FIRST EDITION

COVERS ASP.NET 2.0 / 3.5 WITH jQUERY 1.3.2

51 TIPS TRICKS AND RECIPES

SUPROTIM AGARWAL

Microsoft MVP Most Valuable Professional

with jQuery & ASP.NET

51 Tips, Tricks and Recipes

with jQuery and ASP.NET Controls

Suprotim Agarwal

a2z | Knowledge Visuals

51 Tips, Tricks and Recipes with jQuery and ASP.NET Controls

Bulk Sales

A2Z Knowledge Visuals offers bulk discount on this EBook. For more information, please contact

A2Z Knowledge Visuals Pvt. Ltd at

ebooks@a2zknowledgevisuals.com

Cover Image By Minal Agarwal (http://www.saffronstroke.com)

ABOUT THE AUTHOR

Suprotim Agarwal, an ASP.NET Architecture MVP (**Microsoft Most Valuable Professional**), has been developing applications for over 10 years using Microsoft technologies. After working as a software engineer, manager, architect and a mentor, Suprotim now works as a Consultant to a large number of IT Firms, consulting them on implementing enterprise-level Microsoft solutions and other web-based applications. Not known to many, Suprotim also consults websites on Internet Marketing.

Suprotim is also the founder and primary contributor to DotNetCurry, SQLServerCurry and DevCurry and has written over 700 articles and blog posts. When he is not churning out code, he spends his time checking out new gadgets, playing strategic video games and helps his wife (in eating) some deliciously cooked recipes from Foodatarian

ABOUT THE REVIEWERS

Govind Kanshi strives to create efficient solutions for his customers using his experience and ability to dissect complex technologies available in the market. His experience ranges from working across startup, consulting to Fortune 100 firms as well as working in product delivery. He is comfortable creating and managing teams and is also comfortable working in individual capacity. His interests and expertise lies across Databases, Web applications, Virtualization and Performance tuning.

Malcolm Sheridan (MCSD) is an independent contractor who has been developing business line applications for the last decade. He has extensive experience using various development technologies but now concentrates on and favors the .Net framework and ASP.NET specifically. Most recently, Malcolm has been imperative in developing applications for clients in the telecommunications and banking sectors. In his spare time, he enjoys working on his F# and Silverlight.

CREDITS

Author	Suprotim Agarwal
Technical Reviewers	Govind Kanshi, Malcolm Sheridan
Proof Reader	Carol Nadarwalla
Cover Designer	Minal Agarwal
eBook Layout	Minal Agarwal

Blank Page

TABLE OF CONTENTS

DEDICATION

This eBook is dedicated to my parents whose guidance has got me so far, to my brother Raunak for his encouragement and to my beloved wife Minal, who has patiently provided me endless support throughout the eBook, even when I was punching code during weekends and festivals.

ACKNOWLEDGEMENT

It's interesting to see how ideas come to life. My colleague from Microsoft, Govind Kanshi introduced me to jQuery last year. When I shepherded the idea of writing an eBook on jQuery and ASP.NET controls and requested him to be a reviewer, he was all for it. Thank you Govind for your hard work while reviewing this eBook, for your continuous encouragement and for your support and leadership.

A special note of thanks goes to Malcolm Sheridan, not just because he technically reviewed this eBook, but also for the two chapters he contributed to this eBook. He was prompt in providing feedback despite of having sleepless nights because his two year old sweet daughter Livvy, wouldn't let him sleep! Thank you Malcolm for your good advice, continuous encouragement and for your perspective, which helped me add new sections to this eBook, I hadn't considered earlier.

Big thanks to John Resig and his jQuery team for creating a renaissance in client-side scripting. Thanks to my readers at dotnetcurry.com and devcurry.com and to the thousands of budding programmers, who shared their client-side issues on various forums.

Lastly I would like to thank my family, friends, my mentor Subodh and the MVP community to provide feedback and support throughout this project.

"When we consider a project, we really study it — not just the surface idea, but everything about it. And when we go into that new project, we believe in it all the way. We have confidence in our ability to do it right" – Walt Disney

INTRODUCTION

jQuery is a fast, lightweight JavaScript library that is CSS3 compliant and supports many browsers. The jQuery framework is extensible and supports DOM manipulations, CSS, AJAX, Events, Animations and much more.

Being an ASP.NET Microsoft MVP, I have always had the opportunity to have early access to products and explore new web technologies, first hand. During the month of September 2008, it got me all excited when Microsoft officially announced that jQuery will be part of their official development platform. I have been extensively using jQuery with ASP.NET ever since then. This eBook is the result of my practical experience of using jQuery with ASP.NET controls, all in the pursuit of demonstrating techniques to resolve Client-Side programming challenges, quickly.

WHAT DOES THIS EBOOK CONTAIN?

In this no-nonsense, to-the-point eBook, I show you 51 useful recipes of using jQuery with various ASP.NET controls, each recipe dedicated towards solving a particular problem. So be it tackling ClientID's in MasterPages or be it creating stunning UI animations or avoiding enormous amount of code to solve common client-side programming tasks, this eBook shows you all. Each recipe in this eBook is designed to help you to understand, learn and solve challenges associated with ASP.NET Client-Side Development and to provide you with strategies and anecdotes using jQuery that will make you resolve these issues quickly.

Since the ASP.NET framework contains a plethora of controls, in this eBook, I have included only some of the most frequently used ASP.NET controls. These controls are the: TextBox, CheckBox, RadioButton, Panel, DropDownList, GridView, ImageControl, BulletedList and the Hyperlink control.

WHO IS THIS EBOOK FOR?

This eBook is intended for developers who have been using JavaScript in their ASP.NET applications. This eBook aims at making you more productive, by demonstrating how to write lesser client-side code that works cross-browser. These recipes have been written for real-world developers who need instant solutions to common and not-so-common problems. This book assumes you know jQuery basics. However for those who are new to jQuery, check Appendix 'B' of this book to check out some online resources.

IMPORTANT POINTS TO NOTE BEFORE READING THIS EBOOK

jQuery is a JavaScript framework that can enhance user experience considerably. jQuery can be elegantly used to provide quick and responsive messages to the end user, without wasting CPU cycles on the server to process the same information.

However, you should realize that users can disable JavaScript. Besides providing client-side validations, you should 'always' validate user input on the server side. Here are some additional important points to note while reading this eBook

- Each Recipe comes with a 'Browsers Supported' section where I list the browsers and their versions; the recipe has been tested on. Although the code may work in other browser versions too, I do not provide any support for the same.
- I have included a Live Demo inside each Recipe. It would help you read about a recipe and to be able to test it out quickly.
- The Useful Links section contains a list of links that can provide additional information about the jQuery API or explains a technique used in the recipe.
- Whenever I plan to use a jQuery object more than once, I have prefixed jQuery variables with a '$'. Eg: $tb = $(this). This is a useful convention and offers a slight performance advantage.
- I have used e.preventDefault() in a number of examples to prevent postback from occurring
- Although I have demoed the examples without using a MasterPage, most of the examples work in a MasterPage scenario. To make it work in a MasterPage scenario and to avoid Client-Id issues, I have referenced controls as [id$=controlid]. Check Appendix 'C' for a sample.
- I have demonstrated most of the examples using ASP.NET Server controls. You can replace these controls with matching HTML controls as and where desired. Remember that an ASP.NET control gets rendered as HTML.
- For readability sake, jQuery code is kept inside the ASP.NET page. However in a real application, you should always keep scripts and other resources in separate folders for maintainability.
- Although the eBook uses the jQuery library kept in the Scripts folder of the source code; in production applications, I use the jQuery served off Google's Content Delivery Network.
- As far as possible, I have not included server-side code in most of the recipes. The server-side code has been mentioned in Appendix 'C' of this eBook, with references to the Appendix in the recipe, wherever required.
- Most of the examples work in ASP.NET 2.0, however there are a few recipes in the GridView and AJAX section that use LINQ /JSON and may not work in ASP.NET 2.0.
- jQuery with ASP.NET AJAX is not covered in this eBook.
- Your particular problem area may not be exactly suited to the examples illustrated in this eBook; in fact, it's likely that they won't be the same, and you should adjust your use of the information and recommendations, accordingly.
- Also please note that there are a couple of ways to do the same task. In most of the examples shown in this eBook, I have used the core jQuery library to solve a problem. However jQuery has an extensible architecture which makes it easy to create plug-ins. Appendix 'A' is dedicated to

some useful jQuery plug-ins created by the jQuery community which have been tried and tested and are being used in production applications.

BROWSER SUPPORT

Every effort has been made to make the recipes work in the following modern browsers:

- Internet Explorer 7 and 8
- Firefox 3
- Chrome 2
- Safari 4

WHAT YOU WILL NEED TO TRY OUT THE EXAMPLES

- Visual Studio 2008 (express edition supported)
- Modern Web Browsers - IE 7, IE 8, Firefox 3, Chrome 2, Safari 4
- jQuery 1.3.2

CODE CONVENTIONS

This book uses different styles to represent different kinds of information. Here is a list of them:

ASP.NET Markup is represented like this:

```
<asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine" >
</asp:TextBox>
```

HTML Markup is represented like this:

```
<input name="TextBox1" type="text" id="TextBox1" />
```

- represents Notes

- represents Tips and Tricks

SOURCE CODE

The source code has been mailed to you along with the eBook. If for some reason, you did not receive the source code, please contact us at *ebooks@a2zknowledgevisuals.com*

ERRATA AND FEEDBACK

Every possible attempt has been made to achieve complete accuracy of the content in this eBook and related content. However if there is an errata that you would like to point our attention towards, please drop in a mail at *ebooks@a2zknowledgevisuals.com*. If we come across an error after this eBook has been sent to you, we will announce it on the errata page

Smart readers like you come up with important comments that are very valuable for us. So send in your adulation, criticism and everything in between at *suprotimagarwal@a2zknowledgevisuals.com* or at *ebooks@a2zknowledgevisuals.com*

Section 1 – jQuery and the ASP.NET TextBox

Introduction

The ASP.NET TextBox control is used to gather input from users. The TextBox control has a TextMode property that sets the behavior mode of the TextBox control. The TextMode property can accept the following Values:

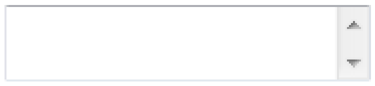
SingleLine – This is the default option and displays a single line input field as shown below

```
<asp:TextBox ID="TextBox1" runat="server" >
</asp:TextBox>
```

The ASP.NET TextBox with TextMode=SingleLine gets rendered in the browser as

```
<input name="TextBox1" type="text" id="TextBox1" />
```

MultiLine – This option displays a multiline input field as shown below



```
<asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine" >
</asp:TextBox>
```

The ASP.NET TextBox with TextMode=MultiLine gets rendered in the browser as

```
<textarea name="TextBox1" rows="2" cols="20" id="TextBox1"></textarea>
```

Password - This option displays a single line input field with masked user input as shown below:

```
<asp:TextBox ID="TextBox1" runat="server" TextMode="Password">
</asp:TextBox>
```

The ASP.NET TextBox with TextMode=Password gets rendered in the browser as

```
<input name="TextBox1" type="password" id="TextBox1" />
```

We will not be covering any recipes with TextMode=Password in this section.

The TextBox control is one of the most frequently used controls in our applications. In this section, I will show you how to use jQuery to solve 12 common client-side scripting challenges while working with the TextBox control.

This section covers the following recipes:

Recipe 1: Access Values from All or Selective TextBoxes using jQuery

Challenge:

You want to access values from ALL or Selective ASP.NET Textboxes on a Button click. The values should be displayed separately on each line. Empty Textboxes should be ignored.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Access Values From Multiple or Selected ASP.NET TextBoxes</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnAll]').click(function(e) {
                e.preventDefault();
                $("#para").text('')
                .append($("input:text").map(function() {
                    return $(this).val() || null;
                }).get().join("<br/>  "));
            });

            $('input[id$=btnSel]').click(function(e) {
                e.preventDefault();
                $("#para").text('')
                .append($("input.selected").map(function() {
                    return $(this).val() || null;
                }).get().join("<br/> "));
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="bigDiv">
        <h2>Access Values From Multiple/Selected ASP.NET TextBoxes</h2><br />
        <asp:TextBox ID="tb1" runat="server" Text="1stTextBox" /><br />
        <asp:TextBox ID="tb2" runat="server" Text="2ndTextBox"/><br />
        <asp:TextBox ID="tb3" runat="server" Text="3rdTextBox"/><br />
        <asp:TextBox ID="tb4" runat="server" Text="1stTextBox-class-selected"
            class="selected" /><br />
        <asp:TextBox ID="tb5" runat="server" Text="4thTextBox"/><br />
        <asp:TextBox ID="tb6" runat="server" Text="2ndTextBox-class-selected"
            class="selected" /><br /><br />
        <asp:Button ID="btnAll" runat="server" Text="Display All"
            ToolTip="Click to display text from all boxes" /><br />
        <asp:Button ID="btnSel" runat="server" Text="Display Selective"
```

```
            ToolTip="Click to display text from boxes with class-selected" />
        <br/>
         Tip: Clicking on the 'Display Selective' button retrieves<br />
        values from only those TextBoxes which have 'class=selected'
        <p id="para"></p>
    </div>
    </form>
</body>
</html>
```

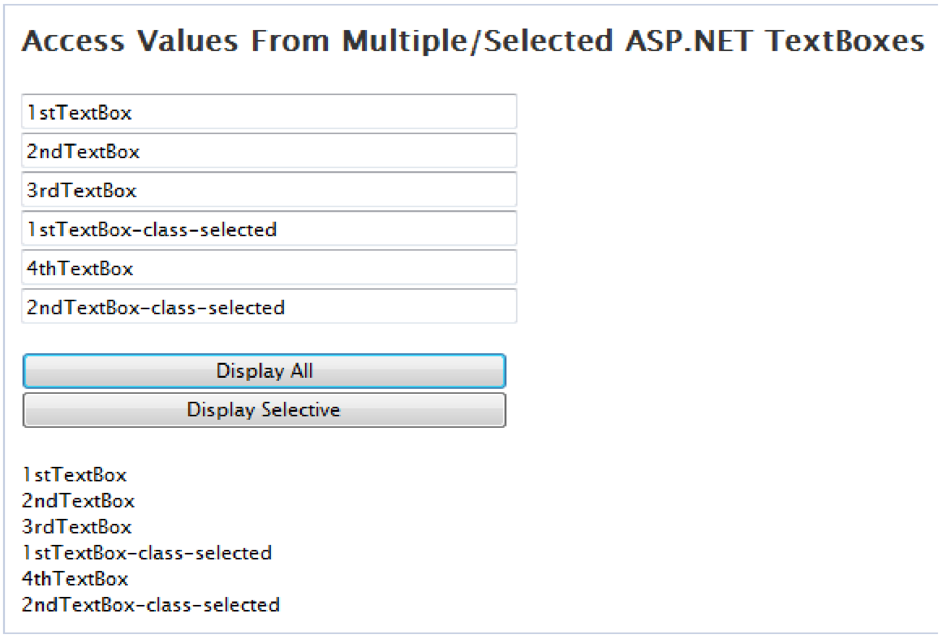
Explanation:

To achieve this requirement, a common technique adopted by developers is to iterate through a set of controls, apply a filter and then collect the results in a new array. *$.map()* is an extremely useful function which saves us from writing these steps. This function transforms an array into another array by using a filter function and then allows us to access individual items of the array and modify them.

In the code shown above, on the first button (btnAll) click, the postback is prevented using *e.preventDefault()*. The content of the paragraph is reset (`$("#para").text('')`) and we then use a Selector to match all input elements of type text (`input:text)`. The *$.map()* method iterates through all the textboxes on the page and invokes a filter function to select the non-empty textboxes. The results are appended to a paragraph (para) and displayed individually using a line break (<br/>).

```
$('input[id$=btnAll]').click(function(e) {
    e.preventDefault();
    $("#para").text('')
    .append($("input:text").map(function() {
        return $(this).val() || null;
    }).get().join("<br/>  "));
});
```

The output will look similar to the screenshot shown below:

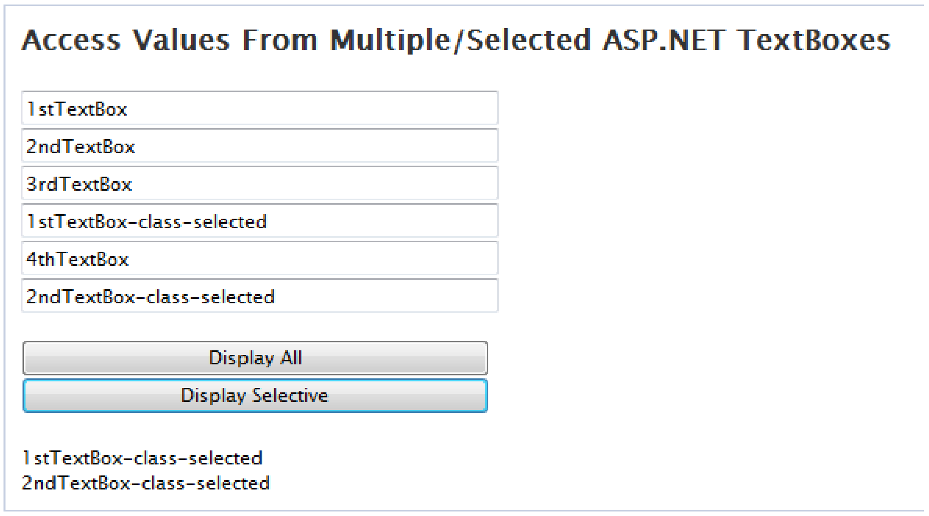

Access Values From Multiple/Selected ASP.NET TextBoxes
1stTextBox
2ndTextBox
3rdTextBox
1stTextBox-class-selected
4thTextBox
2ndTextBox-class-selected
Display All
Display Selective

```
1stTextBox
2ndTextBox
3rdTextBox
1stTextBox-class-selected
4thTextBox
2ndTextBox-class-selected
```

On the second button (btnSel) click, most of the functionality remains the same as described above. The difference is that this time, we use a selector to select all textboxes that have a class of *selected.*

```
$('input[id$=btnSel]').click(function(e) {
    e.preventDefault();
    $("#para").text('')
    .append($("input.selected").map(function() {
        return $(this).val() || null;
    }).get().join("<br/> "));
});
```

The output will be similar to the one shown below:


Access Values From Multiple/Selected ASP.NET TextBoxes
1stTextBox
2ndTextBox
3rdTextBox
1stTextBox-class-selected
4thTextBox
2ndTextBox-class-selected
Display All
Display Selective

```
1stTextBox-class-selected
2ndTextBox-class-selected
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Traversing/map

Recipe 2: Prevent Cut, Copy and Paste Operations in a TextBox using jQuery

Challenge:

Users should be prevented from doing Cut, Copy and Paste operations in an ASP.NET TextBox.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Prevent Cut, Copy and Paste Operations in a TextBox</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
        $('input[id$=tb1]').bind('cut copy paste', function(e) {
                e.preventDefault();
                alert('You cannot ' + e.type + ' text!');
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="bigDiv">
        <h2>Prevent Cut, Copy and Paste Operations in a TextBox</h2>
        <br /><br />
        <asp:Label ID="lblEmail" runat="server"
            Text="Re-Enter your Email Address"></asp:Label>
        <asp:TextBox ID="tb1" runat="server"
            Text="Try copying and pasting text here"
            ToolTip="Try Copy/Cut/Paste in textbox"/>
        <br /><br /><br />
        Tip: Text can only be entered in this box. It cannot be
        copied and pasted from a different textbox
    </div>
    </form>
</body>
</html>
```

Explanation:

This recipe handles a typical requirement where you want the users to 'confirm' an email address and want the user to type it manually, instead of copying and pasting it.

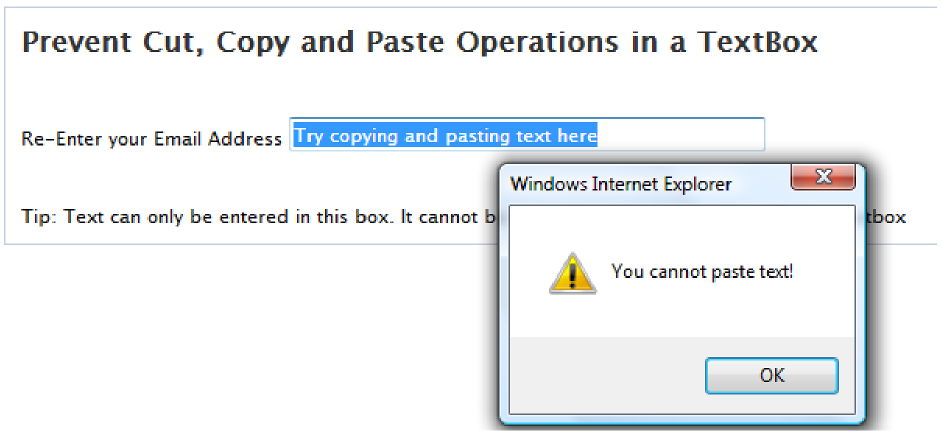
The jQuery *bind()* function binds one or more events to a handler. Observe how convenient it is to list multiple events (cut copy paste) together and bind it to a handler.

```
$('input[id$=tb1]').bind('cut copy paste', function(e) {}
```

If the user performs any of these events on the textbox, the default behavior is prevented using *e.preventDefault()* and the user is alerted. The *e.type* describes the type of event performed.

```
$('input[id$=tb1]').bind('cut copy paste', function(e) {
      e.preventDefault();
      alert('You cannot ' + e.type + ' text!');
});
```

The screenshot shown below represents how the user is informed when he tries to copy text from the textbox.



Prevent Cut, Copy and Paste Operations in a TextBox

Re-Enter your Email Address Try copying and pasting text here

Tip: Text can only be entered in this box. It cannot b … tbox

Text can be edited in this textbox, but cut/copy/paste actions are prevented.

Live Demo

Browsers Supported:

IE7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Events/bind

Recipe 3: Allow Only Alphanumeric Characters in a TextBox using jQuery

Challenge:

Users should be allowed to enter only Alphabets and Numbers in a TextBox. Non-alphanumeric characters should be disallowed. You also want to prevent users from copying and pasting non-alphanumeric characters into the TextBox.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Allow Only Alphanumeric Characters in a TextBox</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input.alpha[$id=tb1]'.bind('keyup blur', function() {
                if (this.value.match(/[^a-zA-Z0-9 ]/g)) {
                    this.value = this.value.replace(/[^a-zA-Z0-9 ]/g, '');
                }
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="bigDiv">
        <h2>Allow Only Alphanumeric Characters in a TextBox</h2>
        <asp:TextBox ID="tb1" runat="server" class="alpha"
            Text="" ToolTip="Try entering Non alphanumeric char"/>
         <br /><br />
         Tip: Examples of some non-alphanumeric characters <br />
         are ~!@~#$%^&* and so on.
    </div>
    </form>
</body>
</html>
```

Explanation:

The code shown above matches a regular expression */[^a-zA-Z0-9]/g* that finds all characters that are not alphabets or numbers. If the user enters a character that is not an alphabet or number, the character is immediately replaced with an empty string. The blur event detects characters when the textbox looses focus whereas the keyup event detects characters after the character has been entered. I have observed people using only the keypress event to handle this requirement, but remember that in IE, keypress does not behave as expected for non-character keys.

As Yehuda Katz says – "*My mantra has always been: keydown/keyup if you want to know the key that was pressed; keypress if you want to know what text was detected*"

If you want to accept only numbers, use */[^0-9]/g* and for accepting only alphabets, use */[^a-zA-Z]/g*. Also note that a user can right click the textbox and choose paste from the context menu. In such cases, the blur event detects the non-alphanumeric characters as soon as the textbox looses focus.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Events/bind

http://docs.jquery.com/Events/keyup

Recipe 4: Make a TextBox ReadOnly at RunTime using jQuery

Challenge:

When the page loads, you want to check if the TextBox has some value in it and make it read-only.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Make TextBoxes ReadOnly at RunTime</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
```

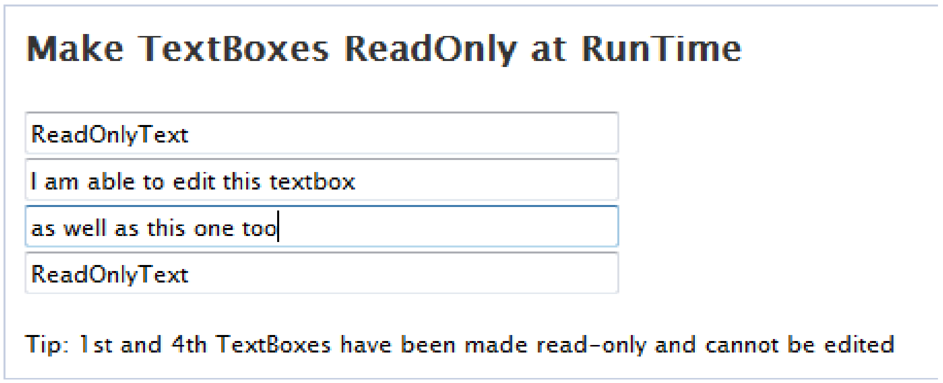
```
        $(function() {
            $('input:text[value!=]').each(function() {
                $(this).attr('readonly', true);
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="bigDiv">
        <h2>Make TextBoxes ReadOnly at RunTime</h2><br />
        <asp:TextBox ID="tb1" runat="server" Text="ReadOnlyText"/><br />
        <asp:TextBox ID="tb2" runat="server" Text=""/><br />
        <asp:TextBox ID="tb3" runat="server" Text=""/><br />
        <asp:TextBox ID="tb4" runat="server" Text="ReadOnlyText" />
        <br /><br />
        Tip: 1st and 4th TextBoxes have been made read-only
        and cannot be edited
    </div>
    </form>
</body>
</html>
```

Explanation:

This is a very common requirement that most developers face on a daily basis. While editing a form, textboxes that have text in it, should be made read-only. The code shown in the example filters the textboxes that have values in it (tb1 and tb4) and applies the 'readonly' attribute to them.

```
$('input:text[value!=]').each(function() {
     $(this).attr('readonly', true);
});
```

When the document loads, the user is able to enter text in the second and third textboxes, but not in the first and fourth, since they are now read-only.



Make TextBoxes ReadOnly at RunTime

ReadOnlyText

I am able to edit this textbox

as well as this one too

ReadOnlyText

Tip: 1st and 4th TextBoxes have been made read-only and cannot be edited

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Core/each

http://docs.jquery.com/Attributes/attr

Recipe 5: Shift Focus to the Next TextBox using the Enter Key using jQuery

Challenge:

You need to tab through the TextBoxes using the Enter Key.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Shift Focus to the Next TextBox when user presses Enter</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input:text:first').focus();
            var $inp = $('input:text');
            $inp.bind('keydown', function(e) {
                var key = (e.keyCode ? e.keyCode : e.charCode);
                if (key == 13) {
                    e.preventDefault();
                    var nxtIdx = $inp.index(this) + 1;
                    $(":input:text:eq(" + nxtIdx + ")").focus();
                }
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Enter text and hit Enter to move to next text box</h2><br />
        <asp:TextBox ID="tb1" runat="server" /><br />
        <asp:TextBox ID="tb2" runat="server" /><br />
        <asp:TextBox ID="tb3" runat="server" /><br />
        <asp:TextBox ID="tb4" runat="server" /><br />
    </div>
    </form>
</body>
</html>
```

Explanation:

This example uses the *bind()* function that binds one or more events to a handler. The key is detected using the keyCode or charCode.

```
var key = (e.keyCode ? e.keyCode : e.charCode);
```

If the key == 13, i.e the Enter key is pressed, the default behavior (postback) is suppressed and instead the index of the current textbox is retrieved. We then add 1 (one) to the index, to shift the focus to the next textbox.

```
if (key == 13) {
        e.preventDefault();
        var nxtIdx = $inp.index(this) + 1;
        $(":input:text:eq(" +nxtIdx + ")").focus();
}
```

This is how the user tabs through the textboxes when the Enter Key is pressed. Some data entry operators find it easier to use the Enter Key to tab through the input controls!

keyCode represents the actual key pressed as a numerical value, whereas *charCode* gives the ASCII/Unicode value of the key press result (for eg: Shift + A). Firefox and other browsers also support *e.which*. IE and Opera do not support *charCode*.

If you plan to add textboxes at runtime and want this code to continue to work with the dynamically added textboxes, use *live()* instead of *bind()*. Remember that in each call, you cannot bind more than one event to *live()*.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Events/bind

http://docs.jquery.com/Events/live

http://www.w3.org/2008/webapps/wiki/KeyCode_and_charCode

Recipe 6: Clone the Contents of a TextBox into another TextBox using jQuery

Challenge:

The contents of the first TextBox have to be cloned into the second TextBox as the user continues typing into the first TextBox. The contents should be cloned even if the user copies and pastes content into the first TextBox.

Solution:

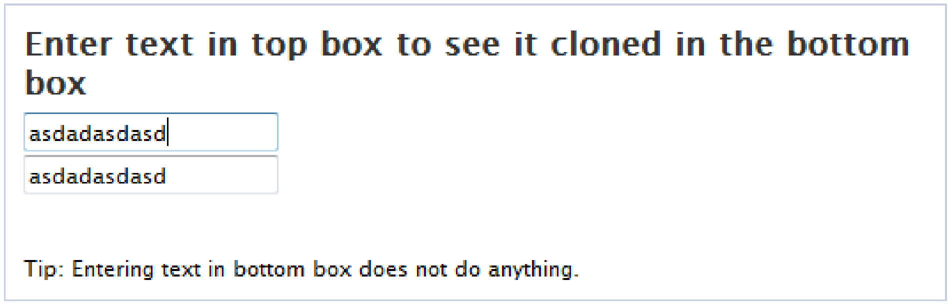
```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Clone the Value of a TextBox as the User Types in it</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=tb1]').keyup(function() {
                var txtClone = $(this).val();
                $('input[id$=tb2]').val(txtClone);
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Enter text in the top textbox to see it cloned in the
        bottom textbox</h2>
        <br />
        <asp:TextBox ID="tb1" runat="server" /><br />
        <asp:TextBox ID="tb2" runat="server" />
        <br /><br /><br />
        Tip: Entering text in bottom box does not do anything.
    </div>
    </form>
</body>
</html>
```

Explanation:

As the user types in the first TextBox, we retrieve the value of the first TextBox and set it to the value of the second TextBox.

```
$('input[id$=tb1]').keyup(function() {
    var txtClone = $(this).val();
    $('input[id$=tb2]').val(txtClone);
});
```

The result is shown here:



Enter text in top box to see it cloned in the bottom box

asdadasdasd

asdadasdasd

Tip: Entering text in bottom box does not do anything.

Since we are capturing the *keyup* event, the contents are cloned into the second textbox, whenever the user pastes text in the first textbox using Ctrl+v. This is however not true when the user right clicks the textbox and chooses paste from the context menu.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Val

Recipe 7: Auto Focus the First TextBox using jQuery

Challenge:

User wants to set focus to the first TextBox, when the page loads. The condition here is that disabled textboxes should be ignored.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Auto Focus the First TextBox</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $("input:text:disabled").css("background-color","gray");
            $("input:text:enabled:first").focus();
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Auto Focus the First TextBox that has Enabled=true</h2>
        <asp:TextBox ID="tb1" runat="server" Enabled="false" /><br />
        <asp:TextBox ID="tb2" runat="server" Enabled="false" /><br />
        <asp:TextBox ID="tb3" runat="server" /><br />
        <asp:TextBox ID="tb4" runat="server" Text="Some Text"/>
        <br />
        <br />
        Tip: The First two textboxes have been disabled.
        You cannot enter text in them.
    </div>
    </form>
</body>
</html>
```

Explanation:

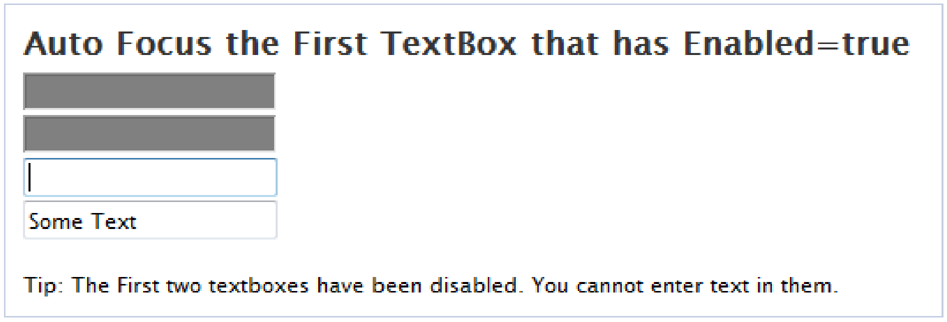
This example is one of the reasons why I just love jQuery Selectors. Look how conveniently we have applied selectors `("input:text:disabled")` to set a background color to the disabled textboxes.

```
$("input:text:disabled").css("background-color","gray");
```

In the next line, we apply another selector (`:enabled:first`) to the TextBox, to set focus on the TextBox that is enabled, which in our case is the third textbox (tb3).

```
$("input:text:enabled:first").focus();
```

The result is as shown below, with the focus on the First TextBox that is not disabled.



Auto Focus the First TextBox that has Enabled=true

Some Text

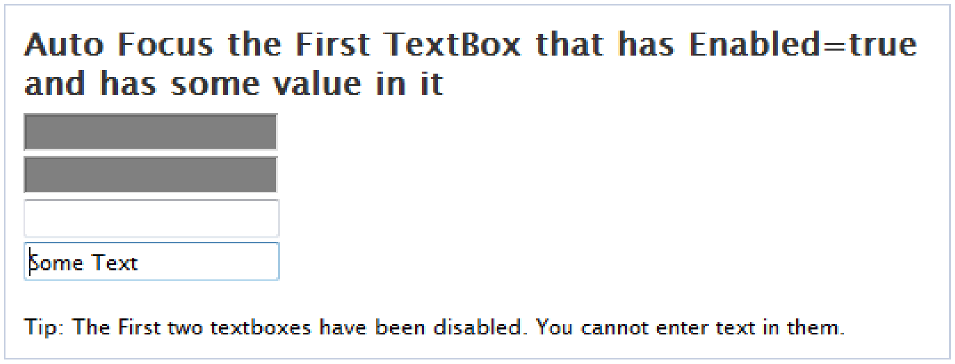
Tip: The First two textboxes have been disabled. You cannot enter text in them.

Live Demo

Similarly, if you want to set focus on the Textbox that is both 'Enabled' and 'has some value in it', use the code shown below, which in our case sets focus to the fourth TextBox (tb4), since this is the only one which qualifies being Enabled and being non-empty.

