brief contents

PART 1  INTRODUCTION TO EXT JS 4.0  ........................................... 1
  1  ■  A framework apart  3
  2  ■  DOM manipulation  28
  3  ■  Components and containers  45

PART 2  EXT JS COMPONENTS .............................................................. 65
  4  ■  Core UI components  67
  5  ■  Exploring layouts  91
  6  ■  Forms in Ext JS  119
  7  ■  The data store  147
  8  ■  The grid panel  168
  9  ■  Taking root with trees  196
 10  ■  Drawing and charting  218
 11  ■  Remote method invocation with Ext Direct  251
 12  ■  Drag-and-drop  269

PART 3  BUILDING AN APPLICATION ................................................. 311
  13  ■  Class system foundations  313
  14  ■  Building an application  337
Envision a scenario where you’re tasked to develop an application with many of the typical user interface (UI) widgets such as menus, tabs, data grids, dynamic forms, and stylized pop-up windows. You want something that allows you to programmatically control the position of widgets, which means it has to have layout controls. You also want detailed and organized centralized documentation to ease your learning curve with the framework. Finally, your application needs to look mature and go into beta phase as quickly as possible, which means you don’t have a lot of time to toy with HTML and CSS. Before entering the first line of code for the prototype, you need to decide on an approach for developing the frontend. What are your choices?

You do some recon on the common popular frameworks and libraries on the market and quickly learn that all of them can manipulate the DOM, but only two of them have mature UI widgets: Yahoo! User Interface (YUI) and ExtJS.
With your first glance at YUI, you might think you needn’t look any further. You play with the examples and notice that they look mature but aren’t exactly professional quality, which means you’ll need to modify CSS. No way. Next, you look at the documentation. It’s centralized and technically accurate, but it’s far from user-friendly. You notice all of the scrolling required to locate a method or class. Some classes are even cut off because the left navigation pane is too small.

In this chapter, we’ll take a good look at ExtJS, and you’ll learn about some of the widgets that compose the framework. After we finish the overview, you’ll download ExtJS and take it for a test drive.

1.1 Looking at Ext JS

To develop a rich internet application (RIA) with a set of rich UI controls, you turn to ExtJS and find that, out of the proverbial box, ExtJS provides a rich set of DOM utilities and widgets. Although you can get excited about what you see in the examples page, it’s what’s under the hood that’s most exciting. ExtJS comes with a full suite of layout management tools to give you full control over organizing and manipulating the UI as requirements dictate. One layer down exist what are known as the Component model and Container model, each playing an important role in managing how the UIs are constructed.

Component and Container models

The Component and Container models play a key role in managing UIs with ExtJS and are part of the reason ExtJS stands out from the rest of the Ajax libraries and frameworks. The Component model dictates how UI widgets are instantiated, rendered, and destroyed in what’s known as the component life cycle. The Container model controls how widgets can manage (or contain) other child widgets. These are two key areas for understanding the framework, which is why we’ll spend a lot of time on these two topics in chapter 3.

Almost all UI widgets in the framework are highly customizable, giving you the option to enable and disable features, override functions, and use custom extensions and plug-ins. One example of a web application that takes full advantage of ExtJS is conjoon. Figure 1.1 shows a screenshot of conjoon in action.

conjoon is an open source personal information manager and can be considered the epitome of web applications developed with ExtJS. It uses just about all of the framework’s native UI widgets and demonstrates how well the framework can integrate with custom extensions such as YouTubePlayer, LiveGrid, and ToastWindow.

You’ve learned that ExtJS can be used to create a full-page web application. It’s quite easy to see that a lot can be achieved using this framework. As you’ll soon learn, the framework is pretty vast, and the API documentation will become your best friend.

Speaking of the API documentation, let’s switch gears and take a glance at it.
Looking at ExtJS

1.1.1 Rich API documentation

With the 4.0 version of the framework, the API documentation is new and improved. When opening the API documentation for the first time, you get a sense of the framework's polish. Unlike competing frameworks, the Ext JS API documentation uses its own framework to present a clean and easy-to-use documentation tool that uses Ajax to provide the documentation.

We’ll explore all of the features of the API and talk about some of the components used in this documentation tool. Figure 1.2 illustrates some of the components used in the Ext JS API documentation application.

The API documentation tool is chock-full of gooey GUI goodness and incorporates six of the most commonly used widgets, including the text input field, tree panel, tab panel, panel, and toolbar with embedded buttons.

**History support**

The Ext JS 4.0 documentation now includes browser history support. This means that you can use the browser’s forward and back buttons to walk up and down your API documentation breadcrumbs.

