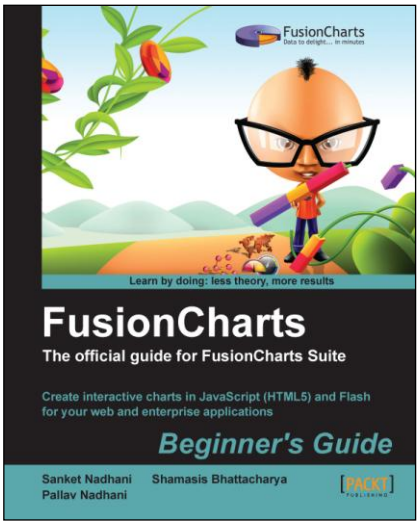
[PACKT] PUBLISHING

FusionCharts Beginner's Guide:

The official guide for FusionCharts Suite

Sanket Nadhani
Shamasis Bhattacharya
Pallav Nadhani



Chapter No. 1
"Introducing FusionCharts"

In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter NO.1 "Introducing FusionCharts"

A synopsis of the book's content

Information on where to buy this book

About the Authors

Sanket Nadhani previously headed Marketing and Sales at FusionCharts, fresh out of college. As part of the role, he had to conceptualize different dashboards and demos and was oft en frustrated with the lack of practical resources on data visualization to learn from. Once he learned them after his daily bouts with tons of data, he started writing about the usability and best practices on the FusionCharts blog that have been well received.

He loves food, beer, and travelling. You can follow him on Twitter at `http://twitter.com/sanketnadhani`.

> I want to thank the wonderful web community for sharing all their learning and experiences so selflessly. A lot of what I know and do today is because of you guys.

Shamasis Bhattacharya has been a part of FusionCharts since 2008. As a JavaScript architect, he heads the JavaScript development team and spends most of his time analyzing, modeling, and coding the FusionCharts JavaScript charting library with attention to smart soft ware design and innovative data visualization countenances.

He writes on his blog `http://www.shamasis.net/` and spends the rest of his time with his wife, Madhumita.

> My part in this book wouldn't have been worth reading had my wife not been around helping me and had not Pallav e-mailed me questioning whether I had been smoking while writing the draft!

Pallav Nadhani is the co-founder and CEO of FusionCharts, and an angel investor. He started FusionCharts at the age of 17 as a way to make pocket money. Today, FusionCharts has around 20,000 customers and 400,000 users in over 110 countries. His entrepreneurial journey has been covered by various magazines such as Forbes, Entrepreneur, Business Today, Economic Times, and numerous blogs and websites. He holds a Masters in Computer Science from The University of Edinburgh and loves traveling, beer, and poker.

He has also worked on the book "*Flash.NET*", *Friends of Ed Publication.*

> I want to dedicate this book to Mom, who has always been my inspiration, and Puja, who has been my cheerful support throughout. I would also like to thank the entire FusionCharts team for taking FusionCharts to where it is today, and Hrishikesh for helping me with this book.

FusionCharts
Beginner's Guide:
The official guide for FusionCharts Suite

As web developers, we build applications that feed on data. We parse it, process it, and report it. Our reports take the form of tables, grids, and diagrams such as charts, gauges, and maps. Parsing and processing are backend tasks that are invisible to the user. The actual reporting of data, however, is a bulk of an experience a user has with our application.

This book is a practical step-by-step guide to using FusionCharts Suite to create delightful web reports and dashboards. After creating your first chart in 15 minutes, you will learn advanced reporting capabilities such as drill-down and JavaScript integration. Finally, you round up the experience by learning reporting best practices including the right chart type selection and practical usability tips to become the data visualization guru among your peers.

What This Book Covers

Chapter 1, Introducing FusionCharts, introduces you to FusionCharts Suite and teaches you how to build your first chart in under 15 minutes. You will learn the XML and JSON data formats that FusionCharts Suite supports, and apply it to build different types of charts.

Chapter 2, Customizing your Chart, brings to you the wide spectrum of customization options you have with FusionCharts Suite, both aesthetically and functionally. You will learn how to customize the chart background and font, control how numbers appear on the chart, and add more context to charts using trendlines.

Chapter 3, JavaScript Capabilities, familiarizes you with the JavaScript programmability of FusionCharts Suite. Using them, you will be able to develop rich and interactive features around your charts and also learn ways to integrate FusionCharts with your web applications.

Chapter 4, Enabling Drill-down on Charts, introduces you to the concept of drill-down in charts, which helps you drill down from a macroscopic view to a more detailed one.

Chapter 5, Exporting Charts, introduces the capability of FusionCharts Suite to be exported as images and PDF documents for use in e-mails and presentations.

Chapter 6, Integrating with Server-side Scripts, explains how to power FusionCharts using server-side technologies such as ASP.NET, PHP, and Java, and drive them through databases.

Chapter 7, Creating Maps for your Applications, introduces you to the interactive maps present in FusionMaps, a part of the FusionCharts Suite. After downloading and setting up FusionMaps, you will be able to create a simple US map and then add drill-down to go from the US map to individual states.

Chapter 8, Selecting the Right Visualization for your Data, takes a step-by-step approach to selecting the right visualization for business dashboards. You start by understanding your dashboard's audience, identify the metrics they need to see, move on to the kind of analysis the metric will need, and finally come to the chart best suited for the case in question. You will also take a closer look at specialized charts such as gauges and Gantt charts.

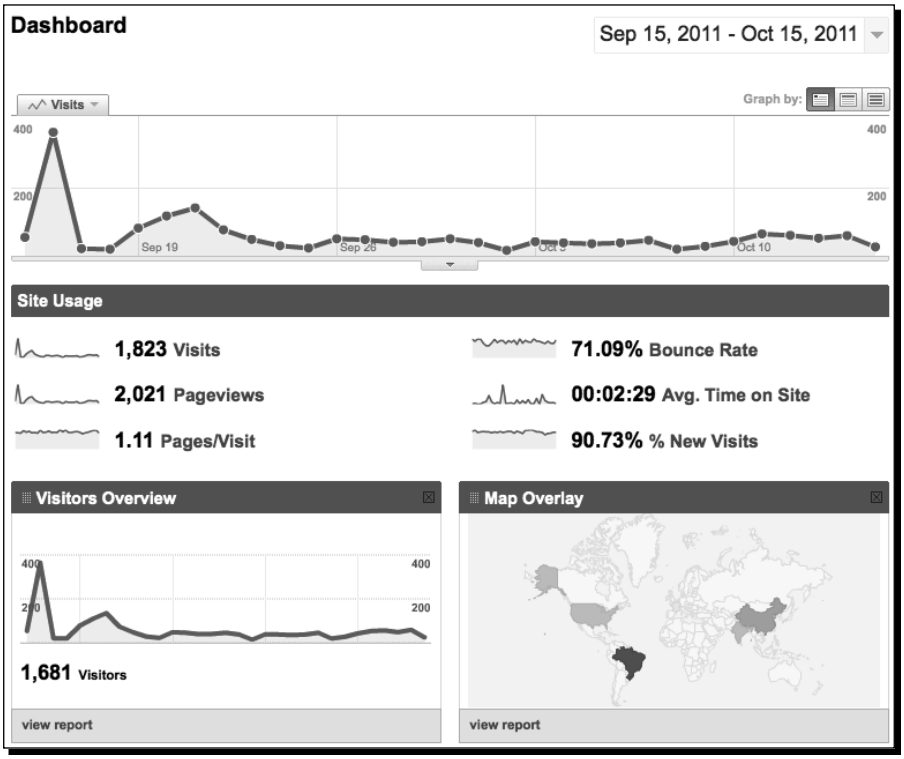
Chapter 9, Increasing the Usability of your Charts, rounds up the experience by introducing simple tips and techniques that can make your charts more usable. From obvious tips such as having descriptive captions, to less obvious ones such as removing excess detail from data, these tips will go a long way in making your dashboards more usable.

1
Introducing FusionCharts

As web developers, we build applications that feed on data. We parse it, process it and report it. Our reports take the form of tables, grids, and diagrams such as charts, gauges, and maps. Parsing and processing are backend tasks that are unseen by the user. The actual reporting of data, however, is the bulk of the experience a user has with our application.

To make our reports interesting and insightful, it is important to provide a highly engaging and functional face to the data in context. While tables, grids, and basic charting are natively supported by most web scripting languages, creating advanced or interactive charts require the use of third-party components. FusionCharts Suite is one such suite of components that help you deliver a delightful experience by aiding the creation of animated and interactive charts, gauges, and maps.

Before we jump in and look at what FusionCharts can do for you, let us see where charts, gauges, and maps can be helpful. Google Analytics, a tool that most web developers swear by, is a beautiful example of effective data presentation. In case you do not know, it is a tool that records a ton of information such as visitor demographics, referrers, advertising, browser information, and so on. With so much data recorded, it is of utmost importance to present it in a compact, yet insightful way, as shown in the following screenshot:



Note how the Dashboard has a very clean and non-cluttered look, despite the large data set it represents. Extra information, which is not required in the main layout, is displayed as tool tips and annotations. Interactive features such as a clickable world map lets you explore the data the way you want to. The selection of chart types is also immaculate, with the line chart showing the trend of the most commonly used metric—the number of visitors to the site. Other relevant metrics such as Pageviews and Pages/Visit are communicated in large text along with sparklines providing a historical context. A world map is used to display where the traffic is coming from.

