ASP.NET jQuery Cookbook Second Edition

jQuery is a lightweight JavaScript library that has changed the landscape of client scripting in web applications. It has popularity with ASP.NET developers and is distributed with Visual Studio and the NuGet package manager.

ASP.NET jQuery Cookbook explores the wide range of utilities that the jQuery library provides. It teaches you the nitty-gritty of plugging in these features in ASP.NET web applications. It covers every aspect of interfacing with the library, right from downloading and including jQuery on web pages to selecting controls, handling events, and creating animations. This book also walks you through DOM traversal and manipulation in ASP.NET and then through visual effects and graphics in ASP.NET sites. It explores advanced features such as posting Ajax requests and writing plugins. It will provide you with all the information you need to use this library confidently with ASP.NET.

Inside the Cookbook...
- A straightforward and easy-to-follow format
- A selection of the most important tasks and problems
- Carefully organized instructions to solve problems efficiently
- Clear explanations of what you did
- Solutions that can be applied to solve real-world problems

What this book will do for you...
- Download and include jQuery in ASP.NET websites and MVC
- Use jQuery selectors with ASP.NET server controls
- Get to know about event handling and DOM traversal
- Deploy jQuery for visual effects and animations
- Develop Ajax-enabled ASP.NET applications
- Create your own plugins
- Solve common problems using less code and cut down your development time

There are over 60 recipes for writing client script in ASP.NET 4.6 applications using jQuery.


Sonal Aneel Allana

Over 60 recipes for writing client script in ASP.NET 4.6 applications using jQuery

ASP.NET jQuery Cookbook
Sonal Aneel Allana

$ 54.99 US
£ 34.99 UK

Prices do not include local sales tax or VAT where applicable

Sonal Aneel Allana

Prices do not include local sales tax or VAT where applicable
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Getting Started with jQuery in ASP.NET'
- A synopsis of the book’s content
- More information on ASP.NET jQuery Cookbook (Second Edition)
About the Author

Sonal Aneel Allana works as a sessional lecturer at the Singapore campus of the University of Newcastle and the University of Hertfordshire. Her teaching areas include degree level courses in e-learning, intelligent systems, robotics, operating systems, and programming in C/C++, .NET, Java, and Android. She is keenly interested in JavaScript frameworks, such as Bootstrap, Node.js, and AngularJS. She has worked in the IT industry for over 10 years in various positions, such as an application developer, project leader, and trainer. She holds a master’s degree in computing from the National University of Singapore and a bachelor's degree in computer engineering from the University of Mumbai. She is certified in security technology and computational neuroscience. She is also the author of the first edition of ASP.NET jQuery Cookbook.
jQuery is a lightweight JavaScript library that has changed the landscape of client scripting in web applications. Developed by John Resig in 2006, it has taken the Web by storm because of its cross-browser compatibility and its ability to get more done with less code. The library is supported by an active community of developers and has grown significantly over the years. Using jQuery eases many client scripting tasks, such as event handling, embedding animations, writing Ajax enabled pages, among many more, and adds to the interactive experience of the end user. Its extensible plugin architecture enables developers to build additional functionalities on top of the core library.

Learning jQuery and using it in ASP.NET applications is an indispensable skill for ASP.NET developers. This book attempts to impart this skill by exploring diverse recipes for fast and easy solutions to some of the commonly encountered problems in ASP.NET 4.6 applications.

What this book covers

Chapter 1, Getting Started with jQuery in ASP.NET, describes recipes to download and include jQuery in ASP.NET 4.6 Web and MVC applications. It discusses the CDN, NuGet Package Manager, as well as debugging the jQuery code in Visual Studio.

Chapter 2, Using jQuery Selectors with ASP.NET Controls, describes various jQuery selectors that can be used to manipulate ASP.NET controls. These selectors can select controls based on the ID, CSS class, HTML tag, attribute, or position in the document.

Chapter 3, Event Handling Using jQuery, describes recipes to handle different types of events, such as mouse, keyboard, and form events. It also explains event delegation and detaching of events.
Chapter 4, *DOM Traversal and Manipulation in ASP.NET*, describes techniques to traverse the document, such as accessing parent, child, or sibling elements. It also teaches manipulation strategies to add and remove elements at runtime.

Chapter 5, *Visual Effects in ASP.NET Sites*, discusses recipes to create different types of animation effects on ASP.NET controls, such as Panel, AdRotator, TreeView, Menu, and GridView. Effects such as enlarging, sliding, and fading are covered in this chapter.

Chapter 6, *Working with Graphics in ASP.NET Sites*, discusses recipes to work with images and explains effects, such as zooming, scrolling, and fading on images. Utilities such as image gallery, image preview, and 5-star rating control are also explored in this chapter.

Chapter 7, *Ajax Using jQuery*, explains how Ajax calls can be made to page methods, web services, WCF services, Web API, MVC controllers, and HTTP handlers.

Chapter 8, *Creating and Using jQuery Plugins*, demonstrates how plugins can be created and included in projects. It also describes how to use the Node Package Manager (NPM) and Bower to download and manage third-party plugins.

Chapter 9, *Useful jQuery Recipes for ASP.NET Sites*, summarizes the book with diverse recipes to solve common real-world problems. You can find this chapter at: https://www.packtpub.com/sites/default/files/downloads/4836OT_Chapter_09.
Getting Started with jQuery in ASP.NET

In this chapter, we will cover the following recipes:

- Downloading jQuery from jQuery.com
- Understanding CDN for jQuery
- Using the NuGet Package Manager to download jQuery
- Adding jQuery to an empty ASP.NET web project using a script block
- Adding jQuery to an empty ASP.NET web project using the ScriptManager control
- Adding jQuery to an ASP.NET Master Page
- Adding jQuery programmatically to a web form
- Understanding the jQuery reference in the default Web Application template
- Hello World in a web project using jQuery
- Bundling jQuery in ASP.NET MVC
- Using a CDN to load jQuery in MVC
- Hello World in ASP.NET MVC using jQuery
- Debugging jQuery code in Visual Studio
Introduction

As a web developer, you often require to include functionalities in your websites that make writing a client script in JavaScript inevitable. Getting the client script to produce the same response for all browsers has always been a challenge. jQuery helps you overcome this difficulty. In essence, jQuery is a powerful JavaScript library that works across all browsers, such as Internet Explorer (IE), Firefox, Safari, Chrome, Opera, iOS, and Android. It takes away the agony that developers face in order to maintain their client scripts across different platforms.

jQuery is popular not only because of its cross-browser support, but also because it is packed with features that developers can plug and play. It has changed the way developers write a client script. In addition to reducing the amount of code that needs to be written, it provides features for traversing the DOM, event handling, building animations, and AJAX, among many more.

