>
<th>Group</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>This is the shared base of Sencha Touch and Ext JS v4, and the bulk of the code for Sencha Touch.</td>
</tr>
<tr>
<td>Layout</td>
<td>A set of managers for visually organizing widgets on screen.</td>
</tr>
<tr>
<td>Utilities</td>
<td>A group of useful odds and ends for the framework.</td>
</tr>
<tr>
<td>Data</td>
<td>Data is the information backbone for Sencha Touch and includes the means for retrieving, reading, and storing data.</td>
</tr>
<tr>
<td>Style</td>
<td>Sencha Touch’s theme is automatically generated via Sass (Syntactically Awesome Style Sheets).</td>
</tr>
<tr>
<td>MVC</td>
<td>Sencha Touch comes with an MVC framework for your application.</td>
</tr>
<tr>
<td>UI Widgets</td>
<td>A collection of visual components that your users will interact with.</td>
</tr>
</tbody>
</table>

The base library for Sencha Touch is known as Sencha Platform. Sencha Platform is based off Ext JS 4.0 but is much improved in many respects. The class system in Sencha Touch resembles that of Ext JS 4, but it’s much more advanced in many ways. For example, it includes a feature known as Config System which allows Sencha Touch to work much more effectively than Ext JS.

The Layout portion of Sencha Touch is implemented by some of the UI widgets and is the code responsible for visually organizing items on the screen. The layouts are responsible for implementing transitional animations if configured to do so.

The Utilities section of the framework is a collection of useful bits of functionality that are often implemented by the framework and can be implemented by you. For instance, the List widget implements XTemplate to paint HTML fragments on screen. The XTemplate is open for you to use to do the same in your own custom widget.

The Data package is a group of classes that gives Sencha Touch the ability to fetch and read data from a myriad of sources, including mobile WebKit’s HTML5 Session, Local, and Database Storage methods. Sencha Touch can read data in a variety of formats, including XML, Array, JSON, and Tree (nested).

The Style area of the framework isn’t something that you typically deal with on a day-to-day basis, but it’s worth mentioning. From the very beginning Sencha Touch has implemented Sass to allow easy style changes to the UI. This means that if you want to change your entire color scheme you can do so with relative ease if you know Sass.

Learn more about Sass
Sass has taken the world of style sheet management by storm and has arguably revolutionized how people style their web pages and apps. To learn more about this utility check out Sass and Compass in Action (Manning, 2013).
The Sencha Touch UI includes an MVC framework that allows developers familiar with that pattern to develop applications within a familiar workspace. It also contains a custom URL routing mechanism and history state support.

The widgets that users will interact with in your application comprise the UI portion of Sencha Touch. When thinking about designing and constructing your applications you have a lot to choose from, which is why it’s a good idea to look at each of them.

1.3 The Sencha Touch UI

The Sencha Touch UI is a rich mixture of widgets that can be displayed on screen for you and your users to interact with. The UI palette is large, and table 1.2 helps you identify the groups of UI components.

After reviewing the groups we’ll dive deeper into each group and discuss the UI components in greater detail.

Table 1.2 The various groups of UI widgets available in the Sencha Touch framework

<table>
<thead>
<tr>
<th>Group</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containers</td>
<td>Widgets that are designed for nothing more than managing other child items. An example of these types of widgets is the Tab panel. Containers typically implement layouts.</td>
</tr>
<tr>
<td>Sheets</td>
<td>Sheets are generally any popup or side-anchored container and appear in a modal fashion, requiring users to interact with the sheet before moving forward. An example of a sheet is the Date Picker widget.</td>
</tr>
<tr>
<td>Views</td>
<td>Views are widgets that implement data stores to display data. The List and Nested List are both views that implement Stores.</td>
</tr>
<tr>
<td>Misc</td>
<td>This collection of widgets ranges from Buttons to Maps to Media.</td>
</tr>
</tbody>
</table>

Now that you have a good overview of the widget groups let’s begin our visual exploration.