You’re probably wondering what all of these are and what they do. Let’s take a moment to discuss these widgets before we move on.
The text input field is a widget that wraps the native browser text input form control, adding features such as validation. In the API documentation, it’s used to perform live searches against the tree panel and is custom styled. We’ll talk more about tab panels in chapter 4.

The tree panel widget displays hierarchical data visually in the form of a tree much like Windows Explorer displays your hard drive’s folders. The tab panel provides a means to have multiple documents or components on the canvas but allows only one to be active at a time, though in the API documentation, it displays only one item.

The panel is a workhorse of Ext JS. It’s flexible and contains many areas to display content, including the dock and the content body. The dock is where items like toolbars are typically placed, and the content body is the area where content or child widgets are typically rendered. In the case of the API documentation, the content body contains the documentation for the framework.

The Toolbar class provides a means to present commonly used UI components such as buttons and menus, but it can also contain, as in this case, any of the Ext.form.Field subclasses. You can think of the toolbar as a place for the common file-edit-view menus that you see in popular operating systems and desktop applications.

Using the API is a cinch. To view a document, click the class node on the tree. Doing so invokes an Ajax request to fetch the documentation for the desired class. Each document for the classes is an HTML fragment (not a full HTML page).

So the documentation is thorough. But what about rapid application development? Can Ext JS accelerate your development cycles?
1.1.2 **Rapid development with prebuilt widgets**

Ext JS can help you jump from conception to prototype because it offers many of the required UI elements already built and ready for integration. Having these UI widgets prebuilt, instead of having to engineer them, saves you a lot of time. In many cases, the UI controls are highly customizable and can be modified to your application’s needs.

1.2 **What you need to know**

Although being an expert in web application development isn’t required to develop with Ext JS, developers should have some core competencies before attempting to write code with the framework.

The first of these skills is a basic understanding of Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS). It’s important to have some experience with these technologies because Ext JS, like any other JavaScript UI library, uses HTML and CSS to build its UI controls and widgets. Although its widgets may look like and mimic typical modern operating system controls, it all boils down to HTML and CSS in the browser.

Because JavaScript is the glue that holds Ajax together, we recommend you have a solid foundation in JavaScript programming. Again, you needn’t be an expert, but you should have a good grasp of key concepts such as arrays, references, and scope. It’s a plus if you’re familiar with object-oriented JavaScript fundamentals such as objects, classes, and prototypal inheritance. If you’re new to JavaScript, you’re in luck. JavaScript has existed nearly since the dawn of the internet. An excellent place to start is W3Schools.com, which offers a lot of free online tutorials and even has sandboxes for you to play with JavaScript online. You can visit them at http://w3schools.com/JS/.

If you need to develop code for the server side, you must have a server-side solution for Ext JS to interact with as well as a way to store data. To persist data, you’ll need to know how to interact with a database or filesystem via your server-side language of choice.

Naturally, the range of solutions available is quite large. For this book, we won’t focus on a specific language. Instead, we’ll use online resources at http://ExtJSinaction.com, where we’ve done the server-side work for you. This way, all you have to focus on is learning Ext JS. Along the way, we’ll provide specific API URLs for you to use.

We’ll begin our exploration of Ext JS with a bird’s-eye view of the framework, where you’ll learn about the categories of functionality.

1.3 **A tour of the Ext JS widgets**

The story of Ext JS main codebase begins in early 2010, during the development of Sencha Touch, the world’s first HTML5 mobile framework (released in November 2010). Sencha Touch brought forth the base underpinnings, known as Sencha Platform (see figure 1.3), which contains many of the critical features that Ext JS and Sencha Touch both use. Such common features include DOM and event management, the Component model, and layouts, all of which we’ll be diving into later in this book.
The Ext JS framework provides not only UI widgets but also a host of other features. These fall into seven major areas of purpose: core, UI components, web remoting, data services, drag-and-drop, draw and charts, and general utilities. Figure 1.4 illustrates the seven areas of purpose.

Knowing what the different areas of purpose are and what they do will give you an edge when developing applications, so we’ll take a moment to discuss them.

**CORE**
The first feature set is the Ext JS core, which comprises many basic features such as Ajax communication, DOM manipulation, and event management. Everything else is dependent on the core of the framework, but the core isn’t dependent on anything else.

**UI COMPONENTS**
The UI components contain all of the widgets that interface with the user.

**WEB REMOTING**
Web remoting is a means for JavaScript to remotely execute method calls that are defined and exposed on the server, which is commonly known as a remote procedure call (RPC). It’s convenient for development environments where you’d like to expose your server-side methods to the client and not worry about all of the fuss of Ajax method management. This package is known as Ext Direct.