This chapter deals with acquiring the library and other supporting files. It aims to cover different aspects of including and using jQuery in ASP.NET 4.6 web application projects, such as web forms and MVCs.

Downloading jQuery from jQuery.com

This recipe explains how to download jQuery on your system along with the version/build to use and the supporting files that are required.

Getting ready

Following are the steps to download jQuery:

1. Launch any web browser and enter the URL http://www.jquery.com to access the jQuery home page:
2. Click on the **Download jQuery** button (highlighted in the preceding screenshot) on the right-hand side of the page. This opens up the download page with a list of available files, as shown in the following screenshot:
Getting Started with jQuery in ASP.NET

How to do it...

jQuery is available in two different major versions at the time of writing:

- Version 1.x
- Version 2.x

Though the Application Programming Interface (API) is the same for both major versions, the difference lies in the support offered for certain browsers. The 2.x line does not support old browsers, such as IE 6, 7, and 8, while the 1.x line continues with this support. So, if the end users of your application will not be using old browsers, you can download the 2.x version.

The jQuery library consists of a single JavaScript (.js) file and can be downloaded in the following formats:

- **Uncompressed format**: This is used in a development environment or when debugging the code.
- **Compressed format**: This is used in a production (that is, release) environment. It is compact and uses low bandwidth. It is commonly referred to as the **minified** version.

To download the file, simply right-click on the required version, 1.x or 2.x, and the required format: uncompressed or compressed. Save the file in a location of your choice as shown in the following screenshot:
Note the following naming convention for the jQuery library:

<table>
<thead>
<tr>
<th>Version</th>
<th>Uncompressed</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.x</td>
<td>jquery-1.x.x.js</td>
<td>jquery-1.x.x.min.js</td>
</tr>
<tr>
<td>2.x</td>
<td>jquery-2.x.x.js</td>
<td>jquery-2.x.x.min.js</td>
</tr>
</tbody>
</table>

The compressed (minified) version is clearly distinct from the uncompressed version because of the `.min.js` extension. The minified file uses code optimization techniques, such as removing whitespaces and comments as well as reducing variable names to one character. This version is difficult to read, so the uncompressed version is preferred when debugging.

On the download page, there is also a map file available with the `.min.map` extension. Sometimes, when bugs appear in the production environment necessitating troubleshooting, the use of the minified file for debugging can be difficult. The map file simplifies this process. It maps the compressed file back to its unbuilt state so that during debugging, the experience becomes similar to using the uncompressed version.

See also...

The Understanding CDN for jQuery recipe.

Understanding CDN for jQuery

A Content Delivery Network (CDN) hosts content for users through large distributed systems. The advantage of using a CDN is to improve the performance. When using a CDN to retrieve the jQuery library, if the files have been downloaded earlier, they will not be re-downloaded. This can help you improve the response time.

How to do it...

The following CDNs are available for jQuery files:

- jQuery’s CDN provided by MaxCDN
- The Google CDN
- The Microsoft CDN
- The CDNJS CDN
- The jsDelivr CDN
To include jQuery on a web page, the URL of the respective CDN can be used so that files can be directly served from the CDN instead of using the local copies. The following table summarizes the respective CDN URLs for jQuery files:

<table>
<thead>
<tr>
<th>CDN</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>jQuery's CDN</td>
<td>Version 2.x:</td>
</tr>
<tr>
<td></td>
<td><a href="http://code.jquery.com/jquery-2.x.x.js">http://code.jquery.com/jquery-2.x.x.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://code.jquery.com/jquery-2.x.x.min.js">http://code.jquery.com/jquery-2.x.x.min.js</a></td>
</tr>
<tr>
<td></td>
<td>Version 1.x:</td>
</tr>
<tr>
<td></td>
<td><a href="http://code.jquery.com/jquery-1.x.x.js">http://code.jquery.com/jquery-1.x.x.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://code.jquery.com/jquery-1.x.x.min.js">http://code.jquery.com/jquery-1.x.x.min.js</a></td>
</tr>
<tr>
<td>The Google CDN</td>
<td>Version 2.x:</td>
</tr>
<tr>
<td></td>
<td><a href="https://ajax.googleapis.com/ajax/libs/jquery/2.x.x/jquery.js">https://ajax.googleapis.com/ajax/libs/jquery/2.x.x/jquery.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://ajax.googleapis.com/ajax/libs/jquery/2.x.x/jquery.min.js">https://ajax.googleapis.com/ajax/libs/jquery/2.x.x/jquery.min.js</a></td>
</tr>
<tr>
<td></td>
<td>Version 1.x:</td>
</tr>
<tr>
<td></td>
<td><a href="https://ajax.googleapis.com/ajax/libs/jquery/1.x.x/jquery.js">https://ajax.googleapis.com/ajax/libs/jquery/1.x.x/jquery.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://ajax.googleapis.com/ajax/libs/jquery/1.x.x/jquery.min.js">https://ajax.googleapis.com/ajax/libs/jquery/1.x.x/jquery.min.js</a></td>
</tr>
<tr>
<td>The Microsoft CDN</td>
<td>Version 2.x:</td>
</tr>
<tr>
<td></td>
<td><a href="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.x.x.js">http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.x.x.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.x.x.min.js">http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.x.x.min.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.x.x.min.map">http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.x.x.min.map</a></td>
</tr>
<tr>
<td></td>
<td>Version 1.x:</td>
</tr>
<tr>
<td></td>
<td><a href="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.x.x.js">http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.x.x.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.x.x.min.js">http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.x.x.min.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.x.x.min.map">http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.x.x.min.map</a></td>
</tr>
</tbody>
</table>
### CDN URL