```
$("input:text[value!=]:enabled:first").focus();
```

So now when you run the application again, the focus is set to the Fourth textbox as shown here:



Auto Focus the First TextBox that has Enabled=true and has some value in it

Some Text

Tip: The First two textboxes have been disabled. You cannot enter text in them.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors

Recipe 8: Click and Highlight Text in a TextBox using jQuery

Challenge:

Text in a TextBox should be automatically selected, when the user clicks inside the TextBox.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Click and Highlight Text</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
       src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=tb1]').click(function() {
                $(this).focus().select();
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
    <h2>Click in the text box to select all the text automatically</h2>
    <br /><br />
        <asp:TextBox ID="tb1" runat="server" Text="Click Here to Select"/>
        <br /><br />
        Tip: If you do not click and just start typing into textbox,
        the effect does not show up
    </div>
    </form>
</body>
</html>
```

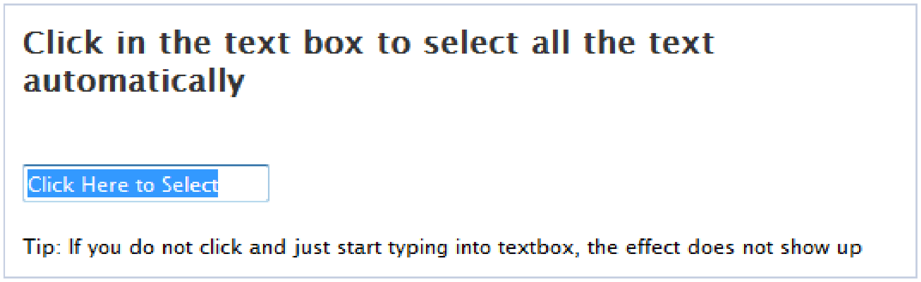
Explanation:

The technique shown over here highlights text by chaining jQuery methods, thereby keeping the code concise.

```
$(this).focus().select();
```

The highlight feature could be used in a scenario where this is a frequent requirement to copy and paste text. The technique shown here saves the user an extra step to first select the text and then copy.

Run the application and click inside the TextBox. The text automatically gets selected. The result is shown here:


Click in the text box to select all the text automatically

Click Here to Select

Tip: If you do not click and just start typing into textbox, the effect does not show up

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://ejohn.org/blog/ultra-chaining-with-jquery/

Recipe 9: Limit Number Of Characters In a Multiline TextBox using jQuery

Challenge:

You want to restrict the user from entering more than 50 characters in an ASP.NET Multiline TextBox. The ASP.NET Multiline TextBox ignores the MaxLength property.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Limit Characters in a Multiline TextBox</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            var limit = 50;

            $('textarea[id$=tb1]').keyup(function() {
                var len = $(this).val().length;
                if (len > limit) {
                    this.value = this.value.substring(0, limit);
                }
                $('#charLeft').text(limit - len + " characters left");
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Type into this textbox which accepts 50 characters overall</h2>
        <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine"/><br />
        <span id="charLeft"></span>
    </div>
    </form>
</body>
</html>
```

Explanation:

In this example, we first capture the keyup event and calculate the number of characters in the textbox.

```
$('textarea[$id=tb1]').keyup(function() {
    var len = $(this).val().length;
```

If the character exceeds the limit of the textbox (50 characters), the additional characters entered by the user is disallowed.

```
if (len > limit) {
    this.value = this.value.substring(0, limit);
}
```

Observe how we are selecting the control using *textarea* `$('textarea[$id=tb1]')`. This is because the ASP.NET Multiline TextBox renders as a *textarea*.

Live Demo

Recommended Approach

Although the solution given above restricts the user from entering more than 50 characters, this behavior can be confusing for users who may not realize that they have reached the TextBox limit. Instead of disallowing extra characters, a slick way of handling this requirement would be to visually indicate to the user when the textbox limit has been exceeded. Then before submitting the form, give the user a chance to remove the extra text. The code shown below changes the background color of the textbox to red when the textbox limit is exceeded. The user is also prevented from submitting the form.

Here's the recommended solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Limit Characters in a Multiline TextBox</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            var limit = 50;
            var $tb = $('textarea[id$=tb1]');
            $tb.keyup(function() {
                var len = $(this).val().length;
                if (len > limit) {
                    $(this).addClass('exceeded');
                    $('#spn').text(len - limit + " characters exceeded");
                }
                else {
                    $(this).removeClass('exceeded');
                    $('#spn').text(limit - len + " characters left");
                }
            });

            $('input[id$=btnSubmit]').click(function(e) {
                var len = $tb.val().length;
                if (len > limit) {
                    e.preventDefault();
                }
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Type into this textbox which accepts 50 characters overall</h2>
        <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine"/><br />
        (This textbox accepts only 50 characters) <br />
        <span id="spn"></span>  <br />
        <asp:Button ID="btnSubmit" runat="server" Text="Submit"/>
        <span id="error"></span>
        <br /><br />
        Tip: Clicking on the Submit button when number of characters are
        less than 50, results in a postback to same page. If you exceed 50
        characters, the exceeded characters are printed below the textbox
        and a postback is prevented.
    </div>
    </form>
</body>
</html>
```

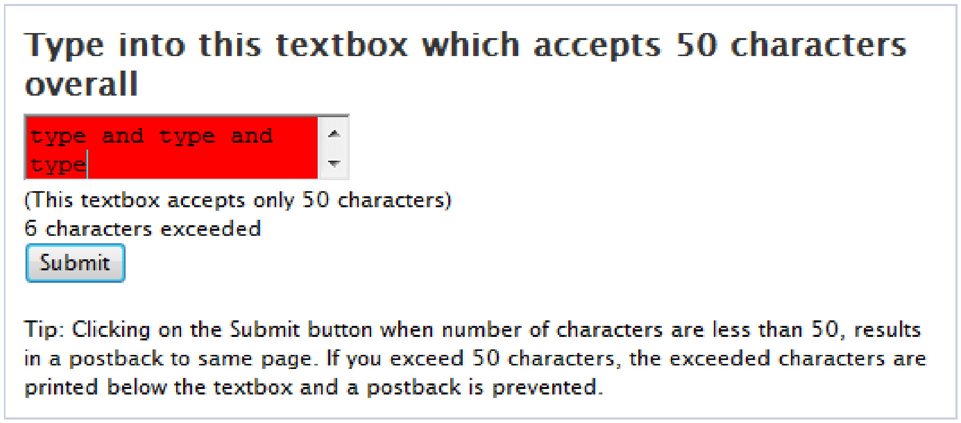
When the number of characters exceeds the textbox limit, we add the *exceeded* class to the textbox, which turns the textbox background to red, indicating the user that the limit has been exceeded.

```
if (len > limit) {
         $(this).addClass('exceeded');
```

The exceeded class declared in Demos.css looks like this

```
.exceeded
{
        background-color:red;
}
```

The result is shown here:



Type into this textbox which accepts 50 characters overall

```
type and type and
type
```

(This textbox accepts only 50 characters)
6 characters exceeded

Tip: Clicking on the Submit button when number of characters are less than 50, results in a postback to same page. If you exceed 50 characters, the exceeded characters are printed below the textbox and a postback is prevented.

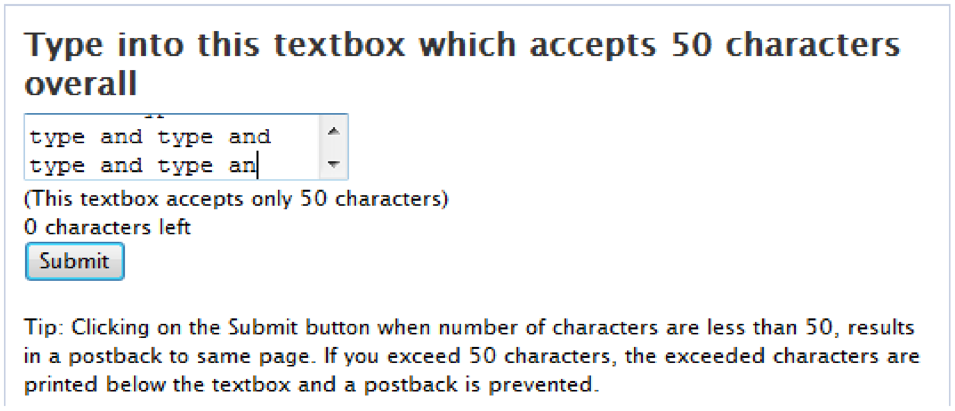
When the user tries and submits the form now, the length of the TextBox is calculated and the user is prevented from submitting the form, since the textbox now exceeds the permissible length.

```
$('input[id$=btnSubmit]').click(function(e) {
    var len = $tb.val().length;
    if (len > limit) {
        e.preventDefault();
    }
});
```

When the user removes the extra text, the *exceeded* class is removed.

```
$(this).removeClass('exceeded');
```

The result is shown below. The form can now be submitted:



Type into this textbox which accepts 50 characters overall

```
type and type and
type and type an
```

(This textbox accepts only 50 characters)
0 characters left

Tip: Clicking on the Submit button when number of characters are less than 50, results in a postback to same page. If you exceed 50 characters, the exceeded characters are printed below the textbox and a postback is prevented.

Check Recipe 3 to see how to use *bind()* to handle mouse paste scenario.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Attributes/addClass

http://docs.jquery.com/Attributes/removeClass

Recipe 10: AutoScroll a Multiline TextBox using jQuery

Challenge:

You want to autoscroll a Multiline TextBox both upwards and downwards.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>AutoScroll a Multiline TextBox</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
   <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $tb = $('textarea[id$=tb1]');

            $('input[id$=btnScroll]').toggle(
            function(e) {
                e.preventDefault();
                scrollArea($tb, $tb[0].scrollHeight);
            },
            function(e) {
                e.preventDefault();
                scrollArea($tb, 0);
            });
        });

        function scrollArea(ctrl, ht) {
            ctrl.animate({ scrollTop: ht }, 1000);
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Scroll the box contents by clicking on the Button</h2>
        <br /><br />
        <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine"
        Text="Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum"
        Rows="5"/>
        <br />
        <asp:Button ID="btnScroll" runat="server" Text="Scroll"/>
        <br /><br />
        Tip: The Text can be scrolled both downwards and upwards.
    </div>
    </form>
</body>
</html>
```

Explanation:

When the user clicks on the button (btnScroll), we toggle the click behavior. On the first click, we cancel the postback by using `e.preventDefault()` and then call a function called scrollArea() passing in the textarea and the scrollHeight. The code `$tb[0].scrollHeight` is for scrolling downwards

```
e.preventDefault();
scrollArea($tb, $tb[0].scrollHeight);
```

When the user clicks the button (btnScroll) again, the postback is cancelled and the scrollHeight is set to 0 (zero). This is done for scrolling upwards.

```
e.preventDefault();
scrollArea($tb, 0);
```

The scrollArea() function accepts the textarea that is to be scrolled as well as the scrollHeight. We then animate the scrollTop property to scroll upwards/downwards depending on the height parameter. The duration of the animation is set to 1000 milliseconds which provides a smooth scrolling effect

```
function scrollArea(ctrl, ht) {
    ctrl.animate({ scrollTop: ht }, 1000);
}
```

The code here assumes that you are not scrolling the textarea manually and then clicking the Scroll button.

Live Demo

Browsers Supported:

IE 7, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Effects/animate#paramsoptions

http://docs.jquery.com/Effects/toggle

Recipe 11: Synchronize Scrolling of Two Multiline TextBoxes using jQuery

Challenge:

Synchronize scrolling of two Multiline TextBoxes.

Solution:

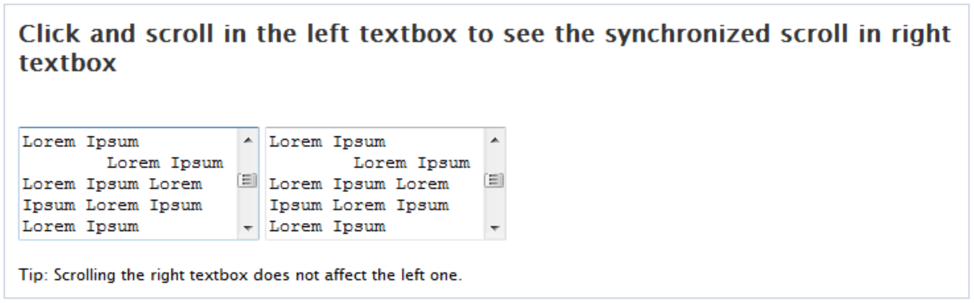
```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Sync two Multiline TextBoxes</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
   <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $tb1 = $('textarea[id$=tb1]');
            var $tb2 = $('textarea[id$=tb2]');
            $tb1.scroll(function() {
                $tb2.scrollTop($tb1.scrollTop());
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="bigDiv">
        <h2>Click and scroll in the left textbox to see the
        synchronized scroll in right textbox</h2><br /><br />
        <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine"
        Text="Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum"
        Tooltip="Click on scrollbar to see the scroll on right box"
        Rows="5"/>
        <asp:TextBox ID="tb2" runat="server" TextMode="MultiLine"
        Text="Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
        Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum"
        Rows="5"/>
        <br /><br />
        Tip: Scrolling the right textbox does not affect the left one.
    </div>
    </form>
</body>
</html>
```

Explanation:

The technique shown above synchronizes the scrolling of TextBox (tb2) with TextBox (tb1). As the user scrolls through the contents of tb1, tb2 automatically scrolls to keep up with the display of tb1.

```
$tb1.scroll(function() {
    $tb2.scrollTop($tb1.scrollTop());
});
```

As shown in the screenshot below, after scrolling 4 lines of the first textbox, the second textbox scrolls automatically to keep up with the first textbox scrolling:



Click and scroll in the left textbox to see the synchronized scroll in right textbox

```
Lorem Ipsum                Lorem Ipsum
        Lorem Ipsum                Lorem Ipsum
Lorem Ipsum Lorem          Lorem Ipsum Lorem
Ipsum Lorem Ipsum          Ipsum Lorem Ipsum
Lorem Ipsum                Lorem Ipsum
```

Tip: Scrolling the right textbox does not affect the left one.

This example is well suited for a requirement when two versions of the same document are to be compared with each other and the user need not have to explicitly scroll through both the versions to compare them.

The code works when the user scrolls using the keyboard or mouse.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Events/scroll

Recipe 12: Create a TextBox Watermark Effect using jQuery

Challenge:

You want to create a watermark effect on your TextBox and display instructions to users, without taking up screen space.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>TextBox WaterMark</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
     <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {

            $(".water").each(function() {
                $tb = $(this);
                if ($tb.val() != this.title) {
                    $tb.removeClass("water");
                }
            });

            $(".water").focus(function() {
                $tb = $(this);
                if ($tb.val() == this.title) {
                    $tb.val("");
                    $tb.removeClass("water");
                }
            });

            $(".water").blur(function() {
                $tb = $(this);
                if ($.trim($tb.val()) == "") {
                    $tb.val(this.title);
                    $tb.addClass("water");
                }
            });
        });

    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
     <h2>TextBox Watermark Demonstration</h2>    <br />
        <asp:TextBox ID="txtFNm" class="water" Text="Type your First Name"
        Tooltip="Type your First Name" runat="server"></asp:TextBox><br />
        <asp:TextBox ID="txtLNm" class="water" Text="Type your Last Name"
        Tooltip="Type your Last Name" runat="server"></asp:TextBox>
         <br /><br />
        <asp:Button ID="btnSubmit" runat="server" Text="Submit" />
        <br /><br />
        Tip: Click on the TextBox to start typing. The watermark
        text disappears.
    </div>
    </form>
</body>
</html>
```

Explanation:

The code shown above adds the “watermark” behavior to controls marked with the ‘class=water’ attribute. When the user loads the page, a watermarked textbox displays a message to the user. As soon as the watermarked textbox receives focus and the user types some text, the watermark goes away. This technique is a great space saver as you can use it to provide instructions to the user, without using extra controls that take up valuable space on your form.

The ‘Tooltip’ attribute applied to the textbox is crucial to this example. The ‘Tooltip’ gets rendered as ‘title’. Observe the code, as we use this ‘title’ property to compare it to the textbox value and remove the watermark css, when the textbox control gains focus

```
$(".water").focus(function() {
    $tb = $(this);
    if ($tb.val() == this.title) {
        $tb.val("");
        $tb.removeClass("water");
    }
});
```

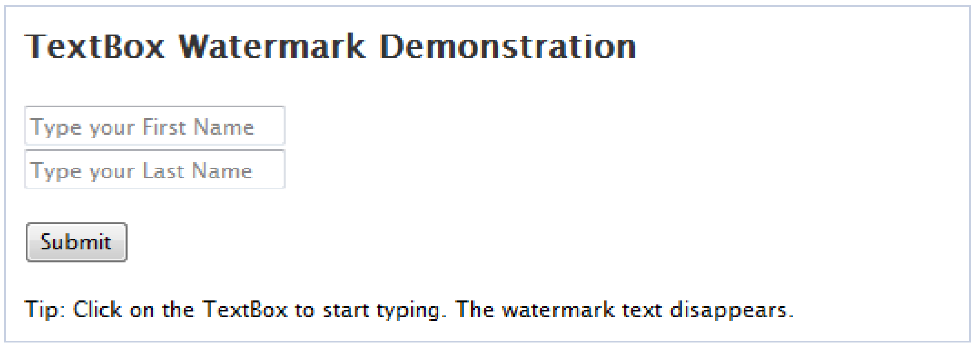
Similarly when the user moves focus to a different control without entering a value in the textbox, we add the watermark css again.

```
$(".water").blur(function() {
    $tb = $(this);
    if ($.trim($(this).val()) == "") {
        $tb.val(this.title);
        $tb.addClass("water");
    }
});
```

The water class declared in Demos.css looks like this:

```
.water
{
     font-family: Tahoma, Arial, sans-serif;
     font-size:75%;
     color:gray;
}
```

When the page loads for the first time, the watermark is visible as shown here:



TextBox Watermark Demonstration

Type your First Name

Type your Last Name

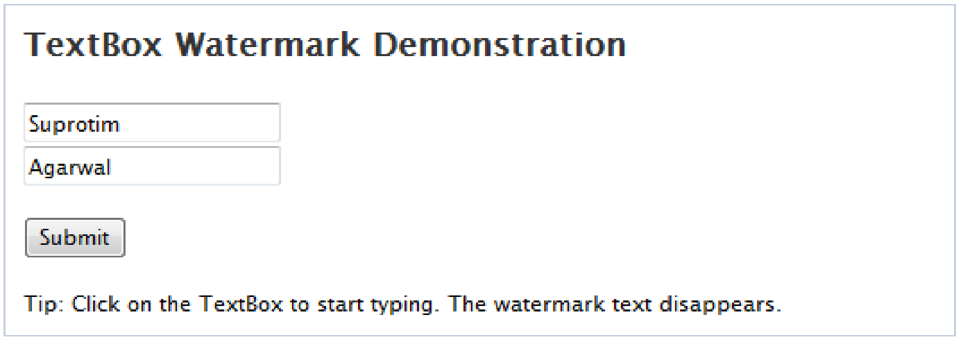
Submit

Tip: Click on the TextBox to start typing. The watermark text disappears.

When the user enters the First/Last Name and submits the form, the watermark behavior is no more needed to be displayed. This is achieved by comparing the 'title' with the 'value' of the textbox. If the 'value' does not match the 'title', this indicates that the user has entered some value in the textboxes and submitted the form. So in this case we remove the watermark appearance.

```
$(".water").each(function() {
    $tb = $(this);
    if ($tb.val() != this.title) {
        $tb.removeClass("water");
    }
});
```

After the user enters the details and submits the form, the result is similar to the one shown here, without the watermark:


TextBox Watermark Demonstration

Suprotim

Agarwal

Tip: Click on the TextBox to start typing. The watermark text disappears.

Thanks to Arnold Matusz for sharing the tip about the tooltip property.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

Create a TextBoxWatermark Extender Functionality using ASP.NET and jQuery

http://docs.jquery.com/Events/focus

http://docs.jquery.com/Events/blur#fn

Section 2: jQuery and the ASP.NET CheckBox

Introduction

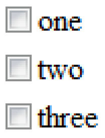
The ASP.NET CheckBox control displays a checkbox in your web forms and allows users to select a true or false condition. The CheckBox control is declared like this on your WebForm:

```
<asp:CheckBox ID="CheckBox1" runat="server" />
```

and gets rendered in the browser as:

```
<input id="CheckBox1" type="checkbox" name="CheckBox1" />
```

Similarly an ASP.NET CheckBoxList control is a list of check boxes that allows you to select multiple checkboxes in a group. The CheckBoxList looks identical to the image shown below:



The CheckBoxList is declared like this:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server">
    <asp:ListItem Text="one"></asp:ListItem>
    <asp:ListItem Text="two"></asp:ListItem>
    <asp:ListItem Text="three"></asp:ListItem>
</asp:CheckBoxList>
```

and gets rendered in the browser as

```
<table id="CheckBoxList1" border="0">
<tr>
<td><input id="CheckBoxList1_0" type="checkbox" name="CheckBoxList1$0" /><label
for="CheckBoxList1_0">one</label></td>
</tr><tr>
<td><input id="CheckBoxList1_1" type="checkbox" name="CheckBoxList1$1" /><label
for="CheckBoxList1_1">two</label></td>
</tr><tr>
<td><input id="CheckBoxList1_2" type="checkbox" name="CheckBoxList1$2" /><label
for="CheckBoxList1_2">three</label></td>
</tr>
</table>
```

In this section, I will show you how to use jQuery to solve five common client-side scripting challenges faced while working with the CheckBox and CheckBoxList control. This section will contain the following recipes:

Recipe 13: Make the CheckBox a Required Field on your WebForm using jQuery

Challenge:

The user should check at least one of the checkboxes, before submitting the form.

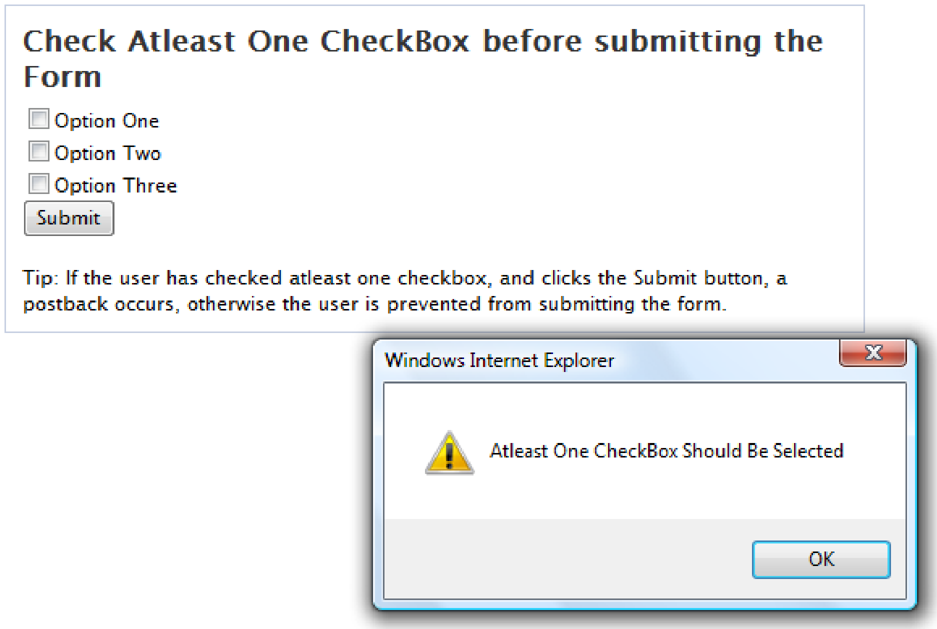
Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Make Sure Atleast One CheckBox is Checked</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnSubmit]').click(function(e) {
                var checked = $(':checkbox:checked').length;
                if (checked == 0) {
                    alert('Atleast One CheckBox Should Be Selected');
                    e.preventDefault();
                }
                else {
                    alert('Postback occured');
                }
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Check Atleast One CheckBox before submitting the Form</h2>
        <asp:CheckBox ID="cb1" runat="server" Text="Option One" /><br />
        <asp:CheckBox ID="cb2" runat="server" Text="Option Two" /><br />
        <asp:CheckBox ID="cb3" runat="server" Text="Option Three" />
        <br /><br />
        <asp:Button ID="btnSubmit" runat="server" Text="Submit"
        ToolTip="Select atleast one checkbox before clicking" />
        <br /><br />
        Tip: If the user has checked atleast one checkbox, and clicks the
        Submit button, a postback occurs, otherwise the user is prevented
        from submitting the form.
    </div>
    </form>
</body>
</html>
```

Explanation:

This code is straightforward. When the button is clicked, we first count the checkboxes that are checked using `$(':checkbox:checked').length`. If the length is 0 (zero), we alert the user and prevent the user from submitting the form using `e.preventDefault();`

The screenshot shown below alerts the user when the user tries submitting the form without selecting a checkbox.



Check Atleast One CheckBox before submitting the Form

☐ Option One
☐ Option Two
☐ Option Three

Tip: If the user has checked atleast one checkbox, and clicks the Submit button, a postback occurs, otherwise the user is prevented from submitting the form.

Windows Internet Explorer

Atleast One CheckBox Should Be Selected

Once the user selects a checkbox, the user is allowed to submit the form.

You should always provide Server-side validations besides these custom client-side validations.

Ideally I should be using a CheckBoxList control over here, but I wanted to demonstrate this example using individual checkboxes.

[Live Demo]

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/checked

Recipe 14: Check/Uncheck All Items in a CheckBoxList using jQuery

Challenge:

You want to check/uncheck all the checkboxes at once, in the easiest possible way.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Check/Uncheck All CheckBoxes At Once</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $chkBox = $("input:checkbox[id$=chkAll]");
            var $tblChkBox = $("table.chk input:checkbox");

            $chkBox.click(function() {
                $tblChkBox
                 .attr('checked', $chkBox
                 .is(':checked'));
            });

            // Unchecks chkAll when a checked CheckBox in cbList
            // is unchecked
            $tblChkBox.click(
            function(e) {
                if (!$(this)[0].checked) {
                    $chkBox.attr("checked", false);
                }
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Check/Uncheck all checkboxes at once using the
            'Do All' checkbox</h2><br />
        <asp:CheckBox ID="chkAll" runat="server"
            Text="Do All"
            ToolTip="Click here to check/uncheck all checkboxes at once"/>
            <br /><hr />
        <asp:CheckBoxList ID="cbList" runat="server" class="chk">
            <asp:ListItem Text="Option One" Value="One" />
            <asp:ListItem Text="Option Two" Value="Two" />
            <asp:ListItem Text="Option Three" Value="Three" />
            <asp:ListItem Text="Option Four" Value="Four" />
            <asp:ListItem Text="Option Five" Value="Five" />
        </asp:CheckBoxList>
        <br /><br />
       Tip: Clicking on any of the individual 'checked' checkboxes,
       unchecks the 'Do All' checkbox.
    </div>
    </form>
</body>
</html>
```

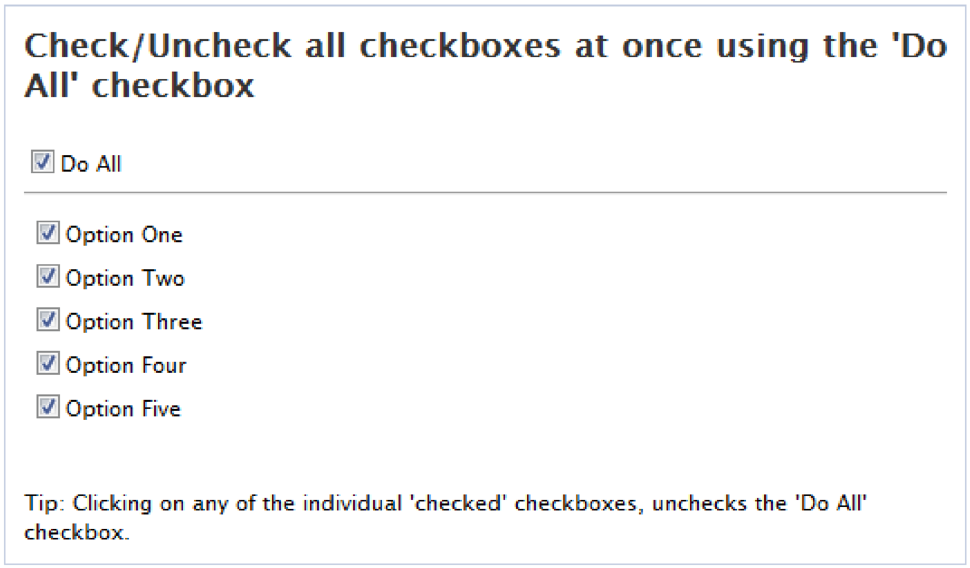
Explanation:

In the Section 2 introduction, we have seen how the CheckBoxList renders as a group of input elements of the type checkbox, inside a HTML table.

In this example, when the user clicks the checkbox (Do All), we reference all input elements of the checkbox type inside the table, and set the 'checked' value of the referenced elements to true or false, based on the checked value of the 'Do All' checkbox.

```
$chkBox.click(function() {
     $tblChkBox
       .attr('checked', $chkBox
       .is(':checked'));
});
```

The output is as shown below:


Check/Uncheck all checkboxes at once using the 'Do All' checkbox

☑ Do All

☑ Option One
☑ Option Two
☑ Option Three
☑ Option Four
☑ Option Five

Tip: Clicking on any of the individual 'checked' checkboxes, unchecks the 'Do All' checkbox.

When the 'Do All' checkbox is unchecked, the CheckBoxList is unchecked too.

The code also takes into account a scenario when a user unchecks any of the individual checkboxes in the CheckBoxList, the 'Do All' checkbox is also unchecked. This is achieved using the following code:

```
$tblChkBox.click(
function(e) {
    if (!$(this)[0].checked) {
        $chkBox.attr("checked", false);
    }
});
```

Thanks to Tim Hobbs for sharing this tip.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

Check/Uncheck all Items in an ASP.NET CheckBox List using jQuery

http://docs.jquery.com/Selectors/checked

Recipe 15: Use a CheckBox to Conditionally Enable or Disable controls using jQuery

Challenge:

You want to use the CheckBox control to enable/disable other controls on the page

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Enable Disable Controls using CheckBox</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $btn = $(':submit[id$=btnSubmit]');
            var $chk = $('input:checkbox[id$=cb1]');

            // check on page load
            checkChecked($chk);

            $chk.click(function() {
                checkChecked($chk);
            });
            function checkChecked(chkBox) {
                if (chkBox.is(":checked")) {
                    $btn.removeAttr('disabled');
                } else {
                    $btn.attr('disabled', 'disabled');
                }
            }
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the Checkbox to Enable the Button control</h2>
        <asp:CheckBox ID="cb1" runat="server"
        Text="Check To Toggle the Enabled status of the Button"
        Tooltip="Click Here To Enable/Disabled the CheckBox"/><br /><br />
        <asp:Button ID="btnSubmit" runat="server" Text="Submit" />
    </div>
    </form>
</body>
</html>
```

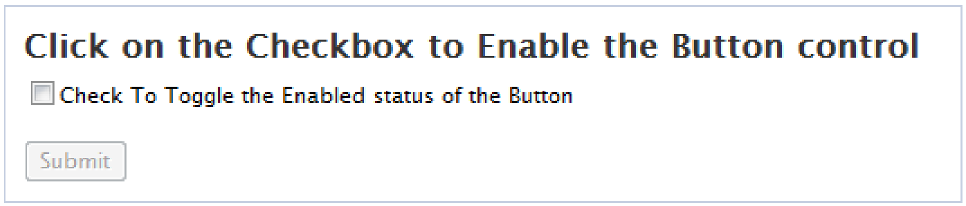
Explanation:

In Recipe 13, we saw how to make the checkbox a required field. The user needed to check at least one of the checkboxes, before submitting the form. In that example, if the user submitted the form without checking any of the checkbox, an alert would popup and the user was prevented from submitting the form.

In this example, we will change our approach and make the checkbox a required field; by enabling the Submit button only after the user has checked the checkbox. When the page loads, we first check if the checkbox is 'checked' and disable the button by using `.attr('disabled', 'disabled');`

```
function checkChecked(chkBox) {
    if (chkBox.is(":checked")) {
        $btn.removeAttr('disabled');
    } else {
        $btn.attr('disabled', 'disabled');
    }
}
```

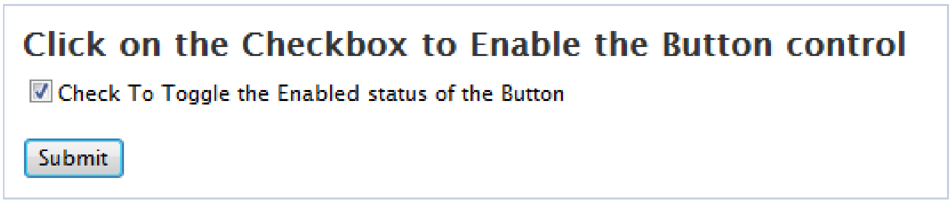
The button is in a disabled state when the page loads, as shown below:


Click on the Checkbox to Enable the Button control
Check To Toggle the Enabled status of the Button
Submit

The button is enabled when the user checks the checkbox as shown in the code below:

```
if (chkBox.is(":checked")) {
    $btn.removeAttr('disabled');
}
```

The output is shown below with the checkbox checked and the button enabled. The user can now submit the form.


Click on the Checkbox to Enable the Button control
☑ Check To Toggle the Enabled status of the Button
Submit

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Attributes/attr#keyvalue

http://docs.jquery.com/Attributes/removeAttr

Recipe 16: Instantly Display the CheckBox Checked Items using jQuery

Challenge:

Display a list of checked items, as and when they are selected by the user.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>List the checked Items</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var arr;
            var checked = $('input:checkbox').click(function(e) {
                trackCheckedBoxes();
            });

            function trackCheckedBoxes() {
                var $checked = $(':checkbox:checked');
                arr = [];
                $checked.each(function() {
                    // for html controls use $(this).val()
                    arr.push($(this).next().text());
                });
                $('#para').html(arr.join('<br/>'));
            }
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Dynamically display the checked options</h2>
        <br />
        <asp:CheckBox ID="cb1" runat="server" Text="Option One" /><br />
        <asp:CheckBox ID="cb2" runat="server" Text="Option Two" /><br />
        <asp:CheckBox ID="cb3" runat="server" Text="Option Three" /><br />
        <br />
        Tip: Choosing the CheckBoxes displays the CheckBox text below
        <br /><br />
        <p id="para"></p>

    </div>
    </form>
</body>
</html>
```

Explanation:

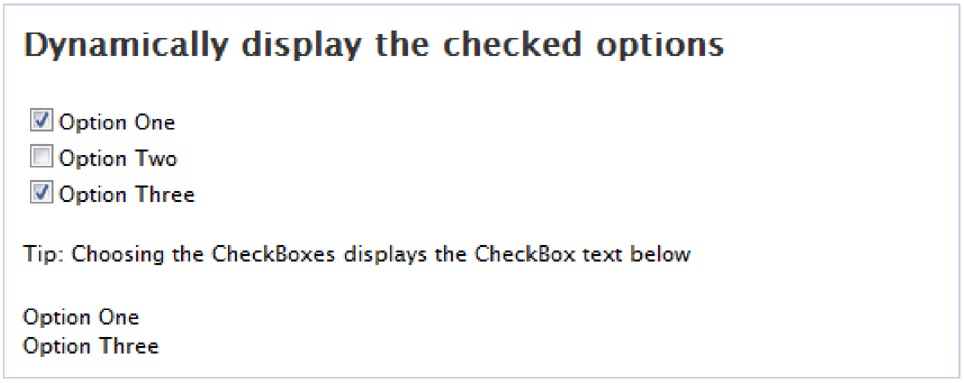
When a checkbox is clicked, a variable ‘checked’ keeps track of the checked checkboxes.