FusionCharts Suite helps you build similar dashboards with a lot more chart types and interactivity. Without further ado, let us proceed and set the goals for this chapter.

In this chapter, you will:

- Learn how to set up FusionCharts
- Build your first chart and configure basic parameters
- Understand the **eXtensible Markup Language** (**XML**) and **JavaScript Object Notation** (**JSON**) data format supported by FusionCharts and different ways to provide it to the chart
- Learn how to build charts with multiple series and axes
- Create advanced charts such as **Combination charts**

What is FusionCharts Suite?

FusionCharts Suite is a collection of four products, each of which help generate different types of charts, gauges, or maps in web applications. These data-visualization components are ideal for use within reports, dashboards, analytics, surveys, and monitors in web and enterprise applications. The visualizations are rendered using both Adobe Flash and JavaScript (HTML5), thereby making the experience seamless across PCs, Macs and a wide spectrum of devices including iPads and iPhones.

The four products in the suite are:

- **FusionCharts XT**: This helps create the 45 most used chart types such as pie, column, bar, area, line, stacked, combination, and advanced ones such as Pareto and Marimekko.
- **FusionWidgets XT**: This helps create **Key Performance Indicators** (**KPI**) and make real-time data in dashboards, monitors, and reports more insightful. It includes a wide variety of charts and gauges such as dial charts, linear gauges, Gantt charts, funnel charts, sparklines, data-streaming column, line, and area charts.
- **PowerCharts XT**: This helps create charts for domain-specific usage such as those in network diagrams, performance analysis, profit-loss analysis, financial planning, stock price plotting, and hierarchical structures.
- **FusionMaps XT**: This consists of over 550 geographical maps, including all countries, US states, and regions in Europe for plotting business data.

All the products are built on a common framework and offer similar ways to use and configure them. To start with, we will create charts using FusionCharts XT and later explore charts of other products in *Chapter 8*, *Selecting the Right Visualization for your Data* and *Chapter 7*, *Creating Maps for your Applications*. Without further ado, let us get started and build our first chart. For that, you will first need to download FusionCharts Suite.

Getting FusionCharts

FusionCharts allows you to download the trial version from its website `http://www.fusioncharts.com`. This trial does not have any feature restriction or an expiry date. The only caveat is that the charts in the evaluation version have *FusionCharts* printed on the chart, which can be removed by purchasing a license of FusionCharts and later just replacing the **Shockwave** (**SWF**) and **JavaScript** (**JS**) files, as we shall see later.

Time for action – downloading and extracting FusionCharts

1. Go to `http://www.fusioncharts.com/download` and fill in your particulars in the download form and click on **Download**.
2. On the next page, you will find links to either download the entire FusionCharts Suite, or individual products from the suite. In this chapter, we will work with FusionCharts XT only and hence will explore that.
3. Once the ZIP file has been downloaded, extract it to a folder on your hard drive, which is conveniently located at `C:\FusionChartsSuite\FusionChartsXT` on Windows or `Users/{YourName}/FusionChartsSuite/FusionChartsXT` on Mac or UNIX based systems. Throughout this book, we will refer to this folder as the **FusionCharts Installation Folder**.

The steps to install and use FusionCharts remain the same, whether you are using the trial or licensed version.

What just happened?

You have now successfully downloaded FusionCharts XT and extracted it in the FusionCharts Installation Folder. We will soon learn how to use these files to build charts. Before that, let us quickly explore the contents of the FusionCharts package.

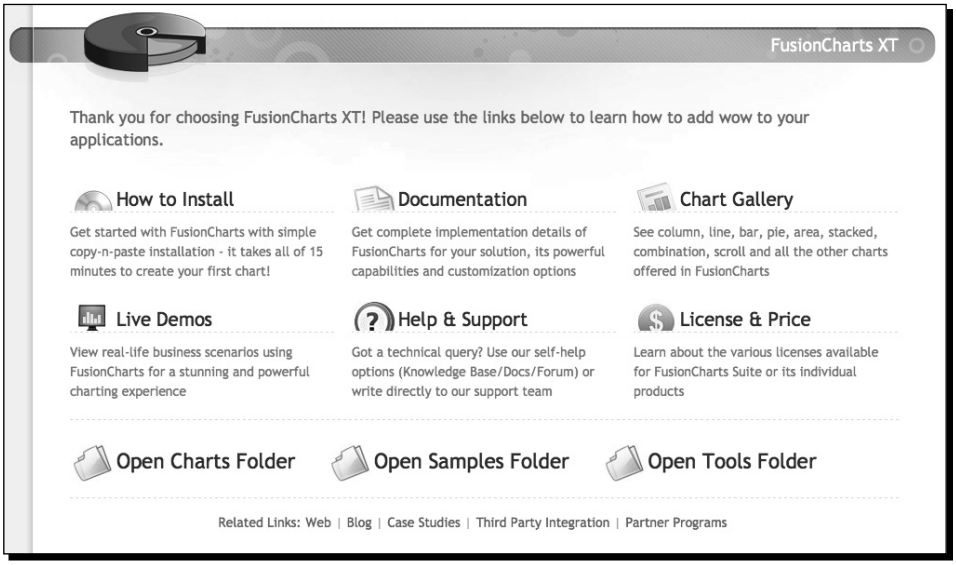
Within the FusionCharts Installation Folder, you will find multiple folders. Some of the folders are internal folders used to store documentation and gallery files, for example, `Contents`, `Gallery`, and so on. The folders that you will mostly use are `Charts`, `Code`, `SourceCode`, `ExportHandlers`, and the `Tools` folder for the following purpose:

Folder Name	What it contains?
`Charts`	Contains all the SWF and JS files that form the core of FusionCharts – we will refer to it as **Core FusionCharts** files. The FusionCharts SWF files have been created using Adobe Flash 8 and need Flash Player 8 (or above) to run. The JavaScript is compatible with IE6 (or above), Firefox, Chrome, Opera, and Safari, including that on iPads and iPhones.
`Code`	Consists of code samples in various programming languages that you can explore, to quickly learn or get started with.
`SourceCode`	Present only with the Enterprise and Enterprise Plus license, this folder contains the source code of FusionCharts in both Flash Source files (`.fla`) and JavaScript files.
`Tools`	Consists of utility applications in three subfolders: `FCDataConverter` helps you convert the FusionCharts XML data to JSON data and vice versa. We will explore this later in the chapter, after we have created our first chart. `FlashPlayerSecuritySetup` contains scripts that help you configure security settings on your local machine only when using FusionCharts with JavaScript. We will explore this later in the book when we build advanced examples of FusionCharts integrated with JavaScript. `XMLGenerator` is a visual interface to generate XML data for FusionCharts. This is primarily intended for non developers and would not be of much help to us.

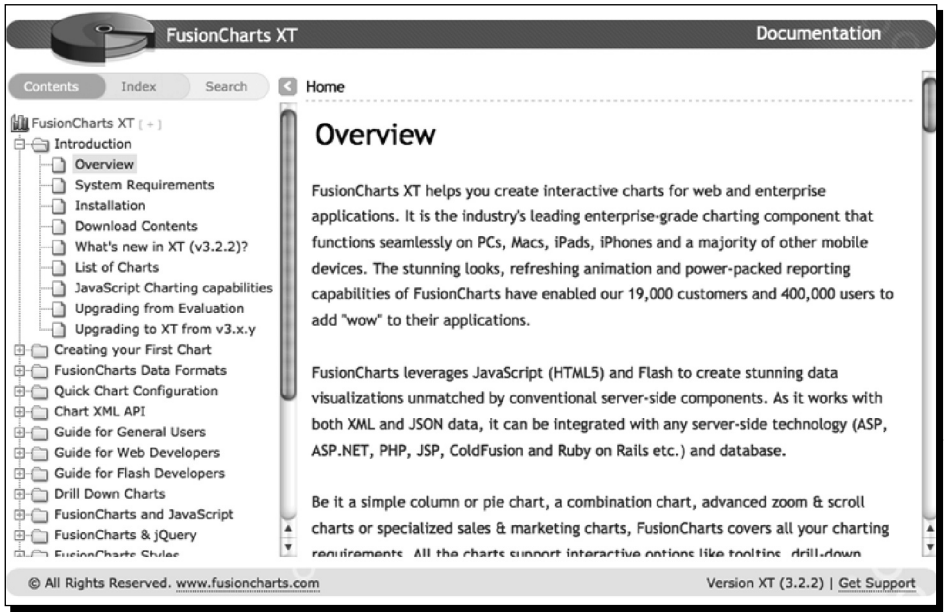
The FusionCharts Installation Folder also contains three files in the root folder:

Filename	What it contains?
`FusionCharts License Agreement.rtf`	Contains the license agreement that governs the usage of FusionCharts. You may want to read through it before using FusionCharts.
`Version.txt`	Contains the detailed version history of FusionCharts XT.
`Index.html`	The main page that you use to start exploring the FusionCharts package.

