Introduction
To
JavaScript
And
jQuery

Overview of jQuery

This document will give you a very rough overview of jQuery, and will show a few examples at the very end. Some of the following sections are taken from the book 'jQuery in Action'

What is jQuery?

jQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML. It was released in January 2006 at BarCamp NYC by John Resig. JQuery is currently used by over 30% of the 10,000 most visited websites, and is the most popular JavaScript library in use today.

jQuery is free, open source software, dual-licensed under the MIT License and the GNU General Public License, Version 2. jQuery's syntax is designed to make it easier to navigate a document, select DOM (Document Object Model) elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. Using these facilities, developers are able to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. This contributes to the creation of powerful and dynamic web pages.

Microsoft and Nokia have announced plans to bundle jQuery on their platforms, Microsoft adopting it initially within Visual Studio for use within Microsoft's ASP.NET AJAX framework and ASP.NET MVC Framework while Nokia has integrated it into their Web Run-Time widget development platform. JQuery is also implemented in MediaWiki since version 1.16.

jQuery UI

jQuery UI provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, built on top of the jQuery JavaScript Library, that you can use to build highly interactive web applications.

Current versions

- jQuery: 1.10.3
- jQuery UI: 1.10.3

Used Version

- jQuery: 1.4.2
- jQuery UI: 1.8.16

Why Use jQuery?

jQuery has quickly won the support of major companies for use in mission-critical applications. Some of jQuery's prominent users include the likes of IBM, Netflix, Amazon, Dell, Best Buy, Twitter, Bank of America, and scores of other prominent companies. Microsoft has even elected to distribute jQuery with its Visual Studio tool, and Nokia uses jQuery on all its phones that include their Web Runtime component.

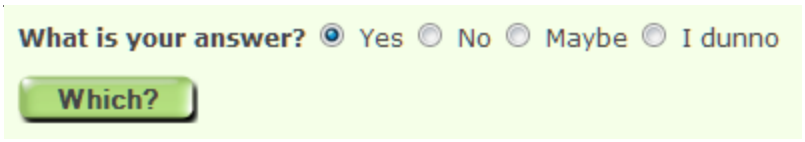
Compared with other toolkits that focus heavily on clever JavaScript techniques, jQuery aims to change the way that web developers think about creating rich functionality in their pages. Rather than spending time juggling the complexities of advanced JavaScript, designers can leverage their existing knowledge of Cascading Style Sheets (CSS), Hypertext Markup Language (HTML), Extensible Hypertext Markup Language (XHTML), and good old straightforward JavaScript to manipulate page elements directly, making rapid development a reality.

Simple Code Example

The HTML code looks like:

```
<label for="radioYes">What is your answer?</label>
<input type="radio" name="RdGrp" id="Yes" value="yes"/> Yes
<input type="radio" name="RdGrp" id="No" value="no"/> No
<input type="radio" name="RdGrp" id="Maybe" value="maybe"
  checked="checked"/> Maybe
<input type="radio" name="RdGrp" id="Confused" value="confused"/> I dunno
<div><button type="button" id="testButton">Which?</button></div>
```

The page looks like:



Using JavaScript to determine which radio button is selected could look like:

```
var checkedValue;
var elements = document.getElementsByTagName('input');
for (var n = 0; n < elements.length; n++) {
  if (elements[n].type == 'radio' &&
    elements[n].name == 'RdGrp' &&
    elements[n].checked) {
    checkedValue = elements[n].value;
  }
}
```

In jQuery this looks like:

```
var checkedValue = $('[name="RdGrp"]:checked').val();
```

We will get into the structure later, but that's so much simpler!!

Unobtrusive JavaScript

You may recall the 'old' days (some of us remember that) before CSS, when we were forced to mix stylistic markup with the document structure markup in our HTML pages. Anyone who's been authoring pages for any amount of time surely does, most likely with less than fondness.

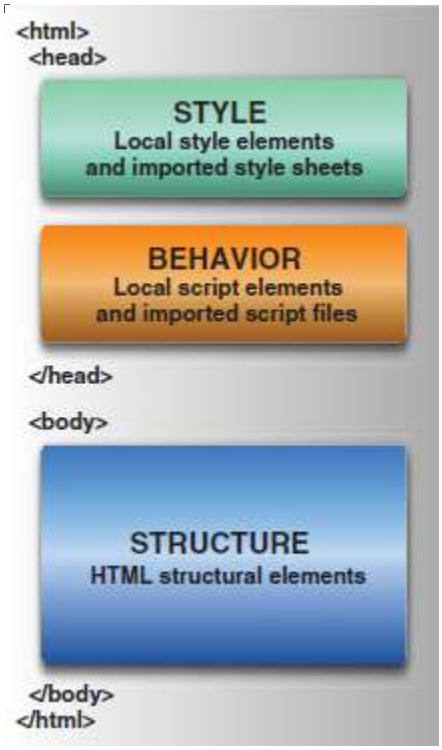
The addition of CSS to our web development toolkits allows us to separate stylistic information from the document structure and gives travesties like the <font> tag the well-deserved boot. Not only does the separation of style from structure make our documents easier to manage, it also gives us the versatility to completely change the stylistic rendering of a page by simply swapping out different style sheets.

Few of us would voluntarily regress back to the days of applying styles with HTML elements; yet markup such as the following is still all too common:

```
<button type="button"
  onclick="document.getElementById('xyz').style.color='red';">
  Click Me
</button>
```

You can easily see that the style of this button element, including the font of its caption, isn't applied via the use of the `<font>` tag and other deprecated style-oriented markup, but is determined by whatever CSS rules (not shown) are in effect on the page. But although this declaration doesn't mix *style* markup with structure, it does mix *behavior* with structure by including the JavaScript to be executed when the button is clicked as part of the markup of the button element via the `onclick` attribute (which, in this case, turns some Document Object Model (DOM) element with the id value of `xyz` red).

Separating behavior from structure



For all the same reasons that it's desirable to segregate style from structure within an HTML document, it's just as beneficial (if not more so) to separate the behavior from the structure.
Ideally, an HTML page should be structured as shown in figure with structure, style, and behavior each partitioned nicely in its own niche.

This strategy, known as **Unobtrusive JavaScript**, was brought into the limelight by the inventors of jQuery and is now embraced by every major JavaScript library, helping page authors achieve this useful separation on their pages. As the library that popularized this movement, jQuery's core is well optimized for producing Unobtrusive JavaScript quite easily. Unobtrusive JavaScript considers any JavaScript expressions or statements embedded in the `<body>` of HTML pages, either as attributes of HTML elements (such as `onclick`) or in script blocks placed within the body of the page, to be incorrect.