```
var checked = $('input:checkbox').click(function(e) {
    trackCheckedBoxes();
});
```

We then iterate the ‘checked’ collection using *$().each()* and add the text of the checkbox to an array (arr). The array is then displayed in a paragraph.

```
function trackCheckedBoxes() {
    var $checked = $(':checkbox:checked');
    arr = [];
    $checked.each(function() {
        // for html controls use $(this).val()
        arr.push($(this).next().text());
    });
    $('#para').html(arr.join('<br/>'));
}
```

Items are displayed in the order in which the checkboxes are displayed on the page:


Dynamically display the checked options

☑ Option One
☐ Option Two
☑ Option Three

Tip: Choosing the CheckBoxes displays the CheckBox text below

Option One
Option Three

An alternative way of achieving this requirement is to use the .serialize() and then loop through the contents and list the checked checkboxes. This technique is not covered here.

I have seen developers using the *change()* event instead of the click(). My advice is to avoid the change event in this case, as in a checkbox scenario; IE fires the event only when the control focus is lost. This behavior could be misleading.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/checked

Recipe 17: Performing Calculations with Check Boxes using jQuery

Challenge:

A requirement demands that users are asked to select services they would like to avail. Each service has a rate displayed. The total service cost should be automatically calculated, as the services are selected.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Calculation with Checked Items</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
```

```
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var total;
            var checked = $('input:checkbox').click(function(e) {
                calculateSum();
            });

            function calculateSum() {
                var $checked = $(':checkbox:checked');
                total = 0.0;
                $checked.each(function() {
                    total += parseFloat($(this).next().text());
                });
                $('#tot').text("Your Total Amount Is: " + total.toFixed(2));
            }
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the checkboxes to total up the values</h2>
         <br /> <br />
        Choose the Add-Ons (in USD): <br />
        <table>
            <tr>
                <td>
                    500 MB Disk Space m/o</td>
                <td>
                    <asp:CheckBox ID="cb1" runat="server" Text="5.23" />
                </td>
            </tr>
            <tr>
                <td>
                    5 GB Bandwidth m/o</td>
                <td>
                    <asp:CheckBox ID="cb2" runat="server" Text="2.20" />
                </td>
            </tr>
            <tr>
                <td>
                    Additional Database  m/o</td>
                <td>
                    <asp:CheckBox ID="cb3" runat="server" Text="8.54" />
                </td>
            </tr>
            <tr>
                <td>
                    Domain Registration</td>
                <td>
                    <asp:CheckBox ID="cb4" runat="server" Text="15.14" />
                </td>
            </tr>
        </table>
        <br />
        <p id="tot"></p>
     </div>
    </form>
</body>
</html>
```

Explanation:

This example builds upon Recipe 16 and shows a practical usage of it. In this example, when a checkbox is clicked, a variable ‘checked’ keeps track of the checked checkboxes.

```
var checked = $('input:checkbox').click(function(e) {
    calculateSum();
});
```

We then iterate the ‘checked’ collection using *$().each()* and add the text of the checkbox to a variable ‘total’.

```
var $checked = $(':checkbox:checked');
total = 0.0;
$checked.each(function() {
    total += parseFloat($(this).next().text());
});
```

Observe how we are accessing the value of a checkbox using `$(this).next().text();`

This is because the checkbox gets rendered like this:

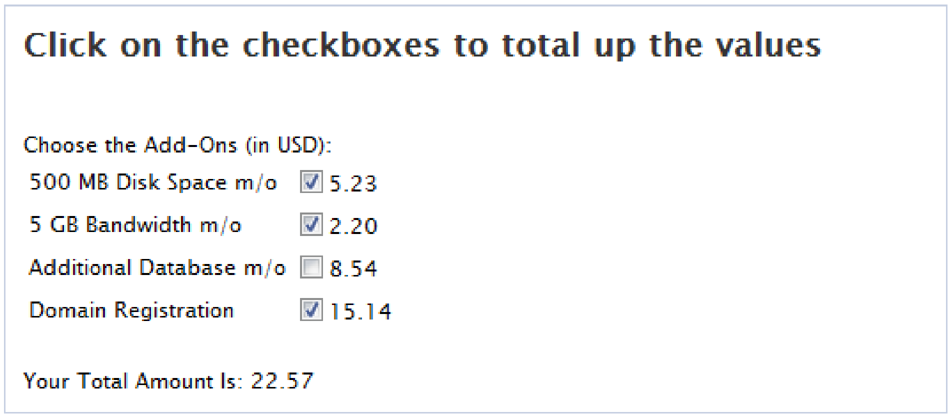
```
<td>
<input id="Checkbox1" type="checkbox" name="cb1" />
<label for="cb1">5.23</label>
</td>
```

To access the value, which is inside the label, we use *next()* which finds the very next sibling of each checkbox, which in our case is the label control. The *parseFloat( )* parses strings into numbers.

The result is displayed in a paragraph (tot) using the *toFixed()* method, which rounds the result to the specified number of decimal places, in our case 2.

```
$('#tot').text("Your Total Amount Is: " + total.toFixed(2));
```

The output is shown here:



Click on the checkboxes to total up the values

Choose the Add-Ons (in USD):

500 MB Disk Space m/o	✓	5.23
5 GB Bandwidth m/o	✓	2.20
Additional Database m/o	☐	8.54
Domain Registration	✓	15.14

Your Total Amount Is: 22.57

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Core/each

http://docs.jquery.com/Traversing/next#expr

Section 3: jQuery and the ASP.NET RadioButton

Introduction

The ASP.NET RadioButton control displays a radio button in your WebForm.

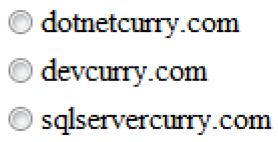
The RadioButton control is declared like this in your WebForm:

```
<asp:RadioButton ID="RadioButton1" Text="Option1"
runat="server" />
```

and gets rendered in the browser as:

```
<input id="RadioButton1" type="radio" name="RadioButton1" value="RadioButton1"
/><label for="RadioButton1">Option1</label>
```

Similarly an ASP.NET RadioButtonList control is used to create a group of radio buttons. The RadioButtonList looks identical to the image shown below:



The RadioButtonList is declared like this in your WebForm:

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server">
    <asp:ListItem Text="dotnetcurry.com" Value="0"></asp:ListItem>
    <asp:ListItem Text="devcurry.com" Value="1"></asp:ListItem>
    <asp:ListItem Text="sqlservercurry.com" Value="2"></asp:ListItem>
</asp:RadioButtonList>
```

and gets rendered in the browser as:

```
<table id="RadioButtonList1" border="0">
<tr>
<td>
<input id="RadioButtonList1_0" type="radio" name="RadioButtonList1" value="0" />
<label for="RadioButtonList1_0">dotnetcurry.com</label>
</td>
</tr>
<tr>
<td>
<input id="RadioButtonList1_1" type="radio" name="RadioButtonList1" value="1" />
<label for="RadioButtonList1_1">devcurry.com</label>
</td>
</tr>
<tr>
<td>
<input id="RadioButtonList1_2" type="radio" name="RadioButtonList1" value="2" />
```

```
<label for="RadioButtonList1_2">sqlservercurry.com</label>
</td>
</tr>
</table>
```

In this section, I will show you how to use jQuery to solve three common client-side scripting challenges faced while working with the RadioButton and RadioButtonList control. This section will contain the following recipes:

Recipe 18: Display the Text and Value of the Selected RadioButton using jQuery

Challenge:

Display the text and value of the selected radio button.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Retrieve the Text and Value of the selected RadioButton</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $radBtn = $("table.tbl input:radio");
            $radBtn.click(function() {
                var $radChecked = $(':radio:checked');
                $("#para").text('')
                    .append("<b>Index: </b>" +
                        $radChecked.val() + "<br/>")
                    .append("<b>Value: </b>" +
                        $radChecked.next().text());
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Select a RadioButton to display its Text and Value</h2><br />
        The site I like the most : <br />
        <asp:RadioButtonList ID="rbl" runat="server" class="tbl"
        ToolTip="Select a Radio Button to see its text and value">
            <asp:ListItem Text="dotnetcurry.com" Value="0"></asp:ListItem>
            <asp:ListItem Text="devcurry.com" Value="1"></asp:ListItem>
            <asp:ListItem Text="sqlservercurry.com" Value="2"></asp:ListItem>
        </asp:RadioButtonList>
        <br />
        <p id="para"></p>
    </div>
    </form>
</body>
</html>
```

Explanation:

Retrieving the index and value of a radio button is a very common requirement in our applications and can be easily achieved using jQuery. In this example, we first retrieve the radio buttons using

```
var $radBtn = $("table.tbl input:radio");
```

Since this is a RadioButtonList, the control gets rendered as a table. When the user clicks on any of the radio buttons, we first store the checked radiobuttons in the `$radChecked` variable to improve selector performance

```
var $radChecked = $(':radio:checked');
```

We then use

`$radChecked.val()` to retrieve the value and `$radChecked.next().text()` to retrieve the text.

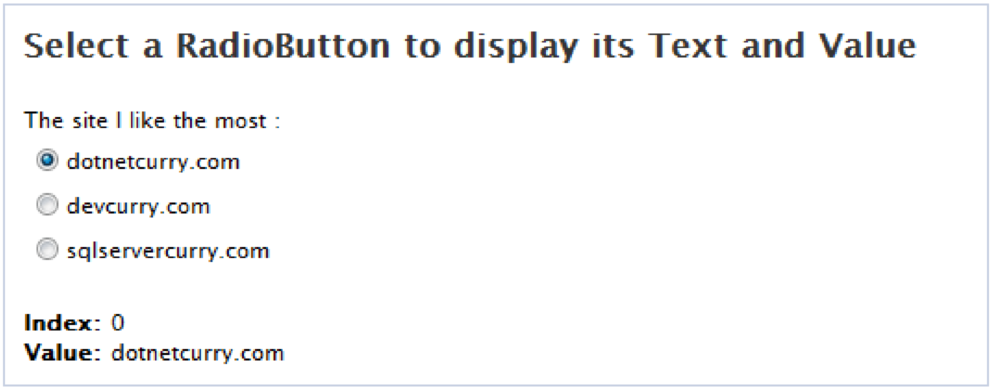
Observe how we are accessing the value of a radiobutton using

`$radChecked.next().text().`

This is because the RadioButton gets rendered like this:

```
<td>
<input id="rbl_0" type="radio" name="rbl" value="0" />
<label for="rbl_0">dotnetcurry.com</label>
</td>
```

To access the value, which is inside the label, we use *next()* which finds the very next sibling of each radiobutton, which in our case, is the label control. The value and text is displayed on the page using a paragraph (para). The output looks similar to the one shown below:


Select a RadioButton to display its Text and Value

The site I like the most :

● dotnetcurry.com
○ devcurry.com
○ sqlservercurry.com

Index: 0
Value: dotnetcurry.com

[Live Demo]

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/checked

Recipe 19: Display Text and Value of Multiple RadioButtonList using jQuery

Challenge:

Display the text and value of the selected radio button across multiple RadioButtonLists.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Display Values from Multiple ASP.NET RadioButtonLists</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            $("#btnSel").click(function(e) {
                e.preventDefault();
                var $selected = $('table.tbl :radio:checked');
                $("#para").text('');
                $selected.each(function() {
                    var $sel = $(this);
                    $("#para")
                    .append("<b>Index: </b>" +
                         $sel.val() + "<br/>")
                    .append("<b>Value: </b>" +
                        $sel.next().text() + "<br/>");
                });
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Select a Radio Button From Each of these lists
            and click the Button</h2>
        <br /><br />
        Developer Site I like the most : <br />
        <asp:RadioButtonList ID="rbl" runat="server" class="tbl">
```

```
            <asp:ListItem Text="dotnetcurry.com" Value="0"></asp:ListItem>
            <asp:ListItem Text="devcurry.com" Value="1"></asp:ListItem>
            <asp:ListItem Text="sqlservercurry.com" Value="2"></asp:ListItem>
        </asp:RadioButtonList>
        <br />
        I am a: <br />
        <asp:RadioButtonList ID="rbl1" runat="server" class="tbl">
            <asp:ListItem Text="Programmer" Value="3"></asp:ListItem>
            <asp:ListItem Text="Designer" Value="4"></asp:ListItem>
            <asp:ListItem Text="Both" Value="5"></asp:ListItem>
        </asp:RadioButtonList>
        <br />
        I visit your sites : <br />
        <asp:RadioButtonList ID="rbl2" runat="server" class="tbl">
            <asp:ListItem Text="2-3 times a week" Value="6"></asp:ListItem>
            <asp:ListItem Text="Once a Month" Value="7"></asp:ListItem>
            <asp:ListItem Text="None of the above" Value="8"></asp:ListItem>
        </asp:RadioButtonList>
        <br />
        <asp:Button ID="btnSel" runat="server" Text="Retrieve"
        Tooltip="Select a radio button from each list and click here"/>
        <br />
        <p id="para"></p>
        Tip: On selecting the radiobuttons and clicking the Button, the index
        and value of each selected radio button is displayed.
    </div>
    </form>
</body>
</html>
```

Explanation:

In Recipe 18 (Display the Text and Value of the Selected RadioButton using jQuery), we saw how to retrieve and display the text and value of the selected radio button. In this example, we will adopt the same technique, but across multiple RadioButtonList. The code remains the same as shown in Recipe 18, except that in this example, since there are multiple RadioButtonList, there could be multiple radio buttons selected; one in each list.

So to handle this scenario, we retrieve a collection of the ‘checked’ radio buttons using

```
var $selected = $('table.tbl :radio:checked');
```

We then iterate through this collection and print the value and text, using the same technique we adopted in Recipe 18.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/checked

Recipe 20: Display and Hide a RadioButton Based on a condition using jQuery

Challenge:

Display/Hide the radio button based on a condition.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Hide a RadioButton based on a condition</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $tbl = $('table.tbl');
            var $selRadio = $("table.tbl input:radio[value='0']");
            $tbl.hide();
            $('.cls input:radio').click(function(e) {
                var whichRad = e.target.id;
                $tbl.show();
                if (whichRad === $("input:radio[id$=rDel]").attr("id")) {
                    $selRadio.hide().next().hide();
                }
              else if (whichRad === $("input:radio[id$=rTake]").attr("id")) {
                    $selRadio.show().next().show();
                }
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Selectively Hide/Show Controls</h2><br />
        Do you want a Delivery or Take Away?<br />
        <asp:RadioButton ID="rDel" runat="server" Text="Delivery"
        GroupName="Mode" class="cls"  />
        <asp:RadioButton ID="rTake" runat="server" Text="Take Away"
        GroupName="Mode" class="cls" />
        <br /><hr />
        Payment Options<br />
        <asp:RadioButtonList ID="rbl" runat="server" class="tbl">
            <asp:ListItem Text="Credit Card" Value="0"></asp:ListItem>
            <asp:ListItem Text="Coupons" Value="1"></asp:ListItem>
            <asp:ListItem Text="Cash" Value="2"></asp:ListItem>
        </asp:RadioButtonList>
    </div>
    </form>
</body>
</html>
```

Explanation:

In this example, we ask the user to choose between two options – ‘Delivery’ or ‘TakeAway’. The CreditCard payment option is shown only if the user opts for ‘TakeAway’.

Since this is a RadioButtonList, the control gets rendered as a table. Since this list has class ‘tbl’ applied to it, hence we reference the control as `$('table.tbl')`.

We first cache the table object and the first radio button in separate variables as shown below:

```
var $tbl = $('table.tbl');
var $selRadio = $("table.tbl input:radio[value='0']");
```

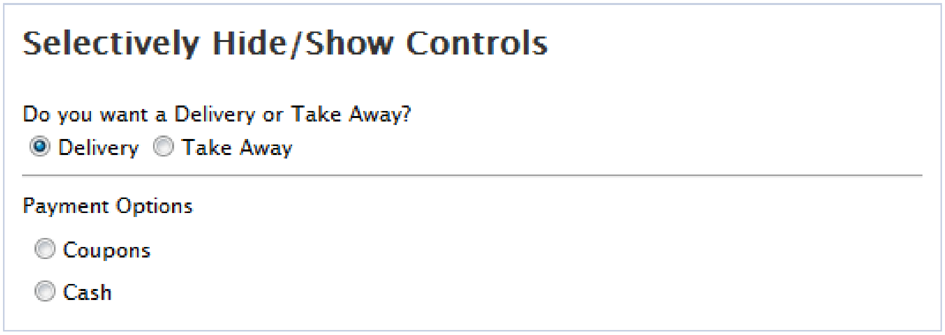
We start by hiding the payment options

```
$tbl.hide();
```

and then detect the radio button clicked (Delivery or TakeAway) by using

```
var whichRad = e.target.id;
```

If the user chooses the ‘Delivery’ option, the ‘CreditCard’ payment radio button is hidden as shown in the screenshot below:


Selectively Hide/Show Controls
Do you want a Delivery or Take Away?
Delivery
Take Away
Payment Options
Coupons
Cash

This is done using the following code:

```
if (whichRad === $("input:radio[id$=rDel]").attr("id")) {
    $selRadio.hide().next().hide();
}
```

You must be wondering why are we calling *hide()* twice. This is because the RadioButton gets rendered as the following markup:

```
<td>
<input id="rbl_0" type="radio" name="rbl" value="0" />

<label for="rbl_0">Credit Card</label>
</td>
```

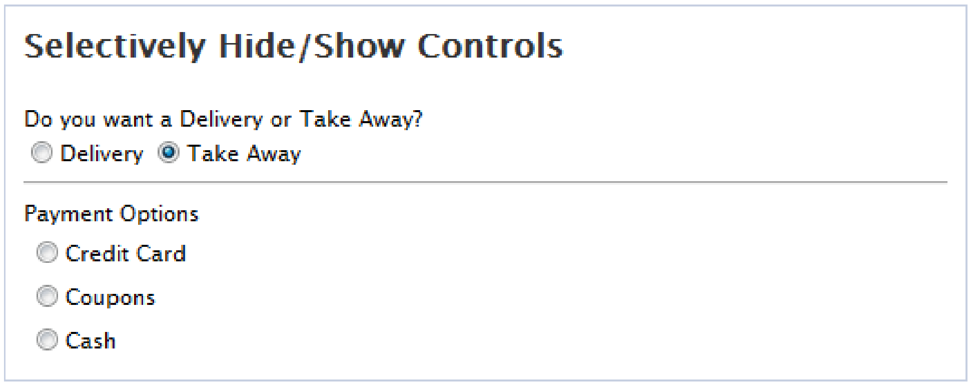
So to hide the RadioButton as well as its label, we use `.hide().next().hide();`

The first *hide()* hides the RadioButton. We use *next()* which finds the very next sibling of each radio button, which in our case is the label control and then call *hide()* again, to hide the label. This is a nice demonstration of method chaining.

Similarly, when the user chooses 'TakeAway', the 'CreditCard' payment radio button is made visible using

```
$selRadio.show().next().show();
```

The output is shown below:


Selectively Hide/Show Controls
Do you want a Delivery or Take Away?
Delivery
Take Away
Payment Options
Credit Card
Coupons
Cash

An alternative way to handle this requirement is to add a class="cc" to the Credit Card radio button and then access it using `$("span.cc")`

[Live Demo]

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://ejohn.org/blog/ultra-chaining-with-jquery/

Section 4: jQuery and the ASP.NET Panel

Introduction

The ASP.NET Panel acts as a container for other controls.

The Panel control is declared like this in your WebForm:

```
<asp:Panel ID="Panel1" runat="server">
</asp:Panel>
```

and gets rendered in the browser as:

```
<div id="Panel1"></div>
```

In this section, I will show you how to use jQuery to create some fancy animations using the Panel control. This section will contain the following recipes:

Recipe 21: Display and Hide Panels with Animation using jQuery

Challenge:

Display and Hide the contents of a Panel with Animation using multiple effects

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Display Hide Panels using jQuery</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $radBtn = $("table.tbl input:radio");
            var $panel = $('div.panel');
            $radBtn.click(function() {
                var value = $(':radio:checked').val();
                switch (value) {
                    case '0':
                        $panel.toggle('slow');
                        break;
                    case '1':
                        if ($panel.is(":hidden")) {
                            $panel.slideDown("fast");
                        } else {
                            $panel.slideUp("fast");
                        }
                        break;
                    case '2':
                        $panel.slideToggle('slow');
                        break;

                    case '3':
                        $panel.animate({
                            height: 'toggle',
                            margin: 'toggle',
                            opacity: 'toggle'
                        }, 500);
                        break;
                    default: ;
                }
            });
        });
    </script>

</head>
<body>
```

```
    <form id="form1" runat="server">
    <div class="smallDiv">
         <h2>Click on a Radio Button to Display/Hide the
            Contents of a Panel</h2>
        <asp:RadioButtonList ID="rbl" runat="server" class="tbl">
            <asp:ListItem Text="Toggle" Value="0"></asp:ListItem>
            <asp:ListItem Text="SlideUpDown" Value="1"></asp:ListItem>
            <asp:ListItem Text="SlideToggle" Value="2"></asp:ListItem>
            <asp:ListItem Text="Animate" Value="3"></asp:ListItem>
        </asp:RadioButtonList>
        <br />
         Tip: Each RadioButton produces a different animation
    </div>
    <br /><br />
    <div class="bigDiv">
        <asp:Panel ID="panelText" runat="server" CssClass="panel">
           Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc
           turpis nunc, placerat ac, bibendum non, pellentesque nec, odio.
           Nulla fringilla aliquet nibh. Donec placerat, massa id commodo
           ornare, justo lectus faucibus leo, in aliquam nisl quam varius
        </asp:Panel>
    </div>
    </form>
</body>
</html>
```

Explanation:

In this sample, I will demonstrate to you 4 different ways of displaying and hiding the content of a Panel.

The selectors are cached into variables as shown below:

```
var $radBtn = $("table.tbl input:radio");
var $panel = $('div.panel');
```

In the first method, we use *toggle()* which toggles the visibility of all matched elements without an animation.

```
$panel.toggle('slow');
```

The second method uses *slideUp()* to hide, and *slideDown()* to display the matched elements in a sliding motion.

```
if ($panel.is(":hidden")) {
    $panel.slideDown("fast");
} else {
    $panel.slideUp("fast");
}
```

The third method simplifies the technique adopted in the second method by using *slideToggle(),* which toggles the visibility of all matched elements using a sliding motion. Since we are using the same radio button to trigger the event, *slideToggle()* is more relevant here, rather than using *slideUp* or *slideDown().*

```
$panel.slideToggle('slow');
```

And finally the fourth method that shows how to make custom animations by using *animate()*. In this example, we are animating some style properties (height, margin and opacity) of the Panel.

```
$panel.animate({
    height: 'toggle',
    margin: 'toggle',
    opacity: 'toggle'
}, 500);
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Effects/toggle

http://docs.jquery.com/Effects/slideUp#speedcallback

http://docs.jquery.com/Effects/slideDown#speedcallback

http://docs.jquery.com/Effects/slideToggle

http://docs.jquery.com/Effects/animate#paramsoptions

Recipe 22: Smooth Cascading Panels using jQuery

Challenge:

Panels should be Expanded/Collapsed one after the other with a Cascading Effect

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Smooth Cascading Effect with ASP.NET Panels</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $cpB = $("div.cpBody");
            var $iShow = $("input.show");
```

```
             var $iHide = $("input.hide");

            $cpB.hide();
            $iHide.hide();
            var panelArray = $cpB.length;
            var temp = -1;

            $iShow.click(function displayPanel() {
                if (temp < panelArray) {
                    $cpB.eq(++temp).fadeIn(2000, function() {
                        displayPanel();
                        if (temp == panelArray) {
                            $iHide.show();
                            $iShow.hide();
                        }
                    });
                }
            });


            $iHide.click(function hidePanel() {
                if (temp >= 0) {
                    $cpB.eq(--temp).fadeOut(2000, function() {
                        hidePanel();
                    });
                    if (temp < 0) {
                        $iHide.hide();
                        $iShow.show();
                        temp = -1;
                    }
                }
            });

        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="divNoBorder">
        <h2>Click on the '+' to Expand and '-' to Collapse the Panels</h2>
        <div class="cpHeader">
            <asp:ImageButton ID="btnShow" ImageUrl="../Images/Show.gif"
                runat="server" OnClientClick="return false;" class="show"/>
            <asp:ImageButton ID="btnHide" ImageUrl="../Images/Hide.gif"
                runat="server" OnClientClick="return false;" class="hide"/>
        </div>

        <asp:Panel ID="panelOne" runat="server" class='cpBody'>
            Panel 1 Content
        </asp:Panel>
        <asp:Panel ID="panelTwo" runat="server" class='cpBody'>
            Panel 2 Content
        </asp:Panel>
        <asp:Panel ID="panelThree" runat="server" class='cpBody'>
            Panel 3 Content
        </asp:Panel>
    </div>
    </form>
</body>
</html>
```

Explanation:

In this example, we are using 3 panels and 2 image buttons. The example starts by caching the following selectors into variables

```
var $cpB = $("div.cpBody");
var $iShow = $("input.show");
var $iHide = $("input.hide");
```

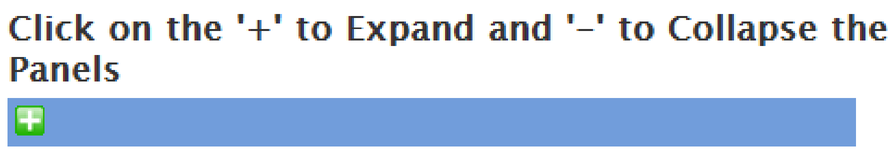
To start with, all the panels with 'class=cpBody' are hidden using

```
$cpB.hide();
```

Similarly one of the image button with 'class=hide', is hidden using

```
$iHide.hide()
```

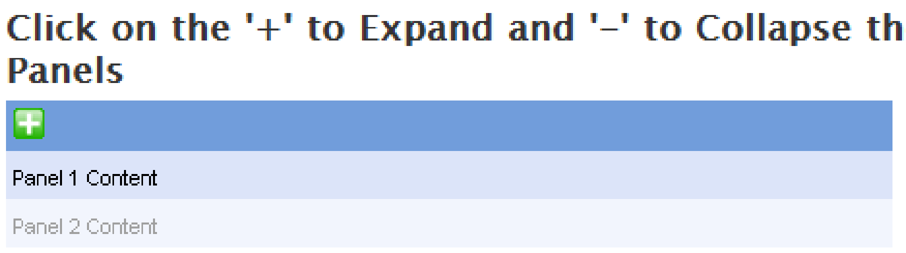
When the page loads, you see only the 'expand' image button (+) as shown below:



On the click of the image button '+', we use a recursive function that loops through the panels (class='cpBody') and displays the panels, one after the other and one at a time, using the *fadeIn()* method. The variables 'panelArray' is used to track the count of panels whereas 'temp' keeps a track of visible panels.

```
$iShow.click(function displayPanel() {
    if (temp < panelArray) {
        $cpB.eq(++temp).fadeIn(2000, function() {
            displayPanel();
```

This is how the panels expand with animation:


Click on the '+' to Expand and '–' to Collapse the Panels

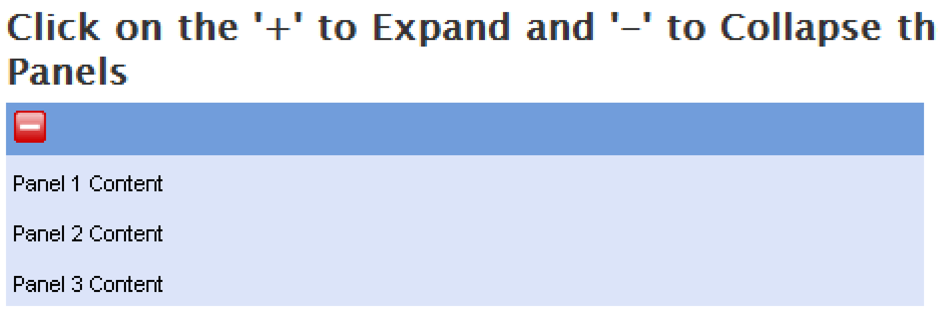
Panel 1 Content

Panel 2 Content

When the variable ‘temp’ equals to the ‘panelArray’, the ‘+’ (Expand) image button is hidden and the ‘-‘ (Collapse) image button is displayed to the user

```
if (temp == panelArray) {
    $iHide.show();
    $iShow.hide();
}
```

The output is shown below:


Click on the '+' to Expand and '–' to Collapse the Panels

Panel 1 Content

Panel 2 Content

Panel 3 Content

Similarly on the click of the ‘–‘ image button, we do a reverse lookup to hide the panels using the *fadeOut()* function.

```
$iHide.click(function hidePanel() {
    if (temp >= 0) {
        $cpB.eq(--temp).fadeOut(2000, function() {
            hidePanel();
        });
        if (temp < 0) {
            $iHide.hide();
            $iShow.show();
            temp = -1;
        }
    }
});
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Effects/fadeIn#speedcallback

http://docs.jquery.com/Effects/fadeOut#speedcallback

http://docs.jquery.com/Selectors/eq#index

Recipe 23: Create a Slide Show with Panels using jQuery

Challenge:

The Panels are to be displayed one after the other, and after a fixed duration, like a slide show. The Panels should have a sliding effect.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Slide Show With Panels</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $divSlide = $("div.slide");
            $divSlide.hide().eq(0).show();
            var panelCnt = $divSlide.length;

            setInterval(panelSlider, 3000);

            function panelSlider() {
                $divSlide.eq(($divSlide.length++) % panelCnt)
                .slideUp("slow", function() {
                    $divSlide.eq(($divSlide.length) % panelCnt)
                        .slideDown("slow");
                });
            }
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="divNoBorder">
         <h2>Slide Show with Panels.
        Each Panel is Visible for 3 seconds.</h2>
        <br /><br />
         <asp:Panel ID="panelOne" runat="server" class='slide'>
            Panel 1 Content Panel 1 Content Panel 1 Content
            Panel 1 Content Panel 1 Content Panel 1 Content
            Panel 1 Content Panel 1 Content Panel 1 Content
            Panel 1 Content Panel 1 Content Panel 1 Content
        </asp:Panel>
        <asp:Panel ID="panelTwo" runat="server" class='slide'>
            Panel 2 Content Panel 2 Content Panel 2 Content
            Panel 2 Content Panel 2 Content Panel 2 Content
            Panel 2 Content Panel 2 Content Panel 2 Content
            Panel 2 Content Panel 2 Content Panel 2 Content
        </asp:Panel>
        <asp:Panel ID="panelThree" runat="server" class='slide'>
           Panel 3 Content Panel 3 Content Panel 3 Content
           Panel 3 Content Panel 3 Content Panel 3 Content
           Panel 3 Content Panel 3 Content Panel 3 Content
           Panel 3 Content Panel 3 Content Panel 3 Content
           Panel 3 Content Panel 3 Content Panel 3 Content
        </asp:Panel>
        <asp:Panel ID="panelFour" runat="server" class='slide'>
            Panel 4 Content
        </asp:Panel>
        <asp:Panel ID="panelFive" runat="server" class='slide'>
            Panel 5 Content
        </asp:Panel>
    </div>
    </form>
</body>
</html>
```

Explanation:

The first step is to cache the selector in a variable as shown below:

```
var $divSlide = $("div.slide");
```

The example then starts off by hiding all the panels except the first one using

```
$divSlide.hide().eq(0).show();
```

We then use the JavaScript setInterval() function to delay the execution of a function (panelSlider) for a specified time, in our case 3000 millisecond (3 seconds).

```
setInterval(panelSlider, 3000);
```

The advantage of a setInterval() function is that it continues to repeat the process of triggering the function at the specified interval, thereby giving it a loop effect.

To pause the slideshow, use the *clearInterval()* function.

The number of panels is stored in a variable panelCnt

```
var panelCnt = $divSlide.length;
```

In the panelSlider() function, we use a simple expression `($divSlide.length++) % panelCnt` that calculates the index to be supplied to the eq() selector and applies the fadeout/fadeIn() animations on the current panel. eq(0) refers to the first panel, eq(1) to the second and so on.

```
function panelSlider() {
    $divSlide.eq(($divSlide.length++) % panelCnt)
    .slideUp("slow", function() {
        $divSlide.eq(($divSlide.length) % panelCnt)
            .slideDown("slow");
}
```

The chart shown below helps you understand what happens at every loop

Loop Number	**Value of** `$divSlide.length`	**Value of** `($divSlide.length++) % panelCnt`
1	5	5mod5 = 0
2	6	6mod5 = 1
3	7	7mod5 = 2
4	8	8mod5=3
5	9	9mod5=4
6	10	10mod5=0
and so on….		

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

https://developer.mozilla.org/En/Window.setInterval

http://docs.jquery.com/Tutorials:Scroll_Up_Headline_Reader

Section 5: jQuery and the ASP.NET DropDownList

Introduction

The ASP.NET DropDownList allows the user to select a single item from a drop-down list.

The DropDownList control is declared like this in your WebForm:

```
<asp:DropDownList ID="DropDownList1" runat="server" >
    <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
    <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
</asp:DropDownList>
```

and gets rendered in the browser as:

```
<select name="DropDownList1" id="DropDownList1">
       <option value="1">Item1</option>
       <option value="2">Item2</option>
</select>
```

In this section, I will show you how to use jQuery to solve five common client-side scripting challenges faced while using the DropDownList control. This section will contain the following recipes:

Recipe 24: Retrieve the Selected Text and Value of a DropDownList using jQuery

Challenge:

You want to display the text and value of the DropDownList item selected by the user

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Get Selected Value and Text From ASP.NET DropDownList</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />

    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('select[id$=DDL]').bind("change keyup", function() {
                $('#para').html(
                "Value: " + $(this).val() +
                "<br />" +
                " Text: " + $('select[id$=DDL] :selected').text());
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Select an Item from the DropDownList to display
        it's text and value </h2>
        <asp:DropDownList ID="DDL" runat="server" >
            <asp:ListItem Text="Select an Item" Value="" />
            <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
            <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
            <asp:ListItem Text="Item4" Value="4"></asp:ListItem>
            <asp:ListItem Text="Item5" Value="5"></asp:ListItem>
            <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
        </asp:DropDownList>
        <br /><br />
        <p id="para"></p>
    </div>
    </form>
</body>
</html>
```

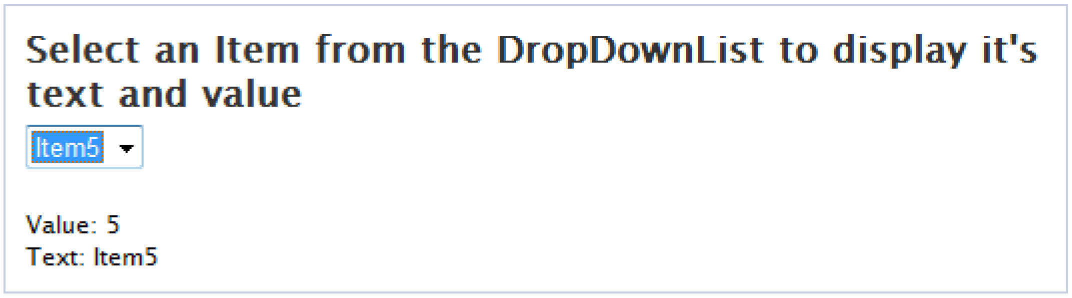
Explanation:

In this example, both the change and keyup event of the DropDownList is captured using *bind()*. According to the W3C specifications, *the change event occurs when a control loses the input focus and its value has been modified since gaining focus*. However IE triggers the 'change' event as soon as the option is selected, whereas Mozilla, Chrome and Safari do not. This behavior of triggering the change event by IE on keyboard scrolling is not according to the W3C specs. This is the reason that we use both the change and keyup event to capture selection, when the user uses the keyboard to scroll through the dropdownlist items.