**DATA SERVICES**
The data services section takes care of all your data needs, which include fetching, parsing, and loading information into stores. With the Ext JS data services classes you can read Array, XML, and JavaScript Serialized Object Notation (JSON), which is a
data format that’s quickly becoming the standard for client-to-server communication. Stores typically feed UI components.

**Draw and Charts**
This all-new package encompasses the Ext JS cross-browser drawing engine compatible with Vector Markup Language (VML) and Scalable Vector Graphics (SVG). With Draw, you can generate your own data visualizations, but its primary purpose is to act as a foundation for the Charting package. The Charting package comes complete with many popular charts including Cartesian (Bar, Line, Column, and so on), Pie, Area, Scatter, and others.

**Get your JSON on!**
Even though JSON has been around for many years, if this is the first time you’ve heard of it we encourage you to visit http://json.org, the go-to source for information on this ubiquitous data exchange format. If you’re interested in learning how to implement JSON in your server-side language of choice, there are a ton of JSON implementations, most of which are documented and explained online. We suggest searching Google using a query like “PHP JSON.”

**Drag-and-Drop**
Drag-and-drop is like a mini-framework inside Ext JS, where you can apply drag-and-drop capabilities to an Ext JS component or any HTML element on the page. It includes all the necessary members to manage the entire gamut of drag-and-drop operations. Drag-and-drop is a complex topic; we’ll spend the entirety of chapters 13 and 14 on this subject alone.

**Utilities**
The utilities section consists of cool utility classes that help you more easily perform some of your routine tasks. An example is Ext.util.Format, which allows you to format or transform data easily. Another neat utility is the CSS singleton, which lets you create, update, swap, and remove style sheets as well as request the browser to update its rule cache.

Now that you have a general understanding of the framework’s major areas of functionality, let’s look at commonly used UI widgets that Ext JS has to offer.

### 1.3.1 Containers and layouts at a glance
Even though we’ll cover these topics in detail in chapter 3, let’s spend a little time here talking about containers and layouts. The terms container and layout are used extensively throughout this book, and we want to make sure you have at least a basic understanding of them before we continue. Afterward, we’ll begin our exploration of visual components of the UI library.

**Containers**
Containers are widgets that can manage one or more child items. A child item is generally any widget or component that’s managed by a container or parent; thus the parent-child paradigm. You’ve already seen this in action in the API. The tab panel is a
CHAPTER 1  A framework apart

container that manages one or more child items, which can be accessed via tabs. Please remember this term, because you’ll use it a lot when you start to learn more about how to use the UI portion of the framework.

LAYOUTS
Layouts are implemented by a container to visually organize the child items in the container’s content body. Ext JS has a whopping 33 layouts in the library! The good news is that you only have to learn 13 of them, which we’ll go into in great detail about in chapter 5, where we show the ins and outs of each layout.

Now that you have a high-level understanding of containers and layouts, let’s look at some containers in action. In figure 1.5 you see two subclasses of Container—Panel and Window—each engaged in parent-child relationships, demonstrating the power of the Container class and various layouts.

The Panel (left) and Window (right) in figure 1.5 each manage two child items. Child Panel 1 of each parent container contains HTML. The children with the title Child Panel 2 manage one child panel each using AutoLayout, which is the default container layout. This parent-child relationship is the crux of all the UI management of Ext JS and will be reinforced and referenced repeatedly throughout this book.

You learned that containers manage child items and use layouts to visually organize them. Now that you have these important concepts down, we’ll see and discuss other containers in action.

1.3.2 Other containers in action
You saw the Panel and Window subclasses used when you learned about Containers. Figure 1.6 shows some other commonly used subclasses of Container.

In figure 1.6 you see the form panel, tab panel, window, toolbar, and field container widgets. The form panel works with the Basic Form class to wrap fields and other child items with a form element. All of these widgets are contained by an instance of Ext.window.Window.

You’ll spend some time building a complex UI in chapter 6, where you’ll learn more about form panels. For now, let’s move on to see what data-presentation widgets the framework has to offer.
1.3.3 Data-bound views

You’ve already learned that the data services portion of the framework is responsible for the loading and parsing of data. Ext JS 4.0 has a lot of widgets that are bound to data stores, known as views. Many of the views that you’ll deploy include the data view, grid panel, and tree panel. If your application requires charts, you’ll be pleased to learn that all of the charts in the framework are also considered views and are bound to data stores. Figure 1.7 shows the Ext JS grid panel in action.