"But how can I instrument the button without the `onclick` attribute?" you might ask. Consider the following change to the button element:

```
<button type="button" id="testButton">Click Me</button>
```

And the code to apply the style is located in the <head> section of the page, and looks like:

```
<script type="text/javascript">
  window.onload = function() {
    document.getElementById('testButton').onclick = function() {
      document.getElementById('xyz').style.color = 'red';
    };
  };
</script>
```

This is still JavaScript and still is fairly complicated, it gets the job done, but the behavior is removed from the HTML structure. jQuery will help accomplish this in a larger scale.

jQuery Fundamentals

At its core, jQuery focuses on retrieving elements from HTML pages and performing operations upon them. If you're familiar with CSS, you're already well aware of the power of selectors, which describe groups of elements by their type, attributes, or placement within the document. With jQuery, we can employ that knowledge, and that degree of power, to vastly simplify your JavaScript.

jQuery Wrapper

To collect a group of elements, we pass the selector to the jQuery function using the simple syntax:

```
$(selector)
```

Or

```
jQuery(selector)
```

Here are a few examples of selectors:

Selector	Results
`$("p:even")`	Selects all even <p> elements
`$("tr:nth-child(1)")`	Selects the first row of each table
`$("body > div")`	Selects direct children of <body>
`$("a[href$= 'pdf ']")`	Selects links to PDF files
`$("body > div:has(a)")`	Selects direct children of <body>-containing links

The possibilities are unlimited.

Utility Functions

The notation for general-purpose functions looks a little odd at first. Below is the function to trim a string:

```
var trimmed = $.trim(someString);
```

If the **`$.`** prefix looks weird to you, remember that `$` is an identifier like any other in JavaScript. Writing a call to the same function using the **`jQuery`** identifier, rather than the `$` alias, may look a bit less odd:

```
var trimmed = jQuery.trim(someString);
```

The Document ready handler

When embracing Unobtrusive JavaScript, behavior is separated from structure, so we're performing operations on the page elements outside of the document markup that creates them. In order to achieve this, we need a way to wait until the DOM elements of the page are fully realized before those operations execute. In the radio group example, the entire body must load before the behavior can be applied.

Traditionally, the **`onload`** handler for the `window` instance is used for this purpose, executing statements after the entire page is fully loaded. The syntax is typically something like:

```
window.onload = function() {
  // do stuff here
};
```

This causes the defined code to execute after the document has fully loaded. Unfortunately, the browser not only delays executing the `onload` code until after the DOM tree is created, but also waits until after all external resources are fully loaded and the page is displayed in the browser window. This includes not only resources like images, but QuickTime and Flash videos embedded in web pages, and there are more and more of them these days. As a result, visitors can experience a serious delay between the time that they first see the page and the time that the `onload` script is executed.

Even worse, if an image or other resource takes significant time to load, visitors will have to wait for the image loading to complete before the rich behaviors become available. This could make the whole Unobtrusive JavaScript proposition a non-starter for many real-life cases.

A much better approach would be to wait only until the document structure is fully parsed and the browser has converted the HTML into its resulting DOM tree before executing the script to apply the rich behaviors. Accomplishing this in a cross-browser manner is somewhat difficult, but jQuery provides a simple means to trigger the execution of code once the DOM tree has loaded (without waiting for external resources). The formal syntax to define such code (using the example above) is as follows:

```
$(document).ready(function() {
  // do stuff here
});
```

A short form of the same functionality looks like:

```
$(function() {
  // do stuff here
});
```

By passing a function to **`jQuery()`** or **`$()`**, we instruct the browser to wait until the DOM has fully loaded (but only the DOM) before executing the code. Even better, we can use this technique multiple times within the same HTML document, and the browser will execute all of the functions we specify in the order that they are declared within the page. In contrast, the window's **`onload`** technique allows for only a single function. This limitation can also result in hard-to-find bugs if any included third-party code uses the **`onload`** mechanism for its own purpose (not a best-practice approach).

Insert HTML Code

```
<html>
  <head>
    <title>Follow me!</title>
    <link rel="stylesheet" type="text/css" href="../styles/core.css"/>
    <script type="text/javascript" src="js/jquery-1.4.2.js"></script>
    <script type="text/javascript">
      $(function(){
        $("<p>Hi there!</p>").insertAfter("#followMe");
      });
    </script>
  </head>
  <body>
    <p id="followMe">Follow me!</p>
  </body>
</html>
```

This example establishes an existing HTML paragraph element named **followMe** in the document body. In the script element within the **<head>** section, we establish a ready handler that uses the following statement to insert a newly created paragraph into the DOM tree after the existing element:

```
$("<p>Hi there!</p>").insertAfter("#followMe");
```

The resulting page will show 'Hi there!' paragraph after the 'Follow me!' paragraph.

Manipulate Table Rows

As a web designer we all have used tables, and all have tried to highlight every other row to easier separate the rows. In the past I would use a class that would set the background color for every other row. If the data was static, and a new row needed to be inserted, all the rows following needed to have the class updated. In jQuery this is very simple. Below is a table with the id of 'demo':

```
<table id="demo">
  <tr><th>Year</th><th>Make</th><th>Model</th></tr>
  <tr><td>1965</td><td>Ford</td><td>Mustang</td></tr>
  <tr><td>1970</td><td>Toyota</td><td>Corolla</td></tr>
  <tr><td>1979</td><td>AMC</td><td>Jeep CJ-5</td></tr>
  <tr><td>1983</td><td>Ford</td><td>EXP</td></tr>
  <tr><td>1985</td><td>Dodge</td><td>Daytona</td></tr>
  <tr><td>1990</td><td>Chrysler</td><td>Jeep Wrangler Sahara</td></tr>
  <tr><td>1995</td><td>Ford</td><td>Ranger</td></tr>
  <tr><td>1997</td><td>Chrysler</td><td>Jeep Wrangler Sahara</td></tr>
</table>
```

The jQuery code to highlight every other row looks like:

```
<script type="text/javascript">
  $(function(){
    // Highlight all odd rows
    $("table#demo tr:nth-child(odd)").addClass("striped");
    // Swap classes
    $("table#demo th").mouseover(swapThem).mouseout(swapThem);
  });

  function swapThem() {
    $('table#demo tr').toggleClass('striped');
  }
</script>
```

The selector **`$("table#demo tr:nth-child(odd)")`**, means that this applies to all odd rows in the table with the id of 'demo'. The **`swapThem()`** function actually does the swapping.

The resulting table looks like:

Year	Make	Model
1965	Ford	Mustang
1970	Toyota	Corolla
1979	AMC	Jeep CJ-5
1983	Ford	EXP
1985	Dodge	Daytona
1990	Chrysler	Jeep Wrangler Sahara
1995	Ford	Ranger
1997	Chrysler	Jeep Wrangler Sahara

The extra line of code will swap the class for all row if the mouse hovers over the header row of the table (not very pretty but it gives you an idea of the power of jQuery), as shown below.

Year	Make	Model
1965	Ford	Mustang
1970	Toyota	Corolla
1979	AMC	Jeep CJ-5
1983	Ford	EXP
1985	Dodge	Daytona
1990	Chrysler	Jeep Wrangler Sahara
1995	Ford	Ranger
1997	Chrysler	Jeep Wrangler Sahara

jQuery UI Widgets and Plug-ins

The jQuery UI has a number of standard widgets that are very easy to implement. Below is the list:

- Accordion
- Autocomplete
- Button
- Datepicker
- Dialog
- Menu
- Progress bar
- Slider
- Spinner
- Tab
- Tooltip

When downloading the jQuery and jQuery UI code from the official web site, it is possible to define which theme and which options that you want download. There are 20+ different themes to pick from, and they all include 174 themed icons. The Themed Icons can be displayed on the test web site under the Demo tab. The interesting thing about the icons is, that they all are located in one graphics file, but in the style sheet each icon is defined as a small section of the graphics file. Since all themes use the same naming convention, we can simply replace the style sheet and add the new graphics file, and we are all set.