Whenever the user selects a new item from the DropDownList, the value is printed using `$(this).val()` and the text is printed using `$('select[id$=DDL] :selected').text()`.

The `:selected` selector matches selected <option> elements. The results are written to a paragraph using `$('#para').html()`

When the user selects an item from the DropDownList, the output looks similar to the one shown below:



Select an Item from the DropDownList to display it's text and value

Value: 5
Text: Item5

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/selected

Recipe 25: Sort the Items of a DropDownList using jQuery

Challenge:

The Items of a DropDownList are displayed in an unsorted format. You want to give the user the functionality to sort the items in an ascending order.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Sort Items of an ASP.NET DropDownList</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnSort]').click(function(e) {
                e.preventDefault();
                var sortedDdl = $.makeArray($('select[id$=DDL] option'))
                .sort(function(o, n) {
                    return $(o).text() < $(n).text() ? -1 : 1;
                });
                $("select[id$=DDL]").html(sortedDdl).val("1");
                $("#para").html("Items were Sorted!");
                $(this).attr("disabled", "disabled");
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the Sort Button to Sort the DropDownList </h2>
        <asp:DropDownList ID="DDL" runat="server" >
            <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
            <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
            <asp:ListItem Text="Item4" Value="4"></asp:ListItem>
            <asp:ListItem Text="Item5" Value="5"></asp:ListItem>
            <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
        </asp:DropDownList>
        <br /><br />
        <asp:Button ID="btnSort" runat="server" Text="Sort" />
        <p id="para"></p>
        <br /><br />
        Tip: Items are sorted in an Ascending order
    </div>
    </form>
</body>
```

```
</html>
```

Explanation:

In the code shown above, when the user clicks on the Sort button, the <option> elements are converted to an array using `$.makeArray()`.

```
$.makeArray($('select[id$=DDL] option'))
```

The JavaScript built-in sort() function is used on this array, which does an in-place sort of the array.

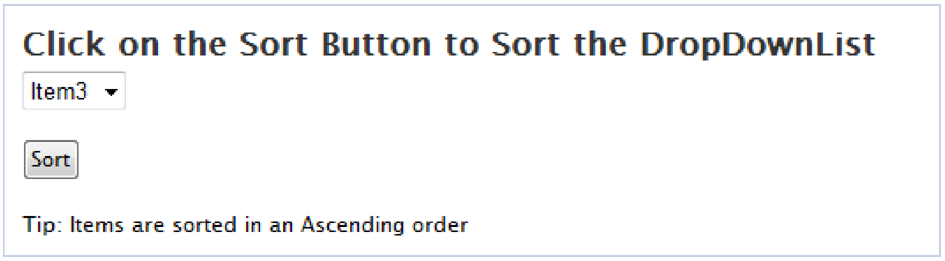
```
$.makeArray($('select[id$=DDL] option'))
.sort(function(o, n) {
    return $(o).text() < $(n).text() ? -1 : 1;
});
```

The final step is to empty the contents of the DropDownList and then use the *.html()* to replace the existing <options> with the sorted <options>.

```
$("select[id$=DDL]").empty().html(sorted)
```

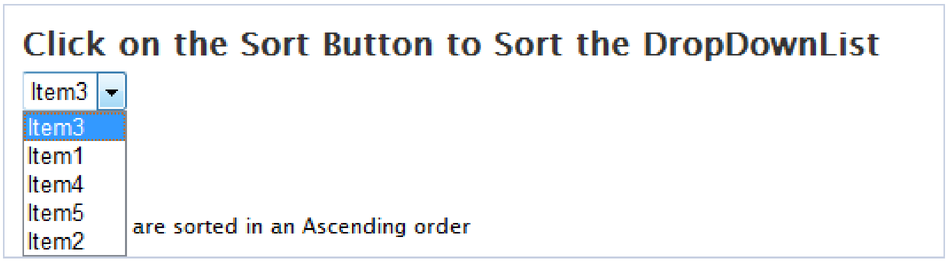
The `val("1")` sets the first option as selected, after the sorting has been done.

When the page loads, the output looks similar to the following screenshot:


Click on the Sort Button to Sort the DropDownList

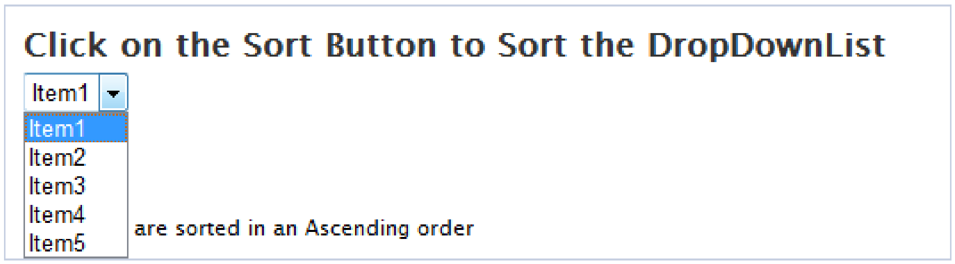
Tip: Items are sorted in an Ascending order

Before clicking on the Sort button, the output is as shown below:


Click on the Sort Button to Sort the DropDownList

are sorted in an Ascending order

After clicking on the Sort Button, the output is as shown below:


Click on the Sort Button to Sort the DropDownList

are sorted in an Ascending order

To sort in a descending order, replace this line

```
$(a).text() < $(b).text()
```

with this:

```
$(a).text() > $(b).text()
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Utilities/jQuery.makeArray

http://docs.jquery.com/Attributes/html#val

Recipe 26: Add Items to the DropDownList using jQuery

Challenge:

Users need to be given the flexibility to add items to the DropDownList at runtime.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Add New Items to the ASP.NET DropDownList</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnAdd]').click(function(e) {
                e.preventDefault();
                // Assuming you get a JSON object from a service
                var lenBefore = $('select[id$=DDL] :all').length;
                var obj = { "6": "Item6", "7": "Item7", "8": "Item8" };
                $.each(obj, function(value, text) {
                    $('select[id$=DDL]').append(
                    $('<option></option>').val(value).html(text))
                });
                var lenAfter = $('select[id$=DDL] :all').length;
                $("#para")
    .html("<b>" + (lenAfter - lenBefore) + "</b>" + " new items were added");
                $(this).attr("disabled", "disabled");
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the Add Button
        to Add New Items to the DropDownList </h2>
        <asp:DropDownList ID="DDL" runat="server" >
            <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
            <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
            <asp:ListItem Text="Item4" Value="4"></asp:ListItem>
            <asp:ListItem Text="Item5" Value="5"></asp:ListItem>
            <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
        </asp:DropDownList>
        <br /><br />
        <asp:Button ID="btnAdd" runat="server" Text="Add New Items"
        ToolTip="Click to Add New Items" /><br />
        <p id="para"></p>
        Tip: Duplicate values are also added.
    </div>
    </form>
</body>
</html>
```

Explanation:

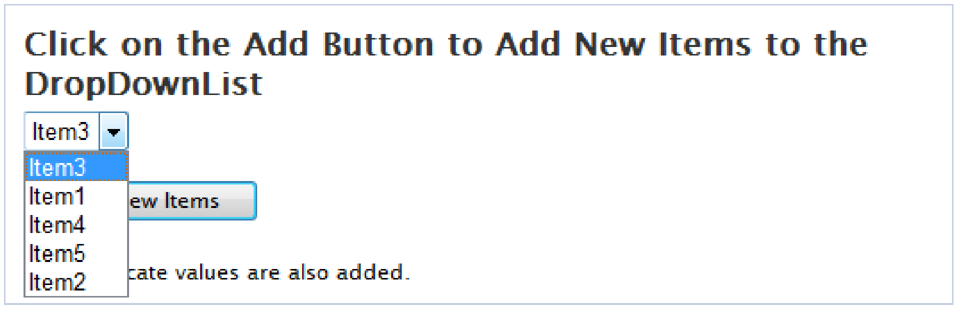
For this example, we are assuming that we have a JSON service which returns an object. You want to consume this object and add it to the DropDownList.

When the user clicks the button, you iterate over the object received from a JSON service and use *append()* to append content to the inside of every matched element (`<option></option>`).

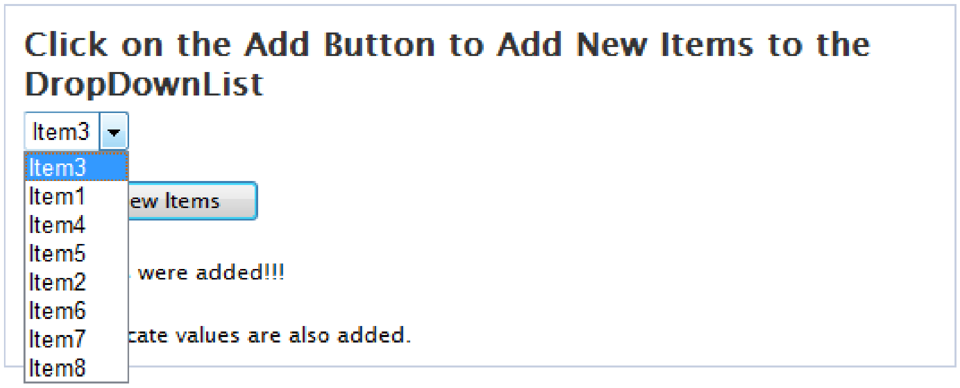
```
var obj = { "6": "Item6", "7": "Item7", "8": "Item8" };
$.each(obj, function(value, text) {
    $('select[id$=DDL]').append(
    $('<option></option>').val(value).html(text))
});
```

We also use the lenBefore and lenAfter variables to print the additional items added to the DropDownList.

Before clicking on the ‘Add New Items’ button, the output looks similar to the following:


Click on the Add Button to Add New Items to the DropDownList

After clicking on the ‘Add New Items’ button, the output is as shown below:


Click on the Add Button to Add New Items to the DropDownList

[Live Demo]

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Utilities/jQuery.each

http://docs.jquery.com/Manipulation/append#content

Recipe 27: Search and Filter Items of a DropDownList using jQuery

Challenge:

Users need to be able to Search and Filter items of a DropDownList

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Search and Filter a DropDownList</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src="../Scripts/jquery-1.3.2.min.js"
        type="text/javascript"></script>

    <script type="text/javascript">
        $(function() {
            var $txt = $('input[id$=txtNew]');
            var $ddl = $('select[id$=DDL]');
            var $items = $('select[id$=DDL] option');

            $txt.keyup(function() {
                searchDdl($txt.val());
            });

            function searchDdl(item) {
                $ddl.empty();
                var exp = new RegExp(item, "i");
                var arr = $.grep($items,
                    function(n) {
                        return exp.test($(n).text());
                    });

                if (arr.length > 0) {
                    countItemsFound(arr.length);
                    $.each(arr, function() {
                        $ddl.append(this);
                        $ddl.get(0).selectedIndex = 0;
                    }
                    );
                }
                else {
                    countItemsFound(arr.length);
                    $ddl.append("<option>No Items Found</option>");
                }
            }

            function countItemsFound(num) {
                $("#para").empty();
                if ($txt.val().length) {
                    $("#para").html(num + " items found");
                }
            }
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Enter Text in the TextBox to Filter the DropDownList </h2>
        <br />
        <asp:TextBox ID="txtNew" runat="server"
            ToolTip="Enter Text Here"></asp:TextBox><br />
        <asp:DropDownList ID="DDL" runat="server" >
            <asp:ListItem Text="Apple" Value="1"></asp:ListItem>
            <asp:ListItem Text="Orange" Value="2"></asp:ListItem>
            <asp:ListItem Text="Peache" Value="3"></asp:ListItem>
            <asp:ListItem Text="Banana" Value="4"></asp:ListItem>
            <asp:ListItem Text="Melon" Value="5"></asp:ListItem>
            <asp:ListItem Text="Lemon" Value="6"></asp:ListItem>
            <asp:ListItem Text="Pineapple" Value="7"></asp:ListItem>
            <asp:ListItem Text="Pomegranate" Value="8"></asp:ListItem>
            <asp:ListItem Text="Mulberry" Value="9"></asp:ListItem>
            <asp:ListItem Text="Apricot" Value="10"></asp:ListItem>
            <asp:ListItem Text="Cherry" Value="11"></asp:ListItem>
            <asp:ListItem Text="Blackberry" Value="12"></asp:ListItem>
        </asp:DropDownList>
        <br />
        <p id="para"></p>
        <br /><br />
        Tip: Items get filtered as characters are entered in the textbox.
        Search is not case-sensitive
    </div>
    </form>
</body>
</html>
```

Explanation:

The example starts by caching the following objects into variables

```
var $txt = $('input[id$=txtNew]');
var $ddl = $('select[id$=DDL]');
var $items = $('select[id$=DDL] option');
```

On the *keyup()* event of the textbox, we are calling the searchDdl() method which accepts the textbox value as its parameter.

```
$txt.keyup(function() {
    searchDdl($txt.val());
});
```

Inside the searchDdl() method, the contents of the DropDownList are emptied and we then use the RegExp object to store the search pattern. In this case, the search modifier is ‘i’ which tells the regular expression to ignore the case of the characters. The expression gets translated to /searchtext/i

Let us analyze this piece of code given below:

```
var arr = $.grep($items,
    function(n) {
        return exp.test($(n).text());
    });
```

The *$.grep()* method finds the elements of an array which satisfy a filter function. The syntax is as follows:

```
$.grep( array, callback, [invert] )
```

where the *array* refers to the set of <option> elements that are searched, and the *callback* refers to the function to process each item against. In this callback function, we call the *.test()* method of the RegExp object, passing it the option item text to see if it matches the characters entered by the user. The *.grep()* this way returns a newly constructed array containing a list of matched items, shown in the code above.

If the array contains matched items, the *$.each()* iterates over this collection(returned by the *$.grep()* method) and fires a callback function on each item, which appends the item to the DropDownList. The number of items that match the search are displayed using countItemsFound() function.

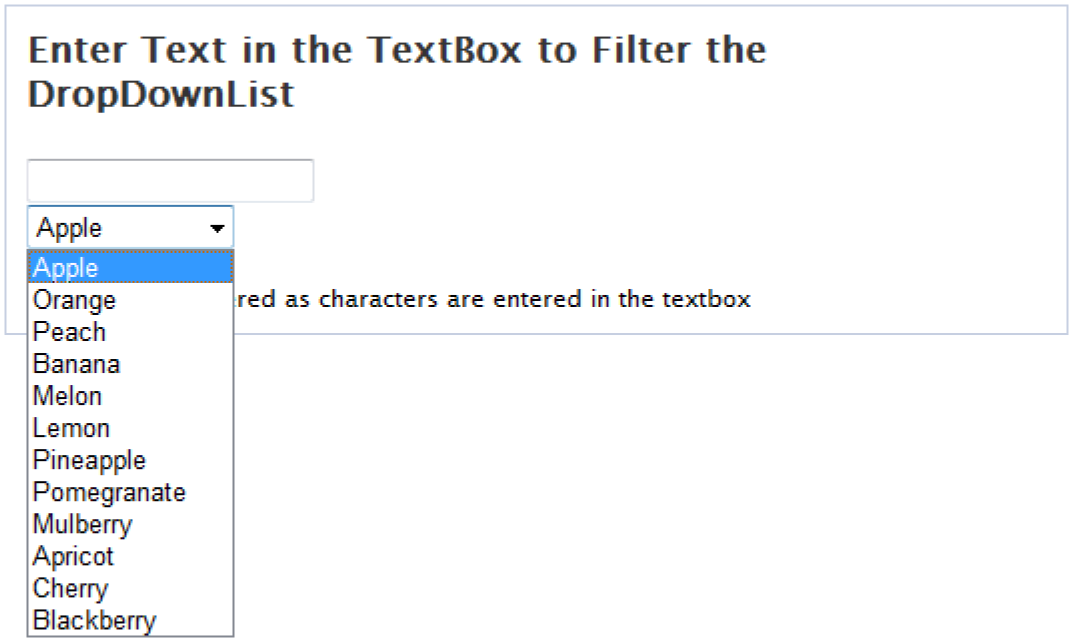
```
if (arr.length > 0) {
    countItemsFound(arr.length);
    $.each(arr, function() {
        $ddl.append(this);
        $ddl.get(0).selectedIndex = 0;
    }
    );
}
else {
    countItemsFound(arr.length);
    $ddl.append("<option>No Items Found</option>");
}
```

If the search does not match the items in the DropDownList, we display ‘No Items Found’ using the following code:

```
$ddl.append("<option>No Items Found</option>");
```

This is how we search and filter through the list.

Before the user type's text in the TextBox, the output looks similar to the following:



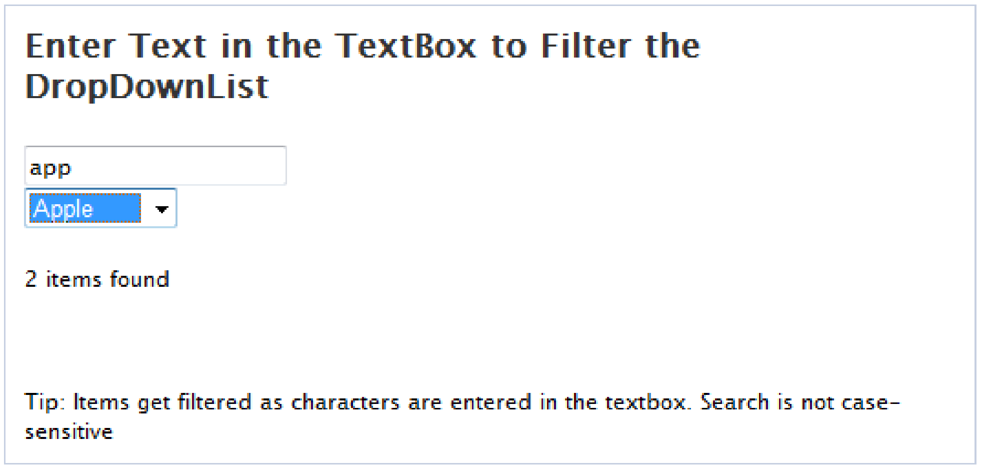
Enter Text in the TextBox to Filter the DropDownList

Apple

Apple
Orange
Peach
Banana
Melon
Lemon
Pineapple
Pomegranate
Mulberry
Apricot
Cherry
Blackberry

red as characters are entered in the textbox

After the user types some text in the TextBox, the output is as shown below:



Enter Text in the TextBox to Filter the DropDownList

app

Apple

2 items found

Tip: Items get filtered as characters are entered in the textbox. Search is not case-sensitive

If you want to reverse the filter condition, use the [invert] attribute of the $.*grep()* method. For e.g. if you want to filter the DropDownList with items that 'do not match' the text entered by the user, use the [invert] attribute as shown below. This attribute takes a Boolean value.

```
var arr = $.grep($items,
    function(n) {
        return exp.test($(n).text());
    }, true);
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Utilities/jQuery.grep

http://www.w3schools.com/jsref/jsref_obj_regexp.asp

Recipe 28: Convert the DropDownList to emulate a MultiSelect ListBox using jQuery

Challenge:

Users want functionality where they can select multiple items of a DropDownList.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Convert DropDownList to MultiSelect Box</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $conv = $('input[id$=btnConvert]');
            var $multi = $('input[id$=btnMulti]');
            $multi.hide();

            $conv.click(function(e) {
                e.preventDefault();
                $('select[id$=DDL]').attr("multiple", "multiple");
                $conv.hide();
                $multi.show();
            });

            $multi.click(function(e) {
                e.preventDefault();
                $("#para").empty();
                $('select[id$=DDL] :selected').each(function(sel) {
                    $("#para").append($(this).text() + "<br />");
                });
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the Button to Convert DropDownList to ListBox</h2>
        <asp:DropDownList ID="DDL" runat="server" >
            <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
            <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
            <asp:ListItem Text="Item4" Value="4"></asp:ListItem>
            <asp:ListItem Text="Item5" Value="5"></asp:ListItem>
            <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
        </asp:DropDownList>
        <br /><br />
        <asp:Button ID="btnConvert" runat="server"
            Text="Convert To MultiSelect List"
            ToolTip="Click to Convert" />
        <asp:Button ID="btnMulti" runat="server"
            Text="Press Ctrl to select multiple items & Click Here"
            ToolTip="Click Here to List Selected Values" />
        <p id="para"></p>
    </div>
    </form>
</body>
</html>
```

Explanation:

Multiple items cannot be selected in a DropDownList unless you change the default behavior of this control by writing custom code. It's the ListBox control that allows the user to select multiple items when its 'SelectionMode' property is set to multiple.

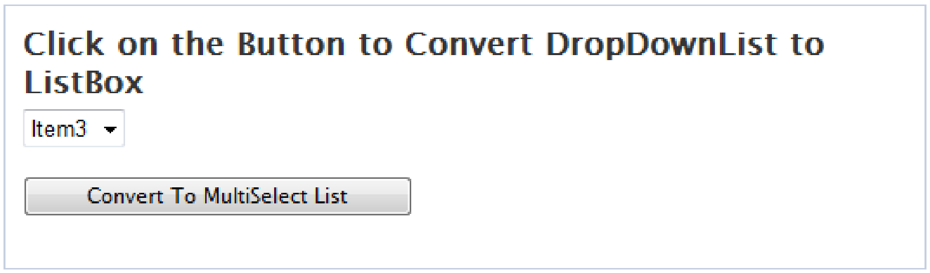
I faced this requirement recently during a client visit. The client was keen on keeping a DropDownList to save screen space. However they also wanted to convert this to a ListBox, if the user wanted to select multiple items. So to solve this requirement client-side, a simple technique adopted was to convert the DropDownList to a MultiSelect ListBox control on the click of a Button. It just requires a single line of code as shown here:

```
$('select[id$=DDL]').attr("multiple", "multiple");
```

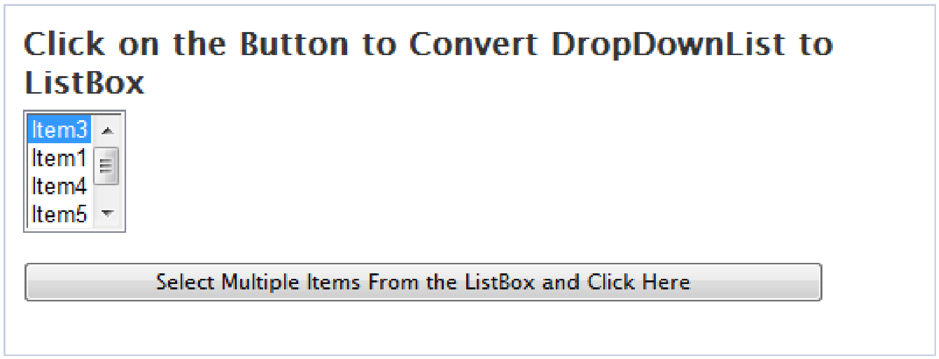
Once the DropDownList is converted into a ListBox control, the user can select multiple items and list them

```
$('select[id$=DDL] :selected').each(function(sel) {
    $("#para").append($(this).text() + "<br />");
});
```

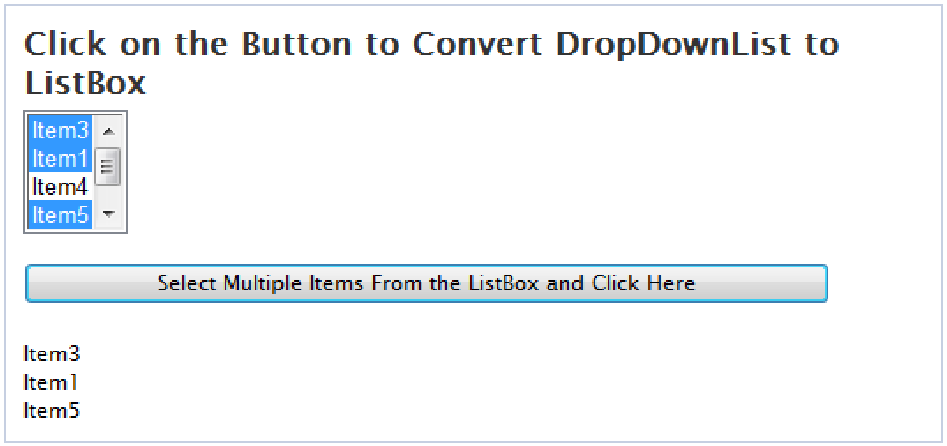
Before the Button is clicked, the output looks like this:


Click on the Button to Convert DropDownList to ListBox

After the Button is clicked, the output is as shown below:


Click on the Button to Convert DropDownList to ListBox

After the user selects multiple values and clicks the Button, the output changes as shown here:


Click on the Button to Convert DropDownList to ListBox

```
Item3
Item1
Item5
```

Use this example with caution. It's always best to choose appropriate controls.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/selected

Section 6: jQuery and the ASP.NET GridView

Introduction:

The ASP.NET GridView control renders its data as a HTML table. This control is one of the most frequently used ASP.NET Controls.

A sample GridView bound to the ObjectDataSource control with paging enabled, is declared in a similar way on your WebForm as shown below:

```
<asp:GridView ID="grdEmployees" runat="server"
    AllowPaging="True" PageSize="2" AutoGenerateColumns="False"
    DataSourceID="ObjectDataSource1">
    <Columns>
        <asp:BoundField DataField="ID" HeaderText="ID" />
        <asp:BoundField DataField="FName" HeaderText="FName" />
        <asp:BoundField DataField="MName" HeaderText="MName"  />
        <asp:BoundField DataField="LName" HeaderText="LName" />
    </Columns>
</asp:GridView>
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
    SelectMethod="GetEmployeeList" TypeName="Employee">
</asp:ObjectDataSource>
```

and gets rendered in the browser as:

```
<table cellspacing="0" rules="all" border="1" id="Table1" style="border-
collapse:collapse;">
<tr>
<th scope="col">ID</th><th scope="col">FName</th><th scope="col">MName</th><th
scope="col">LName</th>
</tr>
<tr>
       <td>1</td><td>John</td><td>&nbsp;</td><td>Shields</td>
</tr>
<tr>
       <td>2</td><td>Mary</td><td>Matthew</td><td>Jacobs</td>
</tr>
<tr>
       <td colspan="4">
       <table border="0">
       <tr>
              <td><span>1</span></td>
              <td><a href="javascript:__doPostBack('GridView1','Page$2')">2</a></td>
              <td><a href="javascript:__doPostBack('GridView1','Page$3')">3</a></td>
              <td><a href="javascript:__doPostBack('GridView1','Page$4')">4</a></td>
              <td><a href="javascript:__doPostBack('GridView1','Page$5')">5</a></td>
              <td><a href="javascript:__doPostBack('GridView1','Page$6')">6</a></td>
              <td><a href="javascript:__doPostBack('GridView1','Page$7')">7</a></td>
```

```
        </tr>
        </table>
        </td>
</tr>
</table>
```

In this section, I will show you how to use jQuery to solve eight common client-side scripting challenges faced while using the GridView control.

Recipe 30 is an exception that shows how to do server-side paging without using the GridView control. This recipe was contributed by Malcolm Sheridan.

This section will contain the following recipes:

Recipe 29: Pass Multiple Values from a GridView to Another Page using jQuery

Challenge:

The user should be able to click on a GridView row and pass multiple values of the row to another page

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Pass Multiple Values from a GridView to Another Page</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
        $(".gv > tbody > tr:not(:has(table, th))")
            .css("cursor", "pointer")
            .click(function(event) {
                var $row = $(this);
                var firstParam = $("td", $row).eq(0).text();
                var secondParam = $("td", $row).eq(1).text();
                var thirdParam = $("td", $row).eq(3).text();
                var baseUrl = "../"
                    var navUrl = baseUrl + "Section6-GridView/R29a-SamplePage.aspx" +
                    "?id=" + firstParam +
                    "&fname=" + secondParam +
                    "&lname=" + thirdParam;
                top.location = navUrl;
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Click on any Table Row to pass the ID, FName and LastName
           of that row to another page</h2><br />
        <asp:GridView ID="grdEmployees" runat="server"
            AllowPaging="True" PageSize="5" AutoGenerateColumns="False"
            DataSourceID="ObjectDataSource1" class="gv">
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" />
                <asp:BoundField DataField="FName" HeaderText="FName" />
                <asp:BoundField DataField="MName" HeaderText="MName"  />
                <asp:BoundField DataField="LName" HeaderText="LName" />
                <asp:BoundField DataField="DOB" HeaderText="DOB"
                    DataFormatString="{0:MM/dd/yyyy}" />
                <asp:BoundField DataField="Sex" HeaderText="Sex" />
            </Columns>
        </asp:GridView>
        <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetEmployeeList" TypeName="Employee">
        </asp:ObjectDataSource>
    </div>
    </form>
</body>
</html>
```

Explanation:

The example starts with a filter applied on the GridView rows

```
$(".gv > tbody > tr:not(:has(table, th))")
```

This filter is required since a GridView does not render a THEAD and a TFOOT (accessibility tags) by default. For the header, the GridView generates TH's inside a TR. Similarly for the footer, the GridView generates a TABLE inside a TR and so on. Hence it is required to use additional filters to exclude header and footer rows while performing operations on the GridView rows. The filter shown above selects only those rows which are inside the TBODY.

When you right click and view source, you may not find a TBODY either. However do note that when the TBODY is not explicitly declared, all modern browsers automatically create the DOM with the TBODY. As a best practice, you should explicitly create the TBODY on the server side as demonstrated by me in this blogpost.

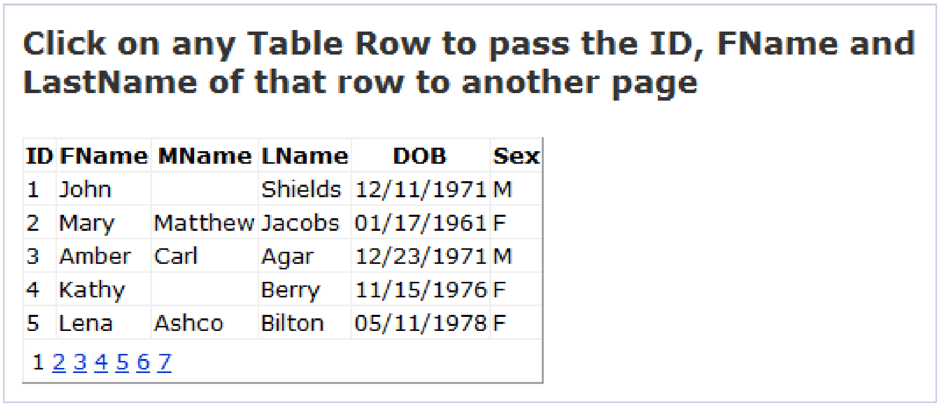
The rest of the example is straightforward. Whenever the user hovers the mouse over a table row, the cursor:pointer css is applied on the filtered rows, to change the default cursor to a hand. We then capture the click on each GridView row (excluding the header and pager row using the filter) and then access a cell of the row using the following code:

```
 var $row = $(this);
 var firstParam = $("td", $row).eq(0).text();
```

The current row object is cached in the '$row' variable and the *eq()* selector is then used to match a 'single cell' of the clicked row by providing the cell index. In the code shown above, the cell index is 0, referring to the first cell in the row. In the same way, we get the values for the second and fourth cell as well. A new navigation url is then formed using these cell values as parameters and then finally the location is set to the new url.

```
var baseUrl = "../"
var navUrl = baseUrl + "Section6-GridView/R29a-SamplePage.aspx" +
    "?id=" + firstParam +
    "&fname=" + secondParam +
    "&lname=" + thirdParam;
top.location = navUrl;
```

When the page loads, you see a screen similar to the following:

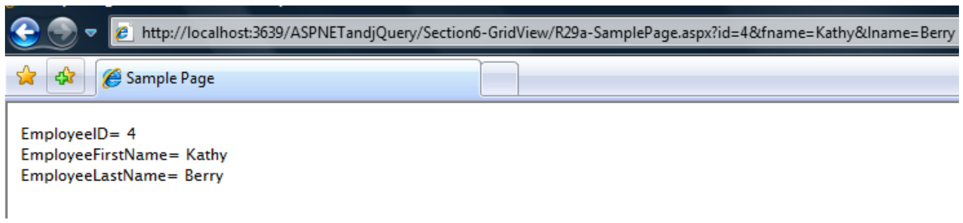


Click on any Table Row to pass the ID, FName and LastName of that row to another page

ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
3	Amber	Carl	Agar	12/23/1971	M
4	Kathy		Berry	11/15/1976	F
5	Lena	Ashco	Bilton	05/11/1978	F

1 2 3 4 5 6 7

Clicking on a row, let us say the 4th row (ID=4), passes the values of the 1st, 2nd and 4th cell as query string variables to the 'R29a-SamplePage.aspx'



http://localhost:3639/ASPNETandjQuery/Section6-GridView/R29a-SamplePage.aspx?id=4&fname=Kathy&lname=Berry
Sample Page

```
EmployeeID= 4
EmployeeFirstName= Kathy
EmployeeLastName= Berry
```

The query string parameters are read using the Request.QueryString[] in SamplePage.aspx.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/eq#index

Recipe 30: Efficient Server Side Paging using jQuery

Challenge:

You want to perform efficient server side paging without using heavy server controls such as the GridView, and you're trying to eliminate a majority of ViewState that accompanies the GridView control.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Efficient Server Side Paging With ASP.NET And jQuery</title>
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script language="javascript" type="text/javascript">
        var pageSize = 10;
        function pageData(e) {
            var skip = e == 1 ? 0 : (e * pageSize) - pageSize;
            $.ajax({
                type: "POST",
                url: "R30-ServerSidePaging.aspx/FetchEmployees",
                data: "{skip:" + skip + ",take:" + pageSize + "}",
                contentType: "application/json; charset=utf-8",
                dataType: "json",
                async: true,
                cache: false,
                success: function(msg) {
                    printEmployees(msg);
                }
            });
            return false;
        }

         $(function() {
             $("#btnFetch").click(function(e) {
                 e.preventDefault();
                 $.ajax({
                     type: "POST",
                     url: "R30-ServerSidePaging.aspx/FetchEmployees",
                     data: "{skip:0,take:" + pageSize + "}",
                     contentType: "application/json; charset=utf-8",
                     dataType: "json",
                     async: true,
                     cache: false,
                     success: function(msg) {
                         var total = msg.d.TotalRecords;
                         if (total > 0) {
                             printEmployees(msg);
                             $("#paging").text("");
                           // Get the page count by dividing the total records
                             // by the page size.  This has to be rounded up
                             // otherwise you might not get the last page
                             var pageTotal = Math.ceil(total / pageSize);
                             for (var i = 0; i < pageTotal; i++) {
                                 $("#paging").append("<a href=\"#\"
onClick=\"pageData(" + (i + 1) + ")\">" + (i + 1) + "</a>&nbsp;");
                             }
```

```
                        }
                        else {
                            $("#paging").text("No records were found.");
                        }
                        $("#totalRecords").text("Total records: " + total);
                    }
                });
            });
        });

        // This function accepts an employee object
        // and prints the results to the div element.
        function printEmployees(msg) {
            $("#result").text("");
            var employees = msg.d.Employees;
            for (var i = 0; i < employees.length; i++) {
                $("#result").append(employees[i].ID + ", ");
                $("#result").append(employees[i].FName + ", ");
                $("#result").append(employees[i].MName + ", ");
                $("#result").append(employees[i].LName + ", ");
                $("#result").append(employees[i].Sex + "<br />");
            }
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Button ID="btnFetch" runat="server" Text="Fetch" />
        <span id="totalRecords"></span>
        <br /><br />
        <div id="result"></div>
        <div id="paging"></div>
    </form>
</body>
</html>
```

Explanation:

This recipe uses the EmployeeData.cs and Employee.cs classes, declared in Appendix ‘C’ of this eBook. The same code can also be found in the App_Code folder in the source code.