The newly refactored GridPanel is a subclass of Panel and presents data in a table-like format, but its functionality extends far beyond that of a traditional table, offering sortable, resizable, and movable column headers and selection models such as RowSelectionModel and CellSelectionModel. You can customize its look and feel and couple it with a paging toolbar to allow large datasets to be segmented and displayed in pages. It contains many features and plug-ins, allowing you to do tasks such as edit by row or cell, or lock a column. The data view shown in figure 1.8 renders photos and other bits of data for various phones on the market.

The DataView class consumes data from a store, paints it onscreen using a class known as XTemplate, and provides a simple selection model. The Ext JS XTemplate is an HTML fragment-generation utility that allows you to create a template with
placeholders for data elements, which can be filled in by individual records in a store and stamped out on the DOM.

**Gone is the list view widget!**

If you’re coming from Ext JS 3, you may wonder where the list view widget is. The simple answer is that the list view, providing faster table rendering in Ext JS 3.0, was removed from 4.0, in favor of refactoring the grid panel for much faster performance.

The grid panel and data view are essential tools for painting data onscreen, but they do have one major limitation: they can show only lists of records and can’t display hierarchical data. This is where the tree panel fills the gap.
13

A tour of the Ext JS widgets

1.3.4 Make like a tree panel and leaf

The tree panel widget is an exception to the list of UI widgets that consume data in that it doesn’t consume data from a data store. Instead, it consumes hierarchical data via the use of the TreeStore class. Figure 1.9 shows an example of an Ext JS tree panel widget. Here, the tree panel is being used to display the parent-child data inside the directory of an installation of the framework.

For Ext JS 4.0, it has been completely rebuilt and is now a close cousin to the grid panel. Figure 1.10 demonstrates the versatility of the new tree panel.
You already saw text fields when we discussed containers. Next, we’ll look at some of the other input fields that the framework has to offer.

### 1.3.5 Form input fields

ExtJS has a palette of eight input fields. They range from simple text fields, as you’ve already seen, to complex fields such as the ComboBox and the HTML Editor. Figure 1.11 shows the ExtJS form field widgets available out of the box.

As you can see in figure 1.11, some of the form input fields look like stylized versions of their native HTML counterparts. The similarities end there, though. With the ExtJS form fields, there’s much more than meets the eye.

Each of the ExtJS fields (except for the HTML Editor) includes a suite of utilities to perform actions such as getting and setting values, marking the field as invalid, resetting, and performing validations against the field. You can apply custom validation to the field via regex or custom validation methods, giving you complete control over the data being entered into the form. The fields can validate data as it’s being entered, providing live feedback to the user.

**TEXTFIELD AND TEXTAREA**

The TextField and TextArea classes can be considered extensions of their generic HTML counterparts that include extra features like validation. The TextField class is the base for many other complex widgets, such as the ComboBox, Number field, and Time field.

![Figure 1.11 The out-of-the-box form elements displayed in an encapsulating window](image)
A tour of the Ext JS widgets

RADIO AND CHECKBOX
Like the text field, radio and checkbox fields are extensions of the out-of-the-box HTML radio and checkbox, but they include all of the Ext JS element management goodness and have convenience classes to assist with the creation of checkbox and radio groups with automatic layout management. Figure 1.12 shows a small sample of how the Ext JS CheckboxGroup and RadioGroup classes can be configured with complex layouts.

HTML EDITOR
The HTML Editor is WYSIWYG, like the text area on steroids. The HTML Editor uses existing browser HTML editing capabilities and can be considered somewhat of a black sheep when it comes to fields. There’s much more to discuss about this field, which we’re going to save for chapter 6. But for now, let’s circle back to ComboBox and its subclass, TimeField.

TRIGGERFIELD FAMILY OF FIELDS
The TriggerField class is the base class responsible for rendering a button to the right of a text field. Its subclasses are broken up into two groups, pickers and spinners. Included in the list of pickers are the ComboBox and the date field. The spinners include the spinner and number fields.

The ComboBox is easily the most complex and configurable form input field. It can mimic traditional option drop-down boxes or can be configured to use remote datasets via the data store. It can be configured to autocomplete text (known as type-ahead) entered by the user and to perform remote or local filtering of data. It can also be configured to use your own instance of an Ext JS XTemplate to display a custom list in the drop-down area, known as the bound list. Figure 1.13 shows an example of a custom ComboBox in action, being used to search the Ext JS forums. The ComboBox here shows information like the post title, date, and author, and a snippet of the post in the list box. Because some of the dataset ranges are so large, it’s configured to use a paging toolbar, allowing users to page through the resulting data.
Because the ComboBox is so configurable, you could also include image references to
the resulting dataset, which can be applied to the resulting rendered data.