1.3.1 Containers

Containers in Sencha Touch do the heavy lifting when it comes to managing widgets inside of widgets. Container is what we like to call the workhorse of Sencha Touch applications because it offers extreme configurability and flexibility. Containers can dock child widgets to their sides or render child widgets inside of their bodies. To see what we mean take a look at figure 1.4.

In figure 1.4 you see a container with three docked items. We have a toolbar docked at the top, List view docked on the left, and another toolbar docked at the bottom. Notice how the top-docked toolbar simply contains a title, whereas the bottom toolbar contains buttons. This shows some of the power and flexibility of the toolbars.

If you need to display screens controlled by a toolbar the Tab panel will get the job done.
1.3.2 Controlling your UI with the Tab panel

The Tab panel (figure 1.5) is a container that automatically sets a top-docked or bottom-docked toolbar for you with automatically generated buttons for every child item. Tapping any of the buttons allows you to “flip” through items known as “cards.”

In figure 1.5 we’ve configured a Tab panel that implements a “slide” transition with two child panels. By selecting Panel 2 in the toolbar Sencha Touch automatically applies the CSS3 transition properties to both child panel elements, allowing for a smooth transition from one panel to another.

The Tab panel does an excellent job of managing the display of items in your screen, but sooner or later you’ll need the ability to accept data input from your users. For that you’ll use the Form panel.

1.3.3 Accepting input with the Form panel

The Form panel is a container that’s typically used to display any of the input fields that Sencha Touch provides and is automatically scrollable. Your fields can be grouped via the FieldSet widget. Figure 1.6 shows an example of a Form panel requiring user input.

In figure 1.6 we have most of the input fields that Sencha Touch offers, with the exception of the Hidden field. With Sencha Touch the Text, Checkbox, URL, Email,
The Sencha Touch UI

Figure 1.5 The Tab panel allows you to configure UIs that can be changed by a tap of a button and includes optional transition animations (from left to right).

Figure 1.6 Form panels are used to display input fields and contain necessary controls to manage the submission of data to your server.
Textarea, Number, Password, and Radio fields all implement native HTML5 input elements, with the addition of styling. Each of these, except for the Radio and Checkbox fields, will force the native slide-in keyboard to appear when focused, allowing users to enter data into the fields.

The Checkbox and Radio fields work similarly to their native web counterparts, except that they’re stylized via Sencha Touch’s own check icon to mimic native application behavior. In this example the Role checkboxes, grouped in a Fieldset, are Radio fields, allowing only one selection in the set.

Next the Spinner field is a custom-styled input field, allowing users to enter numeric values, much like the Number field, with the addition of easy-to-use decrement and increment buttons on each side of the field.

The Slider field implements native Sencha Touch Draggable and Droppable classes, allowing users to input a numeric value, via swipe and tap gestures. The Toggle field extends Slider, allowing users to toggle a field of two values via swipe and tap gestures, much like an on/off toggle switch that you see in various physical devices.

Lastly, the Date Picker and Select fields give your users the ability to choose data from a set. The Date Picker field implements what’s known as a sheet, which is an overlay panel that slides in from the bottom allowing the user to select values via vertical swipe or “flick” gestures. Figure 1.7 shows an example of a date picker displayed on an iPhone.

No matter what the device or its orientation the Date Picker field will always display a sheet forcing selection through this modal overlay. The Date Picker widget may seem familiar to you because it mimics the native iOS Date Picker input widget.

The Select field will display different input widgets based on the device. Figure 1.8 shows our implementation of the Select widget on a phone and on a tablet device.

On the left in figure 1.8 the Select widget is displaying a Picker sheet because Sencha Touch detected that it’s running inside of a phone versus a tablet (right). The difference is that iOS tablets natively display dialog controls for selection.