In the code above, we first create a variable pageSize, to hold the total number of records to display per page.

```
var pageSize = 10;
```

The user will fetch the records by clicking the button btnFetch. We make a *$.ajax()* call using the POST method, to the FetchEmployees() PageMethod and retrieve the results in msg.d. If you are not familiar with the *$.ajax()* call and what msg.d signifies, check the ‘Useful Links’ section of this Recipe.

The 'FetchEmployees' PageMethod declared in the code-behind file looks like this:

```
[WebMethod]
public static EmployeeData FetchEmployees(int skip, int take)
{
    var data = new EmployeeData();
    var emp = new Employee();
    data.Employees = emp.GetEmployeeList()
              .Skip(skip)
                      .Take(take)
                      .ToList();
    data.TotalRecords = emp.GetEmployeeList().Count;
    return data;
}
```

Once the data is retrieved from the server, the Employees data is returned in the msg.d.Employees property, and the total count is fetched using the msg.d.TotalRecords property.

```
success: function(msg) {
     var total = msg.d.TotalRecords;
```

To enable paging, the TotalRecords is divided by the pageSize value to calculate how many paging options need to be displayed.

```
var pageTotal = Math.ceil(total / pageSize);
```

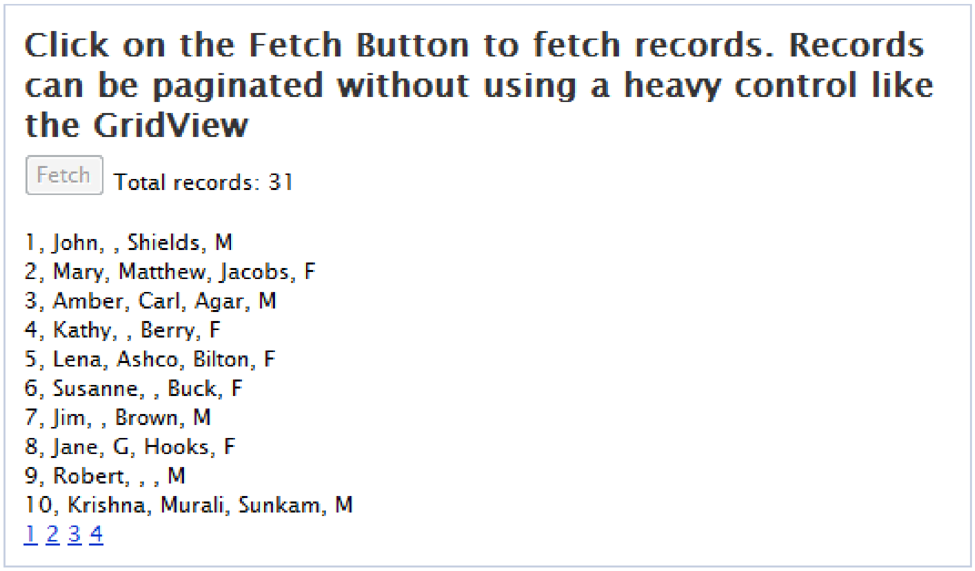
Each time the user clicks on a paging option, they'll run a JavaScript function called pageData. This function uses jQuery's Ajax functionality to call the server side code, but calculates the records to skip and take, by the user's selection.

```
function pageData(e) {
    var skip = e == 1 ? 0 : (e * pageSize) - pageSize;
    $.ajax({
        type: "POST",
        url: "R30-ServerSidePaging.aspx/FetchEmployees",
        data: "{skip:" + skip + ",take:" + pageSize + "}",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        async: true,
        cache: false,
        success: function(msg) {
            printEmployees(msg);
        }
    });
    return false;
}
```

Finally to reduce duplicate code, we have created a function called printEmployees. This function will be responsible for enumerating through each employee record and rendering it in the browser

```
function printEmployees(msg) {
    $("#result").text("");
    var employees = msg.d.Employees;
    for (var i = 0; i < employees.length; i++) {
        $("#result").append(employees[i].ID + ", ");
        $("#result").append(employees[i].FName + ", ");
        $("#result").append(employees[i].MName + ", ");
        $("#result").append(employees[i].LName + ", ");
        $("#result").append(employees[i].Sex + "<br />");
    }
}
```

If you run the code and click on the 'Fetch' button, you'll see the records are displayed ten at a time, as shown below:



Click on the Fetch Button to fetch records. Records can be paginated without using a heavy control like the GridView

```
Total records: 31

1, John, , Shields, M
2, Mary, Matthew, Jacobs, F
3, Amber, Carl, Agar, M
4, Kathy, , Berry, F
5, Lena, Ashco, Bilton, F
6, Susanne, , Buck, F
7, Jim, , Brown, M
8, Jane, G, Hooks, F
9, Robert, , , M
10, Krishna, Murali, Sunkam, M
1 2 3 4
```

When we fetch the employee records, we are enumerating through each record and then appending the results, separated by a comma. This approach of displaying data is not very UI friendly and is not maintainable. In the next Recipe (Recipe 31), we will use Client side templates to tackle this problem.

ASP.NET Date formatting is not handled well in JSON. The DateTime is serialized as an escaped JavaScript Date initializer - /Date(-124003800000)/. To support ASP.NET Date Formatting, use the jMSAjax Plug-in. DotNetCurry contains an article that demonstrates how to use this plug-in.

Live Demo

This chapter was contributed by Malcolm Sheridan.

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

Consume an ASP.NET WebService returning List<> with Dates using jQuery

http://docs.jquery.com/Ajax/jQuery.ajax#options

http://encosia.com/2008/05/29/using-jquery-to-directly-call-aspnet-ajax-page-methods/

Recipe 31: Handling the Master-Detail scenario in a GridView using jQuery

Challenge:

The user wants to handle a Master-Detail scenario using the GridView. When the user clicks on a GridView row, additional details should be displayed in a modal pop-up

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Display Master Details data using jQuery</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
       <link href="http://jqueryui.com/latest/themes/base/ui.all.css"
           rel="stylesheet" type="text/css" />
    <script src="../Scripts/jquery-1.3.2.min.js" type="text/javascript">
    </script>
    <script src="../Scripts/ui.core.js" type="text/javascript"></script>
    <script src="../Scripts/ui.resizable.js" type="text/javascript"></script>
    <script src="../Scripts/ui.dialog.js" type="text/javascript"></script>
    <script src="../Scripts/ui.draggable.js" type="text/javascript"></script>
       <script src="../Scripts/jquery-jtemplates.js" type="text/javascript"></script>

    <script type="text/javascript">
        $(function() {
            $(".gv > tbody > tr:not(:has(table, th))")
            .css("cursor", "pointer")
            .click(function(event) {
                var $row = $(this);
                var firstParam = $("td", $row).eq(0).text();
                $.ajax({
                    type: "POST",
                    url: "R31-MasterDetails.aspx/FetchOrders",
                    data: "{empId:" + firstParam + "}",
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    async: true,
                    cache: false,
                    success: function(msg) {
                        var orders = msg.d;
                        if (orders.length > 0) {
                            CreateTable(orders);
                        }
                        else {
                            $("#tbl").text("No records found");
                        }
                        $("#dialog").dialog({ width: 450, modal: true });
                        $("#dialog").dialog('open');
                    },
                  error: function(XMLHttpRequest, textStatus, errorThrown) {
                        alert(textStatus);
                    }

                });
            });
        });

        function CreateTable(msg) {
            $('#tbl').setTemplateURL('../Template/TemplateOrder.htm',
                                  null, { filter_data: false });
            $('#tbl').processTemplate(msg);
        }

    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Click on a Row to View Details</h2><br />
        <asp:GridView ID="grdEmployees" runat="server"
            AllowPaging="True" PageSize="5" AutoGenerateColumns="False"
            DataSourceID="ObjectDataSource1" class="gv">
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" />
                <asp:BoundField DataField="FName" HeaderText="FName" />
                <asp:BoundField DataField="MName" HeaderText="MName"  />
                <asp:BoundField DataField="LName" HeaderText="LName" />
                <asp:BoundField DataField="DOB" HeaderText="DOB"
                    DataFormatString="{0:MM/dd/yyyy}"/>
                <asp:BoundField DataField="Sex" HeaderText="Sex" />
            </Columns>
        </asp:GridView>
        <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetEmployeeList" TypeName="Employee">
        </asp:ObjectDataSource>
         <br />
    </div>
    <div id="dialog" title="Order Details">
          <div id="tbl"></div>
    </div>
    </form>
</body>
```

```
</html>
```

Explanation:

This recipe uses the Order.cs and Employee.cs classes, declared in Appendix ‘C’ of this eBook. The same classes can also be found in the App_Code folder in the source code.

In Recipe 29, we have already seen the reason for applying the filter shown below on the GridView rows, to select only those rows which are inside the TBODY.

```
$(".gv > tbody > tr:not(:has(table, th))")
```

When the user clicks on a GridView row, the current row object is stored in the ‘row’ variable and the *eq()* selector is then used to match the first cell of the clicked row, by providing the cell index 0.

```
var $row = $(this);
var firstParam = $("td", $row).eq(0).text();
```

We then make a *$.ajax()* call using the POST method, to the FetchOrders() PageMethod and retrieve the results in msg.d. If you are not familiar with the *$.ajax()* call and what msg.d signifies, check the ‘Useful Links’ section of this Recipe. The PageMethod looks like this:

```
    [WebMethod]
    public static List<Orders> FetchOrders(int empId)
    {
        var orderList = new Orders();
        var fetchOrders = orderList.GetOrderDetails()
            .Where(order => order.ID == empId);
        return fetchOrders.ToList();
    }
```

The code in the PageMethod is quite simple. The employee id is passed from the GridView and the Orders taken by that Employee are fetched using the GetOrderDetails() method of the Order class.

Once the data is received, the next step is to display the results in a modal popup. Now usually one would think of looping through the data and create a table dynamically to display the results. Although this looks like a solution, it is not maintainable.

Enter templates! I prefer Client side templates when I come across a similar situation of creating HTML elements on the fly from a data source. These templates help you to separate HTML from your JavaScript. There is a brilliant plugin called jTemplates that makes the work easier for us. Dave from encosia.com has an excellent blog post that shows how to use jTemplates

Create a simple HTML template (TemplateOrder.htm) in the ‘Templates’ folder as shown below:

```
<table>
  <thead>
    <tr>
      <th>OrderID</th>
      <th>Country</th>
      <th>CustomerName</th>
    </tr>
  </thead>
  <tbody>
    {#foreach $T as data}
    <tr>
      <td>{$T.data.OrderID}</td>
      <td>{$T.data.Country}</td>
      <td>{$T.data.CustomerName}</td>
    </tr>
    {#/for}
  </tbody>
</table>
```

Now create a function that accepts the JSON results retrieved from the PageMethod and uses jTemplate's method to load the HTML template into a container, and finally render the JSON result.

```
function CreateTable(msg) {
    $('#tbl').setTemplateURL('../Template/TemplateOrder.htm',
                         null, { filter_data: false });
    $('#tbl').processTemplate(msg);
}
```

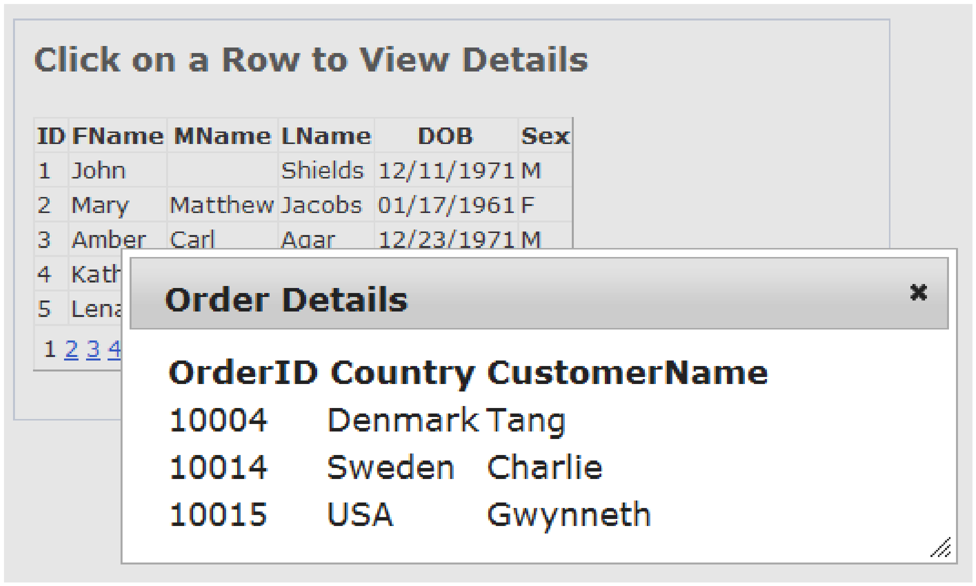
All that's left is to call CreateTable() as shown below:

```
success: function(msg) {
    var orders = msg.d;
    if (orders.length > 0) {
        CreateTable(orders);
    }
```

The results are shown using the jQuery UI Dialog plug-in using the following code:

```
$("#dialog").dialog({ width: 450, modal: true });
$("#dialog").dialog('open');
```

When the user clicks on a GridView row, say ID= 2, we get a modal pop-up similar to the one shown below:



Click on a Row to View Details

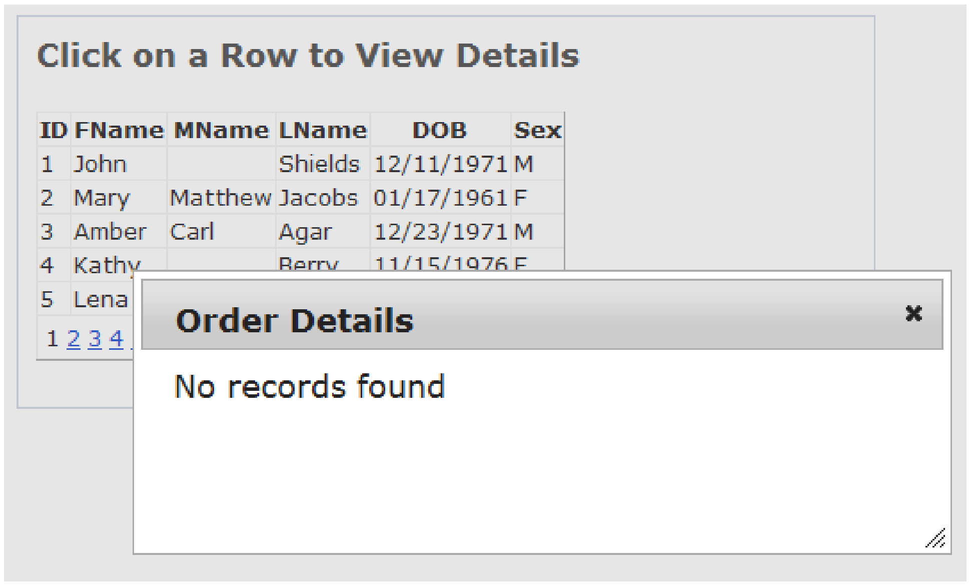
ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
3	Amber	Carl	Agar	12/23/1971	M
4	Kath				
5	Lena				

1 2 3 4

Order Details

OrderID	Country	CustomerName
10004	Denmark	Tang
10014	Sweden	Charlie
10015	USA	Gwynneth

When the Row has no associated records (like for ID=3), you get the 'No records found' message as shown below:



Click on a Row to View Details

ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
3	Amber	Carl	Agar	12/23/1971	M
4	Kathy		Berry	11/15/1976	F
5	Lena				

1 2 3 4

Order Details

No records found

[Live Demo]

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Ajax/jQuery.ajax#options

http://docs.jquery.com/UI/Dialog

http://plugins.jquery.com/project/jTemplates

http://encosia.com/2008/05/29/using-jquery-to-directly-call-aspnet-ajax-page-methods/

http://encosia.com/2009/02/10/a-breaking-change-between-versions-of-aspnet-ajax/

Recipe 32: Click and Retrieve the Value of a GridView Cell using jQuery

Challenge:

You want to click and retrieve data from a GridView cell.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Click and retrieve Cell Value</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            $(".gv > tbody > tr:not(:has(table, th))")
                .css("cursor", "pointer")
                .click(function(e) {
                    $(".gv td").removeClass("highlite");
                    var $cell = $(e.target).closest("td");
                    $cell.addClass('highlite');
                    var $currentCellText = $cell.text();
                    var $leftCellText = $cell.prev().text();
                    var $rightCellText = $cell.next().text();
                    var $colIndex = $cell.parent().children().index($cell);
                    var $colName = $cell.closest("table")
                        .find('th:eq(' + $colIndex + ')').text();
                    $("#para").empty()
                    .append("<b>Current Cell Text: </b>"
                        + $currentCellText + "<br/>")
                    .append("<b>Text to Left of Clicked Cell: </b>"
                        + $leftCellText + "<br/>")
                    .append("<b>Text to Right of Clicked Cell: </b>"
                        + $rightCellText + "<br/>")
                    .append("<b>Column Name of Clicked Cell: </b>"
                        + $colName)
                });

        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Click on any of the table cells to retrieve information
            about the cell and its surrounding elements</h2><br />
        <asp:GridView ID="grdEmployees" runat="server"
             AllowPaging="True" PageSize="5" AutoGenerateColumns="False"
            DataSourceID="ObjectDataSource1" class="gv">
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" />
                <asp:BoundField DataField="FName" HeaderText="FName" />
                <asp:BoundField DataField="MName" HeaderText="MName"  />
                <asp:BoundField DataField="LName" HeaderText="LName" />
                <asp:BoundField DataField="DOB" HeaderText="DOB"
                    DataFormatString="{0:MM/dd/yyyy}"/>
                <asp:BoundField DataField="Sex" HeaderText="Sex" />
            </Columns>
        </asp:GridView>
        <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetEmployeeList" TypeName="Employee">
        </asp:ObjectDataSource>
         <br />
        <p id="para"></p>
    </div>

    </form>
</body>
</html>
```

Explanation:

This recipe uses the Employee.cs class, declared in Appendix 'C' of this eBook. The same code can also be found in the App_Code folder in the source code.

In Recipe 29, we have already seen the reason for applying the filter (shown below) on the GridView rows, to selects only those rows which are inside the TBODY.

```
$(".gv > tbody > tr:not(:has(table, th))")
```

This recipe uses event delegation. Observe how we have used 'e.target' to find out the element that was clicked. This object is cached in the 'cell' variable.

```
var $cell = $(e.target).closest("td");
```

The *closest()* as given in the jQuery documentation, “works by first looking at the current element to see if it matches the specified expression, if so it just returns the element itself. If it doesn't match then it will continue to traverse up the document, parent by parent, until an element is found that matches the specified expression.”

With the help of the ‘cell’ variable, we can use the DOM tree traversal methods like *prev()* and *next()*, to retrieve the value of the immediate ‘preceding’ and ‘following’ elements, respectively.

```
var $leftCellText = $cell.prev().text();
var $rightCellText = $cell.next().text();
```

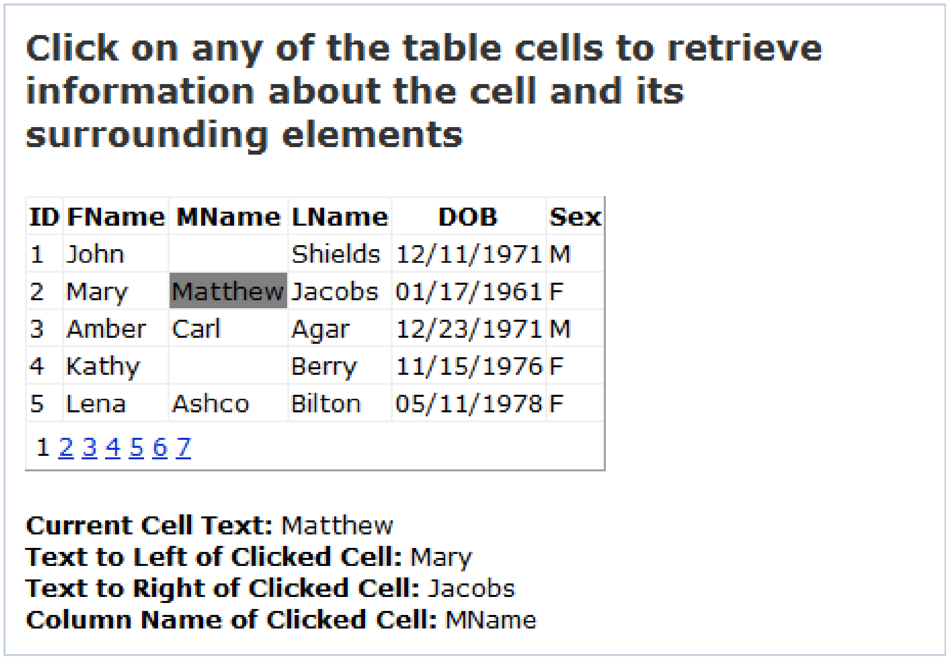
Similarly the column header text is retrieved using the code shown below.

```
var $colIndex = $cell.parent().children().index($cell);
var $colName = $cell.closest("table")
                .find('th:eq(' + $colIndex + ')').text();
```

As shown above, after retrieving the column index, we use the *closest()* method to traverse up the DOM, parent by parent until we find the table element. The Header text is then selected using

```
('th:eq(' + $colIndex + ')').text()
```

When a Cell of the GridView is clicked (for example 2nd Row and 3rd Column), the result is as shown below:



Click on any of the table cells to retrieve information about the cell and its surrounding elements

ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
3	Amber	Carl	Agar	12/23/1971	M
4	Kathy		Berry	11/15/1976	F
5	Lena	Ashco	Bilton	05/11/1978	F

1 2 3 4 5 6 7

Current Cell Text: Matthew
Text to Left of Clicked Cell: Mary
Text to Right of Clicked Cell: Jacobs
Column Name of Clicked Cell: MName

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Traversing

Recipe 33: Highlight and Remove GridView Rows using jQuery

Challenge:

You have a GridView that displays several rows of data on a single page to be printed. The user wants to be able to select and temporarily remove the rows he/she does not want to view or print.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Highlight and Remove Rows in a GridView</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            $(".gv > tbody > tr:not(:has(table, th))")
                .css("cursor", "pointer")
                .click(function() {
                    $(this).toggleClass('highlite');
                });

            $(document).keyup(function(e) {
                var key = (e.keyCode ? e.keyCode : e.charCode);
                if (key == 68) { // D OR d
                    var cnt = $("tr.highlite").length;
                    $("tr.highlite").remove();
                    $("#para")
                    .html("<b>" + cnt + "</b>" + " row(s) deleted!!");

                }
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Click on the Rows to Highlight them. Press 'd' or 'D'
        to Remove the Rows</h2><br />
        <p id="para"></p>
        <asp:GridView ID="grdEmployees" runat="server"
```

```
            AutoGenerateColumns="False"
             DataSourceID="ObjectDataSource1" class="gv">
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" />
                <asp:BoundField DataField="FName" HeaderText="FName" />
                <asp:BoundField DataField="MName" HeaderText="MName"  />
                <asp:BoundField DataField="LName" HeaderText="LName" />
                <asp:BoundField DataField="DOB" HeaderText="DOB"
                    DataFormatString="{0:MM/dd/yyyy}"/>
                <asp:BoundField DataField="Sex" HeaderText="Sex" />
            </Columns>
        </asp:GridView>
        <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetEmployeeList" TypeName="Employee">
        </asp:ObjectDataSource>
        <br /><br />
    </div>
    </form>
</body>
</html>
```

Explanation:

In many applications, I have frequently observed data being shown in the GridView control, without pagination. This is usually done when the records in the GridView are to be printed.

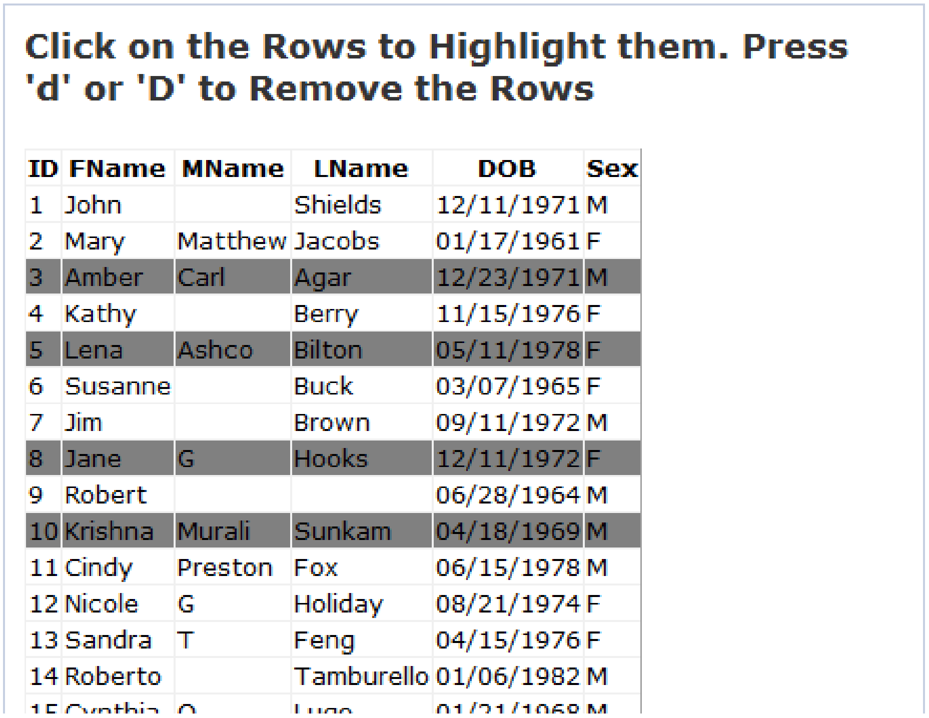
One common requirement in such cases is to be able to prevent some rows from being printed. Although this can be achieved easily while the data is being pulled from the database or any other data source, many a times the user requires this functionality to be available at runtime. A simple way to achieve this requirement is to give the user a functionality to remove some rows on the click of a button and then print the records.

In Recipe 29, we have already seen the reason for applying the filter on the GridView rows to select only those rows which are inside the TBODY. This recipe uses the Employee.cs class, declared in Appendix ‘C’ of this eBook. The same code can also be found in the App_Code folder in the source code.

In this example, we use the *toggleClass()* to toggle the class 'highlite' when a table row is clicked.

```
$(this).toggleClass('highlite');
```

After clicking on the 3rd, 5th, 8th and 10th row, the result is as shown below:



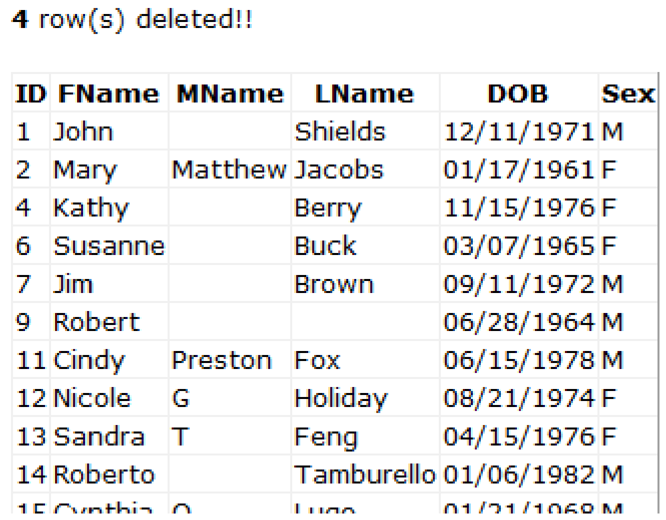
Click on the Rows to Highlight them. Press 'd' or 'D' to Remove the Rows

ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
3	Amber	Carl	Agar	12/23/1971	M
4	Kathy		Berry	11/15/1976	F
5	Lena	Ashco	Bilton	05/11/1978	F
6	Susanne		Buck	03/07/1965	F
7	Jim		Brown	09/11/1972	M
8	Jane	G	Hooks	12/11/1972	F
9	Robert			06/28/1964	M
10	Krishna	Murali	Sunkam	04/18/1969	M
11	Cindy	Preston	Fox	06/15/1978	M
12	Nicole	G	Holiday	08/21/1974	F
13	Sandra	T	Feng	04/15/1976	F
14	Roberto		Tamburello	01/06/1982	M
15	Cynthia	O	Lugo	01/21/1968	M

After the rows have been selected, press the ‘d’ or ‘D’ key to remove the rows.

```
$(document).keyup(function(e) {
    var key = (e.keyCode ? e.keyCode : e.charCode);
    if (key == 68) { // D OR d
        var cnt = $("tr.highlite").length;
        $("tr.highlite").remove();
        $("#para")
        .html("<b>" + cnt + "</b>" + " row(s) deleted!!");

    }
});
```

The key is detected using the keyCode or charCode. If the key == 68, i.e. either ‘d’ or ‘D’ is pressed, the rows that have the ‘highlite’ class applied to them (the ones in Gray) are removed. The *remove()* removes all matched elements from the DOM. After the rows are removed, the output is as shown below:



4 row(s) deleted!!

ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
4	Kathy		Berry	11/15/1976	F
6	Susanne		Buck	03/07/1965	F
7	Jim		Brown	09/11/1972	M
9	Robert			06/28/1964	M
11	Cindy	Preston	Fox	06/15/1978	M
12	Nicole	G	Holiday	08/21/1974	F
13	Sandra	T	Feng	04/15/1976	F
14	Roberto		Tamburello	01/06/1982	M
15	Cynthia	O	Lugo	01/21/1968	M

Printing the GridView control is not covered in this eBook. However you can check my article on dotnetcurry.com to get some ideas on how to print the GridView control. Do not use the technique demonstrated here with large datasets.

This example does not support rowspan.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Attributes/toggleClass

http://docs.jquery.com/Manipulation/remove

Recipe 34: Highlight and Remove GridView Columns using jQuery

Challenge:

You have a GridView that displays several columns of data on a single page, to be printed. The user wants to be able to select and temporarily remove the column he does not want to view or print.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Highlight and Remove Columns in a GridView</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            $(".gv th")
            .css("cursor", "pointer")
                .click(function(e) {
                 var $idx = $(this).parent().children().index($(this));
         $(".gv th:eq(" + $idx + "),.gv tr td:nth-child(" + ($idx + 1) + ")")
                    .toggleClass("highlite");
                });

            $(document).keyup(function(event) {
                var key = event.keyCode || event.charCode || 0;
                if (key == 68) { // D OR d
                    var cnt = $("th.highlite").length;
                    $("td.highlite, th.highlite").remove();
                    $("#para")
                    .html("<b>" + cnt + "</b>" + " column(s) deleted!!");
                }
            });

        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the Column Headings to Highlight them. Press 'd' or 'D'
        to Remove the Columns</h2><br />
        <p id="para"></p>
        <asp:GridView ID="grdEmployees" runat="server"
            AutoGenerateColumns="False"
            DataSourceID="ObjectDataSource1" class="gv">
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" />
                <asp:BoundField DataField="FName" HeaderText="FName" />
                <asp:BoundField DataField="MName" HeaderText="MName"  />
                <asp:BoundField DataField="LName" HeaderText="LName" />
                <asp:BoundField DataField="DOB" HeaderText="DOB"
                    DataFormatString="{0:MM/dd/yyyy}"/>
                <asp:BoundField DataField="Sex" HeaderText="Sex" />
            </Columns>
        </asp:GridView>
        <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetEmployeeList" TypeName="Employee"></asp:ObjectDataSource>
    </div>
    </form>
</body>
</html>
```

Explanation:

This recipe uses the Employee.cs class, declared in Appendix ‘C’ of this eBook. The same code can also be found in the App_Code folder in the source code.

In Recipe 33, we saw how to highlight and remove GridView ‘rows’ using jQuery. The technique adopted for ‘removing’ the GridView columns will be similar to the one shown for the GridView rows. However the technique adopted for Column ‘highlighting’ differs as explained below.

Observe that the click is captured only on the GridView Header using the selector `$(".gv th")`. The current column index is retrieved using

```
var $idx =  $(this).parent().children().index($(this))
```

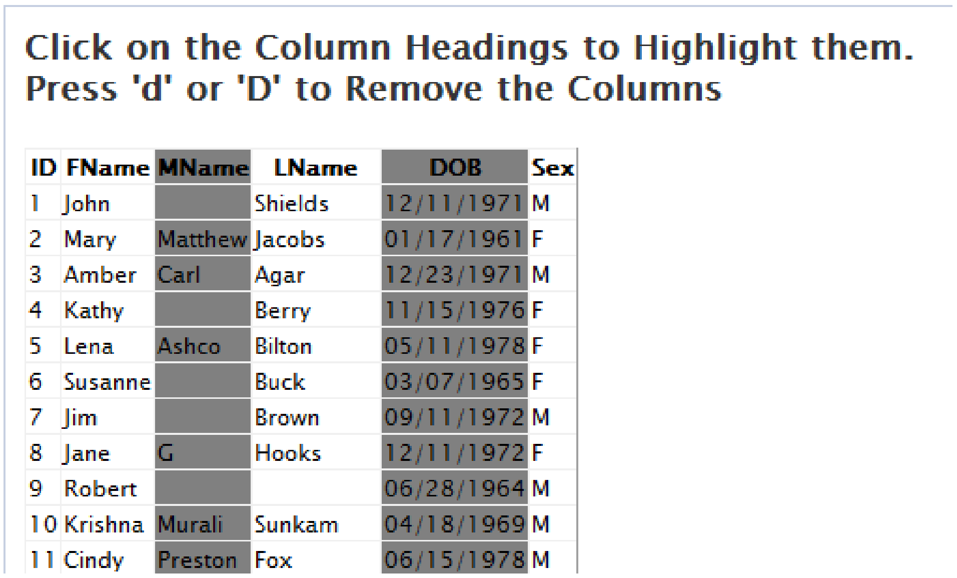
To highlight the header column and subsequent child cells in the same column, we use this line of code

```
$(".gv th:eq(" + $idx + "),.gv tr td:nth-child(" + ($idx + 1) + ")")
.toggleClass("highlite")
```

The `th:eq(" + $idx + ")` selector is used here to match the header cell clicked. Observe the CSS selector `td:nth-child(" + ($idx + 1) + ")` where we add ‘one’ to the column index. This is done since the *:nth-child()* CSS selector is one-based rather than zero-based.