Buttons Widget

Using the downloaded theme will make it very easy to use the selected theme. All we need is to include the following code:

```
// Set style for all buttons with the class of 'button1'
$('.button1').button();
```

Before:

After:

This will actually use the loaded and active theme (there are a number of predefined themes that can be downloaded and modified if necessary).

DatePicker and Accordion Widget

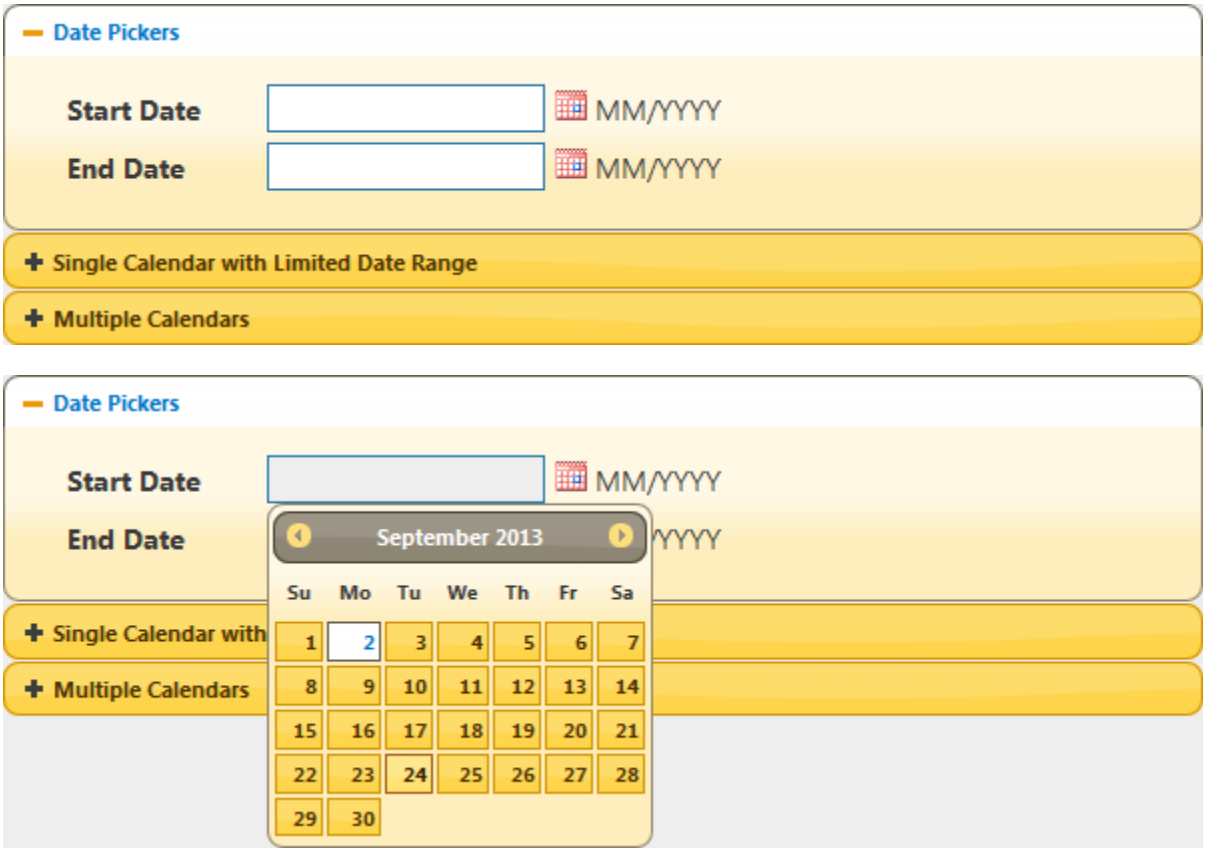
In order to use a popup or static calendar, we can use the DatePicker widget, and the code to create the two data pickers and some static calendars looks like this:

```
<script type="text/javascript">
  $(function(){
    // Calendar Code
    $('#start_date').datepicker({
      showOn: 'button',
      buttonImage: 'img/calendar1.gif',
      buttonImageOnly: true,
      numberOfMonths: 1,
      showAnim: 'fadeIn',
      constrainInput: true,
      duration: 'slow'
    });
    $('#end_date').datepicker({
      showOn: 'button',
      buttonImage: 'img/calendar1.gif',
      buttonImageOnly: true,
      numberOfMonths: 1,
      showAnim: 'slideDown',
      constrainInput: true,
      duration: 'slow'
    });
    $("#datepicker1").datepicker({
      showWeek: true,
      changeMonth: true,
      changeYear: true,
      minDate: '01/01/2009',
      maxDate: '12/31/2011',
      firstDay: 1
    });
    $("#datepicker3").datepicker({
      numberOfMonths: [2,2],
      stepMonths: 4
    });
    // Default date picker dates
    $('#start_date').val('01/01/2010');
    $('#end_date').val(getDate());
    // Accordion code
    $('#accordion').accordion({
      autoHeight: false,
      fillspace: true,
      collapsible: true,
      icons: {
        'header': 'ui-icon-plusthick',
        'headerSelected': 'ui-icon-minusthick'
      }
    });
  });
</script>
```

And the HTML code looks like (code for the accordion effect is marked in red):

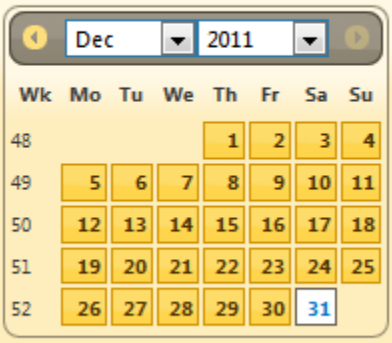
```
<div id="accordion">
  <h2><a href="#">Date Pickers</a></h2>
  <div id="content_panel_1" class="date-panel">
    <span class="label">Start Date</span>
    <input id="start_date" name="date_picker" type="text" /><br />
    <span class="label">End Date</span>
    <input id="end_date" name="date_picker" type="text" />
  </div>
  <h2><a href="#">Single Calendar with Limited Date Range</a></h2>
  <div id="content_panel_2">
    <div id="calendar1" style="font-size:80%;"></div>
  </div>
  <h2><a href="#">Multiple Calendars</a></h2>
  <div id="content_panel_3">
    <div id="calendar4" style="font-size:80%;"></div>
  </div>
</div>
```

All we need are the appropriate jQuery JavaScripts (jQuery 1.4.x and jQuery UI 1.8.x), and the associated style sheet. The result looks like the images below for the three accordion tabs:



+ Date Pickers

− Single Calendar with Limited Date Range

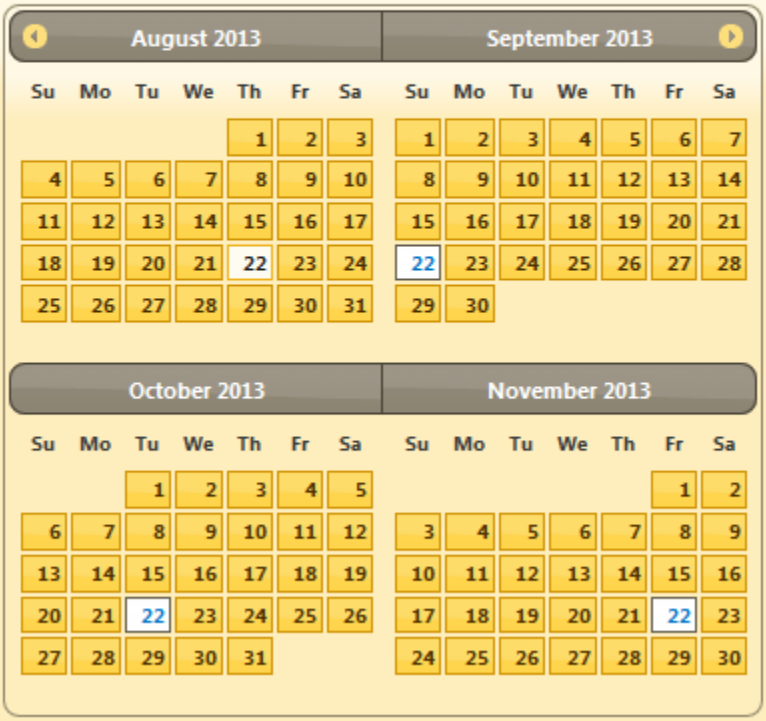


+ Multiple Calendars

+ Date Pickers

+ Single Calendar with Limited Date Range

− Multiple Calendars



A few calendar features are:

- Specify date range (minDate, maxDate)
- Specify button image
- Display more than one month (see above)
- Validate input for valid characters (as based on the date format selected)
- Show the week number
- Specify which is the first day of the week (Sunday or Monday)
- Show week or year dropdowns in header
- And many, many more features

Progress Bar Widget

The progress bar code is as simple as follows:

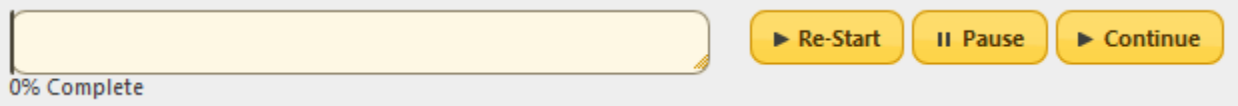
```
<script type="text/javascript">
  $(function(){
    // Progress bar settings and events
    $("#progressbar").progressbar({ value: 0 });
    $("#progressbarWrapper").resizable();
    $("#restart_progress").click(restartProgress); // On button click restart
    $("#continue_progress").click(startProgress); // On button click continue
    $("#pause_progress").click(stopProgress); // On button click pause

    // The buttons
    $('#restart_progress').button({icons:{primary: 'ui-icon-seek-start'}});
    $('#pause_progress').button({ icons:{primary: 'ui-icon-pause'}});
    $('#continue_progress').button({ icons:{primary: 'ui-icon-play'}});
  });

  var StopNow = false;
  function startProgress() {
    setTimeout(updateProgress, 100);
    StopNow = false;
  };

  function restartProgress() {
    setTimeout(updateProgress, 100);
    $('#progressbar').progressbar('value', 0);
    StopNow = false;
  };

  function stopProgress() {
    clearTimeout(updateProgress);
    $('#progressbar').progressbar('stop');
    StopNow = true;
  };

  function updateProgress() {
    var progress = $("#progressbar").progressbar("option", "value");
    if ((progress < 100) && (StopNow == false)) {
      $("#progressbar").progressbar("option", "value", progress + 1);
      setTimeout(updateProgress, 100);
      $("#percentcomplete").html(progress + 1 + "% Complete");
    } else {
      clearTimeout(updateProgress);
    }
  };
</script>
```

The code includes a timer to simulate the progress bar, and it can be started/stopped by clicking the [Restart], [Pause] or [Continue] buttons.

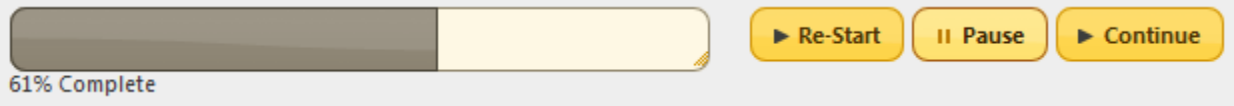
The HTML code looks like this:

```
<table class="example">
  <tr>
    <td>
      <div id="progressbarWrapper" class="ui-widget-default"
      style="height:30px;">
        <div id="progressbar" style="height: 100%"></div>
      </div>
      <div id="percentcomplete" class="val">0% Complete</div>
    </td>
    <td class="button_cell">
      <button id="restart_progress" type="button"
        class="button1">Restart</button>
      <button id="pause_progress" type="button"
        class="button1">Pause</button>
      <button id="continue_progress" type="button"
        class="button1">Continue</button>
    </td>
  </tr>
</table>
```

The progress bar is shown below:



In the example the progress bar starts at 0%. The [Continue] button will start from the current reading, while [Restart] button will reset the reading to 0% and then start.



Slider Widget

As with the other widgets, the slider widget is equally simple.

```
<script type="text/javascript">
  $(function(){
    $("#slider").slider({
      min: 1,
      max: 100,
      value: 50,
      slide: function (event, ui) {
        $('#slidervalue').html('Value: ' + ui.value); }
    });

    $("#sliderrange").slider({
      range: true,
      min: 1,
      max: 100,
      values: [25, 75],
      step: 5,
      slide: function (event, ui) {
        $('#sliderrangevalue').html('Values from ' + ui.values[0] + ' to '
          + ui.values[1]);
      }
    });

    $("#slidervert").slider({
      min: 1,
      max: 100,
      value: 0,
      orientation: 'vertical',
      slide: function (event, ui) {
        $('#slidervertvalue').html('Value: ' + ui.value); }
    });
  });
</script>
```

And the HTML code looks like

```
<table class="example">
  <tr>
    <td valign="top" align="center">
      <h2>Normal Slider</h2>
      <div id="slider"></div>
      <p id="slider_value" class="val">Value: 50</p>
    </td>
    <td rowspan="3" valign="top" align="center">
      <h2>Vertical Slider</h2>
      <div id="slider_vert"></div>
      <p id="slider_vert_value" class="val">Value: 1</p>
    </td>
  </tr>
  <tr>
    <td valign="middle" align="center">
      <h2>Slider Range (Increments of 10)</h2>
      <div id="slider_range"></div>
      <p id="slider_range_value" class="val">Values from 100 to 300</p>
    </td>
  </tr>
  <tr>
    <td valign="bottom" align="center">
      <h2>Big Slider (Increments of 25)</h2>
      <div id="slider_big"></div>
      <p id="slider_big_value" class="val">Value 500</p>
    </td>
  </tr>
</table>
```

A few slider features are:

- Specify range (min, max)
- Specify orientation (horizontal or vertical)
- Setting the range allows us to define a start and end range
- Specify the interval between steps