<table>
<thead>
<tr>
<th>CDN</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The CDNJS CDN</strong></td>
<td>Version 2.x:</td>
</tr>
<tr>
<td></td>
<td><a href="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.x.x/jquery.js">https://cdnjs.cloudflare.com/ajax/libs/jquery/2.x.x/jquery.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.x.x/jquery.min.js">https://cdnjs.cloudflare.com/ajax/libs/jquery/2.x.x/jquery.min.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.x.x/jquery.min.map">https://cdnjs.cloudflare.com/ajax/libs/jquery/2.x.x/jquery.min.map</a></td>
</tr>
</tbody>
</table>

|                | Version 1.x:               |
|                | https://cdnjs.cloudflare.com/ajax/libs/jquery/1.x.x/jquery.js |
|                | https://cdnjs.cloudflare.com/ajax/libs/jquery/1.x.x/jquery.min.js |
|                | https://cdnjs.cloudflare.com/ajax/libs/jquery/1.x.x/jquery.min.map |

<table>
<thead>
<tr>
<th><strong>The jsDelivr CDN</strong></th>
<th>Version 2.x:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="https://cdn.jsdelivr.net/jquery/2.x.x/jquery.js">https://cdn.jsdelivr.net/jquery/2.x.x/jquery.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://cdn.jsdelivr.net/jquery/2.x.x/jquery.min.js">https://cdn.jsdelivr.net/jquery/2.x.x/jquery.min.js</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://cdn.jsdelivr.net/jquery/2.x.x/jquery.min.map">https://cdn.jsdelivr.net/jquery/2.x.x/jquery.min.map</a></td>
</tr>
</tbody>
</table>

|                | Version 1.x:               |
|                | https://cdn.jsdelivr.net/jquery/1.x.x/jquery.js |
|                | https://cdn.jsdelivr.net/jquery/1.x.x/jquery.min.js |
|                | https://cdn.jsdelivr.net/jquery/1.x.x/jquery.min.map |

### Using CDNs for new releases

Note that CDNs may not have the latest files when new versions of the jQuery library are launched since it usually takes a couple of days for third-parties to update their files. In the case of new releases, always check the available version before downloading them.

### How it works...

CDNs consist of servers situated in data centers in strategic locations across the globe. When a client requests a resource from a CDN, the server that is geographically closest to the client processes the request. These servers are also known as edge servers. In addition to this, edge servers have a caching mechanism to serve various assets. All this helps you improve the client's response time.
Using NuGet Package Manager to download jQuery

NuGet is a package manager available with Visual Studio. It simplifies the process of installing and upgrading packages. This recipe demonstrates the use of NuGet to download the jQuery library.

Getting ready

To launch NuGet for a particular project, go to Tools | NuGet Package Manager | Manage NuGet Packages for Solution... as shown in the following screenshot:

Alternatively, right-click on the project in the Solution Explorer tab, and select Manage NuGet Packages.
How to do it...

Perform the following steps to download jQuery using NuGet Manager:

1. In the **NuGet Package Manager**, as shown in the following screenshot, select the **jQuery** package from the left-hand side panel. In the right-hand side panel, select the **Version** that you would like to use in your web project from the drop-down menu. Click on the **Install** button:

   ![NuGet Package Manager](image)

   Searching for packages in NuGet
   If jQuery is not visible in the left-hand side panel, you need to search for it by keying in **jQuery** in the search box in the top left corner of the NuGet Manager screen.

2. Click on **OK** when prompted for confirmation in order to make the required changes to the solution.
How it works...

The NuGet Package Manager downloads the selected version of jQuery in the `Scripts` folder. Any other version existing in the `Scripts` folder is deleted. The `Scripts` folder will look like the following screenshot:

The files downloaded by NuGet are as follows (the version numbers may change in the future):

- The Intellisense file: `jquery-2.1.4.intellisense.js`
- The debug version: `jquery-2.1.4.js`
- The release version: `jquery-2.1.4.min.js`
- The map file: `jquery-2.1.4.min.map`

See also

The Downloading jQuery from jQuery.com recipe

Adding jQuery to an empty ASP.NET web project using a script block

To create ASP.NET 4.6 Web Applications, Visual Studio provides various ready templates such as Empty, Web Forms, MVC, Web API, and so on. This recipe will use the Empty template, which provides the developer with an empty project structure that consists of only the `web.config` file.

Downloading the example code

You can download the example code files for this book 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.
Getting ready

Following are the steps to create a project by using Empty template:

1. Create a new project in Visual Studio by going to **File | New | Project...**, as shown in the following screenshot:

   ![New Project Screenshot]

   **Website or web project?**

   Instead of creating a new project, you can also create a new website. Unlike a project, a website does not contain a collective project file to track individual files in the application. To create a website, go to **File | New | Website...**. This will launch the **New Website** dialog box with the list of available templates. Select the **ASP.NET Empty WebSite** template.
2. This will launch the New Project dialog box, as shown in the following screenshot. From the left-hand side panel, select your desired programming language, Visual C# or Visual Basic, and then, select ASP.NET Web Application from the middle panel:

![New Project dialog box](image.png)

3. Enter WebApplication1 (or any suitable name) in the Name field. Click on the Browse button to go to the desired Location where you would like to save the application. Click on OK.

4. This will launch the Select a template dialog box, as shown in the following screenshot:
5. From **ASP.NET 4.6 Templates**, select **Empty**, and click on **OK**. Visual Studio will create an empty project in the **Solution Explorer** tab, as shown in the following screenshot:
In the remaining recipes, when asked to create a Web Application project using the Empty template, follow the steps listed in this section.