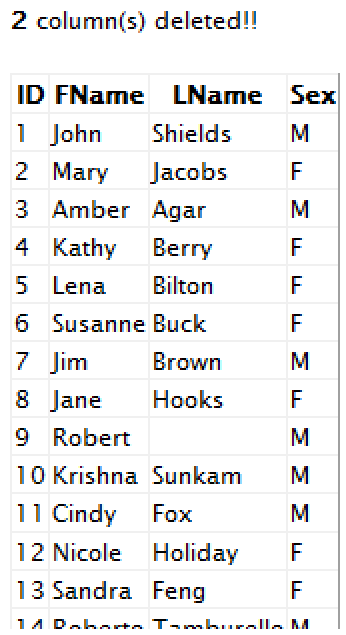
The *toggleClass()* toggles the class 'highlite' when a table column is clicked. To delete the column, press either ‘d’ or ‘D’.

After clicking on the 3rd and 5th column headers, the result is as shown below:


Click on the Column Headings to Highlight them.
Press 'd' or 'D' to Remove the Columns

ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
3	Amber	Carl	Agar	12/23/1971	M
4	Kathy		Berry	11/15/1976	F
5	Lena	Ashco	Bilton	05/11/1978	F
6	Susanne		Buck	03/07/1965	F
7	Jim		Brown	09/11/1972	M
8	Jane	G	Hooks	12/11/1972	F
9	Robert			06/28/1964	M
10	Krishna	Murali	Sunkam	04/18/1969	M
11	Cindy	Preston	Fox	06/15/1978	M

After the columns have been selected, press the 'd' or 'D' key to remove the columns as explained in Recipe 33. After the columns have been removed, the screen will look similar to the following.



2 column(s) deleted!!

ID	FName	LName	Sex
1	John	Shields	M
2	Mary	Jacobs	F
3	Amber	Agar	M
4	Kathy	Berry	F
5	Lena	Bilton	F
6	Susanne	Buck	F
7	Jim	Brown	M
8	Jane	Hooks	F
9	Robert		M
10	Krishna	Sunkam	M
11	Cindy	Fox	M
12	Nicole	Holiday	F
13	Sandra	Feng	F
14	Roberto	Tamburello	M

This example does not support 'colspan'. This example should be used on smaller datasets.

[Live Demo]

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Attributes/toggleClass

http://docs.jquery.com/Manipulation/remove

Recipe 35: How to Search and Highlight a GridView Cell using jQuery

Challenge:

A basic 'Search' feature needs to be added to the GridView. As the user enters text in a TextBox, the text should be searched in the GridView and the cells containing the text should be highlighted. The functionality should work even when pagination is enabled on the GridView.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Search and Highlight GridView</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
            var $txtBox = $('input[id$=txtSearch]');
            $txtBox.keyup(function(e) {
                searchText();
            });

            function searchText() {
                var $txt = $txtBox.val().toLowerCase();
                $(".gv td").removeClass("highlite");
                if ($txt) {
                    $(".gv > tbody > tr > td:not(:has(table, tr))")
                    .filter(function() {
                     return $(this).text().toLowerCase().indexOf($txt) != -1;
                    }).addClass('highlite');
                }
            }

            searchText();
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Type in the TextBox to Search the GridView Cells</h2><br />
        Enter Text Here: <asp:TextBox ID="txtSearch" runat="server" />
        <br /><br />
        <asp:GridView ID="grdEmployees" runat="server"
            AllowPaging="True" PageSize="5" AutoGenerateColumns="False"
            DataSourceID="ObjectDataSource1" class="gv">
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" />
                <asp:BoundField DataField="FName" HeaderText="FName" />
                <asp:BoundField DataField="MName" HeaderText="MName"  />
                <asp:BoundField DataField="LName" HeaderText="LName" />
                <asp:BoundField DataField="DOB" HeaderText="DOB"
                    DataFormatString="{0:MM/dd/yyyy}" />
                <asp:BoundField DataField="Sex" HeaderText="Sex" />
            </Columns>
        </asp:GridView>
        <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetEmployeeList" TypeName="Employee">
        </asp:ObjectDataSource>
        <br />
        Tip: The search also works when you paginate through
        the GridView
    </div>
    </form>
</body>
</html>
```

Explanation:

This recipe uses the Employee.cs class, declared in Appendix 'C' of this eBook. The same code can also be found in the App_Code folder in the source code.

The searchText() method is called during page load, as well as on the keyup event of the TextBox.

```
var $txtBox = $('input[id$=txtSearch]');
$txtBox.keyup(function(e) {
    searchText();
});
```

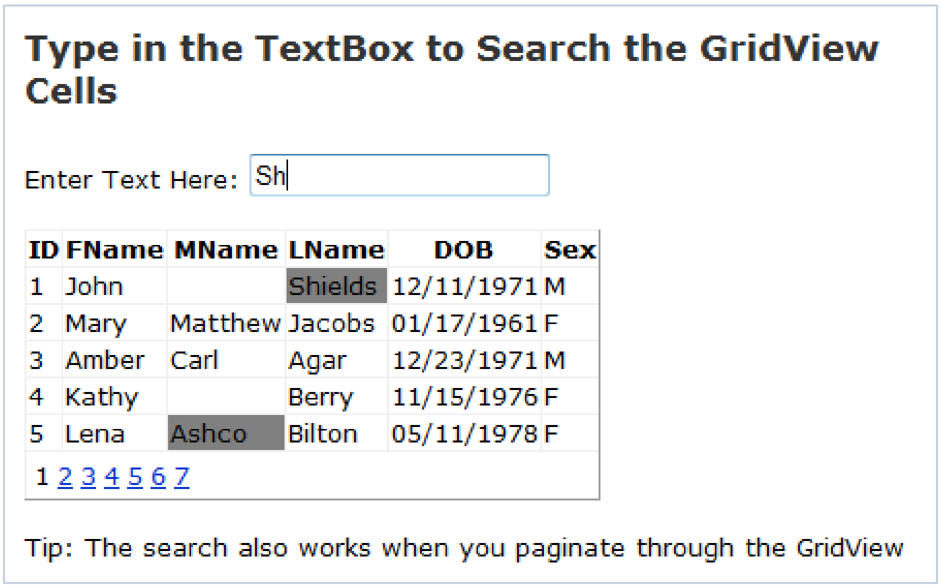
If there is no text in the TextBox, the search does not take place.

As the user types text, we use the *filter()* method to reduce the set of matched TD's to the ones that contain the search text. The CSS selector `".gv > tbody > tr > td:not(:has(table, tr))"` makes sure that the header and footer rows of the GridView are not included in the search.

When the user starts typing text in the TextBox, the text is searched 'everywhere' in the current page of the GridView and the matched elements (cells) are highlighted using *addClass()* as shown below:

```
if ($txt) {
    $(".gv > tbody > tr > td:not(:has(table, tr))")
    .filter(function() {
      return $(this).text().toLowerCase().indexOf($txt) != -1;
    }).addClass('highlite');
}
```

The output after typing two characters 'Sh' is as shown below. The cells having the text 'Sh' get highlighted.



Type in the TextBox to Search the GridView Cells

Enter Text Here: Sh

ID	FName	MName	LName	DOB	Sex
1	John		Shields	12/11/1971	M
2	Mary	Matthew	Jacobs	01/17/1961	F
3	Amber	Carl	Agar	12/23/1971	M
4	Kathy		Berry	11/15/1976	F
5	Lena	Ashco	Bilton	05/11/1978	F

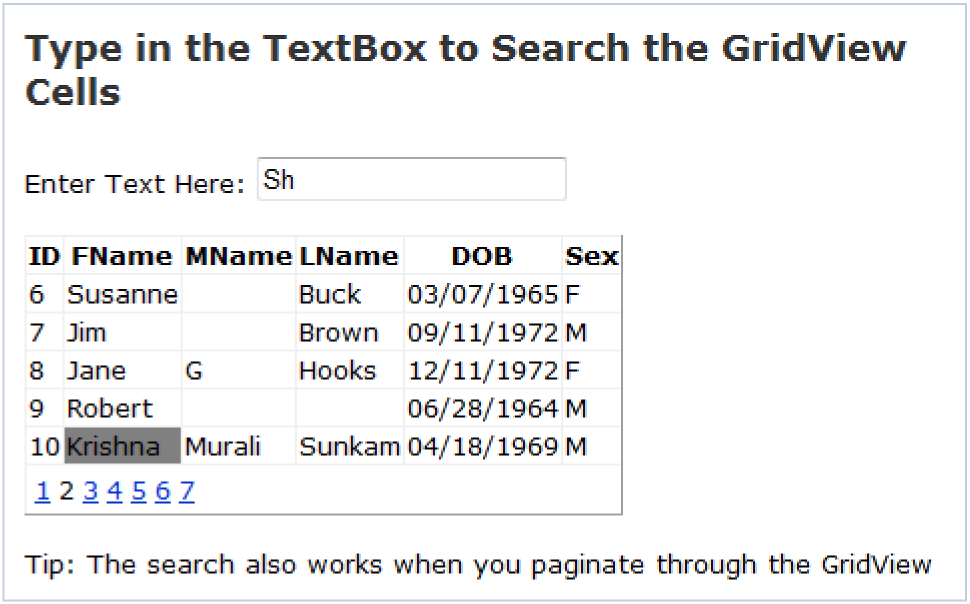
1 2 3 4 5 6 7

Tip: The search also works when you paginate through the GridView

The search works even while paginating, since paging causes a postback which in turn invokes the searchText() method.

```
searchText();
```

So assuming that the textbox still contains ‘Sh’, when the user moves to the second page, the cells having text ‘Sh’ get highlighted. The output is shown below:



Type in the TextBox to Search the GridView Cells

Enter Text Here: Sh

ID	FName	MName	LName	DOB	Sex
6	Susanne		Buck	03/07/1965	F
7	Jim		Brown	09/11/1972	M
8	Jane	G	Hooks	12/11/1972	F
9	Robert			06/28/1964	M
10	Krishna	Murali	Sunkam	04/18/1969	M

1 2 3 4 5 6 7

Tip: The search also works when you paginate through the GridView

We could have used the *:contains* selector here, but remember that *:contains* is case-sensitive. So if you type ‘a’, you will not get ‘A’ in the search result.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Traversing/filter

Recipe 36: Quickly add Events to Controls inside the GridView using jQuery

Challenge:

There are TextBox controls inside the GridView. You want to add events to them at runtime to restrict text entered by the user.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Add Events to Controls inside the GridView </title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

    <script type="text/javascript">
        $(function() {
        $(".gv input.rate").bind('keyup blur', function(e) {
                if (this.value.match(/[^a-zA-Z ]/g)) {
                    this.value = this.value.replace(/[^a-zA-Z ]/g, '');
                }
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>The TextBoxes in the Comments column can accept only
        alphabets. Non-alphabetic characters are rejected</h2><br />
        <asp:GridView ID="grdEmployees" runat="server"
             AllowPaging="True" PageSize="5" AutoGenerateColumns="False"
            DataSourceID="ObjectDataSource1" class="gv">
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" />
                <asp:BoundField DataField="FName" HeaderText="FName" />
                <asp:BoundField DataField="MName" HeaderText="MName"  />
                <asp:BoundField DataField="LName" HeaderText="LName" />
                <asp:BoundField DataField="DOB" HeaderText="DOB"
                    DataFormatString="{0:MM/dd/yyyy}"/>
                <asp:BoundField DataField="Sex" HeaderText="Sex" />
                <asp:TemplateField HeaderText="Comments">
                  <ItemTemplate>
                     <asp:TextBox id="txtComments" runat="server"
                       MaxLength="25" class="rate" />
                  </ItemTemplate>
                </asp:TemplateField>
            </Columns>
        </asp:GridView>
        <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
            SelectMethod="GetEmployeeList" TypeName="Employee">
        </asp:ObjectDataSource>
         <br />
        <p id="para"></p>
    </div>

    </form>
</body>
</html>
```

Explanation:

The technique demonstrated in this recipe is simple yet very effective when you want to add an event to an existing control in the GridView. In this example, we will add the keyup and blur events to the textboxes in the GridView and restrict these textboxes to accept only alphabets. This is done using the *bind()* function as shown below:

```
$(".gv input.rate").bind('keyup blur', function(e) {
    if (this.value.match(/[^a-zA-Z ]/g)) {
        this.value = this.value.replace(/[^a-zA-Z ]/g, '');
    }
});
```

Now when the user tries to enter any character other than an alphabet, the character is replaced with an empty string. However, if the user right clicks the textbox and uses the context menu to paste text, then the non-alphabetic characters are replaced only when the textbox loses focus.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Events/bind

http://www.devcurry.com/2009/03/how-to-quickly-add-key-up-event-to-all.html

Section 7: jQuery and the ASP.NET Image Control

Introduction

The ASP.NET Image Control is used to display an image on your WebForm

The Image control is declared like this in your WebForm:

```
<asp:Image ID="Image1" runat="server" ImageUrl="~/images/1.jpg" />
```

and gets rendered in the browser as:

```
<img id="Image1" src="../images/1.jpg" style="border-width:0px;" />
```

In this section, I will show you how to use jQuery to create some useful effects and animations on the Image control. This section will contain the following recipes:

Recipe 37: Click and Increase the Size of an Image using jQuery

Challenge:

Your page has a number of thumbnails. You want to click and view a larger image when the thumbnail is clicked on.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Click and Increase the Size of an Image</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src='../Scripts/jquery-1.3.2.min.js'
        type='text/javascript'>
    </script>

     <script type="text/javascript">
         $(function() {
             $("img.imgthumb").click(function(e) {
                 var newImg = '<img src='
                               + $(this).attr("src") + '></img>';
                 $('#ladiv')
                    .html($(newImg)
                    .animate({ height: '300', width: '450' }, 1500));
             });
         });

     </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="imgdiv">
        <h2>Click on the thumbnail to view a large image</h2>
        <br />
        <asp:Image ID="imgA" class="imgthumb" runat="server"
            ImageUrl="../images/1.jpg" />
        <asp:Image ID="imgB" class="imgthumb" runat="server"
            ImageUrl="../images/2.jpg" />
        <asp:Image ID="imgC" class="imgthumb" runat="server"
            ImageUrl="../images/3.jpg" />
        <asp:Image ID="Image1" class="imgthumb" runat="server"
            ImageUrl="../images/4.jpg" />
        <hr /><br />
        <div id="ladiv"></div>
    </div>
    </form>
</body>
</html>
```

Explanation:

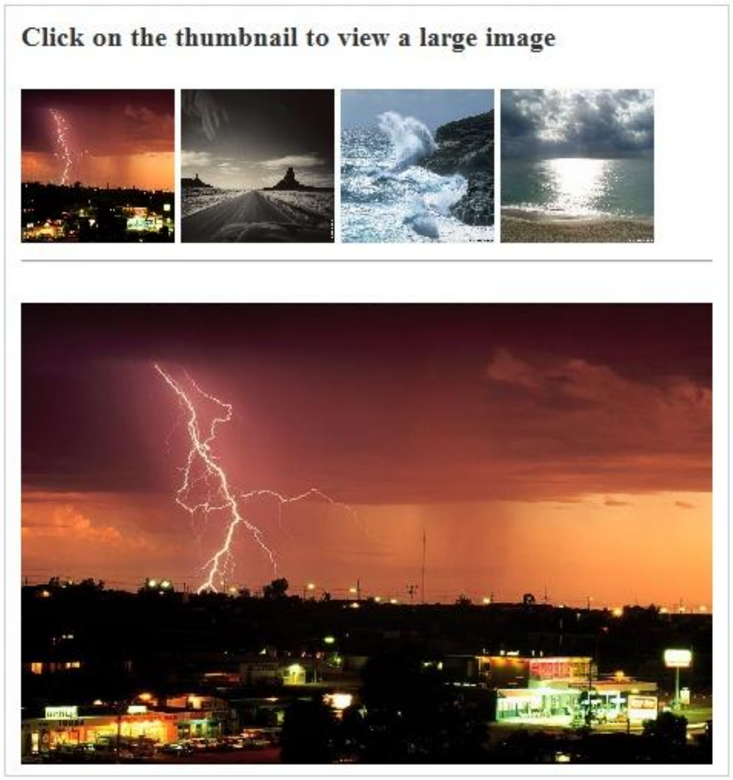
This recipe demonstrates how to increase the size of an image when it is clicked. To give it an Image gallery effect, when a thumbnail is clicked, we create a new image element and set its source, to the source of the clicked thumbnail.

```
var newImg = '<img src=' + $(this).attr("src") + '></img>';
```

The next and final step is to set the html contents of a div element to the 'newImg' variable and then increase the image size, by animating the height and width of the image.

```
$('#ladiv')
    .html($(newImg)
    .animate({ height: '300', width: '450' }, 1500));
```

When you run the application, click on the thumbnail to see a large version of the image with animation, as shown below. Additionally, you can also preload the images for better performance.


Click on the thumbnail to view a large image

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Effects/animate

Recipe 38: Change Image Opacity on MouseOver using jQuery

Challenge:

You want to change the opacity of an image when the user hovers the mouse over an image.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Change Image Opacity on Hover</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('.imgOpa').each(function() {
                $(this).hover(
                    function() {
                        $(this).stop().animate({ opacity: 1.0 }, 800);
                    },
                   function() {
                       $(this).stop().animate({ opacity: 0.3 }, 800);
                   })
                });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <h2>Hover over an Image to change its Transparency level</h2>
        <br />
        <asp:Image ID="Image1" runat="server"
            ImageUrl="../images/1.jpg" class="imgOpa" />
        <asp:Image ID="Image2" runat="server"
            ImageUrl="../images/2.jpg" class="imgOpa" />
        <asp:Image ID="Image3" runat="server"
            ImageUrl="../images/3.jpg" class="imgOpa" />
        <asp:Image ID="Image4" runat="server"
            ImageUrl="../images/4.jpg" class="imgOpa" />
    </div>
    </form>
</body>
</html
```

Explanation:

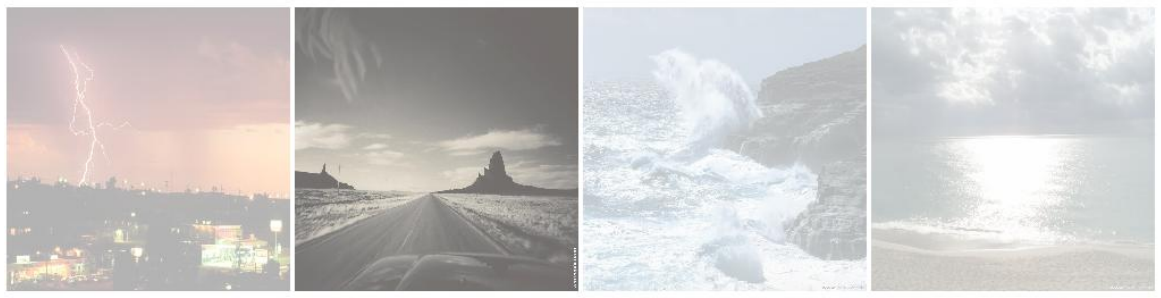
In this example, observe that the images have a class attribute ‘imgOpa’. The definition of the CSS class is as shown here:

```
.imgOpa
{
    height:250px;
    width:250px;
    opacity:0.3;
    filter:alpha(opacity=30);
}
```

When the images are loaded, they are in a semitransparent state. In Firefox, Chrome and Safari, we use the *opacity:n* property for transparency. The opacity value can be from 0.0 - 1.0, where a lower value makes the element more transparent.

In IE 7 and later, we use *filter:alpha(opacity=n)* property for transparency, where the opacity value can be from 0-100.

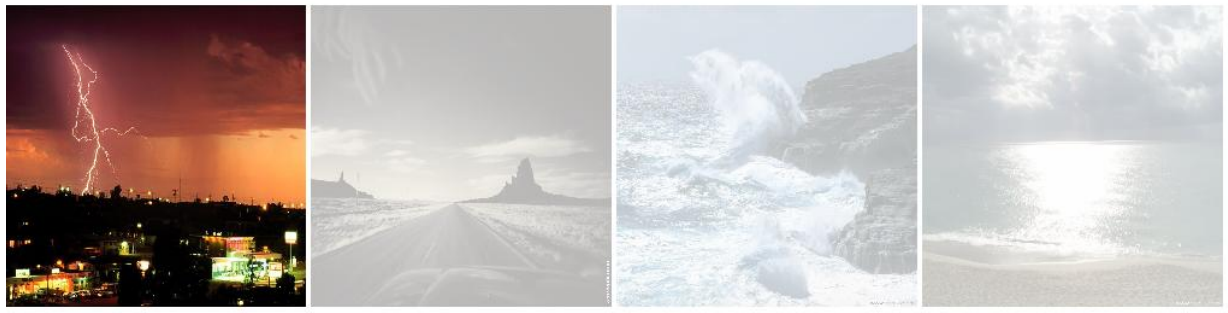
This is how the images appear in a semitransparent when they are loaded:



When the user hovers the mouse over a semitransparent image, we use the jQuery *hover()* method to animate the opacity property from 0.3 to 1.0, thereby creating a cool effect on the images. The code to achieve this effect is shown below:

```
$(this).hover(
    function() {
        $(this).stop().animate({ opacity: 1.0 }, 800);
    },
   function() {
       $(this).stop().animate({ opacity: 0.3 }, 800);
   })
});
```

The screenshot shown below shows how the image will appear when the user hovers the mouse over an image; let us say the first image. The transparency of the image is changed from 0.3 to 1.0.



Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Events/hover

Recipe 39: Create an Image SlideShow using jQuery

Challenge:

You want to create a simple Image SlideShow

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Simple Image SlideShow</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {

            var imgs = [
                   '../images/1.jpg',
                   '../images/2.jpg',
                   '../images/3.jpg',
                   '../images/4.jpg'];
            var cnt = imgs.length;

             var $imageSlide = $('img[id$=imageSlide]');
            // set the image control to the last image
            $imageSlide.attr('src', imgs[cnt-1]);

            setInterval(Slider, 3000);

            function Slider() {
                $imageSlide.fadeOut("slow", function() {
                    $(this).attr('src', imgs[(imgs.length++) % cnt])
                    .fadeIn("slow");
                });
            }
        });

    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Image Slide Show - Image Changes Every 3 Seconds</h2><br />
        <asp:Image ID="imageSlide" runat="server" class="imgdiv" />
    </div>
    </form>
</body>
</html>
```

Explanation:

This recipe shows you how to create a simple image slideshow that works in modern browsers. This example starts by declaring an array containing image paths. We have declared an image control on the page. When the page loads, the image path of this control is set to the last element of the array. I will explain shortly why this is needed.

```
var $imageSlide = $('img[id$=imageSlide]');
// set the image control to the last image
$imageSlide.attr('src', imgs[cnt-1]);
```

We then use the JavaScript *setInterval()* function which delays execution of the Slider function for a specific time, in our case 3000 millisecond(3 seconds). The advantage of a *setInterval()* function is that it continues to repeat the process of triggering the function at a specified interval, thereby sliding the images in a cycle. If you want to pause the slideshow, use the *clearInterval()* function.

Since the Slider function is first called after a 3 seconds delay, hence we explicitly set the image control source to the last image path in the array. If we do not do so, the user does not see an image for the first 3 seconds.

With every loop, we set the image control source to the next element in the array using the expression `imgs[(imgs.length++) % cnt]` and apply the *fadeIn()* and *fadeOut()* effect.

```
function Slider() {
    $(imageSlide).fadeOut("slow", function() {
        $(this).attr('src', imgs[(imgs.length++) % cnt])
        .fadeIn("slow");
    });
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Effects

https://developer.mozilla.org/En/Window.setInterval

Recipe 40: Image Swapping using jQuery

Challenge:

When the user clicks on an image displayed in an ImageControl, you want it to swap with another image

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Swap Image on Click</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
        $('img[id$=imageSwap]').toggle(
          function() {
              $(this).attr("src", "../images/Hide.gif");
          },
          function() {
              $(this).attr("src", "../images/Show.gif");
          }
        );
        });
    </script>
```

```
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the image to swap</h2>
        <br /><br />
        <asp:Image ID="imageSwap" runat="server"
            ImageUrl="~/images/Show.gif" />
    </div>
    </form>
</body>
</html>
```

Explanation:

We often swap images to depict ‘Show/Hide’ or ‘Expand/Collapse’ scenarios. This example shows you how to swap an image when the user clicks on it.

The jQuery *toggle()* method becomes an obvious choice to handle this requirement. In short, the *toggle()* method takes in two handlers as parameters, the first handler to be executed on every 1st, 3rd, 5th (and so on) clicks and the second handler to be execute on the 2nd, 4th, 6th (and so on) clicks.

```
$('img[id$=imageSwap]').toggle(
  function() {
      $(this).attr("src", "../images/Hide.gif");
  },
  function() {
      $(this).attr("src", "../images/Show.gif");
  }
);
```

The image control loads by displaying Show.gif.

When the user clicks on the image (1st click), the image source is swapped to ‘Hide.gif’.

Similarly when the user clicks on the image (2nd click), the image source is swapped back to ‘Show.gif’.

This is how we achieve the swapping behavior. You can also preload the images for better performance.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Effects/toggle

Recipe 41: Add/Remove Hyperlinks on Images using jQuery

Challenge:

Your images have anchor tags around them, making them clickable. You want to remove these hyperlinks when the user performs some action.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Remove Hyperlinks from Images</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnRemove]').click(function(e) {
                e.preventDefault();
                $('a.hyper').each(function() {
                    $(this).replaceWith($(this).children());
                });
                $(this).attr("disabled", "disabled");
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <h2>Click on the Button to Remove Hyperlinks from Images</h2>
        <br /><br />
        <asp:HyperLink ID="linkDnc" runat="server"
        NavigateUrl="http://www.dotnetcurry.com" class="hyper">
            <asp:Image ID="imgA" runat="server"
                ImageUrl="../images/dnc.png" class="imgPreview" />
        </asp:HyperLink>
        <asp:HyperLink ID="linkDevc" runat="server"
        NavigateUrl="http://www.sqlservercurry.com" class="hyper">
             <asp:Image ID="imgB" runat="server"
                ImageUrl="../images/devc.png" class="imgPreview" />
        </asp:HyperLink>
        <asp:HyperLink ID="linkSqlc" runat="server"
        NavigateUrl="http://www.devcurry.com" class="hyper">
```

```
             <asp:Image ID="imgC" runat="server"
                  ImageUrl="../images/sqlc.png" class="imgPreview" />
        </asp:HyperLink>
        <br />
        <asp:Button ID="btnRemove" runat="server" Text="Remove Hyperlinks"
            ToolTip="Click here to remove hyperlinks from Images" />
    </div>

    </form>
</body>
</html>
```

Explanation:

To add a hyperlink to an image, you can wrap the image control inside a hyperlink control as shown below:

```
<asp:HyperLink ID="linkDnc" runat="server"
        NavigateUrl="http://www.dotnetcurry.com" class="hyper">
         <asp:Image ID="imgA" runat="server"
                ImageUrl="~/images/dnc.png" class="imgPreview" />
</asp:HyperLink>
```

This renders in the browser as:

```
<a id="linkDnc" class="hyper" href="http://www.dotnetcurry.com"><img id="imgA"
class="imgPreview" src="../images/dnc.png" style="border-width:0px;" /></a>
```

Here the anchor is the parent of the image tag. So to remove hyperlinks from images, all you have to do is remove and replace the parent tags with the child tags as shown below:

```
$('input[id$=btnRemove]').click(function(e) {
    e.preventDefault();
    $('a.hyper').each(function() {
        $(this).replaceWith($(this).children());
    });
});
```

When you run the application, after clicking the button, the hyperlinks are removed. If you view the page source now, the html appears similar to the following:

```
<img id="imgA" class="imgPreview" style="border-width: 0px;" src="../images/dnc.png"/>
<img id="imgB" class="imgPreview" style="border-width:
0px;" src="../images/devc.png"/>
<img id="imgC" class="imgPreview" style="border-width:
0px;" src="../images/sqlc.png"/>
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Manipulation/replaceWith

Section 8: jQuery and the ASP.NET Bulleted List Control

Introduction

The ASP.NET Bulleted List Control is a control that generates a bulleted list of items. The control is declared in your WebForm as shown below:

```
<asp:BulletedList ID="BulletedList1" runat="server">
    <asp:ListItem Value="dnc">
        DotNetCurry</asp:ListItem>
    <asp:ListItem Value="sql">
        SqlServerCurry</asp:ListItem>
    <asp:ListItem Value="devc">
        DevCurry</asp:ListItem>
    <asp:ListItem Text="st">
        SaffronStroke</asp:ListItem>
</asp:BulletedList>
```

and renders in the browser as:

```
<ul id="BulletedList1">
<li>DotNetCurry</li>
<li>SqlServerCurry</li>
<li>DevCurry</li>
<li>SaffronStroke</li>
</ul>
```

In this section, I will show you how to solve the following three client-side scripting challenges while using the ASP.NET BulletedList Control

Recipe 42: Convert Bulleted List Items to Hyperlinks using jQuery

Challenge:

Your Bulleted List Item Value represents a URL. You want to covert these list items to hyperlinks.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Convert Bulleted List to Hyperlinks</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $("input[id$=btnConvert]").click(function(e) {
                e.preventDefault();
                $('.bullet > li').each(function() {
var link = '<a href="http://www.' + $.trim($(this).text()) + '.com"></a>';
                    $(this).wrapInner(link);
                });
                    $(this).attr("disabled", "disabled");
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <asp:BulletedList ID="list" runat="server" class="bullet">
            <asp:ListItem Value="dnc">
                DotNetCurry</asp:ListItem>
            <asp:ListItem Value="sql">
                SqlServerCurry</asp:ListItem>
            <asp:ListItem Value="devc">
                DevCurry</asp:ListItem>
            <asp:ListItem Text="st">
                SaffronStroke</asp:ListItem>
        </asp:BulletedList>
        <br /><br />
        <asp:Button ID="btnConvert" runat="server"
            Text="Convert to Hyperlinks"
            ToolTip="Click to convert the Bulleted List to Hyperlinks" />
    </div>
    </form>
</body>
</html>
```

Explanation:

The ASP.NET BulletedList control has a ‘DisplayMode’ property that can be used to specify the display behavior to be applied to list items. When the ‘DisplayMode’ is set to ‘Hyperlink’, the list contents are displayed as hyperlinks.

Assuming this property is not set on your control, this recipe demonstrates how to convert BulletedListItems to Hyperlinks at runtime, using client-script. The ASP.NET Bulleted List gets rendered as a group of list items inside an unordered list. So the first step is to loop through each list item inside the unordered list.

```
$('.bullet > li').each(function() {}
```

The next step is to retrieve the text of each list item and construct an html element representing a hyperlink

```
var link = '<a href="http://www.' + $.trim($(this).text()) + '.com"></a>';
```

The final step is to use the *wrapInner()* method to wrap the inner child contents of each list item with the html structure.

```
$(this).wrapInner(link);
```

This converts the List Item from

```
<li>DotNetCurry</li>
```

to

```
<li><a href="http://www.DotNetCurry.com">DotNetCurry</a></li>
```

You can now click on the hyperlink and navigate to the URL, the listitem represents.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Manipulation/wrapInner

Recipe 43: Add New Items to a BulletedList Control and apply CSS on them using jQuery

Challenge:

You want to programmatically add new items to a BulletedList Control at runtime and apply CSS on the newly added items

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Add New Bulleted ListItems</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnAdd]').click(function(e) {
                e.preventDefault();
                // assuming a JSON array coming from a trusted service
                var urlList = ["SaffronStroke", "Foodatarian"];
                var newList = "";
                var bulList = $('ul[id$=list]');
                $.each(urlList, function(index, item) {
                    newList += "<li class='orlist'>" + item + "</li>";
                });
                $(bulList).append(newList);
                $(this).attr("disabled", "disabled");
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Add New Items to the Bulleted List and apply a
            CSS style to the newly added items</h2>
        <asp:BulletedList ID="list" runat="server" class="bullet">
            <asp:ListItem Value="dnc">
                DotNetCurry</asp:ListItem>
            <asp:ListItem Value="sql">
                SqlServerCurry</asp:ListItem>
            <asp:ListItem Value="devc">
                DevCurry</asp:ListItem>
        </asp:BulletedList>
        <br /><br />
        <asp:Button ID="btnAdd" runat="server"
            Text="Add New Items"
            ToolTip="Click to Add New Items to the Bulleted List" />
    </div>
    </form>
</body>
</html>
```

Explanation:

This example starts by assuming that there is a JSON service returning an array. On the click of a button, we consume this array and then use *$.each()* to iterate through the array. During each loop, the content of the array is read and concatenated with new list items li's. The result of the concatenation is stored in the variable ‘newList’.

```
$.each(urlList, function(index, item) {
        newList += "<li class='orlist'>" + item + "</li>";
});
```

Observe that each <li> has a class attribute ‘orlist’ which gives style to the newly added items. This isn’t the jQuery way of doing it, but fits the requirement here. If you want to apply CSS on the new items using jQuery, store the elements in a variable and use the *addClass()* function.

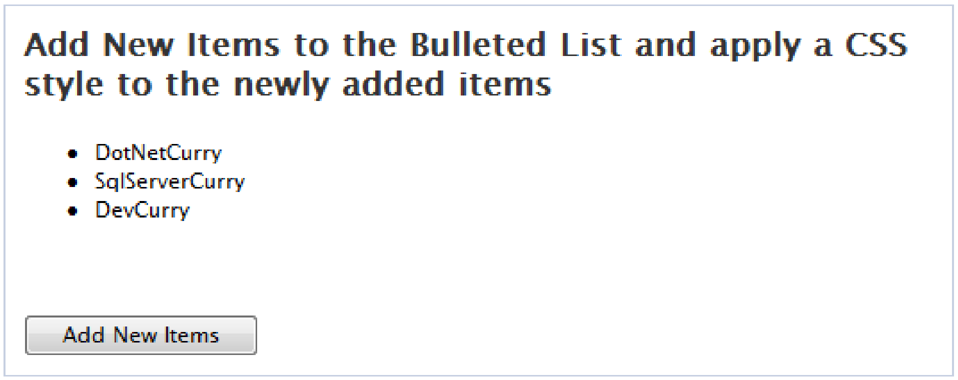
The definition of ‘orlist’ is as shown below:

```
.orlist
{
font-style:italic;
font-weight:bold;
}
```

The last step is to append the contents of the ‘newList’ variable to the BulletedList control and disable the button to prevent the user from adding the same items again.

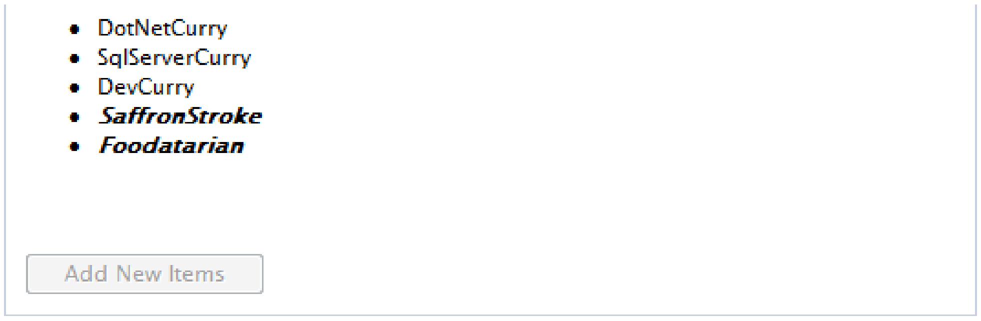
```
$(bulList).append(newList);
$(this).attr("disabled", "disabled");
```

Before the Items are added, the List looks like this:


Add New Items to the Bulleted List and apply a CSS style to the newly added items

- DotNetCurry
- SqlServerCurry
- DevCurry

After the items have been added and CSS applied to the new items, the list looks similar to the following:


- DotNetCurry
- SqlServerCurry
- DevCurry
- ***SaffronStroke***
- ***Foodatarian***

[Live Demo]()

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Utilities/jQuery.each

Recipe 44: Sort Items of a BulletedList Control using jQuery

Challenge:

The BulletedList Control displays items in an unsorted format. You want to sort the items of the BulletedList control and display the sorted list.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Sort Items of a Bulleted List Control</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $("input[id$=btnSort]").click(function(e) {
                e.preventDefault();
                var sortedList = $.makeArray($('.bullet li'))
                    .sort(function(o, n) {
                        return ($(o).text() < $(n).text()) ? -1 : 1;
                        });
                    $('.bullet').html(sortedList);
                    $(this).attr("disabled", "disabled");
                });
            });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="smallDiv">
        <h2>Click on the Button to Sort List Items</h2>
        <asp:BulletedList ID="list" runat="server" class="bullet">
            <asp:ListItem Value="dnc">
                DotNetCurry</asp:ListItem>
            <asp:ListItem Value="sql">
                SqlServerCurry</asp:ListItem>
            <asp:ListItem Value="devc">
                DevCurry</asp:ListItem>
            <asp:ListItem Text="st">
                SaffronStroke</asp:ListItem>
        </asp:BulletedList>
        <br /><br />
        <asp:Button ID="btnSort" runat="server"
            Text="Sort Items"
            ToolTip="Sort the Bulleted List Items in Ascending order" />
    </div>
    </form>
</body>
</html>
```

Explanation:

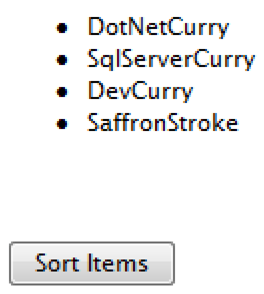
When the user clicks on the Sort button, the BulletedList items are converted to an array using *$.makeArray()*. The JavaScript built-in *sort()* function is used on this array, which does an in-place sort of the array.

```
var sortedList = $.makeArray($('.bullet li'))
  .sort(function(o, n) {
      return ($(o).text() < $(n).text()) ? -1 : 1;
});
```

The final step is to use *.html()* to replace the existing list items with the sorted list items.