Here we are, on the last stop of our UI tour. Now let’s take a peek at some of the
other UI components that work anywhere.

1.3.6 Other widgets

A bunch of UI controls stand out that aren’t major components but that play supporting roles in the grander scheme of a UI. Look at figure 1.14 for a palette of the various widgets rendered onscreen.

You’ve learned how Ext JS can help you get the job done through a large palette of widgets. You’ve learned that you could elect to use Ext JS to build an application without touching an ounce of HTML. You also got a top-down view of the framework, which included a UI tour. All of the material discussed thus far existed for Ext JS 3.0. Let’s take a moment to discuss what’s new in Ext JS 4.0.
1.4 What’s new in Ext JS 4.0

We aren’t exaggerating when we say that Ext JS 4.0 is a revolution for JavaScript frameworks. There are so many enhancements to the framework that it’s sometimes hard to grasp all that’s changed. A lot of the changes are beneath the presentation layer, in the deepest, darkest caverns of the Ext JS codebase, a place where you rarely venture due to its sometimes mind-bending complexity.

Next, we’ll look at some of the most drastic transformations that the framework has undergone. If you have experience in version 3.0, you may have wondered why the size of the framework has grown. You’ll learn the reason in the next few sections.

1.4.1 Poof goes the adapter layer!

Through the use of an adapter layer, Ext JS 2.0 and 3.0 were able to ride on top of the jQuery, Prototype, and YUI libraries. With Ext JS 4.0, this is no longer the case.

Though heralded by developers migrating from those libraries, the adapter layer has always been a source of contention for a number of reasons. The main issue with the adapter layer has been that the versions of the base libraries would change and introduce bugs into Ext JS.

Another well-known issue is the problem of framework namespace collision. Ext JS 1.0–3.0 added to JavaScript by injecting methods into the Function, String, and Array prototypes. Because other libraries took the same action with similar method names, Ext JS trampled on the changes that the base libraries made.

The Sencha development team made sure to prevent such collisions and sources of tension with other libraries by moving said features into the Ext.util namespace as String, Function, and Array singletons. With such changes, the Sencha team decided to remove the adapter layer and make Ext JS work alongside any other library, allowing you to use any version of those libraries without fear that an upgrade of those libraries would cause problems with your Ext JS code.

1.4.2 New class system

Ext JS 4.0 comes with an entirely new class system that includes features such as dependency injection and on-the-fly class loading, a must-have for internet-facing RIAs built with Ext JS 4.0.

Along with dynamic class loading comes the concept of mixins, a modern object-oriented programming pattern that allows for multiple inheritance. This concept has allowed the Sencha development team to be much more creative when developing the framework, reducing the amount of duplicate code while increasing the level of functionality and sometimes the ease of use for some classes and widgets.

Learn about mixins!

If you’re new to the concept of mixins, the following article explains this programming concept very well: http://en.wikipedia.org/wiki/Mixin.
Although the new class system provides many new features, it comes at a cost: new patterns. The new class system promotes vastly different patterns compared to those of Ext JS 3.0 when it comes to instantiation or defining a class. These new patterns can make the learning curve for Ext JS 4.0 steeper, but rest assured that they’ll allow you to be more creative with your application code.

Speaking of classes, Ext JS 4.0 has a completely refactored data class system, which we discuss next.

### 1.4.3 Data package

The all-new data package in Ext JS 4.0 can trace its origins back to Sencha Touch, which used terms such as `model` in place of `record`. The changes to the data package bring functionality and organization far beyond that of Sencha Touch, however.

The Ext JS 4.0 data package incorporates an explosion of classes, and it includes new members such as the LocalStorage proxy and tree store. The LocalStorage proxy allows data to be stored and retrieved using the browser’s local storage feature, whereas the tree store replaces the Ext JS 3.0 tree loader, allowing you to do a lot more with trees than ever before.

The data package comes with added features, such as associations and validations, along with a well-thought-out reorganization of functionality. Figure 1.15 illustrates how features and functions of the data package are segmented and related. We’ll be going over this package in greater detail later on; we just wanted to whet your appetite with some detail.

In Ext JS 4.0, models can directly use proxies, whereas they couldn’t in previous versions of the framework. Likewise, validations and associations are now performed at the model level.

The data package has seen a lot of attention, but the layout namespace has seen a lot of refactoring love as well.

![Figure 1.15 The Ext JS 4.0 data package](image-url)
1.4.4 **Layouts: an explosion of code**

As we discussed earlier, Ext JS 4.0 comes jam-packed with 33 new layout managers, but there are only 13 that you need to be aware of. This is because layouts are broken up into two main areas of functionality: component and container layouts.