**How to do it...**

Following are the steps to include jQuery using script block:

1. JavaScript files are usually placed in a folder named `Scripts` in the web application. So, in the **Solution Explorer** tab, right-click on the project and go to **Add | New Folder** from the menu:

2. Rename the folder to `Scripts`. Now, right-click on the `Scripts` folder, and go to **Add | Existing Item...** as shown in the following screenshot:
3. Now, browse to the location where you have saved the downloaded copy of the jQuery files (refer to the Downloading jQuery from jQuery.com recipe), and click on OK. It is recommended that you add both the uncompressed and compressed versions. The Scripts folder will be updated, as shown in the following screenshot:
4. Next, create a new web form in the project by right-clicking on the project and navigating to **Add | New Item...**. From the dialog box, select **Web Form**, and enter a suitable name for the web form, such as **Default.aspx**:

![Add New Item - WebApplication dialog box]

5. To use jQuery on the web form, simply drag and drop the required jQuery file, that is, uncompressed or compressed on the web form. Or alternatively, include the following `<script>` tag in the `<head>` element:

   For development mode, the code is as follows:

   ```html
   <script src="Scripts/jquery-2.1.4.js"></script>
   ```

   For release mode, the code is as follows:

   ```html
   <script src="Scripts/jquery-2.1.4.min.js"></script>
   ```
Adding jQuery to an empty ASP.NET web project using ScriptManager control

Adding jQuery to a web form using the script block has some disadvantages. If the application is upgraded to use the latest version of jQuery, all the web forms with the `<script>` tag require to be changed. Secondly, switching from the uncompressed version in the development environment to the compressed version in the release environment should be handled manually and is hence error-prone. Using the ASP.NET ScriptManager control helps you overcome this problem. It can also load jQuery directly from CDN instead of using the local copy.

Getting ready

1. Create a new ASP.NET Web Application project using the Empty template by following the steps listed in the Adding jQuery to an empty ASP.NET web project using a script block recipe. Name the project WebApplication2 (or any other suitable name).
2. Follow the steps in the preceding recipe to add the jQuery library (the uncompressed and compressed formats) to the Scripts folder.
3. Follow the steps to add a new web form to the project.

See also

The Downloading jQuery from jQuery.com recipe
How to do it...

Following are the steps to add jQuery to ASP.NET web project using the ScriptManager control:

1. Open the web form in the Design mode.

2. Launch the Toolbox. This can be done in two ways. From the File menu at the top of the page, go to View | Toolbox. Alternatively, use the shortcut keys, Ctrl + Alt + X.

3. Go to Toolbox | AJAX Extensions, and drag and drop the ScriptManager control onto the form:

4. Right-click on the project in the Solution Explorer tab, and go to Add | New Item.... From the dialog box, select Global Application Class. This will add the Global.asax file to the project:
The Global.asax file is an optional file that resides in the root directory of the application and responds to events at the application and session levels, such as the starting and ending an application or session.

5. Open the Global.asax file and include the following namespace at the top of the page:

   For VB, the code is as follows:
   ```vbnet
   Imports System.Web.UI
   ```

   For C#, the code is as follows:
   ```csharp
   using System.Web.UI;
   ```
6. In the Application_Start event in the Global.asax file, add the following code to create a script that maps to jQuery:

For VB, the code is as follows:

```vbnet
Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
        .Path = "/Scripts/jquery-2.1.4.min.js",
        .DebugPath = "/Scripts/jquery-2.1.4.js",
        .CdnSupportsSecureConnection = True,
        .LoadSuccessExpression = "window.jQuery"
    })
End Sub
```

For C#, the code is as follows:

```csharp
protected void Application_Start(object sender, EventArgs e)
{
    {
        Path = "/Scripts/jquery-2.1.4.min.js",
        DebugPath = "/Scripts/jquery-2.1.4.js",
        CdnPath = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js",
        CdnDebugPath = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.js",
        CdnSupportsSecureConnection = true,
        LoadSuccessExpression = "window.jQuery"
    });
}
```

7. Open the Default.aspx web form in the Source mode. Add the following ScriptReference to the ScriptManager control:

```xml
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Scripts>
        <asp:ScriptReference Name="jquery" />
    </Scripts>
</asp:ScriptManager>
```
When using the ScriptManager control to add a reference to the jQuery library, the jQuery code should be placed after the ScriptManager control, that is, after the jQuery reference has been declared; otherwise, the page will throw an error. It is also important to note that the ScriptManager control should reside inside the `<form>` element.

8. To retrieve the jQuery files from CDN, set the EnableCdn property of the ScriptManager control to true, as follows:

```xml
<asp:ScriptManager ID="ScriptManager1" runat="server" EnableCdn="true">
    <Scripts>
        <asp:ScriptReference Name="jquery" />
    </Scripts>
</asp:ScriptManager>
```

### How it works...

This is how the ScriptManager control works:

1. The ScriptManager control can be used to load JavaScript files, such as the jQuery library. This can be done by adding the ScriptReference to jQuery in the ScriptManager control, as follows:

   ```xml
   <asp:ScriptReference Name="jquery" />
   ```

2. However, we require to define this mapping. This can be done in the Global.asax file using a ScriptResourceDefinition object, which exposes the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>This is the release path of the script resource</td>
</tr>
<tr>
<td>DebugPath</td>
<td>This is the development/debug path of the script resource</td>
</tr>
<tr>
<td>CdnPath</td>
<td>This is the release path of the script resource served from a CDN</td>
</tr>
<tr>
<td>CdnDebugPath</td>
<td>This is the development/debug path of the script resource served from a CDN</td>
</tr>
<tr>
<td>CdnSupportsSecureConnection</td>
<td>This indicates whether the HTTPS mode needs to be used to retrieve the resource when the page is accessed using a secure connection</td>
</tr>
<tr>
<td>LoadSuccessExpression</td>
<td>This is the JavaScript expression that detects whether a JavaScript file has been loaded successfully</td>
</tr>
</tbody>
</table>
3. The `ScriptResourceDefinition` object defined in `Global.asax` is named `jquery`. The `ScriptManager` control uses the same name to load the reference on the web form.