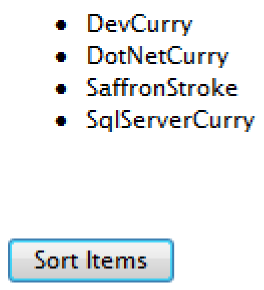
```
$('.bullet').html(sortedList);
```

The BulletedList, before the sorting is done, looks like this:



- DotNetCurry
- SqlServerCurry
- DevCurry
- SaffronStroke

The BulletedList, after the sorting is done, looks like this:



- DevCurry
- DotNetCurry
- SaffronStroke
- SqlServerCurry

To sort in a descending order, replace this line

```
$(o).text() < $(n).text()
```

with this:

```
$(o).text() > $(n).text()
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Utilities/jQuery.makeArray

Section 9: jQuery and ASP.NET Hyperlink Control

Introduction

The ASP.NET Hyperlink Control is used to create a Hyperlink. The control is declared in your WebForm as shown below:

```
<asp:HyperLink ID="HyperLink1" runat="server"
    NavigateUrl="http://www.dotnetcurry.com">
    HyperLink
</asp:HyperLink>
```

and renders in the browser as:

```
<a id="HyperLink1" href="http://www.dotnetcurry.com">HyperLink</a>
```

In this section, I will show you the following three recipes to solve some client-side scripting challenges while using the ASP.NET Hyperlink Control

Recipe 45: Implementing KeyBoard Shortcuts on Hyperlink Control using jQuery

Challenge:

Users should be able to use shortcut keys and invoke click on a Hyperlink

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Implement KeyBoard Shortcuts on Hyperlinks</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $(document).keyup(function(e) {
                var key = (e.keyCode ? e.keyCode : e.charCode);
                switch (key) {
                    case 49:
                        navigateUrl($('a[id$=linkA]'));
                        break;
                    case 50:
                        navigateUrl($('a[id$=linkB]'));
                        break;
                    case 51:
                        navigateUrl($('a[id$=linkC]'));
                        break;
                    default: ;
                }
            });

            function navigateUrl(jObj) {
                window.location.href = $(jObj).attr("href");
                alert("Navigating to " + $(jObj).attr("href"));
            }
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Use Keyboard Keys 1, 2 or 3 to invoke respective
            websites</h2><br />
        <asp:HyperLink ID="linkA" runat="server"
            NavigateUrl="http://www.dotnetcurry.com">
            DotNetCurry</asp:HyperLink><br /><br />
        <asp:HyperLink ID="linkB" runat="server"
            NavigateUrl="http://www.sqlservercurry.com">
             SqlServerCurry</asp:HyperLink><br /><br />
        <asp:HyperLink ID="linkC" runat="server"
            NavigateUrl="http://www.devcurry.com">
            DevCurry</asp:HyperLink><br /><br />
    </div>
    </form>
</body>
</html>
```

Explanation:

Keyboard shortcuts improve productivity by accomplishing tasks more quickly and without much effort. In applications, where the user has to select from a variety of actions to perform, keyboard shortcuts can save on time and effort. If you have used the recent versions of Hotmail, YahooMail or Gmail, you will be quite familiar with these shortcuts.

Implementing a Keyboard shortcut in jQuery is relatively simple as shown here. The code first captures the keyup event and the key is detected using the keyCode or charCode.

```
$(document).keyup(function(e) {
    var key = (e.keyCode ? e.keyCode : e.charCode);
```

In the code shown below, if the key = 49, digit 1 is pressed by the user and the first Hyperlink is auto-clicked. Similarly if the key = 50 or 51, then digit 2 or 3 are pressed respectively and the 2nd or 3rd hyperlink is autoclicked. Once the key matches any of the switch case statement, the function navigateUrl() is called passing in the respective hyperlink control object to the function. So if the user pressed 1, the first hyperlink control object is passed to the function as shown below:

```
switch (key) {
    case 49:
        navigateUrl($('a[id$=linkA]'));
        break;
```

The navigateUrl function looks like this:

```
function navigateUrl(jObj) {
    window.location.href = $(jObj).attr("href");
}
```

The function accepts the hyperlink object and sets the 'window.location.href' to the href attribute of the Hyperlink passed in. This is how we invoke actions with shortcut keys.

keyCode represents the actual key pressed as a numerical value, whereas charCode gives the ASCII/Unicode value of the key press result (for eg: Shift + A). Firefox and other browsers also support 'e.which'. IE and Opera does not support charCode.

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Events/keyup

Recipe 46: Change the URL of a Hyperlink Control using jQuery

Challenge:

The target URL of a hyperlink has to be changed at runtime based on a user action

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Replace Url's at Runtime</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnChangeUrl]').click(function(e) {
                e.preventDefault();
                var $link = $('a[id$=linkA]');
                var oldUrl = $link.attr('href');
                var newUrl = "http://saffronstroke.com";
                $link.attr('href', newUrl);
                $(this).attr("disabled", "disabled");
                $("#para").text("URL Changed from " + oldUrl +
                " to " + newUrl);
            });
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Click on the Button 'Change Url' to change the target URL of
            the Hyperlink</h2><br /><br />
        <asp:HyperLink ID="linkA" runat="server"
            NavigateUrl="http://www.dotnetcurry.com">
            Programming Site</asp:HyperLink><br /><br />
        <asp:Button ID="btnChangeUrl" runat="server" Text="Change Url"
        ToolTip="Click here to Change the URL" />
        <br /><br />
         <p id="para"></p>
    </div>
    </form>
</body>
</html>
```

Explanation:

This recipe demonstrates how to change the target URL of a link at runtime. When the page loads, the target URL of the Hyperlink points to ‘http://www.dotnetcurry.com’.

To change the target URL of the link to a different URL, we access the ‘href’ attribute on the Hyperlink control and set it to a new URL, in our case ‘http://saffronstroke.com’

```
var $link = $('a[id$=linkA]');
var newUrl = "http://saffronstroke.com";
$link.attr('href', newUrl);
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Attributes/attr

Recipe 47: Text Link Ad Rotation using jQuery

Challenge:

You want to rotate a group of Hyperlinks on the same position.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Timer based toggling of hyperlink</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            var anch = $("a.dev");
            $(anch).hide().eq(0).show();
            var cnt = anch.length;

            setInterval(linkRotate, 3000);

            function linkRotate() {
               $(anch).eq((anch.length++) % cnt).fadeOut("slow", function() {
                    $(anch).eq((anch.length) % cnt)
                    .fadeIn("slow");
                });
            }
        });
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>The Hyperlink and Text shown below changes after 3 seconds
        </h2><br />
        <asp:HyperLink ID="linkA" runat="server" class="dev"
            NavigateUrl="http://www.dotnetcurry.com">
            DotNetCurry</asp:HyperLink>
        <asp:HyperLink ID="linkB" runat="server" class="dev"
            NavigateUrl="http://www.sqlservercurry.com">
            SqlServerCurry</asp:HyperLink>
        <asp:HyperLink ID="linkC" runat="server" class="dev"
            NavigateUrl="http://www.devcurry.com">
            DevCurry</asp:HyperLink>
    </div>
    </form>
</body>
</html>
```

Explanation:

People who have been monetizing their sites need no introduction to TextLinkAds. In simple words, Text Link Ads are Hyperlinks sponsored by Advertisers. When a user clicks on these hyperlinks, they are sent to the sponsor's website.

In this recipe, I will demonstrate how to rotate multiple hyperlinks or TextLinkAds on the same position. We begin by hiding all the hyperlinks with class="dev" and then display the first one.

```
$(anch).hide().eq(0).show();
```

We then use the JavaScript setInterval() function to delay the execution of a function (linkRotate) for a specific time, in our case 3000 millisecond (3 seconds), as shown below:

```
setInterval(linkRotate, 3000);
```

The advantage of the setInterval() function is that it continues to repeat the process of triggering the function at the specified interval, thereby giving it a loop effect.

In the linkRotate() function, we use a simple expression `(anch.length++) % cnt` that calculates the index to be supplied to the eq() selector and applies the fadeout/fadeIn() animations on the current hyperlink. eq(0) refers to the first link, eq(1) to the second link and so on.

```
function linkRotate() {
    $(anch).eq((anch.length++) % cnt).fadeOut("slow", function() {
        $(anch).eq((anch.length) % cnt)
        .fadeIn("slow");
    });
}
```

The chart shown below helps you understand what happens at every loop

Loop Number	**Value of `anch.length`**	**Value of `(anch.length++) % cnt`**
1	3	3mod3 = 0
2	4	4mod3 = 1
3	5	5mod3 = 2
4	6	6mod3=0
and so on….		

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Selectors/eq#index

Section 10: jQuery, AJAX and ASP.NET

Introduction

The jQuery library has an excellent AJAX support and is suitable for communicating with the server behind the scenes, without refreshing a page. In this section, I will show you how to use the jQuery library to consume ASP.NET Web services that return JSON formatted data, call PageMethods and submit a form behind the scenes, without using ASP.NET AJAX.

ASP.NET AJAX is a very useful framework for building highly interactive and responsive web applications. However this eBook does not cover jQuery with ASP.NET AJAX.

This chapter uses some server-side code. To keep the server-side code to the minimum in these chapters, the code has been declared in Appendix ‘C’ of this eBook. The same code can also be found in the App_Code folder of the source code.

This section covers the following 5 recipes.

Recipe 48: Consume an ASP.NET JSON Web Service using jQuery

Challenge:

A JSON Serialized ASP.NET Web Service has to be consumed in your ASP.NET application using jQuery

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Consume ASP.NET Web Service returning JSON</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script src="../Scripts/jquery-1.3.2.min.js"
        type="text/javascript"></script>
    <script src="../Scripts/jquery-jtemplates.js"
            type="text/javascript"></script>

    <script type="text/javascript">
        $(function() {
            $('input[id$=btnPull]').click(function(e) {
                e.preventDefault();
                $.ajax({
                    type: "POST",
                    url: "../Services/EmployeeList.asmx/FetchEList",
                    data: "{}",
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    success: function(msg) {
                        var orders = msg.d;
                        if (orders.length > 0) {
                            CreateTable(orders);
                        }
                        else {
                            $("#tbl").text("No records found");
                        }
                    },
                   error: function(XMLHttpRequest, textStatus, errorThrown) {
                        alert(textStatus);
                    }
                });


            });
        });

        function CreateTable(msg) {
            $('#tbl').setTemplateURL('../Template/TemplateEmployee.htm',
                             null, { filter_data: false });
            $('#tbl').processTemplate(msg);
        }
    </script>
</head>
```

```
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Click on the Button to consume data from an ASP.NET JSON
            WebService</h2>
        <br /><br />
        <asp:Button ID="btnPull" runat="server" Text="Retrieve Data"
            ToolTip="Click Here to Retrieve Data From a Service" />
        <div id="tbl"></div>
    </div>

    </form>
</body>
</html>
```

Explanation:

This recipe demonstrates how to consume an ASP.NET Web Service (EmployeeList.asmx) that is JSON Serialized. The data source for this web service is List<Employees> in the Employee.cs class, declared in Appendix 'C' of this eBook. The same code can also be found in the App_Code folder in the source code.

If you look at the EmployeeList.cs file, you will observe that the method has been decorated with the [WebMethod] attribute to allow calls from client script

```
    [WebMethod]
    public List<Employee> FetchEList() {
        var emp = new Employee();
        return emp.GetEmployeeList();
    }
```

The FetchEList() method calls the GetEmployeeList() method on the Employee class which returns a List<Employee>.

If a method is not marked with [ScriptMethod] attribute, the method will be called by using the HTTP POST command and the response will be serialized as JSON. The 'Useful Links' section of this recipe contains a post by Scott Guthrie that explains why a request should be a POST request.

To consume this web service using jQuery *$.ajax()*, two important points to note is that the request should be a POST request and the request's content-type must be 'application/json; charset=utf-8'. The code structure looks similar to the following:

```
$.ajax({
    type: "POST",
    url: "../Services/EmployeeList.asmx/FetchEList",
    data: "{}",
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    success: function(msg) {

    },
    error: function(XMLHttpRequest, textStatus, errorThrown) {
    }
});
```

Once the Ajax method is completed, the success function will be executed.

```
success: function(msg) {
    var orders = msg.d;
    if (orders.length > 0) {
        CreateTable(orders);
    }
    else {
        $("#tbl").text("No records found");
    }
}
```

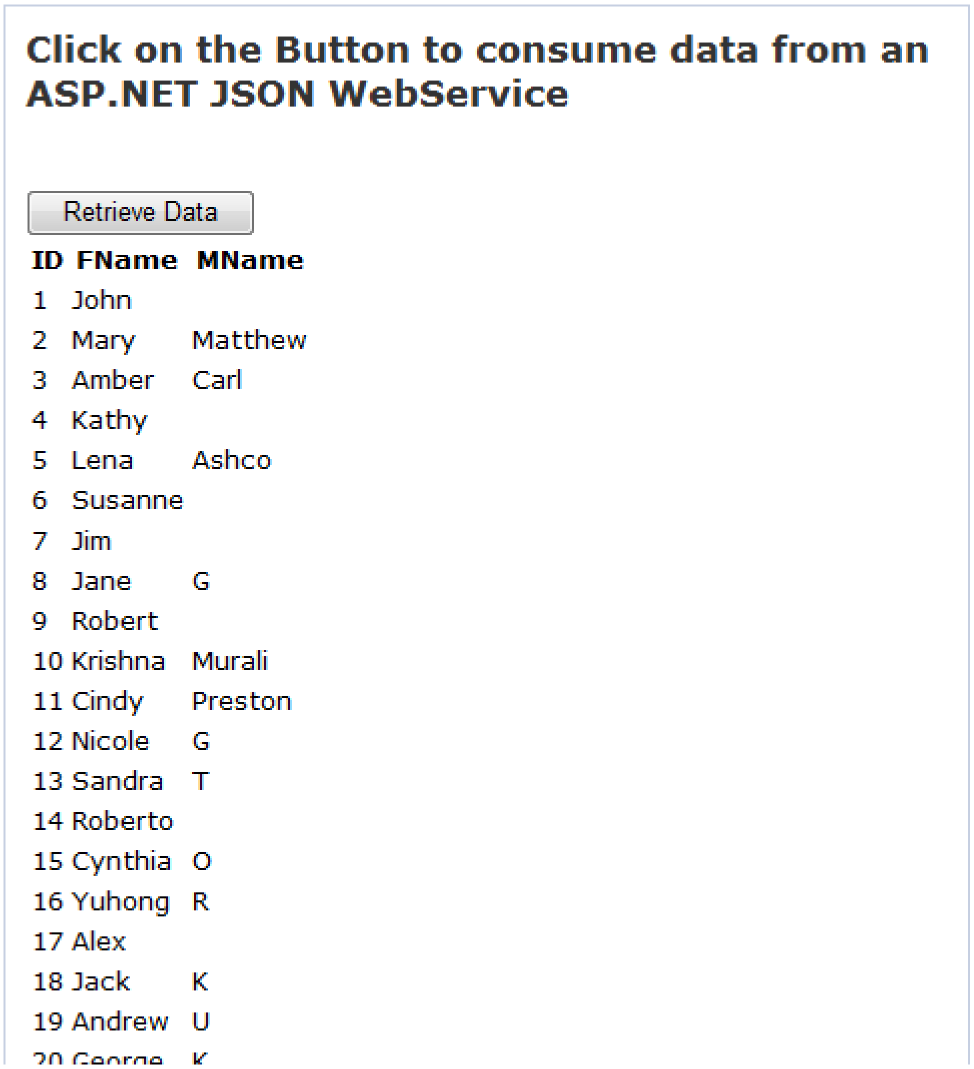
The CreateTable function accepts the JSON results retrieved from the Web Service and uses a jTemplate method to load the HTML template into a container, and finally render the JSON result. The technique of creating HTML elements on the fly from a data source using jTemplates and displaying the results remains the same as described in Recipe 30.

```
function CreateTable(msg) {
    $('#tbl').setTemplateURL('../Template/TemplateEmployee.htm',
                null, { filter_data: false });
    $('#tbl').processTemplate(msg);
}
```

The HTML template structure (TemplateEmployee.htm) used here is as shown below:

```
<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>FName</th>
      <th>MName</th>
    </tr>
  </thead>
  <tbody>
    {#foreach $T as data}
    <tr>
      <td>{$T.data.ID}</td>
      <td>{$T.data.FName}</td>
      <td>{$T.data.MName}</td>
    </tr>
    {#/for}
  </tbody>
</table>
```

When you run the application, and click the button, the results will look similar to the one shown below:


Click on the Button to consume data from an ASP.NET JSON WebService

ID	FName	MName
1	John	
2	Mary	Matthew
3	Amber	Carl
4	Kathy	
5	Lena	Ashco
6	Susanne	
7	Jim	
8	Jane	G
9	Robert	
10	Krishna	Murali
11	Cindy	Preston
12	Nicole	G
13	Sandra	T
14	Roberto	
15	Cynthia	O
16	Yuhong	R
17	Alex	
18	Jack	K
19	Andrew	U
20	George	K

[Live Demo]()

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://weblogs.asp.net/scottgu/archive/2007/04/04/json-hijacking-and-how-asp-net-ajax-1-0-mitigates-these-attacks.aspx

http://msdn.microsoft.com/en-us/library/system.web.script.services.scriptmethodattribute.aspx

Recipe 49: jQuery Tabs and Lazy Loading

Challenge:

You want to lazy load data into tabs using jQuery.

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>jQuery Lazy Loading Tabs</title>
    <link href="../CSS/jquery-ui-1.7.2.custom.css" rel="stylesheet"
        type="text/css" />
    <script src="../Scripts/jquery-1.3.2.min.js"
        type="text/javascript"></script>
    <script src="../Scripts/jquery-ui-1.7.2.custom.min.js"
        type="text/javascript"></script>
    <script src="../Scripts/jquery-jtemplates.js"
        type="text/javascript"></script>

    <script type="text/javascript">
        $(document).ready(function() {
            // Tabs
            $('#tabs').tabs({ selected: -1 });
            $('#tabs').bind('tabsshow', function(event, ui) {
                var tabSelected = "#tab" + ui.index + "Selected";
                $(tabSelected).val("isClicked");
            });

            $('#tabs').bind('tabsselect', function(event, ui) {
                var callMethod;
                var tabSelected = "#tab" + ui.index + "Selected";
                if ($(tabSelected).text() == "") {
                    switch (ui.index) {
                        case 0:
                            callMethod = "R49-LazyTabs.aspx/GetEmployees";
                            break;
                        case 1:
                            callMethod = "R49-LazyTabs.aspx/GetProducts";
                            break;
                        case 2:
                            callMethod = "R49-LazyTabs.aspx/GetOrders";
                            break;
                    }

                    $.ajax({
                        type: "POST",
                        url: callMethod,
                        data: "{}",
                        contentType: "application/json; charset=utf-8",
                        dataType: "json",
                        async: true,
                        success: function(msg) {
                        switch (ui.index) {
                               case 0:
                                var employees = msg.d;
                                if (employees.length > 0) {
                                    CreateTable(employees,
                                    "../Template/TemplateEmployee.htm",
                                    $("#tabs-0"));
                                }
                                break;
                            case 1:
                                var products = msg.d;
                                if (products.length > 0) {
                                    CreateTable(products,
                                    "../Template/TemplateProduct.htm",
                                    $("#tabs-1"));
                                }
                                break;
                            case 2:
                                var orders = msg.d;
                                if (orders.length > 0) {
                                    CreateTable(orders,
                                    "../Template/TemplateOrder.htm",
                                    $("#tabs-2"));
                                }
                                break;
                        }
                        },
                   error: function(XMLHttpRequest, textStatus, errorThrown) {
                            alert(textStatus);
                        }
                    });
                }
            });

            //hover states on the static widgets
            $('#dialog_link, ul#icons li').hover(
                           function() { $(this).addClass('ui-state-hover'); },
                           function() { $(this).removeClass('ui-state-hover'); }
                           );
        });

        function CreateTable(msg, url, ctrl) {
            $(ctrl).setTemplateURL(url,
                         null, { filter_data: false });
            $(ctrl).processTemplate(msg);
        }
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div id="tabs">
```

```
        <ul>
            <li><a href="#tabs-0">Employees</a></li>
            <li><a href="#tabs-1">Products</a></li>
            <li><a href="#tabs-2">Orders</a></li>
        </ul>
        <div id="tabs-0">
            <input type="hidden" id="tab0Selected" />
        </div>
        <div id="tabs-1">
            <input type="hidden" id="tab1Selected" />
        </div>
        <div id="tabs-2">
            <input type="hidden" id="tab2Selected" />
        </div>
    </div>
    </form>
</body>
</html>
```

Explanation:

This recipe uses the Employee.cs, Order.cs and Product.cs classes declared in Appendix 'C' of this eBook. The same classes can also be found in the App_Code folder in the source code. This example also calls three webmethods listed in the code-behind class of this WebForm. The methods are declared in the following way:

```
    [WebMethod]
    public static List<Employee> GetEmployees()
    {
        var employee = new Employee();
        return employee.GetEmployeeList();
    }

    [WebMethod]
    public static List<Product> GetProducts()
    {
        var product = new Product();
        return product.GetProductList();
    }

    [WebMethod]
    public static List<Orders> GetOrders()
    {
        var order = new Orders();
        return order.GetOrderList();
    }
```

To start with, the code sets the selected tabs to none:

```
$('#tabs').tabs({ selected: -1 });
```

For the lazy loading to work, when the user selects a tab, the *tabsshow* event is triggered. We then add a binding to that event to update the value in a hidden field, in the selected tab. Once this field has a value, we know the tab has been loaded and therefore it will not be loaded again, as you'll see in the code further down in the example:

```
$('#tabs').bind('tabsshow', function(event, ui) {
       var tabSelected = "#tab" + ui.index + "Selected";
      $(tabSelected).val("isClicked");
});
```

When the user selects a tab, the *tabsselect* event will be triggered. We are binding to that event to execute the server side code:

```
$('#tabs').bind('tabsselect', function(event, ui)
```

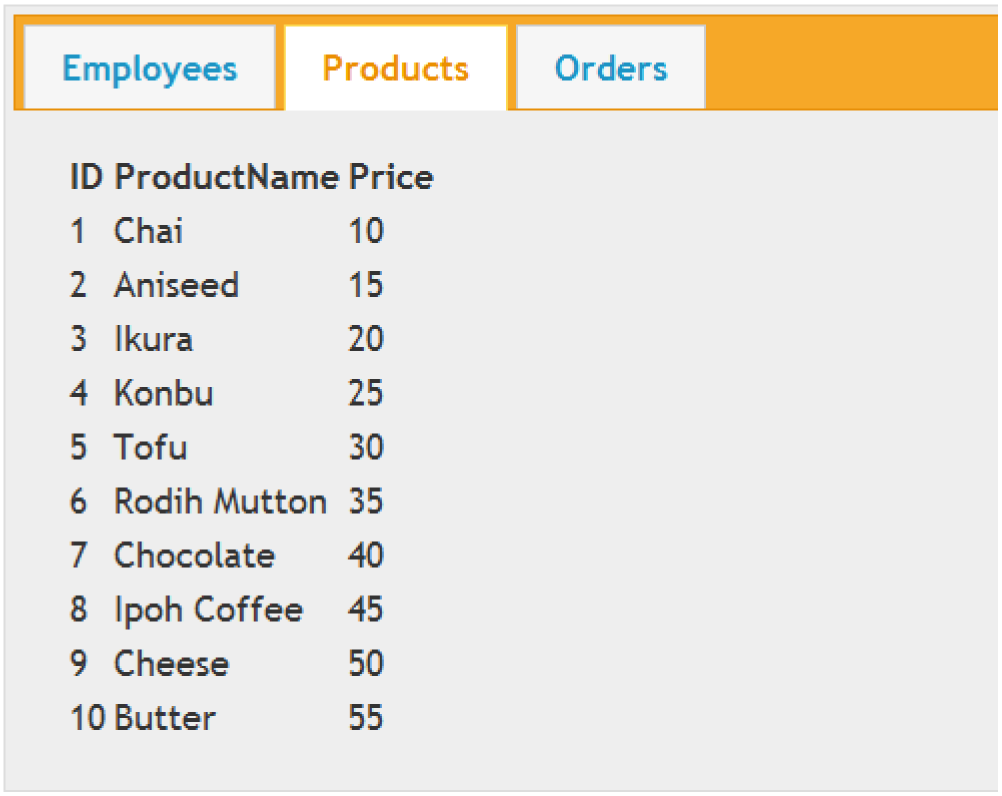
We first check the value of the hidden field in the selected tab, and if it's empty, we go ahead and find which tab was selected. This is required because there are three server side events and the correct method needs to be executed:

```
var tabSelected = "#tab" + ui.index + "Selected";
if ($(tabSelected).text() == "") {
    switch (ui.index) {
        case 0:
            callMethod = "R49-LazyTabs.aspx/GetEmployees";
            break;
        case 1:
            callMethod = "R49-LazyTabs.aspx/GetProducts";
            break;
        case 2:
            callMethod = "R49-LazyTabs.aspx/GetOrders";
            break;
    }
```

Once the Ajax method is completed, the success function will be executed. The technique of creating HTML elements on the fly from a data source using jTemplates and displaying the results remains the same, as described in Recipe 30 and 49. The templates can be found in the Templates folder in the source code or in Appendix 'C' of this eBook.

```
switch (ui.index) {
case 0:
    var employees = msg.d;
    if (employees.length > 0) {
        CreateTable(employees,
        "../Template/TemplateEmployee.htm",
        $("#tabs-0"));
    }
    break;
case 1:
    var products = msg.d;
    if (products.length > 0) {
        CreateTable(products,
        "../Template/TemplateProduct.htm",
        $("#tabs-1"));
    }
    break;
case 2:
    var orders = msg.d;
    if (orders.length > 0) {
        CreateTable(orders,
        "../Template/TemplateOrder.htm",
        $("#tabs-2"));
    }
    break;
}
```

The output will be similar to the one shown here:



```
ID ProductName Price
1  Chai         10
2  Aniseed      15
3  Ikura        20
4  Konbu        25
5  Tofu         30
6  Rodih Mutton 35
7  Chocolate    40
8  Ipoh Coffee  45
9  Cheese       50
10 Butter       55
```

ASP.NET Date formatting is not handled well in JSON. The DateTime is serialized as an escaped JavaScript Date initializer - /Date(-124003800000)/. To support ASP.NET Date Formatting, use the jMSAjax Plug-in. DotNetCurry contains an article that demonstrates how to use this plug-in.

Live Demo

This chapter was contributed by Malcolm Sheridan.

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Ajax/jQuery.ajax#options

Consume an ASP.NET WebService returning List<> with Dates using jQuery

Recipe 50: Populate a ListBox from JSON results using jQuery

Challenge:

The ASP.NET ListBox has to be populated using data coming from a JSON Serialized ASP.NET Web Service

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Populate a ListBox from JSON Results</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
        $(function() {
            var firstParam = 'F';
            var radBtn = $("table.tbl input:radio");
            var lBox = $('select[id$=lb]');
            $(radBtn).click(function() {
                lBox.empty();
                var firstParam = $(':radio:checked').val();
                $.ajax({
                    type: "POST",
                    url: "../Services/EmployeeList.asmx/FetchEmpOnGender",
                    data: "{empSex:\"" + firstParam + "\"}",
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    success: function(msg) {
                        var gender = msg.d;
                        if (gender.length > 0) {
                            var listItems = [];
                            for (var key in gender) {
                                listItems.push('<option value="' +
                                key + '">' + gender[key].FName
                                + '</option>');
                            }
                            $(lBox).append(listItems.join(''));
                        }
                        else {
                            alert("No records found");
                        }
                    },
                     error: function(XMLHttpRequest, textStatus, errorThrown) {
                        alert(textStatus);
                    }
                });
            });
        });
    </script>


</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Click on a Radio Button to retrieve 'Gender' based data</h2>
        <br /><br />
        <asp:RadioButtonList ID="rbl" runat="server" class="tbl"
        ToolTip="Click on this RadioButton to retrieve data">
            <asp:ListItem Text="Male" Value="M"></asp:ListItem>
            <asp:ListItem Text="Female" Value="F"></asp:ListItem>
        </asp:RadioButtonList><br />
        <asp:ListBox ID="lb" runat="server"></asp:ListBox>
    </div>
    </form>
</body>
</html>
```

Explanation:

The technique of calling the JSON serialized service in this Recipe remains the same as discussed in Recipe 48 (Consume an ASP.NET JSON Web Service using jQuery). However in this recipe, I will additionally demonstrate the following using *$.ajax()* :

- Pass parameters to the web service
- Loop through the JSON result and select a single field
- Populate a ListBox using data from that field

This recipe uses an ASP.NET Web Service (EmployeeList.asmx) that is JSON Serialized. The data source for this web service is List<Employees> in the Employee.cs class, declared in Appendix ‘C’ of this eBook. The same can also be found in the App_Code folder in the source code.

This example contains two radio buttons ‘Male’ and ‘Female’ which when selected, passes in the gender information to the web service. Depending on the input received, the web service method filters and returns data for the gender requested. The Web Method looks like this:

```
    [WebMethod]
    public List<Employee> FetchEmpOnGender(char empSex)
    {
        var emp = new Employee();
        var fetchEmp = emp.GetEmployeeList()
            .Where(m => m.Sex == empSex);
        return fetchEmp.ToList();
    }
```

Observe how we are passing in the gender parameter using a 'special formatting' in the *$.ajax()* call

```
data: "{empSex:\"" + firstParam + "\"}"
```

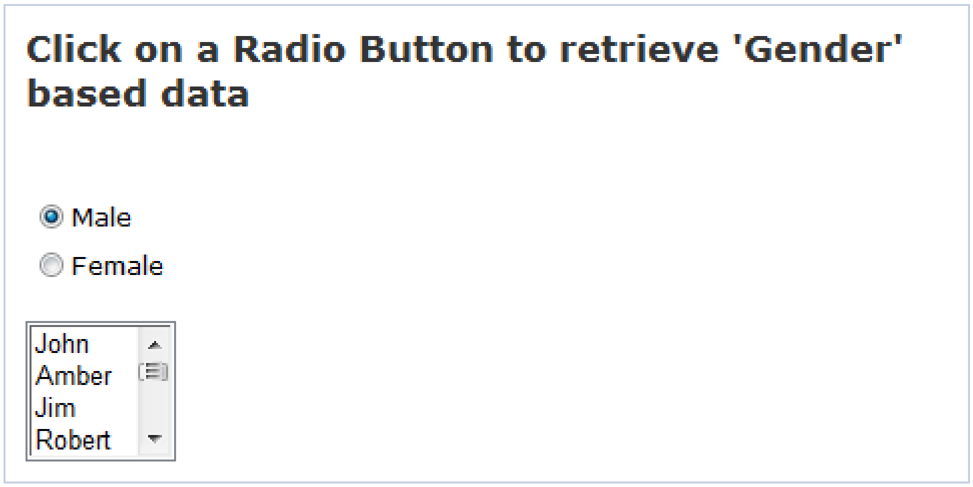
Once we receive the results, we loop through it and populate an array 'listItems' with the FirstName values by passing in a key to the FName value (`gender[key].FName)`)

```
success: function(msg) {
    var gender = msg.d;
    if (gender.length > 0) {
        var listItems = [];
        for (var key in gender) {
            listItems.push('<option value="' +
            key + '">' + gender[key].FName
            + '</option>');
        }
```

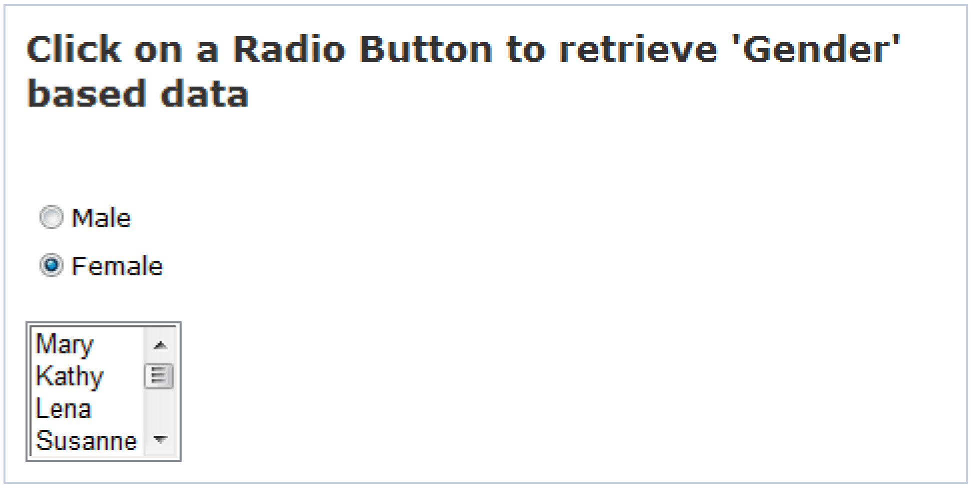
The final step is to populate the ListBox using the populated array

```
$(lBox).append(listItems.join(''));
```

When you run the application, click on the radio button, to retrieve the FirstName of Employees. Clicking on the 'Male' radio button fetches the following results:



Whereas clicking on the ‘Female’ radio button fetches the following results:



Click on a Radio Button to retrieve 'Gender' based data

Mary
Kathy
Lena
Susanne

[Live Demo]

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://docs.jquery.com/Ajax/jQuery.ajax#options

Recipe 51: Serialize only Selected Elements of a Form and Submit it using jQuery

Challenge:

You want to submit form data in the background without using ASP.NET AJAX and at the same time prevent some form fields from being submitted

Solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Serialize Form Data</title>
    <link href="../CSS/Demos.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
        src='../Scripts/jquery-1.3.2.min.js'>
    </script>

    <script type="text/javascript">
```

```
        $(function() {
            $('input[id$=btnSubmit]').click(function(e) {
                e.preventDefault();
                var params = $('form[id$=form1]').serializeArray();
                $.post("R51a-ReadValues.aspx",
                   $(params).filter(function() {
                       return !({ "txtLName": 1, "__VIEWSTATE": 1,
"__EVENTVALIDATION": 1 })[this.name];
                   }),
                   function(data) {
                       //do something
                   },
                   "json"
               );
                alert('Form data submitted in the background');
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div class="tableDiv">
        <h2>Enter data and click on 'Serialize and Submit' button
        to send form data to a different page without a postback</h2>
        <br /><br />
        <asp:Label ID="lblFName" runat="server"
            Text="FirstName: "></asp:Label>
        <asp:TextBox ID="txtFName" runat="server">
            </asp:TextBox><br />
        <asp:Label ID="lblMName" runat="server"
            Text="MidName: "></asp:Label>
        <asp:TextBox ID="txtMName" runat="server">
            </asp:TextBox><br />
        <asp:Label ID="lblLName" runat="server"
            Text="LastName: "></asp:Label>
        <asp:TextBox ID="txtLName" runat="server">
            </asp:TextBox><br />
        <asp:Button ID="btnSubmit" runat="server"
            Text="Serialize and Submit" />
    </div>
    </form>
</body>
</html>
```

Explanation:

$.post() is a very useful method for sending a simple POST request to the server using AJAX. The signature of *$.post()* is as follows:

$.post(url, [data], [callback], [type]) where url is the URL where the request is sent, [data] is the data sent with the request, [callback] is a function to execute when the request is complete and [type] is the format in which content has been returned (xml, json)

So a simple piece of code like the one shown below can send the form data to another page using AJAX.