The component and container layouts play two completely different roles in the framework. Component layouts are responsible for arranging the HTML for components, whereas container layouts are responsible for managing the location and size of child components.

While we’re on the topic of layouts, let’s shed some light on the new docking system that Ext JS 4.0 brings to the table.

1.4.5 **New docking system**

Originating in Sencha Touch, panels in Ext JS can have widgets such as toolbars arranged on the outside of the area known as the *content body*, affording more UI arrangement flexibility than ever before with this widget. Figure 1.16 shows three toolbars docked on the top, bottom, and left of a panel. This arrangement wasn’t possible with any previous versions of Ext JS without deep nesting of containers and layouts. This is all made possible via the component layout known as Dock.

Though using the Dock layout is something that you might be able to envision taking full advantage of, if your application uses grid panels what we’re about to discuss next might excite you.

1.4.6 **Grid panel improvements**

The Sencha development team literally worked night and day on features like the grid panel, and the results show, especially after taking a good look at what’s changed since Ext JS 3.0.
Features new to the grid panel include what’s known as the *infinite grid*, which allows you to paginate through large datasets without having to include a paging toolbar. Other new features include a reorganization of the namespace for better grouping of classes (see figure 1.17).

The grid area of code has been segmented by groups of code, including column types, plug-ins, and features. Even though not technically in the grid namespace, dataStore is a supporting class for GridPanel, so we included it in figure 1.17.

This level of organization of the grid package means that you have more flexibility in configuring grid panels, allowing Ext JS to implement only code that’s required. For instance, if you want to allow cell editing, you include the CellEditing plug-in in your grid panel configuration. Likewise, if you want to include drag-and-drop functionality, include the DragDrop plug-in.

Other bits of functionality were migrated to the so-called *feature namespace*, which is somewhat similar to plug-ins. We don’t want to muddy the waters with details of how features work, but it’s good to note that grid goodies like row grouping and providing a summary row of your data can be engaged only if you desire them to be.

As you just learned, the grid panel endured a lot of changes. The story of major change doesn’t end here. The tree panel has undergone some serious changes as well!

### 1.4.7 Tree panel now closer to grids

The code for the Ext JS tree panel has stayed relatively the same for Ext JS versions 1.0 through 3.0, but the Ext JS 4.0 tree panel code has been completely rewritten. Applying a family tree analogy to the difference between the grid and tree panels in prior Ext JS versions, we could say that they were, at best, third cousins. In Ext JS 4.0, they’re siblings!

As illustrated in figure 1.18, the grid and tree panels are siblings because they share the same superclass, meaning they share the same base code. The good news is that once you learn one of the two, the learning curve is reduced for the other. Having the grid and tree panels share the same base code means that you can have things like columns in your tree views.
That said, the tree panel doesn’t contain a lot of the functionality that the grid panel sports, such as the summary row plug-in or column locking. In addition, the tree panel must use the `TreeStore` class from the data package to manage and display hierarchical data.

We just covered two of the major data-bound views that have been with the framework since its early days. Next we’ll tackle the all-new charting package.

### 1.4.8 Draw and charts

Charts were first introduced in Ext JS 3.0 with relatively little fanfare. There were two reasons for this. The first is that they were Flash-based charts repackaged from the YUI library. The second is that upgrades to the YUI packaged charts often lagged behind a few revisions, frustrating developers.

With Ext JS 4.0, the YUI charting package was tossed and rebuilt from scratch in two major sections. The first is Ext Draw, which is a mini-framework inside Ext JS that has its roots in lessons learned by RaphaelJS, a Sencha labs project for drawing in the browser using Vector Markup Language (VML), Scalable Vector Graphics (SVG), or Canvas.

The second is the charting package, which uses Ext Draw as a base. With the new charting package come two new graphs: scatter and radar. Figure 1.19 shows the radar chart.

We’ve discussed many elements from the UI widgets, but there are others under the hood that are worth mentioning.

![Radar Chart](image)

**Figure 1.19** Ext JS contains new charts that don’t use Flash.
1.4.9 **New CSS styling architecture**

Ext JS uses Sass (Syntactically Awesome Stylesheets) to allow both the Sencha development team and users to create custom themes. 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).

---

Custom style sheets and widgets enable you to develop applications with Ext JS. They need something to tie them together, and with Ext JS 4.0, Sencha has delivered such a tool.