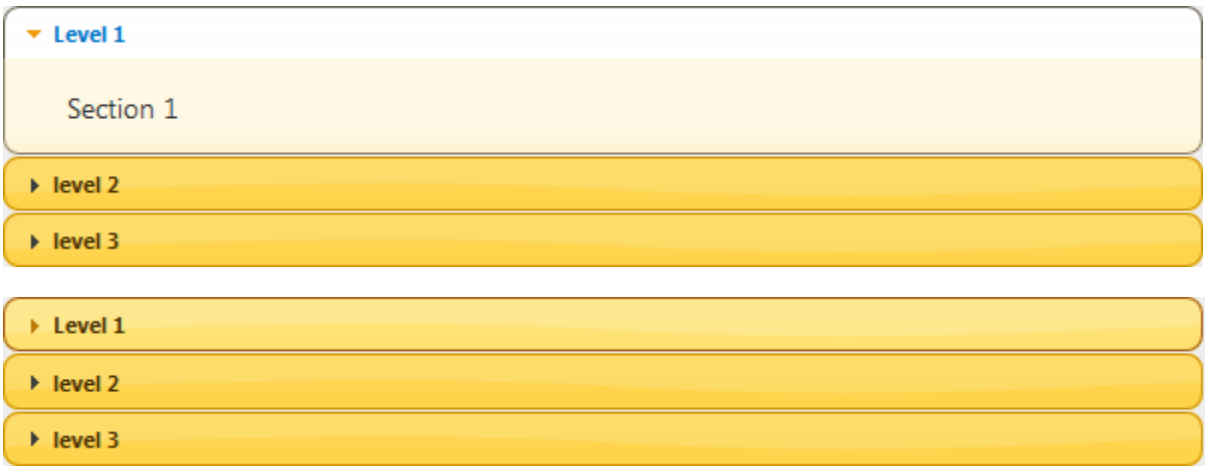
Accordion Widget

We have shown an example of the accordion widget earlier. The accordion widget is meant to be used to open one section at a time.

```
<script type="text/javascript">
  $(function(){
    // Accordion code
    $('#accordion').accordion({
      autoHeight: false,
      fillspace: true,
      collapsible: true
    });
  });
</script>
```

And the HTML code looks like:

```
<div id="accordion">
  <h2><a href="#">Date Pickers</a></h2>
  <div id="content_panel_1" class="date-panel">
    Section 1
  </div>
  <h2><a href="#">Single Calendar with Limited Date Range</a></h2>
  <div id="content_panel_2">
    Section 2
  </div>
  <h2><a href="#">Multiple Calendars</a></h2>
  <div id="content_panel_3">
    Section 3
  </div>
</div>
```



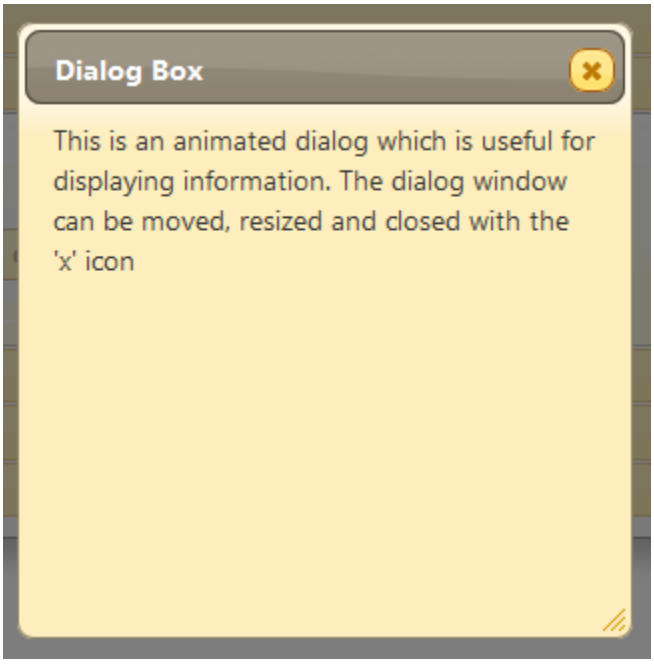
Dialog Box Widget

The dialog box will allow us to create popup windows with dynamic content. At the same time the rest of the page can be inactive until the popup window is closed

```
<script type="text/javascript">
  $(function () {
    $('#dialog').dialog({
      autoOpen: false,
      modal: true,
      hide: 'hide'
      width: 300,
      height: 300
    });

    $('#opener1').click(function () {
      $('#dialog').dialog('open');
      $('#dialog').html("<p>This is an animated dialog which is useful for
        displaying information. The dialog window can be moved, resized and
        closed with the 'x' icon</p>");
      $('#dialog').dialog({
        title: "Dialog Box",
        hide: 'blind'
      });
      return false;
    });

    $('#opener2').click(function () {
      $('#dialog').html("Open the window with new content");
      $('#dialog').dialog({
        show: 'slide',
        title: "Different Header",
        hide: 'explode'
      });
      $('#dialog').dialog('open');
      return false;
    });
  });
</script>
```

And the HTML code looks like:

```
<button id="opener1" class="button1">Open Dialog1</button>
<button id="opener2" class="button1">Open Dialog2</button>
<div id="dialog"></div>
```

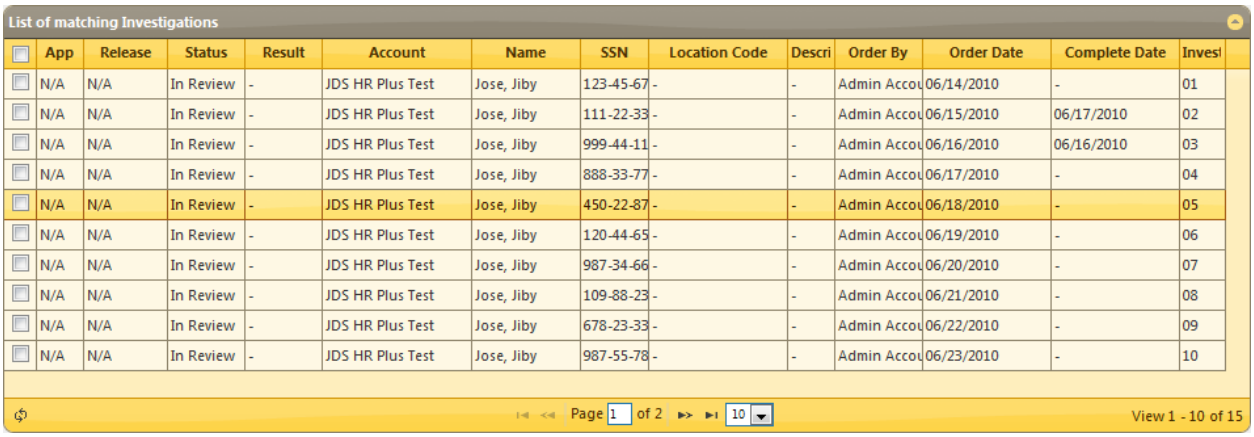


A few dialog features are:

- Specify height and width
- Specify maximum height and width
- Specify if the rest of the page should be inactive (modal = true)
- Specify the show and hide effects
- Specify if the dialog box should be resizable
- Specify the title of the dialog box
- Specify Modal. If set to 'true', that the page is locked and only the dialog box is active.