When you run `Index.html`, you will see a page similar to the following screenshot:

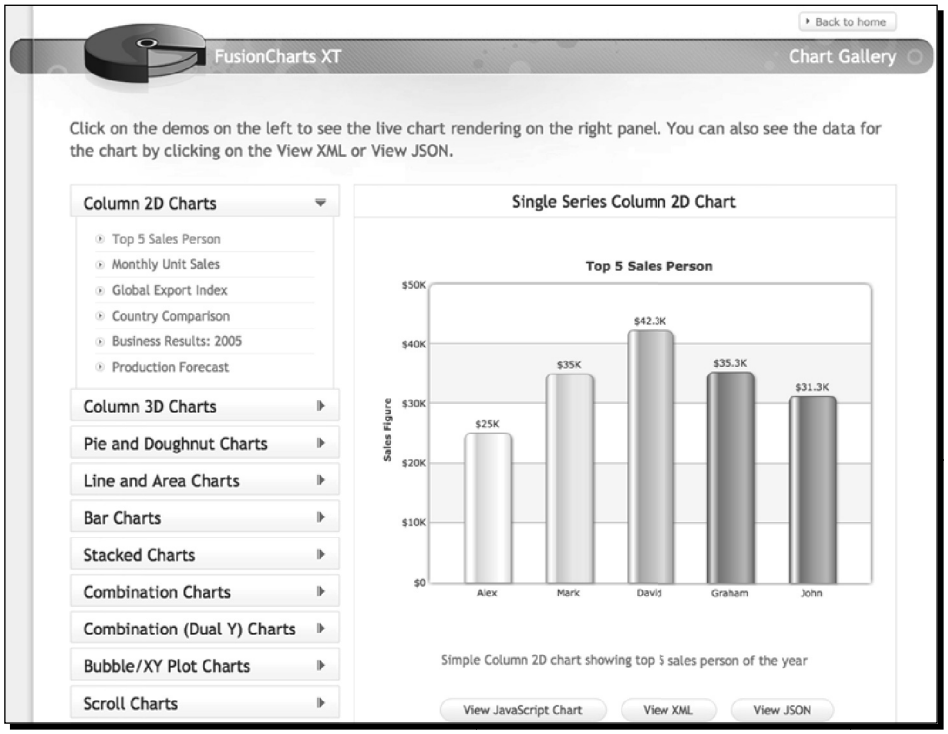


When you click on the **Documentation** link, it opens the documentation for FusionCharts XT, as shown in the following screenshot:



This documentation is an exhaustive resource for FusionCharts including sections for beginners, chart parameter lists, API references, and sections on advanced charting. In this book, we will not repeat, but rather refer to the APIs and parameters explained in the documentation.

From `Index.html`, you can also explore all the chart types present in FusionCharts XT by clicking on **Chart Gallery**. It presents a list of chart types and multiple examples for each, as shown in the following screenshot. We recommend you spend some time exploring this, as this is a good learning resource to get a real-world feel of the charts and understand what you can create once you are familiar with FusionCharts.



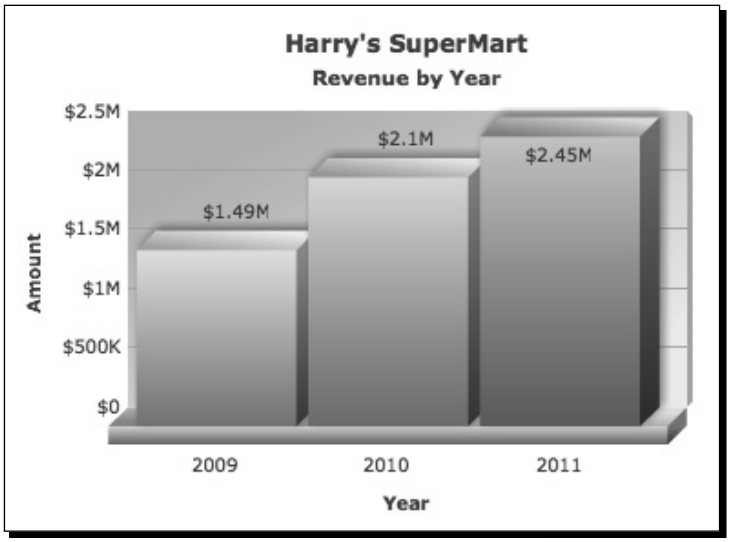
The **Live Demos** section, accessible from `Index.html`, lets you explore sample dashboards and examples created using FusionCharts—both offline and online.

Now that you have had a taste of what FusionCharts can do for you, it is time to create your own chart, your first chart using FusionCharts.

Creating your first chart

In our examples, we will create charts for a fictional supermarket, Harry's SuperMart, so that Harry, the owner of the supermarket, can make more sense out of his data. Harry's SuperMart, with 11 stores located in four states in the US, offering over 2,000 types of products and a customer base of around 25,000, records an intensive amount of data, which when presented effectively gives a lot of actionable insights. We will learn how to build meaningful charts that can facilitate this. For our first chart, let us build a simple **Revenue by Year** chart.

Once completed, the chart should look similar to the following screenshot:



Steps to create a chart using FusionCharts

Fundamentally, for each chart you build, you should ask yourself the following questions to ensure that the chart serves a meaningful purpose, as opposed to just being a fancy object on the page:

- Who will view this chart and why will this data interest him? This person is the end user.
- What type of chart is best suited to represent this data? Are there any alternate charts that we can use?
- Is this chart part of a bigger report/dashboard, or standalone? This helps us decide how to split information across multiple charts.

For our first chart that we build, Harry is the end user. This chart lets him compare the revenues of this year against the last two years. We would plot this data on a 3D Column chart, as Harry uses this to compare the revenues instead of seeing the overall trend. Had Harry wanted to see the trend of revenues over multiple years, we would have used an area or line chart. Also, to keep things simple, we will build this as a standalone chart.

Thereafter, technically, there are three steps to build a chart using FusionCharts:

1. Set up FusionCharts for the entire application, typically done only once per application.
2. Encode the data for the chart, either in XML or JSON format.
3. Write the HTML and JavaScript code to include the chart in a web page.

Let us cover them one-by-one.

Time for action – set up FusionCharts for our first chart

1. Create a folder on your hard-drive to centrally store all the examples that we will build iteratively. If you are working on a web server, you can create this under the root folder of the web application. Let us name it as `LearningFusionCharts`. You can give it any other name as well.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

2. Create a subfolder called `FusionCharts` within this folder. This folder will contain all the SWF and JavaScript files of FusionCharts, which are the **FusionCharts Core** files. If you are working on a web server, create this folder under the root of the web application, so that the entire web application can conveniently access this.
3. Copy all the SWF and JS files from the `Charts` folder under the FusionCharts Installation Folder (where you had earlier downloaded and extracted the FusionCharts ZIP file) to the newly created `FusionCharts` folder. This step completes the installation of FusionCharts for your application.
4. Create another folder under `LearningFusionCharts` and name it as `FirstChart`. This will be used to store the XML data and the HTML file for our first chart.

Upgrading the FusionCharts version, or converting from trial to licensed

If you are upgrading to a newer version of FusionCharts, or converting from evaluation to a licensed version, all you need to do is copy the SWF and JS files from the new or licensed version and overwrite the existing files in the FusionCharts folder.

What just happened?

You just installed FusionCharts. It involved copying of all the SWF and JavaScript files of FusionCharts, which are the FusionCharts Core files. If you intend to plot just a subset of chart types, you can select only those SWF files and paste them here. However, copying all files makes it easier in the future whenever you need to create a new chart type in your application. Each SWF file is used to plot a particular type of chart in Flash and the name of the file represents the chart type. You can find the complete list of charts in **FusionCharts Documentation** | **Introduction** | **List of Charts**. For our first chart, we are going to use `Column3D.swf` to plot a 3D Column chart.

The FusionCharts folder also contains six JavaScript files that aid in embedding and configuring charts, along with rendering them in JavaScript when viewed on devices that do not support Flash. These files are as follows:

Filename of the JavaScript Class	Purpose
`FusionCharts.js`	This is the main JavaScript class for FusionCharts, which helps you embed charts in your web pages in a user-friendly way, and offers functionalities such as updating chart data, retrieving chart data, supporting multiple data formats, and event handling.
`FusionCharts.HC.js`	This framework contains code to render FusionCharts in JavaScript.
`FusionCharts.HC.Charts.js`	Contains chart specific code to render FusionCharts XT in JavaScript.
`jquery.min.js`	Minified jQuery framework used by FusionCharts class for internal functions.
`FusionCharts.jqueryplugin.js`	FusionCharts jQuery class that lets you embed FusionCharts using jQuery syntax.
`FusionChartsExportComponent.js`	The charts generated by FusionCharts can be exported as images or PDFs in the browser itself, using a module called Client-side Export Component, as we will see later. This JavaScript file provides interfaces to link the Client-side Export Component to the charts.

While creating your chart, as you will soon see, you just need to include `FusionCharts.js` in your page. The other files such as `FusionCharts.HC.js`, `FusionCharts.HC.Charts.js`, and `jquery.min.js` are dynamically loaded by the code in `FusionCharts.js`.

With the basic setup in place, let us focus on the data for our chart.

Time for action – creating XML data for our first chart

1. Create an empty XML file within the `FirstChart` folder named as `Data.xml`. This can be done using your text editor (Notepad on Windows, or TextEdit on Mac). To do so, while saving an empty text file, rename the extension to `.xml`.

2. Write the following XML code in the file and save it:

```
<chart caption='Harry&apos;s SuperMart' subcaption='Revenue by
  Year' xAxisName='Year' yAxisName='Amount' numberPrefix='$'>
  <set label='2009' value='1487500' />
  <set label='2010' value='2100600' />
  <set label='2011' value='2445400' />
</chart>
```

3. Check whether the XML is valid by opening `Data.xml` in Internet Explorer or Firefox. If the browser shows the XML properly, you are good to go. Otherwise, review the error message and fix the error in XML accordingly.