As you’ve just seen, Sencha Touch offers quite a few wrapped native HTML5 input fields as well as a few custom widgets. Because we’ve been talking about the Date Picker and Select fields implementing sheets let’s look at the various Pickers and Sheets that Sencha Touch offers, outside of the Form panel.
13

1.3.4 Sheets and pickers

We’ve already seen the Date picker and Picker classes implemented via their associated form input widgets. Sencha Touch provides you with a widget called a sheet, which is a floating modal panel that animates into view, grabbing the user’s attention and focus. Figure 1.9 shows a sheet in action.

In figure 1.9 you can see a sheet with top- and bottom-docked toolbars, managing a scrollable List view. You can configure such a UI because Sheet is a subclass of Panel and includes all of the UI goodness that Panel provides.

What’s neat about the Sheet widget is that it’s orientation-aware. This means that flipping the device while the application code is executing immediately causes the sheet to render in landscape mode, as illustrated in figure 1.10.

The story about Sheet doesn’t end here. It has three subclasses: ActionSheet, MessageBox, and Picker. You’ve already seen Picker and its subclass, Date picker, but you haven’t seen ActionSheet and MessageBox.

We’re throwing a lot of new names at you, so to help with any confusion see the simple inheritance model diagram (figure 1.11).

Out of the box the ActionSheet widget allows you to easily render buttons in a sheet, rendered in a vertical stack. Because ActionSheet is a subclass of Sheet, which extends Panel, you can add pretty much anything you want to the stack of widgets. Figure 1.12 shows an example of an ActionSheet widget rendered with a custom HTML title.
Such an ActionSheet could be used to request action from users, requiring that they choose an action via one of the buttons. In this case you’re asking the user to choose one of three options, and with the custom title you’re providing a hint along with the actionable button set.

MessageBox is a subclass of Sheet that provides Sencha Touch–styled alert-like functionality to your applications. Figure 1.13 illustrates the three most common uses of MessageBox, including alert (top left), confirm (bottom left), and prompt (top right). Each of these dialogs appears with smooth CSS3 transitions and mimics their native counterparts.

The key differences among the three are apparent. The alert MessageBox widget is designed to alert the user of some condition and only displays one button. The confirm dialog allows the user to make a decision by tapping on a button, enabling a branch of logic to execute. Lastly, the MessageBox prompt asks the user for direct input.

You’ve seen all that Sheet and its subclasses have to offer. Next let’s look at the various data-bound views that Sencha Touch provides.
1.3.5 Data-bound views

If you’re an Ext JS developer you might be surprised to learn that Sencha Touch only provides three data-bound views and that this list excludes a GridPanel. At your disposal you have Data View, List, and NestedList. We’ll begin with the most basic, Data View, and work our way to the most complex.

Data View is a widget that binds to a data store to render data on screen. It gives you 100% control over how you’ll render your data. Figure 1.14 is an example of a simple Data View widget displaying a set of names, beginning with the last name. Here we have a stylized Data View widget rendering data from a data store, which contains a list of names. This example rendered only names to keep it simple but you can use Data Views to render anything imaginable and to allow for user interaction.
With the Data View widget you’re completely responsible for a lot of work, including defining the XTemplate that’ll be used to stamp out the HTML fragments, as well as styling how the items are rendered on screen. If you’re like us and want the look and feel of a native list the List class is at your disposal. Figure 1.15 shows a List widget, rendering the exact same data as in the previous example.

The difference between this example and the previous one should be clear. What’s not as obvious is that the level of effort required to create this List view is orders of magnitude less than creating the earlier Data view. You’ll see what we mean by this in chapter 6, when we tackle List views head on.

In addition to allowing for a native application look and feel the List view has three more key features, shown in figure 1.16.

With a few minor tweaks you’re able to transform a simple List view into a grouped List view. The grouped list in figure 1.16 has what’s known as a grouping bar, which is a separator between items in the list. The Sencha team has been able to get this list to work nearly identically to native grouped lists, and it includes optional disclosure icons, as well as an index bar for fast searching with a single finger swipe.