Data Grid Plug-in

We will be using data grids on the new web site. Below is a data grid created using jQuery, where the data shown is retrieved from an XML file (as an example):



List of matching Investigations

App	Release	Status	Result	Account	Name	SSN	Location Code	Descri	Order By	Order Date	Complete Date	Invest
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	123-45-67	-	-	Admin Accou	06/14/2010	-	01
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	111-22-33	-	-	Admin Accou	06/15/2010	06/17/2010	02
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	999-44-11	-	-	Admin Accou	06/16/2010	06/16/2010	03
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	888-33-77	-	-	Admin Accou	06/17/2010	-	04
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	450-22-87	-	-	Admin Accou	06/18/2010	-	05
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	120-44-65	-	-	Admin Accou	06/19/2010	-	06
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	987-34-66	-	-	Admin Accou	06/20/2010	-	07
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	109-88-23	-	-	Admin Accou	06/21/2010	-	08
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	678-23-33	-	-	Admin Accou	06/22/2010	-	09
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	987-55-78	-	-	Admin Accou	06/23/2010	-	10

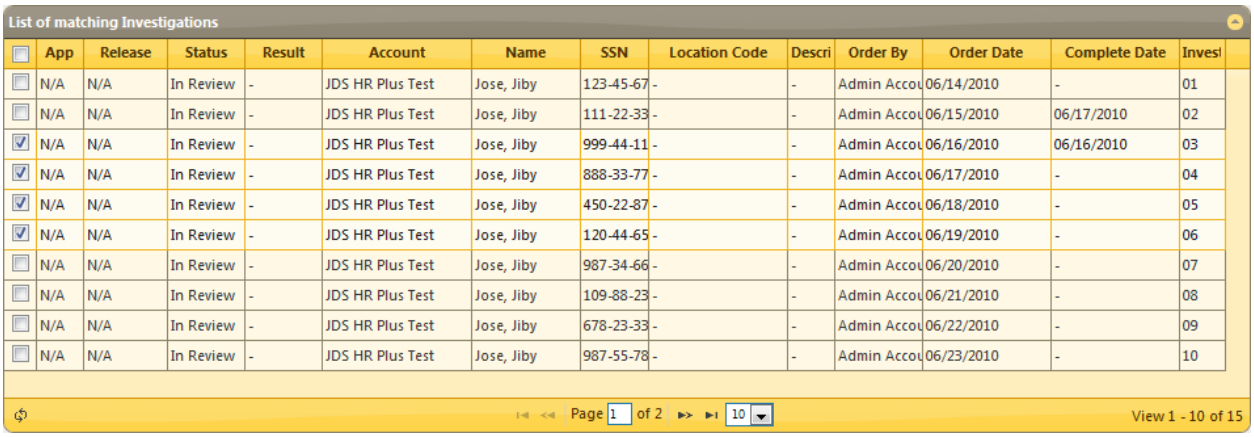
Page 1 of 2 10 View 1 - 10 of 15

The data grid shown above is using version 3.7.2 of jqGrid, and the displayed data is based on an XML file

A few data grid features are:

- Sort data
- Paging
- Remove columns
- Resize columns
- Select rows per page from drop down box
- Reload the data
- Select multiple records (see second screen shot)
- Search for data (not used here)
- Edit, add or delete rows (not used here)
- MVC compatible
- Supports AJAX

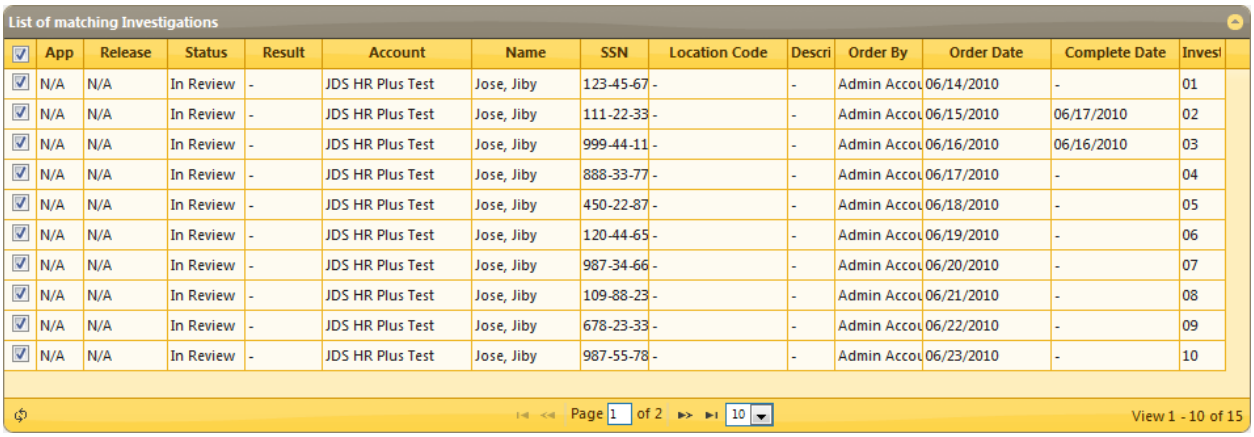
Select Multiple Rows


List of matching Investigations

App	Release	Status	Result	Account	Name	SSN	Location Code	Descri	Order By	Order Date	Complete Date	Invest
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	123-45-67	-	-	Admin Accou	06/14/2010	-	01
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	111-22-33	-	-	Admin Accou	06/15/2010	06/17/2010	02
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	999-44-11	-	-	Admin Accou	06/16/2010	06/16/2010	03
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	888-33-77	-	-	Admin Accou	06/17/2010	-	04
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	450-22-87	-	-	Admin Accou	06/18/2010	-	05
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	120-44-65	-	-	Admin Accou	06/19/2010	-	06
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	987-34-66	-	-	Admin Accou	06/20/2010	-	07
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	109-88-23	-	-	Admin Accou	06/21/2010	-	08
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	678-23-33	-	-	Admin Accou	06/22/2010	-	09
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	987-55-78	-	-	Admin Accou	06/23/2010	-	10