What just happened?

Here, we have encoded the data, as shown in the following table, to an XML format supported by FusionCharts:

Year	Revenue
2009	$1,487,500
2010	$2,100,600
2011	$2,445,400

Each chart in FusionCharts is powered by data. This data could be static and hand-coded as we will build in this example, or dynamically generated by live scripts that are connected to databases or web services which we will explore later in *Chapter 6*, *Integrating with Server-side Scripts*. FusionCharts can accept this data in two formats—XML and JSON. Both are commonly used formats for data exchange on the Web, with XML being easy on the human eyes.

The XML format that we just created is called single-series XML in FusionCharts parlance, as we are plotting just one series of data. Later in this chapter, we will explore multi-series charts that let you compare more than one series of data, for example, revenue split across Food and Non-Food products for each year across last three years.

All FusionCharts XML files start with the `<chart>` element. The attributes of the `<chart>` element help you configure the functional and cosmetic properties of the chart. In our example, we have defined the chart caption, subcaption, axis titles, and the currency prefix for numbers on the chart, as in the following line of code:

```
<chart caption='Harry&apos;s SuperMart' subcaption='Revenue by Year'
  xAxisName='Year' yAxisName='Amount' numberPrefix='$'>
```

For each chart type, there are hundreds of optional attributes that you can define. If these are not defined, the chart assumes the default values for each of them.

Special characters in XML need to be encoded

XML documents can contain non-ASCII characters or special characters. However, these need to be encoded before they are provided in the XML document. In our example, note how we have encoded the apostrophe in Harry's to `Harry&apos;s`. Had we not done that, the XML document would have been an invalid one and raised errors when opened in a browser.

Each row of data to be plotted on the chart is represented by the `<set>` element. The `label` attribute defines the text label for each data point, and the `value` attribute defines its numerical value to be plotted. There are additional attributes that can be defined for the `<set>` element, for example, user-defined colors, which we will explore in later chapters. An important thing to note is how the $ prefix or comma separators have been stripped off the revenue numbers, before encoding them as a value for the `<set>` element, that is, $1,487,500 has been converted to 1487500, as shown in the following line of code:

```
<set … value='1487500' />
```

This is necessary as FusionCharts can interpret only standard numeric values.

While the standard attributes for the `<chart>` and `<set>` elements are common across chart types, many chart types have special features that are controlled by attributes that are specific for the chart. You can explore a list of all such attributes for each chart in the documentation of FusionCharts, under the section **Chart XML API**.

With both the basic setup and data in place, we are just one step away from seeing our chart live—writing the HTML and JavaScript to embed this chart, which we will do next.

Time for action – Writing the HTML and JavaScript code to embed the chart

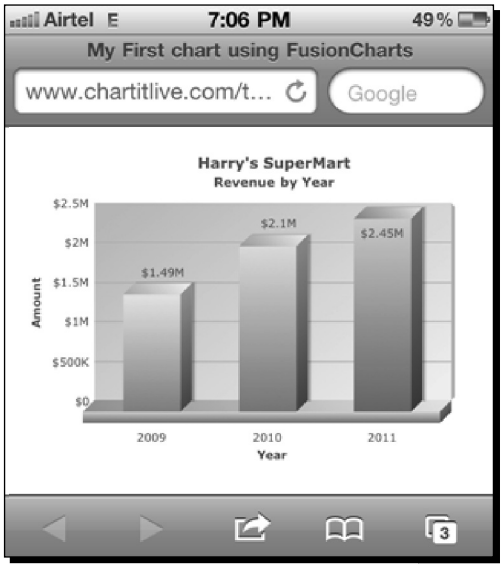
1. Create an empty HTML file within the `FirstChart` folder named as `FirstChart.html`.

2. Paste the following code in the file and save it:

```
<html>
  <head>
    <title>My First chart using FusionCharts</title>
    <script type="text/javascript"
      src="../FusionCharts/FusionCharts.js">
    </script>
  </head>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
    <script type="text/javascript">
    <!-- var myChart =
      new FusionCharts("../FusionCharts/Column3D.swf",
      "myChartId", "400", "300", "0", "1" );
      myChart.setXMLUrl("Data.xml");
      myChart.render("chartContainer");//-->
    </script>
  </body>
</html>
```

3. Open it in a web browser. You should see your first chart coming to life, as shown in the following screenshot. Refresh the browser to experience the animation again, or hover over the columns to see tooltips.

4. If you have access to an iPad or iPhone, open this example using the device. To do so, upload the entire `LearningFusionCharts` to a server that can be accessed over the Internet. Now point the browser in the device to `http://Your_Website_URL/FirstChart/FirstChart.html`. You will be able to see the same chart, but this time, rendered using JavaScript. The following screenshot shows a rendering of the chart within Safari in an iPhone. Tap on the columns to see the tool-tips.



What just happened?

You just created your first chart, that's what happened! This chart renders using Adobe Flash on devices that support it, and automatically switches to JavaScript rendering on devices such as iPads and iPhones. The beauty of the solution is that no additional code or configuration is required to do this.

Let us break down our HTML and JavaScript code into digestible chunks. To create charts using FusionCharts in your page, you first need to include the FusionCharts JavaScript library (`FusionCharts.js`), as in the following lines of code:

```
<script type="text/javascript" src="../FusionCharts/FusionCharts.js">
  </script>
```

Note that you only need to include `FusionCharts.js` in your code. The other files required for FusionCharts, namely `FusionCharts.HC.js`, `FusionCharts.HC.Charts.js`, and `jquery.min.js` are dynamically loaded by code in `FusionCharts.js`.

Next, we create a DIV as a placeholder where the chart would be rendered. We give the DIV an ID—`chartContainer`. This is done using the following code:

```
<div id="chartContainer">FusionCharts will load here!</div>
```

The DIV carries a placeholder text `FusionCharts will load here!` which will be displayed if there is an error in your JavaScript code, or `FusionCharts.js` or the chart SWF file could not be loaded. If you see this text instead of the chart, you know what to fix.

Following this, we initialize a chart by invoking the FusionCharts JavaScript constructor, using the following code:

```
var myChart = new FusionCharts("../FusionCharts/Column3D.swf",
  "myChartId", "400", "300", "0", "1" );
```

To this constructor, we pass the following parameters in order:

1. **Path and filename of the chart SWF**: The first parameter contains the path and filename of the chart SWF file. We have used the relative path to the SWF file, which is recommended.
2. **ID of the chart**: Each chart on the page needs a unique ID. This ID is different from the ID of the container DIV. As we will learn later, this ID is used to get a reference of the chart for manipulation using advanced JavaScript.
3. **Width and height in pixels**: Each chart needs to be initialized with width and height, specified either in pixels (specified in numeric format, without appending `px`) or percentage. In this example, we have used pixels. You can also set it to `%` values as in the following code:

   ```
   var myChart = new FusionCharts("../FusionCharts/Column3D.swf",
     "myChartId", "100%", "100%", "0", "1" );
   ```

   The FusionCharts JavaScript class will automatically convert the % dimensions to pixel dimensions, with respect to the parent container element in HTML, DIV in this case, and pass it to the chart.

4. **Whether to start the chart in Debug mode**: While developing your charts, if you face any issues, you can initialize them in debug mode by setting this parameter to `1`. The Debug mode gives you behind-the-scenes information on where the data is loaded from, errors, and so on. In our example, we are rendering the chart in normal mode, by setting this parameter to `0`.
5. In previous versions of FusionCharts, you had to manually set the last parameter to `1`, if you wanted FusionCharts to communicate with JavaScript. Now that FusionCharts is very well integrated with JavaScript, this parameter is a mandatory `1`.

Alternate compact constructor method

A chart can also be initialized using the `static render()` method of the FusionCharts class, as shown below.

```
<script type="text/javascript">
  <!--var myChart = FusionCharts.render
  ("../FusionCharts/Column3D.swf", "myChartId",
    "400", "300",
  "chartContainer", "Data.xml"); // -->
</script>
```

There are additional possible syntaxes of this constructor and are detailed in **FusionCharts Documentation** | **FusionCharts and JavaScript** | **Constructor methods**.

Once the chart is constructed, we tell the chart where to source data from. We use a relative path to `Data.xml`, as it is stored in the same folder.

```
myChart.setXMLUrl("Data.xml");
```

If you recall, FusionCharts accepts data in two formats – XML and JSON – either provided as a string or a URL that points to the data file. In our example, we have used XML as the data format, which is stored in `Data.xml`. So, we use the `setXMLURL()` function to pass the URL of the XML data file to the chart.

What if the XML data file was stored in another location or subdomain?

If your data file was stored in a different folder, you would have to specify the relative path to the folder and then the filename, for example, `../Source/Data/MyData.xml`. We do not recommend specifying absolute URLs, because, if you move your web page or data file to another domain, cross-domain security issues would crop up and the chart would stop working.

Flash Player's sandbox security model blocks loading of files across different sub-domains. If you need to load your XML data from another subdomain, you will have to create a Cross domain policy XML file, as explained at `http://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html`.