1.4.10 **New MVC architecture**

One of the things that Ext JS has lacked is a solid pattern for developing applications with the framework. This isn’t the case with Ext JS 4.0. Using the lessons learned with Sencha Touch, Ext JS 4.0 comes with a solid MVC architecture that lets you develop code using the tried-and-true MVC pattern. We’ll go over this in great detail in the last two chapters of this book.

The new stuff for Ext JS 4.0 doesn’t apply just to what can be used in the browser. The framework comes with other tools that you can use in your application build process.

1.4.11 **Bundled packaging tool**

Earlier, you learned that Ext JS 4.0 comes with a dynamic class-loading system. The class loader is a great solution for internet-based Ext JS applications, but intranet-based applications often have higher demands on response times, which is why Sencha now includes its popular JSBuilder packaging and minification tool, the same tool it uses to build and package Ext JS and Sencha Touch.

We’ve spent a lot of time looking at what’s new in the framework. It’s time that you download it and begin using it.

1.5 **Downloading and configuring**

Even though downloading Ext JS is a simple process, configuring a page to include Ext JS isn’t as simple as referencing a single file in HTML. Now you’ll learn about configuration, the folder hierarchy, and what folders are and what they do.

The first thing you need to do is get the source code. To do so, visit www.sencha.com/products/ExtJS/download/. The downloaded file will be the SDK in zip format, which weighs in at over 30 MB in size. We’ll explain why this file is so large in a moment. Now extract the file to a place where you serve JavaScript. To use Ajax and view the documentation without having to visit sencha.com, you’re going to need a web server. We typically use Apache configured locally on our computer, which is free and cross-platform, but IIS for Windows will do.
If you’re like us, you probably checked the size of the files extracted from the downloaded SDK zip file. If your jaw dropped, feel free to pick it back up. Yes, over 30 MB is rather large for a JavaScript framework. Pay no attention to the size for now; figure 1.20 shows what was extracted.

Looking at the contents of the SDK, you see a lot of stuff. The reason there are so many folders and files is that the downloadable package contains a few copies of the entire codebase and CSS. It’s done this way because you have the freedom to build or use Ext JS any way you see fit. Table 1.1 explains what each of the folders is and what each one does.

Table 1.1  The contents of the Ext JS SDK

<table>
<thead>
<tr>
<th>Folder</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>build</td>
<td>Contains the necessary scripts to use JSBuilder to concatenate and minify your application code.</td>
</tr>
<tr>
<td>builds</td>
<td>Contains three different builds of the Ext JS framework. First is the sandbox version, where you can run Ext JS 4.0 inline with Ext JS 3.0 to mitigate migration risk. The core contains DOM management and various utilities in the framework. The last item is Ext JS foundation, which is the base of the Ext JS framework.</td>
</tr>
<tr>
<td>docs, overview, and welcome</td>
<td>The docs folder holds the full API documentation, and the overview directory contains a quick introduction to the framework. The welcome folder contains the necessary resources to support the framework’s splash screen, which you make visible by double-clicking index.html.</td>
</tr>
</tbody>
</table>
Although there are quite a few files and folders in the distribution, you need only a few of them to get the framework running in your browser. Now is a good time to look at using Ext JS for the first time.

1.6 Take it for a test drive

For this exercise you’re going to create an instance of Ext.form.Panel, which will be rendered inside an Ext.window.Window. The form panel will contain two text input fields and a button to provide some feedback once clicked. The following listing demonstrates how you’ll bootstrap the application code.

Listing 1.1 Creating hello_world.html

```html
<link rel="stylesheet" type="text/css"
     href="/ExtJS/resources/css/ext-all.css" />
<script type="text/javascript"
        src="/ExtJS/ext-all-debug.js"></script>
<script type="text/javascript" src='hello_world.js'></script>
```

Listing 1.1 shows the HTML markup for a typical Ext JS–only setup, which includes the concatenated CSS file, ext-all.css, and the required JavaScript file, ext-all-debug.js. Last, it includes your soon-to-be-created hello_world.js file.

The listing uses /ExtJS as the absolute path to the framework code. Be sure to change it if your path is different. Create a script tag pointing to the hello_world.js file, which will contain your main JavaScript code.
Next you’re going to create the hello_world.js file in two phases. The first, shown in the next listing, is the construction of the form panel and its related child components.