Page 1 of 2 10 View 1 - 10 of 15

Select All Rows

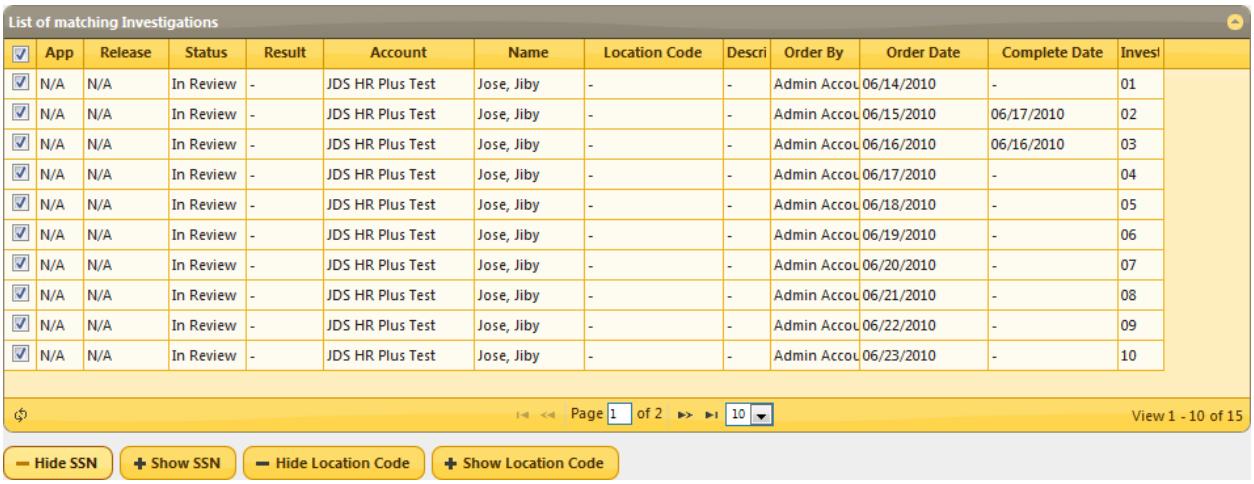

List of matching Investigations

App	Release	Status	Result	Account	Name	SSN	Location Code	Descri	Order By	Order Date	Complete Date	Invest
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	123-45-67	-	-	Admin Accou	06/14/2010	-	01
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	111-22-33	-	-	Admin Accou	06/15/2010	06/17/2010	02
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	999-44-11	-	-	Admin Accou	06/16/2010	06/16/2010	03
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	888-33-77	-	-	Admin Accou	06/17/2010	-	04
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	450-22-87	-	-	Admin Accou	06/18/2010	-	05
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	120-44-65	-	-	Admin Accou	06/19/2010	-	06
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	987-34-66	-	-	Admin Accou	06/20/2010	-	07
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	109-88-23	-	-	Admin Accou	06/21/2010	-	08
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	678-23-33	-	-	Admin Accou	06/22/2010	-	09
N/A	N/A	In Review	-	JDS HR Plus Test	Jose, Jiby	987-55-78	-	-	Admin Accou	06/23/2010	-	10

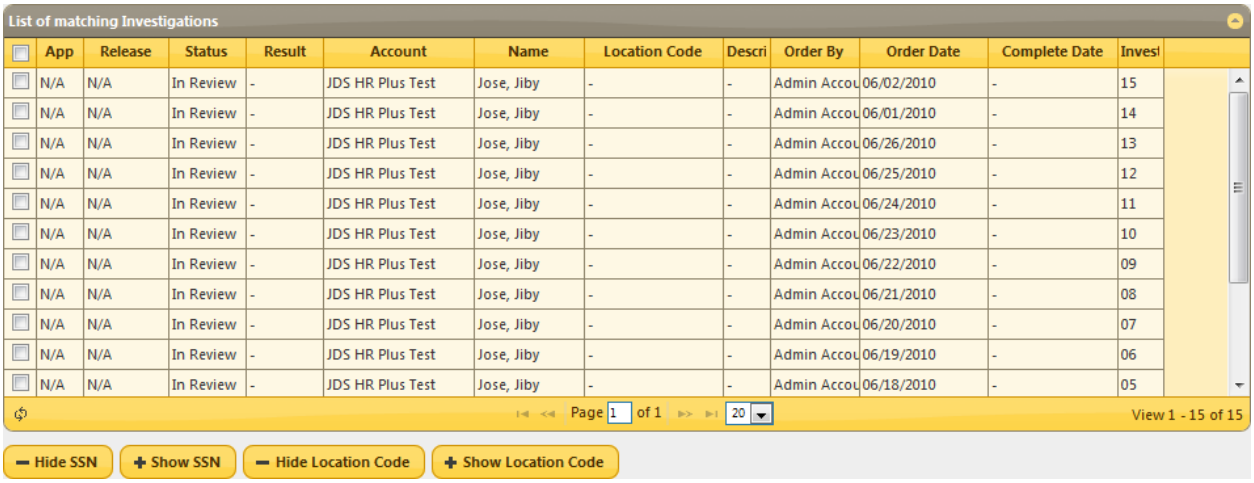
Page 1 of 2 10 View 1 - 10 of 15

All these features and many more are standard data grid features. DevExpress has a grid view control that can be bound directly to a database.

Remove Columns



Change the number of rows per page, which will add a scroll bar as required:



The code for the data grid can be complicated:

```
$.jgrid.no_legacy_api = true;
$.jgrid.useJSON = true;

$("#data_grid").jqGrid({
  url: 'data/investigations.xml',
  datatype: "xml",
  colNames:["App","Release","Status","Result","Account","Name","SSN","Location
    Code","Description","Order By","Order Date","Complete Date","Investigation"],
  colModel:[
    {name:"Application",index:"Application",width:20,
      xmlmap:"ItemAttributes>Application"},
    {name:"Release",index:"Release",width:38,xmlmap:"ItemAttributes>Release"},
    {name:"Status",index:"Status",width:35,xmlmap:"ItemAttributes>Status"},
    {name:"Result",index:"Result",width:35,xmlmap:"ItemAttributes>Result"},
    {name:"Account",index:"Account",width:70,xmlmap:"ItemAttributes>Account"},
    {name:"Name",index:"Name",width:50,xmlmap:"ItemAttributes>Name"},
    {name:"SSN",index:"SSN",width:30,xmlmap:"ItemAttributes>SSN"},
    {name:"Code",index:"Code",width:65,xmlmap:"ItemAttributes>Code",hidden:false},
    {name:"Description",index:"Description",width:20,
      xmlmap:"ItemAttributes>Description"},
    {name:"OrderBy",index:"OrderBy",width:40,xmlmap:"ItemAttributes>OrderBy"},
    {name:"OrderDate",index:"OrderDate",width:60,xmlmap:"ItemAttributes>OrderDate",
      sorttype:"date"},
    {name:"CompleteDate",index:"CompleteDate",width:60,
      xmlmap:"ItemAttributes>CompleteDate",sorttype:"date"},
    {name:"Investigation",index:"Investigation",width:20,
      xmlmap:"ItemAttributes>Investigation"}],
  height:250,
  width: 950,
  rowNum:10,
  rowList:[5,10,20,30],
  pager: '#table_pager',
  sortname: 'Investigation',
  viewrecords: true,
  loadonce: true,
  sortorder: 'desc',
  multiselect: true,
  xmlReader: {
    root : 'Items',
    row: 'Item',
    repeatitems: false,
    id: 'Investigation'
  },
  caption: "List of matching Investigations",
  editurl:"index.html"
});
```

```
// Pager section
$("#data_grid").jqGrid('navGrid','#table_pager',{
  edit:false,
  add:false,
  del:false,
  search:false
});

// Button section
$("#hssn").click( function() {
  jQuery("#data_grid").jqGrid('hideCol',["SSN"]);
  });
$("#sssn").click( function() {
  jQuery("#data_grid").jqGrid('showCol',["SSN"]);
  });
$("#hlc").click( function() {
  jQuery("#data_grid").jqGrid('hideCol',["Code"]);
  });
$("#slc").click( function() {
  jQuery("#data_grid").jqGrid('showCol',["Code"]);
  });
```

The HTML code looks like:

```
<table id="data_grid"></table>
<div id="table_pager" ></div>
<div style="padding: 10px 0px;">
  <button id="hssn">Hide SSN</button>
  <button id="sssn">Show SSN</button>
  <button id="hlc">Hide Location Code</button>
  <button id="slc">Show Location Code</button>
</div>
```