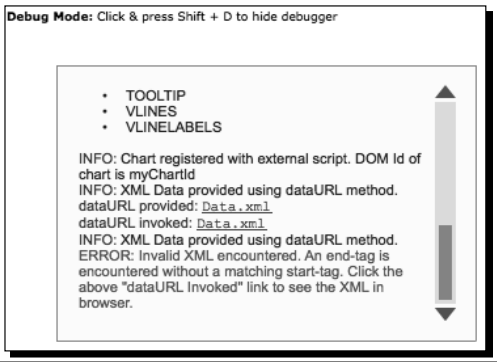
Finally, to render the chart in the `DIV` that you had earlier created, you invoke the `render()` function and pass to it the ID of the `DIV`.

```
myChart.render("chartContainer");
```

Do remember that each chart and DIV needs to have its own unique ID.

What to look for if your chart is not rendered?

If you do not see any chart, there could be multiple reasons behind it. You should check for the following, based on what you see in your browser:

What do you currently see instead of the chart?	Corrective measures you should take
"FusionCharts will load here!" text that you had placed in the container DIV	Check whether the `FusionCharts` folder is present in the `LearningFusionCharts` folder and contains all JavaScript files required for FusionCharts. Check whether you have provided the correct relative path to `FusionCharts.js` in the page `FirstChart.html`. Check for errors in your JavaScript code that you have written to embed the chart. Use the browser's developer tools to check this. Ensure that you have given different IDs for container `DIV`, chart JavaScript variable and the chart object in the constructor.
Empty white area instead of the chart	Check whether you have copied `Column3D.swf` to the `FusionCharts` folder. Check whether the relative path provided to `Column3D.swf` in FusionCharts constructor is correct.
"Error in loading data"	Check whether `Data.xml` is present within the `FirstChart` folder Check whether the path specified to `Data.xml` is correct in the `setXMLUrl()` method.
"Invalid data"	Check for the validity of XML data in `Data.xml` by opening it in a browser or an XML editor. Or, you can also switch the debug mode of chart to ON by changing the last but one parameter in constructor to `1`. That will highlight the error in XML, as shown in the following screenshot: 

With these measures, you should be able to locate the error and get your chart working. Before we move ahead to explore the other aspects of FusionCharts, let us understand how FusionCharts automatically switches between Flash and JavaScript mode.

Converting the chart to a pure JavaScript chart

By default, FusionCharts renders its charts using Adobe Flash. However, as you have seen earlier, when you view the chart on iPads or iPhones, FusionCharts automatically switches to JavaScript rendering, as Flash is not supported on those devices. This is internally checked by `FusionCharts.js`, and the auto-loaded files `FusionCharts.HC.js`, `FusionCharts.HC.Charts.js`, and `jquery.min.js` then aid in rendering the chart using JavaScript, using the same datasource and configuration.

FusionCharts also provides an option to entirely skip Flash rendering and use JavaScript as the default rendering, irrespective of the device. This feature can be very nifty for developers who want to develop JavaScript-only applications or even frameworks. Let us quickly see how to attain that.

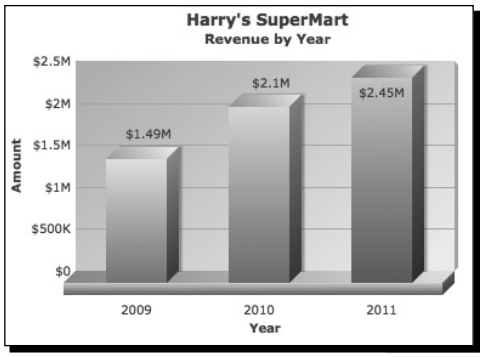
Time for action – creating JavaScript only charts

1. Create a copy of our `FirstChart.html` in the same location and name it as `JavaScriptChart.html`.

2. Add the following lines of code, as highlighted, before the constructor.

```
<html>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
    <script type="text/javascript">
      <!--FusionCharts.setCurrentRenderer('javascript');
        var myChart =
        new FusionCharts("../FusionCharts/Column3D.swf",
        "myChartId", "400", "300", "0", "1" );
      myChart.setXMLUrl("Data.xml");
      myChart.render("chartContainer");// -->
    </script>
  </body>
</html>
```

3. Open the page in a browser. You should see the same chart as earlier, but this time rendered using JavaScript. It has animations and interactivity similar to the Flash version as shown in the following screenshot:



How different is the Flash rendering from JavaScript rendering?

The JavaScript version of FusionCharts behaves similar to the Flash version, offering most of the functional and cosmetic configuration. With the exception of a true 3D chart, all charts look and behave almost the same as their Flash counterpart. There are minor visual differences in the JavaScript version such as the width of columns, effect of shadows, handling of long x-axis labels, appearance of scroll bars, and so on. A detailed list of such differences are present in **FusionCharts Documentation** | **Introduction** | **JavaScript Charting Capabilities** | **How different is JavaScript charts from Flash charts?**

What just happened?

You just converted the previous chart to a pure JavaScript chart, irrespective of the device it is now viewed on. The following snippet of code instructs FusionCharts to switch the rendering mode to JavaScript:

```
FusionCharts.setCurrentRenderer('javascript');
```

If your page contains multiple charts, this setting applies to all such charts that are defined after this line of code. Hence, if you declare this at the beginning, all the charts in the page will render using JavaScript. You would not need to declare the same for each chart in the page.

JavaScript cannot access data stored on your hard drive in some browsers

Some browsers restrict JavaScript from accessing the local filesystem due to security reasons. Hence, the JavaScript charts, when running from your local hard drive, would not be able to access XML or JSON data provided as a URL. However, when run from a server, including localhost, they will run fine. An alternate method to get JavaScript charts working locally is to use the Data String method, which we will explore in the next section.

Have a go hero – build a dashboard for Harry

In this example, you created a standalone Column 3D chart. How about inching towards building a complete dashboard? To do so, convert the existing chart to a Column 2D chart and add the following three charts to this page. In addition, specify different width and height for each chart to accommodate the amount of data it presents, and also place them in order of importance of the chart to Harry.

- A Line 2D chart, using `Line.swf`, comparing monthly revenues for this year. For this, you need to create an XML data with the `<set>` element for each month of the year.
- A Pie 2D chart, using `Pie2D.swf`, showing the composition of expenses of this year split under these categories: Salary, Cost of Goods, Marketing Costs, Overheads, and Administration.
- A Column 2D chart, using `Column2D.swf`, showing the top five salespersons for the year.

All these charts use the same single-series XML format that you had earlier created. Remember to provide a different ID for each chart and its container `DIV`. Also, do not forget to encode special characters such as `&` (ampersand) or `'` (apostrophe) in XML.

Once you are done, let us explore the other way to provide XML data to FusionCharts—as a string, instead of providing a URL, for example, `Data.xml`.

Using the Data String method to provide data

As we had mentioned earlier, there are two ways to provide data to FusionCharts – either as a URL to the datasource (**Data URL method**), or as a string (**Data String method**). Till now, we have used the former method by invoking the `setXMLUrl()` method on the chart instance and providing `Data.xml` as the URL. In order to pass the XML as a string to the chart, we can use the `setXMLData()` method, as explained next.

Time for action – embedding XML in the web page and using the Data String method

1. Create a copy of our `FirstChart.html` in the same location and name it as `DataStringMethod.html`.
2. Change the following lines in code, as highlighted:

```
<html>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
```

```
    <script type="text/javascript">
      <!-- var myChart =
        new FusionCharts("../FusionCharts/Column3D.swf",
        "myChartId", "400","300", "0", "1" );
        myChart.setXMLData("<chart
          caption='Harry&apos;s SuperMart'
          subcaption='Revenue by Year' xAxisName='Year'
          yAxisName='Amount' numberPrefix='$'>\
        <set label='2009' value='1487500' />\
        <set label='2010' value='2100600' />\
        <set label='2011' value='2445400' />\
      </chart>");
      myChart.render("chartContainer");// -->
    </script>
  </body>
</html>
```

3. Open the page in a browser. You should see the same chart as earlier, but this time using data embedded in the page, and not `Data.xml`.

What just happened?

You just used the Data String method of FusionCharts to power up your chart using XML data embedded in the page, instead of reading it from `Data.xml`. This was done by invoking the `setXMLData()` method on the chart instance.

```
myChart.setXMLData("<chart caption='Harry&apos;s SuperMart'
  subcaption='Revenue by Year' xAxisName='Year' yAxisName='Amount'
  numberPrefix='$'>\
  <set label='2009' value='1487500' />\
  <set label='2010' value='2100600' />\
  <set label='2011' value='2445400' />\
  </chart>");
```

The entire XML string is passed to this method. Note how we are using the \ characters in JavaScript to split the XML data string into multiple lines for enhanced readability. Make sure there are no trailing spaces, when using this approach.

You can also define a JavaScript string variable, store XML data in it, and then assign the variable reference to the chart instance, as shown in the following code snippet:

```
<html>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
    <script type="text/javascript"><!-- var strData =
      "<chart caption='Harry&apos;s SuperMart' subcaption='Revenue by
      Year' xAxisName='Year' yAxisName='Amount' numberPrefix='$'>" +
```

```
        "<set label='2009' value='1487500' />" +
        "<set label='2010' value='2100600' />" +
        "<set label='2011' value='2445400' />" + "</chart>";
        var myChart = new FusionCharts("../FusionCharts/Column3D.swf",
        "myChartId", "400", "300", "0", "1" );
        myChart.setXMLData(strData);
        myChart.render("chartContainer");// -->
    </script>
  </body>
</html>
```

In the previous example, we had stored the entire XML string in the variable `strData`, and then passed its reference to the `setXMLData()` method, instead of the XML string directly.