The Data View and List widgets are designed to display data in a linear set. But there are times when you want to display nested data. For that you’ll need to use the NestedList widget, shown in figure 1.17.

In figure 1.17 we’ve set up a NestedList for the selection of a food item. There are two main categories: Drinks and Food (left). We chose “Drinks,” which brought us to three subcategories (center). We then chose “Sports Drinks,” which led us to the last
Figure 1.16 A grouped List view, sporting an index bar

Figure 1.17 A NestedList widget in action used in a navigational manner
section of items, which is a list of specific sports drinks (right). All of this is done with the slide animation.

You’ve just explored the Data View, List, and NestedList widgets. Next let’s examine the Map and Media widgets.

### 1.3.6 Maps and Media

With the rapid-expansion world of mobile applications integrating maps into your applications can provide a huge boost in productivity for your users. To meet this growing demand Sencha Touch integrates Google Maps to supercharge your location-aware applications. Figure 1.18 shows the Sencha Touch Map widget in action.

The Sencha Touch Map widget literally wraps Google Maps, allowing your application code to manage the Google Maps instance as if it was native to Sencha Touch. This means that the Map widget can take part in layouts and has normalized events, as well as an interface method to easily update the map’s coordinates.

Another growing demand for mobile applications is the ability to play audio and video content. HTML5 natively has video and audio tags that bring this functionality to Mobile Safari but Sencha Touch makes it easier to use. Figure 1.19 shows the Media widget displaying a video on a tablet.

Just like the Map widget, the Media widget uses familiar interface methods, and it’s easily configurable to play audio and video in your applications.

We’ve just completed our UI walkthrough. Before we wrap up this chapter we want to talk about thinking like a mobile developer. This conversation will be especially helpful to you if this is your first dive into the world of mobile application development. We know you’re itching to get down to coding so we won’t hold you very long.

### 1.4 Thinking like a mobile developer

If you’re like us you’re making the transition from Ext JS to Sencha Touch. Making the transition to mobile from desktop application development poses thought-process challenges that must be overcome if you plan to build successful apps. Here are some points you need to think about before moving forward with your application development.
1.4.1 Think lightweight

When spec’ing out or developing your mobile app you must think “lightweight” or your app is destined to run into performance issues. If you’ve made the transformation from a native desktop application developer to a desktop-web application developer it’s likely that you’ve encountered this issue during your transition, because native desktop applications can handle much more of a burden than desktop-web applications.

Due to the reduced computing power of mobile devices the mobile browser is limited in many ways when compared to its desktop counterpart. This is why thinking “lightweight” is paramount for a successful application.

Our suggestion is to try to reduce the amount of data as well as the complexity of the screen size as much as possible. Reducing the user interaction models is also a plus since complex user interaction models bog down mobile-web applications.

1.4.2 Remember—it’s a browser!

Many developers are tasked with converting native applications to Sencha Touch–powered web applications. Often during the conversion process they experience performance issues, and Sencha Touch is blamed.
It’s during these times that we tell developers caught in this cycle to remember that the application they’re developing is running inside of a browser and thus has limited power relative to native-compiled mobile applications. Just as native desktop applications can handle more difficult tasks than desktop-web applications, native mobile apps have more muscle than mobile-web applications.

We believe that entering the conversion process with this in mind helps you set realistic expectations with your customers.

1.4.3 **Throw away what you don’t need**

With the reduced power of mobile devices comes an increase of responsibility to keep things as clean as possible and reduce DOM clutter and bloat. For desktop-web applications this isn’t as critical, but for mobile web it’s extremely critical.

This means that when placing items such as ActionSheets in the Document Object Model (DOM) you must take care to destroy them when they are no longer needed. DOM bloat is the enemy of performance.