4. In the development/debug mode, the script is served from `DebugPath` while in the release mode, it is served from `Path`.

---

### Running in development/debug and release modes

To run the application in the development/debug mode, set the `debug` attribute of the `<compilation>` element in the web.config to `true` as follows:

```xml
<system.web>
    <compilation debug="true"/>
    ...
</system.web>
```

When the `debug` attribute is set to `false`, the application will run in the release mode.

---

5. If `EnableCdn` is set to `true`, the script is served from the CDN path, that is, from `CdnDebugPath` in the development/debug mode and `CdnPath` in the release mode.

6. The `LoadSuccessExpression` property renders an inline script to load the library from the local path in the event of a CDN failure. By right-clicking on the web page and viewing the source, note that the `ScriptManager` control adds a fallback mechanism when the CDN is unavailable and files are served locally instead:

```html
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.js" type="text/javascript"></script>  
<script type="text/javascript">  
(function($){  
    //<![CDATA[  
    (window.$) || document.write("<script type="text/javascript" src="scripts/jquery-2.1.4.js"></script>"); //]]>  
}(jQuery));  
</script>
```

---

### See also

- The Adding jQuery to an empty ASP.NET web project using a script block recipe
- Master Pages are used to achieve a uniform look and feel in the website. They maintain a consistent layout across all the content pages. Including jQuery in the Master Page ensures that all the content pages using that Master Page will also have the library included by default. This recipe will demonstrate how this can be done.
A Master Page is an ASP.NET file with the .Master extension. It has a @Master directive at the top of the layout instead of the @Page directive in an ordinary .aspx page.

Getting ready

1. Create a new ASP.NET Web Application project using the Empty template by following the steps listed in the Adding jQuery to an empty ASP.NET web project using a script block recipe. Name the project WebApplicationWithMaster (or any other suitable name).

2. Follow the steps in the previous recipe to add the jQuery library (the uncompressed and compressed formats) to the Scripts folder.

3. In the Solution Explorer tab, right-click on the project, and go to Add | New Item.... This will launch a dialog box, as shown in the following screenshot. From the dialog box, select Web Forms Master Page. Name the Master Page Default.Master, and click on Add:
4. To add a web form—that is, a content page—to the project, right-click on the project in the **Solution Explorer** tab again, and navigate to **Add | New Item...**. From the dialog box, this time select **Web Form with Master Page**, as shown in the following screenshot. Name the web form **Default.aspx**, and click on **Add**:

![Add New Item - WebApplicationWithMaster](https://via.placeholder.com/150)

5. This will launch a dialog box so that you can select the Master Page. From the dialog box, as shown in the following screenshot, select the Master Page to be associated with the content page, and click on **OK**:

![Select a Master Page](https://via.placeholder.com/150)
To incorporate jQuery in an ASP.NET Master Page, follow these steps:

1. Open the Default.Master Master Page in the Source mode, and add a reference to the jQuery library using either the `<script>` block (refer to the Adding jQuery to an empty ASP.NET web project using a script block recipe) or the ScriptManager control (refer to the Adding jQuery to an empty ASP.NET web project using the ScriptManager control recipe), as shown in the following screenshot:
When using the `<script>` block, the jQuery reference should preferably be placed in the `<head>` element.

When using the `ScriptManager` control, the control should preferably be placed in the `<form>` element before the `ContentPlaceHolder` in which the jQuery code will be added later to the content pages. The `Global.asax` file should also be updated in order to add the required `ScriptResourceDefinition`, as described in the Adding jQuery to an empty ASP.NET web project using the ScriptManager control recipe.

2. The required jQuery code can now be added to the `ContentPlaceHolder` (with ID = "ContentPlaceHolder1") in the `Default.aspx` web form.

**How it works...**

On running the application, when the `Default.aspx` content page is loaded, the HTML markup from the Master page adds the reference to the jQuery library. This makes the content page jQuery-ready so that any jQuery code can be executed.

To check whether the jQuery reference has been added to the page, run the project and launch `Default.aspx` in the browser. Right-click on the page in the browser window and select View Source. The jQuery reference will be seen on the page, as shown in the following screenshot:

```html
<script src="/Scripts/jquery-1.1.4.js" type="text/javascript"></script>
<script type="text/javascript">
</script>
```

**See also**

The Adding jQuery to an empty ASP.NET web project using the ScriptManager control recipe
Adding jQuery programmatically to a web form

In addition to adding jQuery to web forms using the script block and the ScriptManager control, the code-behind file can also emit the required script code. This recipe will demonstrate how this can be done.

Getting ready

1. Create an ASP.NET Web Application project by navigating to File | New | Project | ASP.NET Web Application. Select the Empty template. Name the project WebApplicationWithPageLoad (or any other suitable name).
2. Add a new Web Form to the project and name it Default.aspx.
3. Add the jQuery library files to the Scripts folder.
4. From the Solution Explorer tab, navigate to Default.aspx.vb (VB) or Default.aspx.cs (C#), which is the code-behind file for the web form. Open this file.

How to do it...

In the Page_Load event handler of Default.aspx.vb, use the RegisterClientScriptInclude method to generate a script block on the page, as follows:

For VB, the code is as follows:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
End Sub
```

For C#, the code is as follows:

```
protected void Page_Load(object sender, EventArgs e)
{
}
```
Getting Started with jQuery in ASP.NET

How it works...

The `RegisterClientScriptInclude` method requires two parameters: the key and URL. It adds the script block with the path to the jQuery library in the `<form>` element, as shown in the following screenshot. The `Page.ResolveUrl` method is used to return a URL relative to the site root:

```html
<body>
    <form method="post" action="Default.aspx" id="form1">
        <div class="aspNetHidden">
            <input type="hidden" name="__VIEWSTATE" value="wzvezu1h3wbc13mz7pHf062FHe185yjeb41mL308H9LX22cAN6W7b4a4e1HuZleS0Z4K0c8Lli/smQr4ntK4xKvZHjyEuSg74gKc="/>
        </div>
    </form>
    <script src="/Scripts/jquery-2.1.4.js" type="text/javascript"></script>
</body>
```

Since the jQuery library is added to the `<form>` element, all the jQuery code should be written in the `<form>` element instead of the `<head>` element, preferably toward the end of the page before closing the `<form>` element.

See also

The Adding jQuery to an empty ASP.NET web project using a script block recipe

Understanding jQuery reference in the default web application template

So far, all examples have used the Empty template for the ASP.NET Web Application project. When using a non-empty built-in web application template, ASP.NET adds a reference to the jQuery library in the Master Page using the ScriptManager control. This recipe walks you through the important details of this mapping.

How to do it...

Here are the steps to create an ASP.NET web application using the default web application template:

1. Create a new project by navigating to File | New | Project.... From the dialog box, select ASP.NET Web Application. Name the project DemoWebApplication (or any other suitable name), and click on OK.

2. A new dialog box will be launched. Select Web Forms from the available templates. Note that the Web Forms checkbox is checked by selecting the Web Forms template (refer to the following screenshot) and click on OK as shown in the following screenshot:
3. Open the `Site.Master` Master Page in the Source mode, as shown in the following screenshot:
4. Notice that the ScriptManager control that is added to the `<form>` element has the following reference to jQuery:

```xml
<asp:ScriptReference Name="jquery" />
```

### How it works...

When you follow the preceding steps, this is how the web application is mapped to the jQuery library:

1. The ScriptManager control switches the jQuery library between the development and release versions, depending on the debug attribute of the `<compilation>` element in web.config:

```xml
<compilation debug="true"/>
```

2. When the debug attribute is true, the uncompressed version is used. When debug is false, the minified version is used.

3. The default template is shipped with the AspNet.ScriptManager.jquery package. This package adds the following ScriptMappings to jQuery in the PreApplicationStart method of the application as follows:

   For C#, the code is as follows:

   ```csharp
   string str = "2.4.1";
   {
       Path = "~/Scripts/jquery-" + str + ".min.js",
       DebugPath = "~/Scripts/jquery-" + str + ".js",
       CdnPath = "http://ajax.aspnetcdn.com/ajax/jQuery/jquery-" + str + ".min.js",
       CdnDebugPath = "http://ajax.aspnetcdn.com/ajax/jQuery/jquery-" + str + ".js",
       CdnSupportsSecureConnection = true,
       LoadSuccessExpression = "window.jQuery"
   });
   ```

   The default Web Forms template adds the Microsoft CDN URL, as shown in the preceding code.

4. When the EnableCdn property of the ScriptManager control is set to true, CdnPath and CdnDebugPath are used in release and development modes, respectively, to serve scripts from the Microsoft CDN:

```xml
<asp:ScriptManager runat="server" EnableCdn="true"/>
```
5. However, if the CDN is down or if the application is offline, the ScriptManager control will include a fallback mechanism to serve the local copy of jQuery, as shown in the following screenshot:

```html
<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.1.4.js" type="text/javascript"></script>
<script type="text/javascript">(window.jQuery)||(document.write('<script type="text/javascript" src="Scripts/jquery-2.1.4.js"></script>');)
</script>
```

6. To change the CDN to another, for example Google CDN, we need to change ScriptResourceMapping in the RegisterBundles method in BundleConfig, as shown in the following code. This module/class is located in the App_Start folder:

For VB, the code is as follows:

```vbnet
    .Path = "/Scripts/jquery-2.1.4.min.js",
    .DebugPath = "/Scripts/jquery-2.1.4.js",
    .CdnSupportsSecureConnection = True,
    .LoadSuccessExpression = "window.jQuery"})
```

For C#, the code is as follows:

```csharp
    Path = "/Scripts/jquery-2.1.4.min.js",
    DebugPath = "/Scripts/jquery-2.1.4.js",
    CdnPath = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js",
    CdnDebugPath = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.js",
    CdnSupportsSecureConnection = true,
    LoadSuccessExpression = "window.jQuery"
});
```
7. By running the page and viewing the source in the browser window, note that Microsoft CDN is replaced with Google CDN as required:

```html
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.js" type="text/javascript"></script>
<script type="text/javascript">
(function(window, $) {document.write('<script type="text/javascript" src="Scripts/jquery-1.4.js"></script>');});//[]
</script>
```

8. Open the `Global.asax` page to view the registration of bundles in the `Application_Start` event handler as follows:

   For VB, the code is as follows:
   ```vbnet
   BundleConfig.RegisterBundles(BundleTable.Bundles)
   ```

   For C#, the code is as follows:
   ```csharp
   BundleConfig.RegisterBundles(BundleTable.Bundles);
   ```

See also

The Adding jQuery to an empty ASP.NET web project using the ScriptManager control recipe

**Hello World in a web project using jQuery**

Until now, all recipes have demonstrated different ways to add the jQuery library to web pages. This is the first step in making the page jQuery-ready. In this recipe, let's move on to the next step: writing the jQuery code inside a script block to manipulate controls in a web form. We will display a simple Hello World message on the web page by manipulating a `Label` control on a web form.

**Getting ready**

1. Create a Web Application project by going to File | New | Project | ASP.NET Web Application. Select the Empty template. Name the project HelloWorld (or any other suitable name).

2. Add a new Web Form to the project.

3. Add the jQuery library files to the Scripts folder.

5. Open the web form in the Design mode and drag and drop a Label control by navigating to the Toolbox | Standard controls. Change the properties of the Label control as follows:

<asp:Label ID="lblMessage" runat="server" Text=""></asp:Label>

How to do it...

If a jQuery reference is added to the <head> element, then include the following <script> block in the <head> element. Otherwise, include the <form> element, preferably before the <form> tag is closed:

<script type="text/javascript">
$(document).ready(function () {
  var fontStyle = "Arial";
  var fontSize = 28;
  $("#<%=lblMessage.ClientID%>").css("font-family", fontStyle);
  $("#<%=lblMessage.ClientID%>").css("font-size", fontSize);
  $("#<%=lblMessage.ClientID%>").text("Hello World!!");
});
</script>

How it works...

Following are the steps to print Hello World!! in a web project using jQuery:

1. In the preceding jQuery code, the $ symbol is used to instantiate the jQuery object.
2. The .ready() function is triggered when the DOM is ready. It is commonly used to execute the required jQuery code on the page.
3. The Label control can be accessed from the jQuery code using ASP.NET's ClientID property and jQuery's #identifier selector.
4. Using the `.css()` property of the jQuery object, the font style, size, and text of the Label control are manipulated so that the following output is displayed on running the application:

![Output Image]

See also

The Hello World in ASP.NET MVC using jQuery recipe

Bundling jQuery in ASP.NET MVC

Model View Controller (MVC) is a design pattern that separates design (Model), presentation (View), and action (Controller). Because of its popularity with developers, Visual Studio provides ready templates that are used to create MVC projects.

Similar to web forms, jQuery can be included in MVC views using the `<script>` tag. In this example, however, let's take a look at the use of bundling for this purpose.

Bundling helps you reduce the number of HTTP requests made by the browser. It is a feature that allows style sheets, JavaScript, or other files to be combined together in a single file called a bundle. This combined file can be downloaded as one unit using a single HTTP request.
Getting ready

1. Launch a new ASP.NET Web Application project in Visual Studio using the **Empty** template. Ensure that the **MVC** checkbox is checked, as shown in the following screenshot:
2. This will create a project with MVC folders. Right-click on the Controllers folder in the Solution Explorer tab, and go to Add | Controller... as shown in the following screenshot:

3. This will launch the Add Scaffold dialog box. Select MVC 5 Controller – Empty, and click on the Add button:
4. On being prompted to add a name for the controller, type **HomeController** and click on the **Add** button:

![Add Controller Window](image)

5. Next, open the **HomeController** in the source mode, and right-click on the **Index** action method, as shown in the following screenshot. Click on **Add View...** as shown in the following screenshot:

![HomeController Class](image)
6. This will launch the **Add View** dialog box. From the **Template** field, select **Empty (without model)**. Uncheck the **Use a layout page** option and click the **Add** button to continue:

![Add View Dialog Box](image)

In the remaining recipes, when asked to create a MVC application, follow steps 1 to 6 as mentioned earlier.

7. To use bundling, we need to install the ASP.NET Web Optimization package. This can be done from NuGet. From the **File** menu, launch NuGet by navigating to **Project** | **Manage NuGet Packages**. Select **Microsoft.AspNet.Web.Optimization** from the list of available packages. If the package is not visible, search for **web.optimization**, as shown in the following screenshot. Click on the **Install** button to download and install the latest version:

![Manage NuGet Packages](image)
Lastly, create a Scripts folder in the project and include the jQuery library files in the folder.

**How to do it...**

Follow these steps to bundle jQuery in ASP.NET MVC:

1. Open the BundleConfig class in the App_Start folder in the MVC project. If the file does not exist, create a new module (VB)/class (C#) in the App_Start folder, and name it BundleConfig.vb/BundleConfig.cs.

2. In BundleConfig.vb/BundleConfig.cs, add a namespace to System.Web.Optimization at the top of the file:

   For VB, the code is as follows:
   ```vbnet
   Imports System.Web.Optimization
   ```

   For C#, the code is as follows:
   ```csharp
   using System.Web.Optimization;
   ```
3. Register and configure a bundle for jQuery in the RegisterBundles method in BundleConfig as follows:

For VB, the code is as follows:

```vbnet
Public Module BundleConfig
    Public Sub RegisterBundles(ByVal bundles As BundleCollection)
        bundles.Add(New ScriptBundle("~/Scripts/jquery").Include("~/Scripts/jquery-{version}.js"))
    End Sub
End Module
```

For C#, the code is as follows:

```csharp
public class BundleConfig
{
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/Scripts/jquery").Include("~/Scripts/jquery-{version}.js"));
    }
}
```

4. To enable bundling in the development mode (optional), add the following code to the RegisterBundles method:

For VB, the code is as follows:

```vbnet
BundleTable.EnableOptimizations = True
```

For C#, the code is as follows:

```csharp
BundleTable.EnableOptimizations = true;
```

5. In the Global.asax file, include the namespace for System.Web.Optimization, as shown in step 2 mentioned previously. Then, register the bundle in the Application_Start method as follows:

For VB, the code is as follows:

```vbnet
BundleConfig.RegisterBundles(BundleTable.Bundles)
```

For C#, the code is as follows:

```csharp
BundleConfig.RegisterBundles(BundleTable.Bundles);
```
6. Now, open the Index view and include the namespace for System.Web.Optimization, as shown in the following code:

For VB, the code is as follows:
```vbnet
Imports System.Web.Optimization
```

For C#, the code is as follows:
```csharp
using System.Web.Optimization
```

7. Next, add the script reference for jQuery to the view in the `<head>` element as follows:
```csharp
@Scripts.Render("~/Scripts/jquery")
```

How it works...

Bundling jQuery in ASP.NET MVC can be done by following these steps:

1. The wildcard string used for bundling jQuery `~/Scripts/jquery-{version}.js` includes the development as well as the minified versions. The `.vsdoc` file, which is used by IntelliSense, is not included in the bundle.

2. When the debug mode is on, the corresponding debug version is used. In the release mode, the minified version is bundled.

3. On running the view in a browser, the bundled file can be seen on viewing the source in the browser window, as shown in the following HTML markup:

```html
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Index</title>
  <script src="/Scripts/jquery?umgG9RT7xo0nFK-4qvkXoehmlbyGmA553tmz-Wto5bS5s1c1"></script>
</head>
</body>
```
Getting Started with jQuery in ASP.NET

See also

The Using a CDN to load jQuery in MVC recipe

Using CDN to load jQuery in MVC

Because of the advantages of using CDN in web applications, bundling also supports the loading of files directly from CDN. This recipe will explain how a MVC project can be configured to use CDN.

Getting ready

This recipe is a continuation of the previous recipe, Bundling jQuery in ASP.NET MVC. So, follow all the steps described in the previous recipe.

How to do it...

Following are the steps to load jQuery in MVC:

1. In the BundleConfig module/class, modify the RegisterBundles method in order to set the UseCdn property to true, as shown in the code snippet in step 2.

2. Declare the required CDN path, and add a ScriptBundle with two parameters: the virtual path of the bundle and the CDN path, as follows:

   For VB, the code is as follows:
   ```vbnet
   Public Module BundleConfig
       Public Sub RegisterBundles(ByVal bundles As BundleCollection)
           bundles.UseCdn = True
           Dim cdnPath As String =  "http://ajax.aspnetcdn.com/ajax/
                                     jQuery/jquery-2.1.4.min.js"
           bundles.Add(New ScriptBundle("~/Scripts/jquery", cdnPath).
                        Include("~/Scripts/jquery-{version}.js"))
       End Sub
   End Module
   ```

   For C#, the code is as follows:
   ```csharp
   public class BundleConfig
   {
       public static void RegisterBundles(BundleCollection bundles)
       {  
   ```
bundles.UseCdn = true;
string cdnPath = "http://ajax.aspnetcdn.com/ajax/jQuery/
jquery-2.1.4.min.js";
bundles.Add(new ScriptBundle("~/Scripts/jquery", cdnPath).
Include("~/Scripts/jquery-{version}.js");
}
}

**How it works...**

Following are the steps to load jQuery in MVC using CDN:

1. By setting the UseCdn property, serving of bundled scripts from the CDN is enabled.
2. In the development mode, the application retrieves files from the local Scripts folder. In the release mode, the CDN path is used to serve the bundled scripts.
3. However, there is a possibility that the CDN is down. Hence, a fallback mechanism is required so that the scripts are served locally in such a scenario. This can be done by adding the following `<script>` block in the required view:

```csharp
@Scripts.Render("~/Scripts/jquery")
<script type="text/javascript">
    if (typeof jQuery == 'undefined') {
        var e = document.createElement('script');
        e.src = '@Url.Content("~/Scripts/jquery-2.4.1.js")';
        e.type = 'text/javascript';
        document.getElementsByTagName("head")[0].appendChild(e);
    }
</script>
```

**See also**

The *Hello World in ASP.NET MVC using jQuery* recipe

**Hello World in ASP.NET MVC using jQuery**

This recipe demonstrates how to write a simple jQuery code to display Hello World in the ASP.NET MVC project.

**Getting ready**

Use the *MyMvcApplication* project created in the *Bundling jQuery in ASP.NET MVC* recipe.
Getting Started with jQuery in ASP.NET

How to do it...

Following are the steps to write simple jQuery code:

1. Open the Index view, and add the following markup to the <body> element:
   ```html
   <div id="divMessage">
   </div>
   ```

2. In the <head> element, include the following jQuery code:
   ```javascript
   $(document).ready(function () {
   var fontStyle = "Arial";
   var fontSize = 28;
   $('#divMessage').css("font-family", fontStyle);
   $('#divMessage').css("font-size", fontSize);
   $('#divMessage').text("Hello World!!");
   });
   ```

3. Right-click on the Index view, and select View in Browser (Internet Explorer).

How it works...

Following are the steps to print Hello World in ASP.NET MVC using jQuery:

1. The $ symbol is used to instantiate the jQuery object.
2. The .ready() function is triggered when the DOM is ready. It is commonly used to execute the required jQuery code on the page.
3. The HTML <div> element with id = "divMessage", which is used to display the Hello World message, can be accessed using its ID with jQuery's #identifier selector—that is, using the #divMessage selector.
4. Using the .css() property of the jQuery object, the font style, size, and text of the <div> element are manipulated so that the following output is displayed on running the application:
Debugging jQuery code in Visual Studio

Debugging is inevitable for resolving bugs in the code during the development phase. Sometimes, bugs also slip into production. Visual Studio provides support for developers to debug the JavaScript code in the same manner as the server-side code. However, there is a limitation and debugging in Visual Studio can only be done using the Internet Explorer browser at present.

See also

The Bundling jQuery in ASP.NET MVC recipe
Getting ready

1. To enable debugging for a particular project, both the project properties and web.config must be updated. To update the project properties, right-click on the project in the Solution Explorer tab, and select Properties. Go to the Web tab, and select the ASP.NET checkbox in the Debuggers section, as shown in the following screenshot:

2. In the web.config file, go to the configuration/system.web/compilation element. If the element does not exist, add a new node. To enable debugging, the debug property of the <compilation> node should be set to true, as follows:

```xml
<compilation debug="true" ... />
```

How to do it...

Debugging jQuery code in Visual Studio can be done by performing the following steps:

1. The first step in debugging is to define breakpoints in the JavaScript code, where the execution will be halted so that variables, program flow, and so on can be inspected. To define breakpoints, just click on the left-hand side gray margin in the source code. Each breakpoint is represented by a small red circle, as shown in the following figure:
2. Press F5, or navigate to Debug | Start Debugging, to start running the application in the debug mode. The execution will stop at the first breakpoint that it comes across, as shown in the following screenshot:
3. To launch the Watch window in order to observe the values of variables during runtime, go to **Debug | Windows | Watch**. This will display the window, as shown in the preceding screenshot.

4. You will also be able to see a window showing the breakpoints by navigating to **Debug | Windows | Breakpoints**. The result is shown in the following screenshot:

![Breakpoints Window](image)

5. To trace the code line by line, press **F11** or navigate to **Debug | Step Into** at each line. To skip to the next breakpoint, press **F5**.

6. Press **Shift + F5** to stop debugging.

   Make sure that you turn off debugging before launching the application in the production environment. An application that has debugging enabled has a slower performance since debugging generates additional information to enable the debugger to display the contents of variables. It also outputs more information to the call stack, which can become a security issue in the production environment.

**See also**

The *Hello World in a web project using jQuery recipe*
Where to buy this book

You can buy ASP.NET jQuery Cookbook (Second Edition) from the Packt Publishing website.

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

Click here for ordering and shipping details.