When using this method to provide data, if your chart is not working or reporting `Invalid data`, check for the following:

- Make sure that the quotation marks specified in JavaScript to provide parameters and in XML to provide attributes are different. Otherwise, it will result in a JavaScript syntax error. To keep things easy to remember, use double quotation marks for JavaScript, and single quotation marks for XML attributes.
- Ensure that special characters such as `'`, `"`, `&`, `<`, and `>` present as XML attribute values are encoded to `&apos;`, `&quot;`, `&`, `<`, and `>` respectively.

Now that you are familiar with both the ways of providing XML data to FusionCharts, let us explore the other data format supported by FusionCharts—JSON.

Using JSON data with FusionCharts

JSON is a lightweight and simple data format derived from JavaScript. The data structure is language-independent, with encoders and parsers available for virtually every programming language. FusionCharts allows you to provide JSON data to the chart either as a URL using the `setJSONUrl()` method or as a string using the `setJSONData()` method. Before we use these methods, let us convert our previous data from XML format to JSON format to understand this format. To do this, we will use the **FusionCharts Data Format Conversion Tool** that comes in the FusionCharts download package. Perform the following steps:

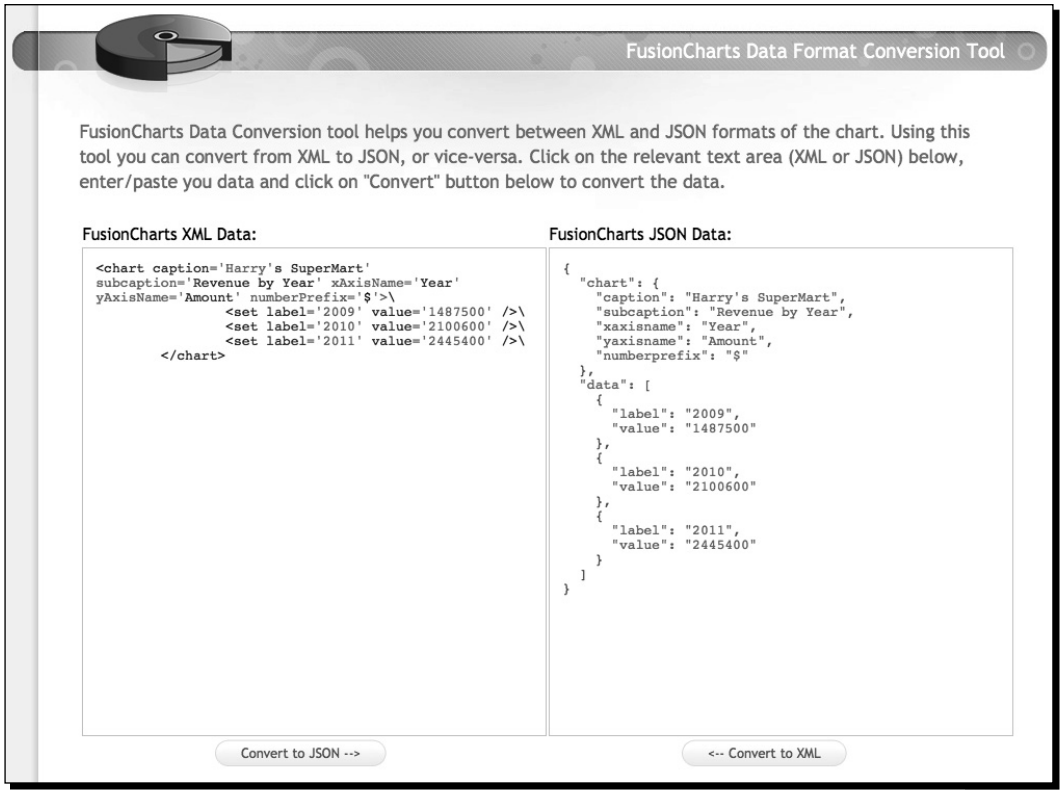
Time for action – converting FusionCharts XML format to JSON

1. Launch the `FCDataConverter` tool from the FusionCharts Installation Folder | `Tools` | `FCDataConverter` | `Index.html`.

2. Once the page has finished loading, in the text area on the left, titled **FusionCharts XML Data**, paste the XML data that we had previously created for **Revenue by Year** chart.

3. Click on the **Convert to JSON** button present below it.

4. In the text area on the right, you will now see the JSON equivalent of the XML data, as shown in the following screenshot:



What just happened?

Using the FusionCharts Data Format Conversion Tool, you just converted the previous XML data into JSON format. It reads as in the following code snippet:

```
{
  "chart": {
    "caption": "Harry's SuperMart",
    "subcaption": "Revenue by Year",
    "xaxisname": "Year",
    "yaxisname": "Amount",
```

```
        "numberprefix": "$"
      },
      "data": [
        {
          "label": "2009",
          "value": "1487500"
        },
        {
          "label": "2010",
          "value": "2100600"
        },
        {
          "label": "2011",
          "value": "2445400"
        }
      ]
    }
```

Similar to XML, the `chart` object contains attributes that let you configure functional and cosmetic aspects of the chart.

In the most general form, chart attributes represent the following JSON format:

```
"attributeName" : "Value"
```

For example, `"xAxisName" : "Year"`

The attributes can occur in any order and values can be specified either using double quotes or single, for example, `xAxisName:'Year'`. However, you need to ensure that the same attribute is not defined twice for any element, as it results in an invalid XML.

Escaping of special characters is not compulsory in the JSON URL format

When using the JSON data format, special characters are not encoded to XML entities. Instead, they are escaped in JavaScript using `\`. However, this is not mandatory when using JSON URL as data, as JavaScript loads the JSON data and directly parses attributes as string literals. Hence, in our example, Harry's SuperMart does not need to be encoded as we had done in the XML format.

However, if you have a mismatch of JavaScript enclosing quotes and JSON attribute quotes, as we will see in our next example, escaping is required.

Next, the array `data` contains all the data points to be plotted on the chart. For example, in XML, `label` attribute for each data point defines its text label, and the `value` attribute represents its numerical value. Each element in the `data` array is an unnamed object defined in the following format:

```
{ "label: "Jan", "value" : "17400", "otherAttribute" : "value"}
```

With the JSON format understood, let us also look at how to use the `setJSONUrl()` and `setJSONData()` methods.

Time for action – powering a chart using JSON data stored in a file

1. Create a file `Data.json` in the `FirstChart` folder.
2. Paste the previously converted JSON in this file and save it.
3. Create a copy of `FirstChart.html` in the same folder and name it as `JSONDataURL.html`.
4. Change the following lines of code, as highlighted:

```
<html>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
    <script type="text/javascript"><!-- var myChart =
      new FusionCharts("../FusionCharts/Column3D.swf",
      "myChartId","400", "300", "0", "1" );
      myChart.setJSONUrl("Data.json");
      myChart.render("chartContainer");// -->
    </script>
  </body>
</html>
```

5. View the page in the browser. You should see the same chart as the previous one.

What just happened?

You just configured your chart to use JSON data as URL, instead of XML. If you do not see a chart, however, your browser might be restricting JavaScript to load local files. In that case, you will have to switch to the JSON Data String method, as explained in the next section.

Time for action – powering a chart using JSON data embedded in the page

1. Create a copy of `DataStringMethod.html` in the `FirstChart` folder and name it as `DataStringMethodJSON.html`.

2. Change the following lines of code, as highlighted:

```
<html>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
    <script type="text/javascript">
      <!--var myChart =
      new FusionCharts("../FusionCharts/Column3D.swf",
      "myChartId","400", "300", "0", "1" );
      myChart.setJSONData('{\"chart": {\
      "caption": "Harry\'s SuperMart",\
      "subcaption": "Revenue by Year",\
      "xaxisname": "Year",\
      "yaxisname": "Amount",\
      "numberprefix": "$"\
      },\"data": [{\
      "label": "2009",\
      "value": "1487500"\
      },{\
      "label": "2010",\
      "value": "2100600"\
      },{\
      "label": "2011",\
      "value": "2445400"\
      }]}');
      myChart.render("chartContainer");// -->
    </script>
  </body>
</html>
```

3. View the page in the browser. You should see the same chart as the previous one.

What just happened?

You changed the `setXMLData()` function to the `setJSONData()` function and provided JSON data instead of XML data. Also, note how the apostrophe in `Harry's SuperMart` was escaped in JavaScript so as to form `Harry\'s SuperMart`. Otherwise, there would have been a conflict of quotes leading to invalid JavaScript syntax.

You can also provide the JSON data to the `setJSONData()` method as an object, instead of a string, as shown in the following code:

```
<html>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
    <script type="text/javascript">
    <!--  var myChart =
      new FusionCharts("../FusionCharts/Column3D.swf",
      "myChartId", "400", "300", "0", "1" );
      myChart.setJSONData({
        "chart": {
          "caption": "Harry\'s SuperMart",
          "subcaption": "Revenue by Year",
          "xaxisname": "Year",
          "yaxisname": "Amount",
          "numberprefix": "$"
        },
       "data": [{
       "label": "2009",
       "value": "1487500"
      },{
          "label": "2010",
          "value": "2100600"
      },{
        "label": "2011",
        "value": "2445400"
      }]});
      myChart.render("chartContainer");// -->
    </script>
  </body>
</html>
```

Here, we have converted the JSON string to a JavaScript object by removing the enclosing string quotation marks and even the \ character that was used for concatenating the string distributed across multiple lines. And that does it all!