### Listing 1.2 Creating hello_world.js

```javascript
var tpl = Ext.create('Ext.Template', [
  'Hello {firstName} {lastName}!
  ' Nice to meet you!
]);

var formPanel = Ext.create('Ext.form.FormPanel', {
  itemId : 'formPanel',
  frame : true,
  layout : 'anchor',
  defaultType : 'textfield',
  defaults : {
    anchor : '-10',
    labelWidth : 65
  },
  items : [
    {
      fieldLabel : 'First name',
      name : 'firstName'
    },
    {
      fieldLabel : 'Last name',
      name : 'lastName'
    }
  ],
  buttons : [
    {
      text : 'Submit',
      handler : function() {
        var formPanel = this.up('#formPanel'),
        vals = formPanel.getValues(),
        greeting = tpl.apply(vals);
        Ext.Msg.alert('Hello!', greeting);
      }
    }
  ]
});
```

Listing 1.2 shows the code needed to configure a form panel that contains two input fields and a button. First, you create an instance of Ext.Template 1, which you’ll use later to create a dynamic dialog text body. Next, you create an instance of Ext.form.FormPanel 2, which contains two text input fields 3 and a button 4. The button is configured with a handler that uses the template you configured earlier and values from the form panel to display an Ext.Msg alert dialog 5.

You’re almost done with the “Hello world” example. Your form hasn’t been rendered onscreen yet. For this, you need to call Ext.onReady. You’ll also wrap the form panel inside the window to demonstrate the flexibility of the framework in the next listing.
CHAPTER 1  A framework apart

Listing 1.3  Putting it all together

Ext.onReady(function() {

    Ext.create('Ext.window.Window', {
        height   : 125,
        width    : 200,
        closable : false,
        title    : 'Input needed. ',
        border   : false,
        layout   : 'fit',
        items    : formPanel
    }).show();

});

Listing 1.3 contains code to render your form panel inside an Ext JS window. You first call `Ext.onReady` and pass in an anonymous function, which gets executed when Ext JS deems that the browser is ready to have the DOM manipulated. Inside this anonymous function is where you create your `Ext.window.Window` instance, which contains your `FormPanel` instance. Figure 1.21 shows the example rendered with the child form panel.

Figure 1.22 shows the “Hello world” example rendered onscreen. To exercise the Submit button handler you need to enter data in the two input fields and click the Submit button. If you’ve done everything correctly, you should see the `Ext.Msg` alert dialog using the data that you placed in the form.

There you have it! You just used Ext JS to render a form panel with related input fields and a button inside an Ext JS window. Though this example was simple in nature, it shows you the power of Ext JS.

1.7   Summary

In this introduction to Ext JS, you learned how it can be used to build robust web applications. You also learned how it measures up against other popular frameworks.
on the market and that it’s the only UI-based framework to contain UI-centric support classes such as the Component, Container, and Layout models.

You explored many of the core UI widgets that the framework provides and learned that the many prebuilt widgets help rapid application development efforts. We also investigated some of the changes that Ext JS 4.0 has implemented, such as all-new non-Flash charts and the MVC package.

Finally, you saw how to download and set up the framework with each base framework. You created a “Hello world” example of how to use an Ext JS window to render a form panel with a button that displays an Ext.Msg alert dialog with some simple JavaScript.

In the chapters to follow, you’ll explore how Ext JS works from the inside out. This knowledge will empower you to make the best decisions when building well-constructed UIs and better enable you to use the framework effectively. This will be a fun journey.
Ext JS IN ACTION, Second Edition
Garcia • Grisogono • Andresen

Ext JS is a mature JavaScript web application framework that provides modern UI widgets and an advanced MVC architecture. It helps you manage tedious boilerplate and minimize hand-coded HTML and browser incompatibilities.

Ext JS in Action, Second Edition starts with a quick overview of the framework and then explores the core components by diving into complete examples, engaging illustrations, and clear explanations. You’ll feel like you have an expert guide at your elbow as you learn the best practices for building and scaling full-featured web applications.

What’s Inside

• Building professional web apps with Ext JS
• Stamping out DOM fragments with templates
• Customizing and building Ext widgets
• Masterful UI design

A working knowledge of JavaScript is assumed. No prior experience with Ext JS is required.

Jay Garcia is a well-known member of the Ext JS community and a contributor to the framework. He wrote Sencha Touch in Action. Grgur Grisogono founded SourceDevCon in London, UK and Split, Croatia. Jacob Andresen is a consultant specializing in large scale internet applications.

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

“A must-have book to learn Ext JS properly.”
—Loiane Groner, Citibank

“A guided tour for an action-packed journey from novice to expert.”
—Jeet Marwah, gen-E

“Master every detail of Ext JS 4.0 and how to build desktop-like UIs for the web.”
—Efran Cobisi
Microsoft MVP, MCSD

“The missing link between experimenting with Ext JS and being productive with it.”
—Raul Cota
Virtual Materials Group

$49.99 / Can $52.99 [INCLUDING eBook]