**Mobile Safari will crash**

Mobile Safari will crash if your application causes it to run out of memory. This will simply cause the application to disappear from the user without warning.

Sencha Touch widgets come with a complete three-phase lifecycle, allowing you to easily destroy components, thus removing items from the DOM and freeing up crucial resources.

Along with the destruction of items that are no longer needed you should only instantiate what’s needed. We often find hugely nested components being instantiated when only a single component is needed for a particular action. To keep things safe think conservatively.

1.4.4 **“finger” !== “mouse”**

Part of transitioning to mobile development involves understanding the user interaction models and how they relate to browser events. Table 1.3 describes some of the most common user gestures, alongside their desktop counterparts (when applicable).

**Table 1.3 Comparing touch gesture events with desktop mouse gesture events**

<table>
<thead>
<tr>
<th>Mobile</th>
<th>Desktop</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>touchstart</td>
<td>mousedown</td>
<td>The initial point at which a touch is detected in the UI.</td>
</tr>
<tr>
<td>touchend</td>
<td>mouseup</td>
<td>Signals the end of a touchstart event.</td>
</tr>
<tr>
<td>tap</td>
<td>click</td>
<td>A tapstart and tapend event for a single target.</td>
</tr>
</tbody>
</table>
Summary

Always test all of your complex interaction models with the physical platforms that you’re targeting for your applications. It’s only then that you can truly see how they’ll react during events like pinch, swipe, and drag.

1.4.5 Reduce the data

When developing your applications you have to remember to reduce the amount of data you’re sending to the browser. If you find yourself pushing megabytes of data to the server for a single Ajax request reconsider your approach.

Along these lines, also aim for a reduction in data complexity. Remember that these devices are relatively low-powered and any time spent manipulating complex data could be spent allowing the user to interact with the application. Tasking your mobile application to deal with deeply nested and complicated data structures is highly discouraged.

Server-side developers will have to work harder

Often deeply nested data structures are passed to clients because of the amount of work involved for the server-side developers to reduce complexity. We’d much rather have our server-side developers work harder than impact the performance of the client and thus avoid a negative view of our mobile applications.

Through your application development iterations we suggest placing yourself in the shoes of your end user. Remember, mobile applications should be quick and responsive.

1.5 Summary

We covered quite a bit in this first chapter, beginning with a high-level discussion about what Sencha Touch is and the problems that it aims to solve for the mobile-web application space.

You then took a deep dive into the world of the Sencha Touch UI widget set and learned about what’s offered out of the box. You explored some of the differences between widgets, such as the Data View and List view.
Finally, you learned how you should think about mobile applications and some of the limitations that mobile devices pose.

In the next chapter we’ll begin our deep dive into Sencha Touch, beginning with where to get the framework, and then we’ll inspect its contents. After you’ve become familiar with setting up a basic Sencha Touch app page you’ll develop a quick application with the framework.
The Sencha Touch framework makes it easy to build cross-platform mobile apps using HTML5 and JavaScript. It offers numerous features that mimic native mobile APIs and an MVC architecture that feels right at home for application developers. So you get the power and richness of native apps and the convenience of standard web tools.

Sencha Touch in Action is a complete guide for developers of native-quality mobile Sencha Touch applications. You’ll explore real-world examples as you master this impressive framework from the ground up. The book shows you good practices for mobile web development, from widget implementation to the structure of MVC applications.

What's Inside

- Covers Sencha Touch 2
- Build on your existing web dev skills
- Create mobile web apps that feel like native apps
- Extend enterprise apps to mobile clients

The book requires basic JavaScript skills. It assumes no experience with Sencha Touch or Ext JS.

Jay Garcia is a popular speaker, Sencha community advocate, and author of Ext JS in Action. Anthony De Moss is a professional web and mobile developer. Mitchell Simoens is a Sencha developer supporting the Sencha Touch and Ext JS products.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/SenchaTouchinAction