```
 $.post("R51a-ReadValues.aspx", $('form[id$=form1]').serialize());
```

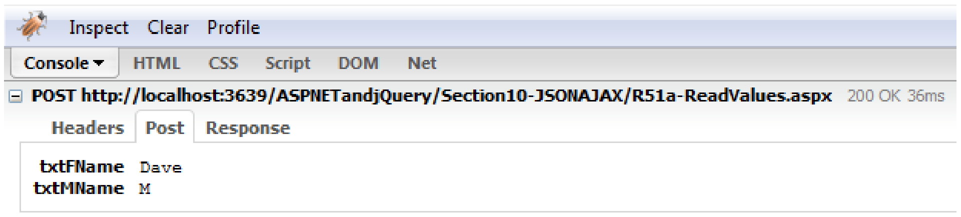
How very Ajax!

Now what if you want to prevent some fields from being submitted to the server? So apart from the form fields, there is the ViewState and EventValidation that also gets posted to the server. Let us save some bandwidth by exploring how to prevent these fields as well as some other Form fields too for demonstration purposes (like the lastname) from being posted to the server. Observe this piece of code:

```
var params = $('form[id$=form1]').serializeArray();
$.post("R51a-ReadValues.aspx",
   $(params).filter(function() {
       return !({ "txtLName": 1, "__VIEWSTATE": 1, "__EVENTVALIDATION": 1
})[this.name];
   }),
   function(data) {
       //do something
   },
   "json"
);
```

In the code above, we use serializeArray() to serialize all forms and form elements. A JSON structure is returned which is stored in the variable 'params'. When you use *$.post()* to submit the form fields, we use the *filter()* function to filter out the txtLName, __VIEWSTATE and __EVENTVALIDATION fields. Imagine the performance increase when we are able to reduce the data posted. Now when you run the application, enter data in the text fields and submit data, you will see that only two fields are posted – txtFName and txtNName.

Since the form is being posted in the background, you will not see the postback happening. However the output can be seen using FireBug as shown below.



```
POST http://localhost:3639/ASPNETandjQuery/Section10-JSONAJAX/R51a-ReadValues.aspx 200 OK 36ms
txtFName Dave
txtMName M
```

Live Demo

Browsers Supported:

IE 7, IE 8, Firefox 3, Chrome 2, Safari 4

Useful Links:

http://www.west-wind.com/weblog/posts/472329.aspx

http://docs.jquery.com/Ajax/jQuery.post

http://docs.jquery.com/Ajax/serializeArray

Appendix A: Useful jQuery Plug-ins

One of the powerful features of the jQuery library is its extensibility. jQuery supports a plugin architecture that allows you to extend the jQuery library and create reusable components. The jQuery Plugin respository is an excellent place to look out for plugins.

In this section, I will showcase some useful jQuery plugins created by the jQuery Community. The description for these plugins has been taken directly from the jQuery site/creator's site.

jQuery UI Plugins Library

jQueryUI - jQuery UI provides a comprehensive set of core interaction plugins, UI widgets and visual effects that use a jQuery-style, event-driven architecture and a focus on web standards, accessibility, flexible styling, and user-friendly design. This library is created and maintained by the jQuery UI team

Images Plugins

Here is a list of plugins you can use to manipulate images on client-side:

ImageOverlay - Allows you to display additional information in a translucent pane above an image that gracefully appears when a user hovers over the image

jCrop - jCrop is the quick and easy way to add image cropping functionality to your web application. It combines the ease-of-use of a typical jQuery plugin with a powerful cross-platform DHTML cropping engine that is faithful to familiar desktop graphics applications.

jLoupe - jLoupe adds a "loupe" or "magnifying glass" effect to images. Applying the plugin is as easy as adding the class "jLoupe" to the images you want to enable the effect on.

Magnify - Magnify allows you to easily create a "magnifier" on your images. You just need to create a small and large image which are proportionally sized, configure your markup accordingly and let Magnify do the rest

Reel - Reel is a jQuery plugin which takes an image tag and makes it a live "projection" of pre-built animation frames sequence. Its aim is to provide a 360° view of something or someplace. Like a turntable or a panorama for instance

SlideShow and ImageGallery plugins

Here is a list of plugins you can use to create slideshows:

Cycle - The jQuery Cycle Plugin is a lightweight slideshow plugin. It supports pause-on-hover, auto-stop, auto-fit, before/after callbacks, click triggers and many transition effects.

jOmino - Wrap your html elements in a jQuery collection and animate them with a dominoes in motion effect.

PanelGallery - A simple image gallery where the images are sliced and faded in one piece at a time.

PikaChoose - PikaChoose is a image gallery plugin. You place images into an unordered list structure and let PikaChoose do the rest for you

Pliable Photo Gallery - Easy embeddable JavaScript based photo gallery. Great for showing off a small group of images. Guiding principles: simple, less requirements, more conventions

SlicedImageSlider - This plug-in provides sliced Image Navigation. The plugin loops through the given images and displays each image partitioned into n number of slices. It also has other options like direction in which you want the image to slide

Dialogs plugins

Here is a list of plugins you can use to create fancy modal dialogs:

BlockUIPlugin (v2) - The jQuery BlockUI Plugin lets you simulate synchronous behavior when using AJAX, without locking the browser. When activated, it will prevent user activity with the page (or part of the page) until it is deactivated. BlockUI adds elements to the DOM to give it both the appearance and behavior of blocking user interaction

ColorBox - A light-weight, customizable lightbox

DOMWindow - jQuery plugin used to create DOM windows

FancyBox - FancyBox is tool for displaying images, html content and multimedia in a Mac-style "lightbox" that floats overtop of web page

jqDialog - jqDialog is a small dialog plugin that provides smooth, persistent, non-intrusive alternatives for alert(), confirm() and prompt(). There is also a notify() dialog that pops up and goes away in X seconds

jqModal - jqModal is a plugin for jQuery to help you display notices, dialogs, and modal windows in a web browser. It is flexible and tiny, akin to a "Swiss Army Knife", and makes a great base as a general purpose windowing framework

Validation plugins

Here is a list of plugins you can use to validate form inputs:

Numeric - Allows only valid characters (i.e. numbers) to be typed into a text box. Can take negative numbers and a decimal point. You can supply a callback that runs when focus is lost and the value in the text box is not a valid number.

Password Validation - This plugin extends the jQuery validation plugin

Validation – Validate forms like you have never been validating before

HTML Table Plugins

Here is a list of plugins you can use to manipulate tabular data:

ColumnHover - A jQuery-plugin that highlights whole columns in a table when hovering over them. It supports tables with colspans and rowspans too

ColumnManager - A jQuery-plugin to toggle the visibility of table columns (collapsing and expanding them) and to save the state until the next visit. It supports tables with colspans and rowspans too

FixedTableHeader - Fixed Table Header plugin is fixing header row of table without div overflow. It is fix header row of table when scroll down the page. You can use fixedtableheader plugin with ASP.NET DataGrid or GridView

FlexiGrid - A lightweight but rich data grid with resizable columns and a scrolling data to match the headers, plus an ability to connect to an xml based data source to load the content

FloatHeader - A plugin that makes the header of a table floating when the original header isn’t visible. The plugin uses the thead tag as the header for a table

jQBinder - JQBinder is a light-weight template engine that allows client-side data binding between an HTML template (embedded in HTML file itself) and JSON data (which is typically asynchronously fetched from the server).

jqGrid - The grid is a Ajax-enabled JavaScript control that provides solution for representing tabular data on the web. Since the grid is client-side solution and loading data dynamically through Ajax callbacks, it can be integrated with any server side technology

PicNetTableFilter - This plugin adds a row to the section of a table and adds filters that allows real time filtering of tabular data

TableSorter - Tablesorter is a jQuery plugin for turning a standard HTML table with THEAD and TBODY tags into a sortable table without page refreshes

Other Plugins

Here is a list of other miscellaneous plugins you can use:

Accessible RIA - jQuery Accessible RIA, a collection of strictly WAI WCAG 2.0 and WAI ARIA conform web applications based on the popular Java-Script framework jQuery (using the UI Widget Factory).

Accordion - This plugin creates an accordion menu. It works with nested lists, definition lists, or just nested divs. Options are available to specify the structure, if necessary, the active element (to display at first) and to customize animations. The navigation-option automatically activates a part of the accordion based on the current location (URL) of the page.

Autocomplete - Autocomplete an input field to enable users quickly finding and selecting some value, leveraging searching and filtering

Calculation - The Calculation plug-in is designed to give easy-to-use jQuery functions for commonly used mathematical functions.

ColorPicker - This rudimentary color picker plugin was designed for use in small UIs

Corner - Rounded corners without images. Plus lots of other corner adornments

DatePicker - A jQuery plugin that attaches a popup calendar to your input fields or shows an inline calendar for selecting individual dates or date ranges

DivCorners - This plugin aims to create an easy way to add border layouts to screen content. Create rounded corners, drop shadows, halos, beaks—the list is endless. It includes three functions that can be called after jquery.pack.js and jquery.divcorners.js are loaded.

DotNet jQuery - DNJ is a .Net framework integrating JQuery in ASP.Net applications and Visual Studio Designer. The project is aiming to mix the power of JQuery and Visual Studio designer capabilities to make the use of JQuery in ASP.Net applications easier

FormAdvisor - Little plugin to describe form inputs in friendly way

FormPlugin - The jQuery Form Plugin allows you to easily and unobtrusively upgrade HTML forms to use AJAX. The main methods, ajaxForm and ajaxSubmit, gather information from the form element to determine how to manage the submit process.

FullCalendar - Uses AJAX to fetch events on-the-fly for each month and is easily configured to use your own feed format (an extension is provided for Google Calendar). It is visually customizable and exposes hooks for user-triggered events (like clicking or dragging an event)

HoverPulse - A plugin which grows and then restores an element's size in response to the mouse hovering over it

ImageTickBox - Small and unobtrusive plugin to change any input checkbox to an image version while retaining full functionality! It even pre-loads the images to keep it snappy

jEditable – Makes in place editing simple. Can be used with Wiki, Textile, Markdown, BBCode, ReST etc renderers. Features automatic sizing of input elements, automatic sending element id attribute and more.

jHighlighter - A useful plugin to highlight elements on your site

jQLog - A logging framework plugin for jQuery that can be configured and extended allowing you to "roll your own" logging framework.

jReject - jReject provides a simple, robjust, light-weight way to display rejections based on a the browser, specific browser version, specific platforms, or rendering engine.

jSlider - Transform a listbox (select box) in a slider.

LockSubmit - Small JQuery function which disables a submit button on the onClick event, while still having the button's name/value passed to the form

MaskedInput - It allows a user to more easily enter fixed width input where you would like them to enter the data in a certain format (dates,phone numbers, etc).

NotNow - notNow allows to postpone certain operation(function) through a certain period once

PrettyDate - This plugin, originally written by John Resig, provides clientside date formatting in the style of Twitter’s timeline: “just now”, “5 minutes ago”, “yesterday”, “2 weeks ago”. This release extends John’s original release, providing support for internationalization and improving the API a bit

ScrollTo - With this plugin, you will easily scroll overflowed elements, and the screen itself. It gives you access to many different options to customize and various ways to specify where to scroll.

SexyCaptcha - Sexy Captcha is a truly ajaxified, sexy captcha module. It's visually stimulating, functional and returns immediate feedback on whether the user's answer was correct

SimpleWaterMark - A simple plug-in that populates text form fields with default information until they are focused on to. Upon losing focus, the plug-in reapplies the default text if the field is empty.

StyleSwitcher - The jQuery Styleswitcher widget aims to be a fast, accessible widget that will allow developers to easily switch between stylesheets, switch colors and disable images with a minimal number of clicks.

TabSlideOut - Create a side tab that expands content for a feedback form or contact info. Make your own image to use as a tab and apply this plugin to any div to hide the content off the right, left, top or bottom of the screen, It animates to show the content, when the handle/tab is clicked

TagArea - Turns a textarea or text input field into a 'tagarea' which alows the user to enter a list of tags/labels and displays them nicely (a lot like the the Tumblr tag field).

TextResizer - The Text Resizer plugin attempts to solve one problem: that of resizing text on demand by the user

Timers - jQuery timers is an attempt to combine jQuery's concise chaining programming style with the awkward style in which timed events are coded in JavaScript to produce a friendlier and more intuitive timed event system

TreeView - Lightweight and flexible transformation of an unordered list into an expandable and collapsable tree, great for unobtrusive navigation enhancements. Supports both location and cookie based persistence

Appendix B: jQuery Learning Resources

This small appendix points you to some valuable books and online resources for learning jQuery

Books

There are some excellent jQuery books to help you master the framework. Here are some I strongly recommend:

Learning jQuery 1.3

jQuery UI 1.6 : The User Interface Library for jQuery

Beginning JavaScript and CSS Development with jQuery

Online Resources

Here are some online resources to help you learn more about jQuery

Official Blogs

http://docs.jquery.com/Main_Page

http://docs.jquery.com/Frequently_Asked_Questions

http://docs.jquery.com/Tutorials

http://blog.jquery.com/

http://ejohn.org/

http://docs.jquery.com/Sites_Using_jQuery

http://www.learningjquery.com/

Some other blogs and Tutorials

Using jQuery with ASP.NET - A Beginner's Guide

http://encosia.com/2008/09/28/avoid-this-tricky-conflict-between-aspnet-ajax-and-jquery/

http://net.tutsplus.com/articles/web-roundups/jquery-for-absolute-beginners-video-series/

http://www.learningjquery.com/category/levels/beginner http://www.jqueryfordesigners.com/

http://www.learnjquerynow.com/

http://www.smashingmagazine.com/tag/jquery/

My Websites

jQuery and ASP.NET articles

jQuery and ASP.NET Tips

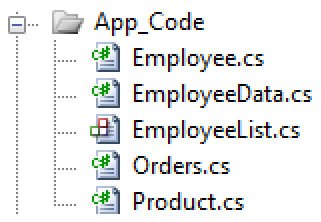
Appendix C: Server-side Code and Other Resources

In some chapters of this eBook, we have used server-side code to populate data. We have also used some HTML Templates and CSS files. This appendix will contain code definitions for all these files:

Server-side Code

App_Code

The App_Code folder in the source code contains a list of classes that are used in some data-bound control. Here's a screenshot of the classes used:



App_Code
Employee.cs
EmployeeData.cs
EmployeeList.cs
Orders.cs
Product.cs

The files have been created using C#. To convert it into VB.NET, you can use this online code convertor tool.

The definition of all the files in the App_Code folder is given here:

Employee.cs

```
using System;
using System.Collections.Generic;

public class Employee
{
    public int ID { get; set; }
    public string FName { get; set; }
    public string MName { get; set; }
    public string LName { get; set; }
    public DateTime DOB { get; set; }
    public char Sex { get; set; }

    public List<Employee> GetEmployeeList()
    {
        List<Employee> empList = new List<Employee>();
        empList.Add(new Employee() { ID = 1, FName = "John", MName = "", LName =
"Shields", DOB = DateTime.Parse("12/11/1971"), Sex = 'M' });
```

```
        empList.Add(new Employee() { ID = 2, FName = "Mary", MName = "Matthew", LName
= "Jacobs", DOB = DateTime.Parse("01/17/1961"), Sex = 'F' });
        empList.Add(new Employee() { ID = 3, FName = "Amber", MName = "Carl", LName =
"Agar", DOB = DateTime.Parse("12/23/1971"), Sex = 'M' });

        empList.Add(new Employee() { ID = 4, FName = "Kathy", MName = "", LName =
"Berry", DOB = DateTime.Parse("11/15/1976"), Sex = 'F' });
        empList.Add(new Employee() { ID = 5, FName = "Lena", MName = "Ashco", LName =
"Bilton", DOB = DateTime.Parse("05/11/1978"), Sex = 'F' });
        empList.Add(new Employee() { ID = 6, FName = "Susanne", MName = "", LName =
"Buck", DOB = DateTime.Parse("03/7/1965"), Sex = 'F' });
        empList.Add(new Employee() { ID = 7, FName = "Jim", MName = "", LName =
"Brown", DOB = DateTime.Parse("09/11/1972"), Sex = 'M' });
        empList.Add(new Employee() { ID = 8, FName = "Jane", MName = "G", LName =
"Hooks", DOB = DateTime.Parse("12/11/1972"), Sex = 'F' });
        empList.Add(new Employee() { ID = 9, FName = "Robert", MName = "", LName = "",
DOB = DateTime.Parse("06/28/1964"), Sex = 'M' });
        empList.Add(new Employee() { ID = 10, FName = "Krishna", MName = "Murali",
LName = "Sunkam", DOB = DateTime.Parse("04/18/1969"), Sex = 'M' });
        empList.Add(new Employee() { ID = 11, FName = "Cindy", MName = "Preston",
LName = "Fox", DOB = DateTime.Parse("06/15/1978"), Sex = 'M' });
        empList.Add(new Employee() { ID = 12, FName = "Nicole", MName = "G",
LName = "Holiday", DOB = DateTime.Parse("08/21/1974"), Sex = 'F' });
        empList.Add(new Employee() { ID = 13, FName = "Sandra", MName = "T",
LName = "Feng", DOB = DateTime.Parse("04/15/1976"), Sex = 'F' });
        empList.Add(new Employee() { ID = 14, FName = "Roberto", MName = "",
LName = "Tamburello", DOB = DateTime.Parse("01/06/1982"), Sex = 'M' });
        empList.Add(new Employee() { ID = 15, FName = "Cynthia", MName = "O",
LName = "Lugo", DOB = DateTime.Parse("01/21/1968"), Sex = 'M' });
        empList.Add(new Employee() { ID = 16, FName = "Yuhong", MName = "R",
LName = "Li", DOB = DateTime.Parse("08/22/1979"), Sex = 'M' });
        empList.Add(new Employee() { ID = 17, FName = "Alex", MName = "",
LName = "Shoop", DOB = DateTime.Parse("03/01/1972"), Sex = 'M' });
        empList.Add(new Employee() { ID = 18, FName = "Jack", MName = "K",
LName = "Brown", DOB = DateTime.Parse("04/11/1978"), Sex = 'M' });
        empList.Add(new Employee() { ID = 19, FName = "Andrew", MName = "U",
LName = "Gibson", DOB = DateTime.Parse("08/21/1977"), Sex = 'M' });
        empList.Add(new Employee() { ID = 20, FName = "George", MName = "K",
LName = "Wood", DOB = DateTime.Parse("07/15/1972"), Sex = 'M' });
        empList.Add(new Employee() { ID = 21, FName = "Eugene", MName = "",
LName = "Miller", DOB = DateTime.Parse("09/13/1974"), Sex = 'M' });
        empList.Add(new Employee() { ID = 22, FName = "Russell", MName = "",
LName = "Gorgi", DOB = DateTime.Parse("08/19/1978"), Sex = 'M' });
        empList.Add(new Employee() { ID = 23, FName = "Katie", MName = "",
LName = "Sylar", DOB = DateTime.Parse("08/21/1978"), Sex = 'M' });
        empList.Add(new Employee() { ID = 24, FName = "Michael", MName = "M",
LName = "Bentler", DOB = DateTime.Parse("11/26/1977"), Sex = 'M' });
        empList.Add(new Employee() { ID = 25, FName = "Barbara", MName = "L",
LName = "Duffy", DOB = DateTime.Parse("05/29/1972"), Sex = 'M' });
        empList.Add(new Employee() { ID = 26, FName = "Stefen", MName = "J",
LName = "Northup", DOB = DateTime.Parse("01/26/1972"), Sex = 'M' });
```

```
        empList.Add(new Employee() { ID = 27, FName = "Shane", MName = "",
LName = "Nay", DOB = DateTime.Parse("02/21/1978"), Sex = 'M' });
        empList.Add(new Employee() { ID = 28, FName = "Brenda", MName = "",
LName = "Lugo", DOB = DateTime.Parse("08/18/1981"), Sex = 'F' });
        empList.Add(new Employee() { ID = 29, FName = "Shammi", MName = "I",
LName = "Rai", DOB = DateTime.Parse("03/13/1968"), Sex = 'M' });
        empList.Add(new Employee() { ID = 30, FName = "Rajesh", MName = "H",
LName = "Vyas", DOB = DateTime.Parse("04/19/1969"), Sex = 'M' });
        empList.Add(new Employee() { ID = 31, FName = "Gabe", MName = "P",
LName = "Lloyd", DOB = DateTime.Parse("08/21/1971"), Sex = 'M' });
        return empList;
    }
}
```

EmployeeData.cs

```
using System.Collections.Generic;

public class EmployeeData
{
    public List<Employee> Employees { get; set; }
    public int TotalRecords { get; set; }
}
```

EmployeeList.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Services;
using System.Web.Script.Services;


[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[ScriptService]
public class EmployeeList : System.Web.Services.WebService {

    [WebMethod]
    public List<Employee> FetchEList() {
        var emp = new Employee();
        return emp.GetEmployeeList();
    }

    [WebMethod]
    public List<Employee> FetchEmpOnGender(char empSex)
    {
        var emp = new Employee();
        var fetchEmp = emp.GetEmployeeList()
            .Where(m => m.Sex == empSex);
        return fetchEmp.ToList();
    }
}
```

Order.cs

```
using System.Collections.Generic;

public class Order
{
    public int OrderID { get; set; }
    public string Country { get; set; }
    public string CustomerName { get; set; }
    public int ID { get; set; }

    public List<Order> GetOrderList()
    {
        List<Order> orderList = new List<Order>();
        orderList.Add(new Order() { OrderID = 10001, Country = "France", CustomerName
= "Maria", ID = 4 });
        orderList.Add(new Order() { OrderID = 10002, Country = "USA",  CustomerName =
"Laurence", ID = 6 });
        orderList.Add(new Order() { OrderID = 10003, Country = "Sweden", CustomerName
= "Victoria", ID = 1 });
        orderList.Add(new Order() { OrderID = 10004, Country = "Denmark", CustomerName
= "Tang", ID = 2 });
        orderList.Add(new Order() { OrderID = 10005, Country = "Sweden", CustomerName
= "Martin", ID = 1 });
        orderList.Add(new Order() { OrderID = 10006, Country = "USA", CustomerName =
"Mario", ID = 1 });
        orderList.Add(new Order() { OrderID = 10007, Country = "Denmark", CustomerName
= "Doshi", ID = 7 });
        orderList.Add(new Order() { OrderID = 10008, Country = "France", CustomerName
= "Marc", ID = 8 });
        orderList.Add(new Order() { OrderID = 10009, Country = "USA", CustomerName =
"Rahul", ID = 7 });
        orderList.Add(new Order() { OrderID = 10010, Country = "Denmark", CustomerName
= "Tim", ID = 9 });
        orderList.Add(new Order() { OrderID = 10011, Country = "France", CustomerName
= "Parth", ID = 13 });
        orderList.Add(new Order() { OrderID = 10012, Country = "USA", CustomerName =
"Malcolm", ID = 11 });
        orderList.Add(new Order() { OrderID = 10013, Country = "France", CustomerName
= "Itzia", ID = 12 });
        orderList.Add(new Order() { OrderID = 10014, Country = "Sweden", CustomerName
= "Charlie", ID = 2 });
        orderList.Add(new Order() { OrderID = 10015, Country = "USA", CustomerName =
"Gwynneth", ID = 2 });
        orderList.Add(new Order() { OrderID = 10016, Country = "France", CustomerName
= "Patrick", ID = 5 });
        orderList.Add(new Order() { OrderID = 10017, Country = "Sweden", CustomerName
= "David", ID = 4 });
        orderList.Add(new Order() { OrderID = 10018, Country = "Denmark", CustomerName
= "Stephen", ID = 17 });
        orderList.Add(new Order() { OrderID = 10019, Country = "France", CustomerName
= "Nikhil", ID = 11 });
```

```
        orderList.Add(new Order() { OrderID = 10020, Country = "USA", CustomerName = 
"Carlo", ID = 15 });
        return orderList;
    }
}
```

Product.cs

```
using System.Collections.Generic;

public class Product
{
    public int ID { get; set; }
    public string ProductName { get; set; }
    public double Price { get; set; }

    public List<Product> GetProductList()
    {
        var productList = new List<Product>();
        productList.Add(new Product() { ID = 1, ProductName = "Chai", Price = 10 });
        productList.Add(new Product() { ID = 2, ProductName = "Aniseed", Price = 15 
});
        productList.Add(new Product() { ID = 3, ProductName = "Ikura", Price = 20 });
        productList.Add(new Product() { ID = 4, ProductName = "Konbu", Price = 25 });
        productList.Add(new Product() { ID = 5, ProductName = "Tofu", Price = 30 });
        productList.Add(new Product() { ID = 6, ProductName = "Rodih Mutton", Price = 
35 });
        productList.Add(new Product() { ID = 7, ProductName = "Chocolate", Price = 40 
});
        productList.Add(new Product() { ID = 8, ProductName = "Ipoh Coffee", Price = 
45 });
        productList.Add(new Product() { ID = 9, ProductName = "Cheese", Price = 50 
});
        productList.Add(new Product() { ID = 10, ProductName = "Butter", Price = 55 
});
        return productList;
    }
}
```

CSS and HTML Templates

CSS

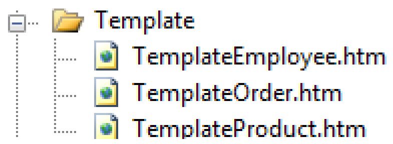
Almost all the examples used in the eBook use the CSS file Demos.css. The definition of this file is as given below:

```
body
{
background-color:White;
}
```

```
body.col
{
  margin-top:0px;
  background-color:#F4F9FE;
}

body.plain
{
 font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana, Arial;
font-size:12px;
}


.bigDiv
{
    background-color:White;
    font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana, Arial;
       font-size:12px;
       color:#000000;
    width:710px;
    margin-left:auto;
    margin-right:auto;
    padding:10px;
    border:solid 1px #c6cfe1;
}

.smallDiv
{
    background-color:White;
    font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana, Arial;
       font-size:12px;
       color:#000000;
    width:510px;
    margin-left:auto;
    margin-right:auto;
    padding:10px;
    border:solid 1px #c6cfe1;
}

.tableDiv
{
    background-color:White;
    font-family: "Lucida Grande", Verdana, Arial;
       font-size:13px;
       color:#000000;
    width:460px;
    margin-left:auto;
    margin-right:auto;
    padding:10px;
    border:solid 1px #c6cfe1;
}

.divNoBorder
{
    background-color:White;
    font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana, Arial;
       font-size:12px;
       color:#000000;
    width:510px;
    margin-left:auto;
    margin-right:auto;
    padding:10px;
}
.wrapper
{
    background-color:White;
    width:300px;
    color:Orange;
}
.exceeded
{
        background-color:red;
}

.water
{
     font-family: Tahoma, Arial, sans-serif;
     font-size:75%;
     color:gray;
}

.highlite
{
    background-color:Gray;
}

a{color:#0033CC;}

.bigDiv input{      font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana,
Arial;
       font-size:12px;
       width:300px;
}

.smallDiv input{    font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana,
Arial;
       font-size:12px;
}

.imgOpa
{
    height:250px;
    width:250px;
    opacity:0.3;
    filter:alpha(opacity=30);
}

.imgthumb
{
    height:100px;
    width:100px;
}

.imgPreview
{
    height:180px;
    width:370px;
}
.imgdiv
{
    background-color:White;
    margin-left:auto;
    margin-right:auto;
    padding:10px;
    border:solid 1px #c6cfe1;
    height:500px;
    width:450px;
}

h2{     color:#333333;
        margin-bottom:5px;
        font-size:18px;
}

div.box{
        border:solid 1px #c6cfe1;
        background:#dfe4ee;
        padding:10px;
        color:#333333;
        margin-bottom:10px;
        width:350px;
}


div.mainnews
{
    font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana, Arial, Helvetica,
sans-serif;
        font-size:12px;
        color:#000000;
        margin:10px auto;

        }

div.boxnews h1 {
        font-size:14px;
        color:#000000;
        padding-bottom:4px;
}
div.boxnews p{padding-bottom:14px;}

div.boxnews input .input-text{
        border:1px solid #3b6e22;
        color:#666666;
        width:210px;
}

div.boxnews label{
        display:block;
        margin-bottom:10px;
        color:#555555;
}

div.boxnews label span{
        display:block;
        float:left;
        padding-right:6px;
        width:50px;
        text-align:left;
        font-weight:bold;
}

.orlist
{
font-style:italic;
font-weight:bold;
}


.cpHeader
{
    color: white;
    background-color: #719DDB;
    font: bold 11px auto "Trebuchet MS", Verdana;
    font-size: 12px;
    cursor: pointer;
    width:450px;
    height:18px;
    padding: 4px;
}
.cpBody
{
    background-color: #DCE4F9;
    font: normal 11px auto Verdana, Arial;
    border: 1px gray;
    width:450px;
    padding: 4px;
    padding-top: 7px;
}


#divDemo
{
    background-color:#F0F0F0;
    font-family:"Lucida Sans Unicode", "Lucida Grande", Verdana, Arial;
        font-size:13px;
        color:#000000;
    margin-left:auto;
    margin-right:auto;
    padding:10px;
    border:solid 1px #666666;
}

.w1
{
    width:710px;
}

.w2
{
    width:510px;
}

.w3
{
    width:460px;
}
```

HTML Templates

HTML templates separate HTML from JavaScript. These templates have been used extensively in most of our GridView and Ajax examples. The folder structure is as shown below:



The definition of all the files in the Template folder is given here:

TemplateEmployee.htm

```
<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>FName</th>
      <th>MName</th>
    </tr>
  </thead>
  <tbody>
    {#foreach $T as data}
    <tr>
      <td>{$T.data.ID}</td>
      <td>{$T.data.FName}</td>
      <td>{$T.data.MName}</td>
    </tr>
    {#/for}
  </tbody>
</table>
```

TemplateOrder.htm

```
<table>
  <thead>
    <tr>
      <th>OrderID</th>
      <th>Country</th>
      <th>CustomerName</th>
    </tr>
  </thead>
  <tbody>
    {#foreach $T as data}
    <tr>
      <td>{$T.data.OrderID}</td>
      <td>{$T.data.Country}</td>
      <td>{$T.data.CustomerName}</td>
    </tr>
    {#/for}
  </tbody>
</table>
```

TemplateProduct.htm

```
<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>ProductName</th>
      <th>Price</th>
    </tr>
  </thead>
  <tbody>
    {#foreach $T as data}
    <tr>
      <td>{$T.data.ID}</td>
      <td>{$T.data.ProductName}</td>
      <td>{$T.data.Price}</td>
    </tr>
    {#/for}
  </tbody>
</table>
```

How to make a Recipe work with MasterPages

In this subsection, I will demonstrate how to convert a Recipe to make it work in a MasterPage scenario. Make sure you have the source code of this eBook with you, to understand what's being explained here. We will convert the recipe R6-TextBoxCloning.aspx kept in Section1 and make it work with a MasterPage.

Step 1: Create a Master Page (MasterPage.master) and add references to the jQuery Library and our custom css file (Demos.css) as shown below:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs"
Inherits="MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Master Page</title>
    <script src="Scripts/jquery-1.3.2.min.js"
type="text/javascript"></script>
    <link href="CSS/Demos.css" rel="stylesheet" type="text/css" />
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

Step 2: Create a Content Page and add the form content from R6-TextBoxCloning.aspx to this page as shown below:

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <!-- jQuery code will come here -->
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
   <div class="smallDiv">
        <h2>Enter text in the top textbox to see it cloned in the
        bottom textbox</h2>
        <br />
        <asp:TextBox ID="tb1" runat="server" /><br />
        <asp:TextBox ID="tb2" runat="server" />
        <br /><br /><br />
        Tip: Entering text in bottom box does not do anything.
    </div>
</asp:Content>
```

Step 3: The last step is to add the jQuery code in your Content Page as shown below:

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <script type="text/javascript">
        $(function() {
            $('input[id$=tb1]').keyup(function() {
                var txtClone = $(this).val();
                $('input[id$=tb2]').val(txtClone);
            });
        });
    </script>
</asp:Content>
```

Recommended Approach

Instead of adding the jQuery code directly into your Content page, create a separate JavaScript file (call it txtClone.js) and add the jQuery code to it:

```
$(function() {
    $('input[id$=tb1]').keyup(function() {
        var txtClone = $(this).val();
        $('input[id$=tb2]').val(txtClone);
    });
});
```

Then reference this JavaScript file (txtClone.js) in your Content Page as shown below:

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <script src="Scripts/txtClone.js" type="text/javascript"></script>
</asp:Content>
```

To see this code in action, check the source code that came along with this eBook. The source code has a sample folder called ‘jQueryWithMasterPageSample’ which demonstrates the technique explained in this section.

To use jQuery on a MasterPage that uses ASP.NET AJAX, check this link.