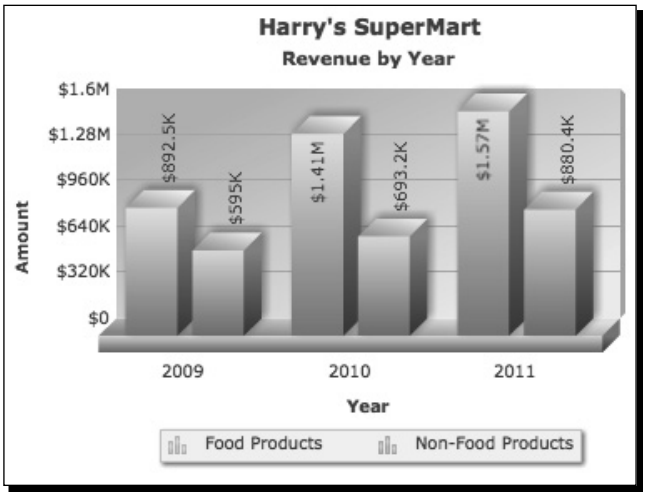
Bingo! You are now adept with the basics of FusionCharts. You have learned how to create a FusionCharts, provide XML or JSON data as either URL or string, and even render the chart using pure JavaScript. Now, we are all set to explore additional charts in FusionCharts. First, we will create a chart with more than one series of data, called a multi-series chart in FusionCharts parlance.

Creating charts with multiple series

In our previous example, we had built a Column 3D chart with three columns, each column representing the revenue for a specific year. Now, Harry needs to see how the revenue is split across food products and non-food products, each year. He needs this to monitor growth of both the segments over the years. The data for this example is provided in the following table:

Year	Sales of Food Products	Sales of Non-Food Products
2009	892500	595000
2010	1407400	693200
2011	1565000	880400

The sum of food products and non-food products adds up to the total revenue per year, which we had earlier plotted. The set of data points representing one of these segments, says food-products, is a **data series**, or a **dataset** in FusionCharts XML terminology. We have two data series in our next chart that would be rendered side-by-side, as in the following screenshot:



Time for action – creating a multi-series chart

1. Create a copy of `FirstChart.html` in the `FirstChart` folder and name it as `MultiSeriesChart.html`.

2. Change the following lines of code, as highlighted:

```
<html>
  <body>
    <div id="chartContainer">FusionCharts will load here!</div>
    <script type="text/javascript">
     <!-- var myChart =
       new FusionCharts("../FusionCharts/MSColumn3D.swf",
       "myChartId", "400", "300", "0", "1" );
       myChart.setXMLUrl("MSData.xml");
       myChart.render("chartContainer");// -->
    </script>
  </body>
</html>
```

3. Create a copy of `Data.xml` in the same folder and rename the copy to `MSData.xml`. Write the following XML in this file and save it.

```
<chart caption='Harry&apos;s SuperMart' subcaption='Revenue by
Year'
  xAxisName='Year' yAxisName='Amount' numberPrefix='$'
  rotateValues='1'>
  <categories>
    <category label='2009' />
    <category label='2010' />
    <category label='2011' />
  </categories>
  <dataset seriesName='Food Products'>
    <set value='892500' />
    <set value='1407400' />
    <set value='1565000' />
  </dataset>
  <dataset seriesName='Non-Food Products'>
    <set value='595000' />
    <set value='693200' />
    <set value='880400' />
  </dataset>
</chart>
```

4. Open `MultiSeriesChart.html` in your browser. You should now see a chart with two series of columns, as we had planned to build.

What just happened?

In `MultiSeriesChart.html`, we have changed the SWF file from `Column3D.swf` to `MSColumn3D.swf` to render a multi-series chart. FusionCharts uses separate SWF files to plot charts with single series of data (**single-series**) and those with more than one (**multi-series**). Names of SWFs that plot multiple series start with a prefix of `MS`. We have also pointed XML URL to the newly created `MSData.xml`, which contains data in multi-series format for this chart, in the `setXMLUrl()` method. These are the only changes required in the web page.

The bulk of changes are in the XML data file to adapt it to the multi-series format. Let us review them. The multi-series chart XML begins with the `<chart>` element, similar to single-series and all the other charts in FusionCharts. You can provide attributes for the `<chart>` element to configure the functional and cosmetic properties of the chart. In this example, building on top of our earlier single-series chart, we have introduced a new attribute `rotateValues='1'`, that rotates the data values on the chart to accommodate more numbers. The other attributes remain the same.

Next, we have the `<categories>` element that is applicable when you are plotting multi-series charts only. Each child `<category>` element of the `<categories>` element represents an x-axis label (also called the data point's label). The label attribute of the `<category>` element lets you specify them as string values. In our chart, we are comparing the segment-wise sales for 2009, 2010, and 2011, and hence have them as x-axis labels.

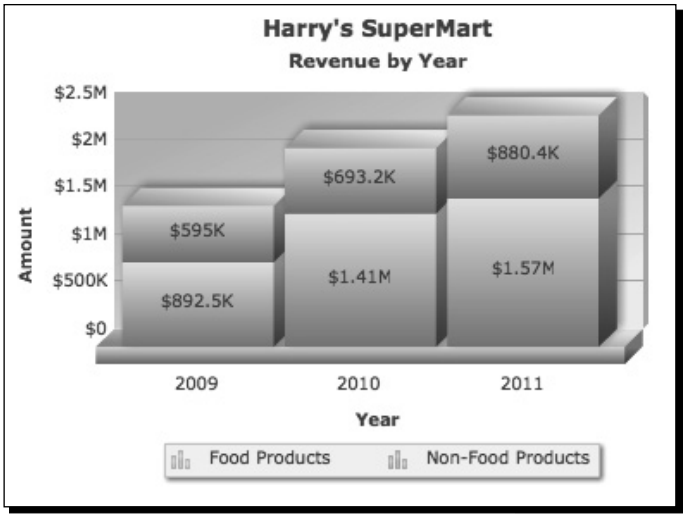
```
<categories>
  <category label='2009' />
  <category label='2010' />
  <category label='2011' />
</categories>
```

All that is now left to be defined is data for both the data series. Each series is represented by a `<dataset>` element and attributes of this element lets you specify a custom color for the series, and whether to show or hide data values. To define individual data points within a series, children `<set>` elements are added with a value attribute containing a data point value.

```
<dataset seriesName='Food Products'>
  <set value='892500' />
  <set value='1407400' />
  <set value='1565000' />
</dataset>
```

The data point labels or x-axis labels in `<category>` elements are matched to their respective data point values in the `<set>` element, based on the order of the definition in XML, that is, the first `<category>` element provides the label for the first `<set>` element and so on. To plot a normal chart, the number of `<set>` elements within each `<dataset>` should be equal to the number of defined `<category>` elements.

The previous XML can also be used to plot a stacked chart where columns are placed on top of each other as opposed to side-by-side. The stacked charts are used when the constituents of a data series are relevant along with the sum of all such constituents. While these charts are suited for the comparison of sum, comparison of the constituents against each other is best portrayed by the multi-series chart. To build our stacked chart, we need to change the chart SWF to `StackedColumn3D.swf`. In the XML, we only make a small change to not rotate the data values by removing the `rotateValues='1'` attribute from the `<chart>` element. This results in a chart as shown in the following screenshot:



Can multi-series charts be used to plot only one series of data?

Yes, with a few changes. In a single series chart, all the columns are colored differently. In contrast, in a multi-series chart, all the columns of a data series are of the same color, unless explicitly configured not to. Secondly, the multi-series charts show a visual legend indicating the color of each data series. The key in this legend is clickable and lets the end users show or hide the data series.

When plotting a single series of data using a multi-series chart, you need to provide only one `<dataset>` element to contain these data points.

Can the number of <set> and <category> elements mismatch?

No. If you have more `<category>` elements than `<set>` elements in any data series, FusionCharts fails to find data for those additional `<category>` elements and plots an empty space on the chart for the missing data points. In contrast, if the number of `<set>` elements are more than the number of defined `<category>` elements, FusionCharts ignores them, as these data points do not have respective x-axis labels defined for them, and hence would be without context.

In real-life scenarios, you may have missing or non-existent data within a data series. FusionCharts provides a method for you to specify that. Let us consider an example.

Representing missing or non-existent data on the chart

Let us consider a hypothetical example of building this on top of our last example. What if Harry started selling food products only in 2010, when he added refrigeration capabilities? However, non-food products were still sold in 2009. Hence, we would need to tell FusionCharts that data for food products in 2009 is missing or nonexistent. This can be done by specifying an empty `<set/>` element as in the following XML:

```
<chart caption='Harry&apos;s SuperMart' subcaption='Revenue by Year'
  xAxisName='Year' yAxisName='Amount' numberPrefix='$'
  rotateValues='1'>
  <categories>
    <category label='2009' />
    <category label='2010' />
    <category label='2011' />
  </categories>
  <dataset seriesName='Food Products'>
    <set />
    <set value='1407400' />
    <set value='1565000' />
  </dataset>
  <dataset seriesName='Non-Food Products'>
    <set value='595000' />
    <set value='693200' />
    <set value='880400' />
  </dataset>
</chart>
```

This instructs the chart to render an empty space instead of the column representing food-products in 2009, to indicate missing or non-existent data. This method of providing missing data is applicable to all charts in FusionCharts.

Can missing data be replaced by providing zero values?

You might wonder that if there was no data for 2009, why could we not specify the value as `0`. It would mean that Harry's SuperMart was not able to sell any food products in 2009, as opposed to the fact that it did not stock food products then. Both have different meanings and need to be visualized accordingly.

Now that you have an understanding of how to plot multiple series on a chart for comparison, let us consider an extension of multi-series charts—combination charts.

Combination charts

Combination charts let you combine two or more chart types in a single chart, for example, a column chart series and a line chart series. This is done either to highlight specific data series by rendering, or to mix different types of data series on the same chart. There are two types of combination charts possible in FusionCharts.

The first type, called **Single Y-axis combination charts**, have a single y-axis and all the data series conform to it. Some charts in this category are `MSCombi2D.swf`, `MSCombi3D.swf`, `MSColumnLine3D.swf`, and so on. As an example, if you wish to plot the actual revenue versus the projected revenue on a chart, as both the series have the same unit (currency) and magnitude, we plot them against the same axis. However, as the focus is on the actual revenue, it should be plotted using column or area, and the projected revenue can be plotted using lines to show it as guidance.

The second type, called **Dual Y-axes combination charts**, have two y-axes, each having its own units and magnitude. These charts have the abbreviation `DY` in their name, for example, `MSCombiDY2D.swf` or `MSColumn3DLineDY.swf`. Consider a chart where you are plotting the revenue of a company versus the units (quantity) sold. As both the series represent different units, they need to be plotted on different axes. The primary axis can represent the sales, and the second used for units sold. Conventionally, you would represent the sales using columns plotted against the primary axis on the left side of the chart, and the units sold using lines plotting against the secondary axis on the right side of the chart.

Let us build an example of both these charts.

Time for action – the chart showing the actual versus the projected revenue

1. Create a copy of `MultiSeriesChart.html` in the same folder and name it as `ActualVsProjected.html`.
2. Change the reference of the chart SWF file in the embedding code to `MSCombi2D.swf`. This SWF renders a 2D combination chart with a single y-axis.
3. Change the XML URL to `ActualVsProjected.xml` in the chart embedding code.
4. Create an XML file with the name `ActualVsProjected.xml` in the same folder and write the following data in it:

```
<chart caption='Harry&apos;s SuperMart' subcaption='Revenue by
Year'
  xAxisName='Year' yAxisName='Amount' numberPrefix='$'>
  <categories>
    <category label='2009' />
    <category label='2010' />
    <category label='2011' />
  </categories>
  <dataset seriesName='Actual Revenue'>
    <set value='1487500' />
    <set value='2100600' />
    <set value='2445400' />
  </dataset>
  <dataset seriesName='Projected Revenue' renderAs='Line'
    dashed='1' showValues='0' color='666666'>
    <set value='1216500' />
    <set value='2043400' />
    <set value='2292400' />
  </dataset>
</chart>
```

5. Open `ActualVsProjected.html` in your browser. You should see a chart similar to the following screenshot:

What just happened?

We just created a combination chart to compare the actual revenue of Harry's SuperMart to the projected revenues, which were set as a target. For a change, we have plotted it as a 2D chart using `MSCombi2D.swf`. The XML data is similar in structure to the multi-series chart. The three years that form the x-axis labels are provided as `<category>` elements. There are two data series in the chart, one representing the actual revenue, and the other, projected revenue. There are a few new attributes that we have added to the `<dataset>` element of the data series representing the projected revenue, as the following line of code:

```
<dataset seriesName='Projected Revenue' renderAs='Line' dashed='1'
  showValues='0' color='666666'>
```

The first attribute `renderAs='Line'` instructs FusionCharts to render this series as a line series. The other possible values are `Column` and `Area` for this chart. As this line in the chart reflects guidance of revenues, we have visually indicated this by plotting it as a dashed line using the attribute `dashed='1'`. Thereafter, we have turned off the display of the data values for this series using `showValues='0'` to avoid cluttering of too many data values. Finally, we have provided a custom color for this series using `color='666666'`.

Simple, isn't it? Let us now build the second type of combination chart, to compare the revenues versus units of products sold.

Time for action – a chart showing revenue versus units sold

1. Create a copy of `ActualVsProjected.html` in the same folder and name it `RevenueVsUnits.html`.

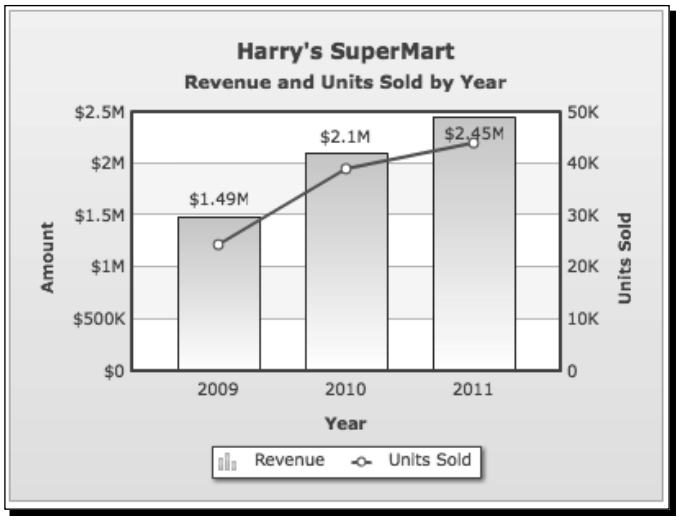
2. Change the reference of the chart SWF file in embedding code from `MSCombi2D.swf` to `MSCombiDY2D.swf`, to use a chart with dual axes.

3. Change the XML URL to `RevenueVsUnits.xml` in the chart embedding code.

4. Create an XML file with the name `RevenueVsUnits.xml` in the same folder and write the following data in it:

```
<chart caption='Harry&apos;s SuperMart' subcaption='Revenue and
  Units
  Sold by Year' xAxisName='Year' PYAxisName='Amount'
  SYAxisName='Units Sold' numberPrefix='$'>
  <categories>
    <category label='2009' />
    <category label='2010' />
    <category label='2011' />
  </categories>
  <dataset seriesName='Revenue'>
    <set value='1487500' />
    <set value='2100600' />
    <set value='2445400' />
  </dataset>
  <dataset seriesName='Units Sold' parentYAxis='S' renderAs='Line'
    showValues='0' color='666666'>
    <set value='24355' />
    <set value='38998' />
    <set value='43987' />
  </dataset>
</chart>
```

5. Open `RevenueVsUnits.xml` in your browser. You should see a chart similar to the following screenshot:



What just happened?

We just enabled Harry to compare the revenues of Harry's SuperMart for the last three years, along with the units sold. This chart gives him a perspective on how the revenues are affected by the units sold. Note how this chart has two y-axes, one on the left called primary axis, and one on the right called secondary axis. Each axis has its title defined using two new attributes of the `<chart>` element, `PYAxisName` for the primary axis, and `SYAxisName` for the secondary axis. The attribute `YAxisName` is not applicable to this chart as there is no common y-axis.

```
<chart caption='Harry&apos;s SuperMart' subcaption='Revenue and Units
  Sold by Year' xAxisName='Year' PYAxisName='Amount'
  SYAxisName='Units Sold' numberPrefix='$'>
```

The `numberPrefix` attribute gets applied to the primary axes. If we had to specify a number prefix for the secondary axes, we would use the attribute `sNumberPrefix`.

There are two data series in the chart, the first representing the revenue, and the other containing data on the units sold. The second series, representing quantity, has a new attribute `parentYAxis='S'` that lets you configure whether this data series is plotted against the primary axis (`parentYAxis='P'`, by default) or the secondary axis (`parentYAxis='S'`). In our example, as the units sold is to be plotted against the secondary axis, on the right, we have set `parentYAxis='S'`, the other attributes remaining the same as before.

```
<dataset seriesName='Units Sold' parentYAxis='S' renderAs='Line'
  showValues='0' color='666666'>
```

If you were using a 3D chart such as `MSColumn3DLineDY.swf`, the `renderAs` attribute is not required, as the chart can only plot columns on the primary axis and lines on the secondary. Just setting `parentYAxis='Y'` plots the data series as a line against the secondary y-axis.

Summary

In this chapter, we learned how to create the basic charts using FusionCharts that form the building blocks for a large dashboard or a reporting application.

Specifically, we covered:

- How easy it is to download and set up FusionCharts for your application.
- How to create a chart and different methods to provide data to the chart, either in XML or JSON format.
- Different types of charts having one or more series. Single series were created to compare revenues across multiple years. Multi-series charts were created to compare the breakdown of this revenue into two segments –food products and non-food products.
- Combination charts that let you plot multiple types of series on the same chart. We built examples to plot the actual revenues versus the projected revenues, and also the revenues versus the quantity sold.

Now that we've learned how to build charts, we are ready to explore detailed features offered by each chart and how to customize them to your needs, which is the topic of the next chapter.

Where to buy this book

You can buy FusionCharts Beginner's Guide from the Packt Publishing website: `http://www.packtpub.com/fusion-charts-the-official-beginners-guide/book`.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com

For More Information:
www.packtpub.com/fusion-charts-the-official-beginners-guide/book