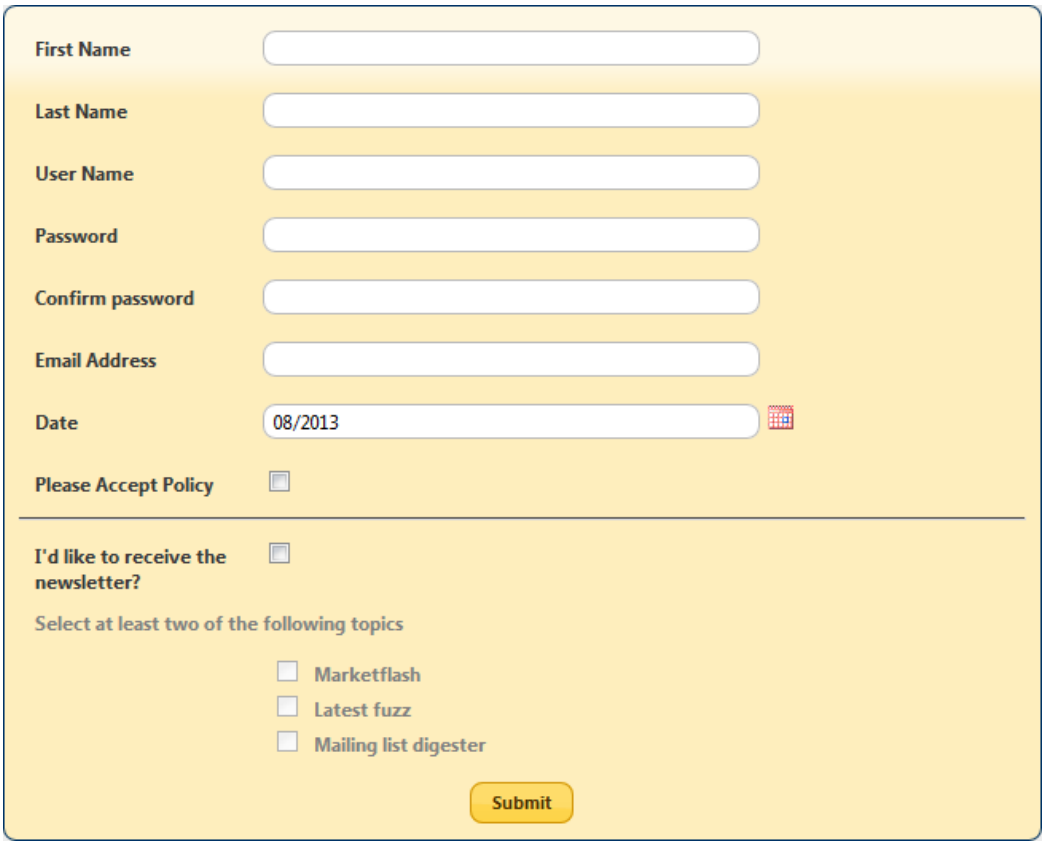
The XML code for one row looks like (with one row only):

```
<Investigations>
  <Items>
    <Request>
      <IsValid>True</IsValid>
      <ItemSearchRequest>
        <SearchIndex>Investigations</SearchIndex>
      </ItemSearchRequest>
    </Request>
    <Item>
      <DetailPageURL>javascript:showResult('01');</DetailPageURL>
      <ItemAttributes>
        <View>
          <a class="underline" href="javascript:alert('01');">View</a>
        </View>
        <Application>N/A</Application>
        <Release>N/A</Release>
        <Status>In Review</Status>
        <Result>-</Result>
        <Account>JDS HR Plus Test</Account>
        <Name>Jose, Jiby</Name>
        <SSN>123-45-6789</SSN>
        <Code>-</Code>
        <Description>-</Description>
        <OrderBy>Admin Account</OrderBy>
        <OrderDate>06/14/2010</OrderDate>
        <CompleteDate>-</CompleteDate>
        <Investigation>01</Investigation>
      </ItemAttributes>
    </Item>
  </Items>
</Investigations>
```

Form Validation Plug-in

It is possible to standardize the form validation throughout the web site. This should be in regards to functionality as well as design. Again we can use a jQuery plug-in that makes it easy to make efficient scripts on a form page.

Below is an example of a form page:



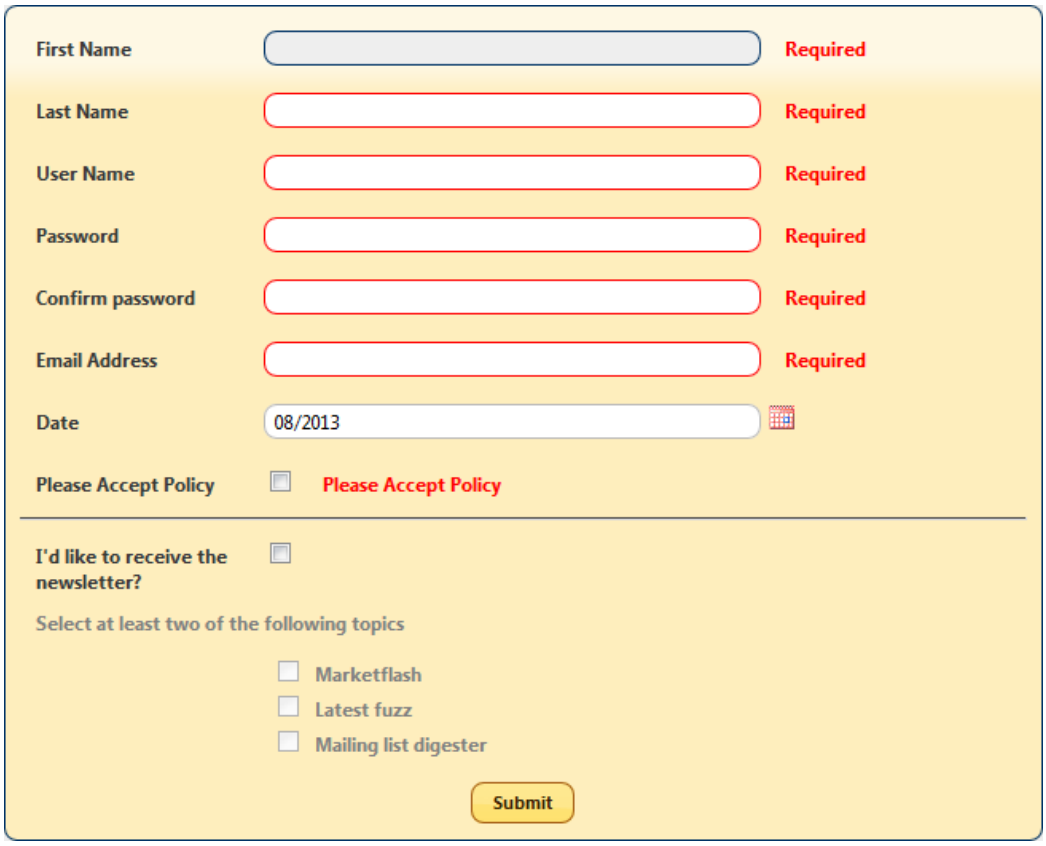
The validation script is as simple as this:

```
$().ready(function() {
  // Validate form
  $("#signupForm").validate({
    rules: {
      firstname: { required: true, minlength: 5, maxlength: 10 },
      lastname: { required: true, minlength: 5 },
      username: { required: true, minlength: 8 },
      password: { required: true, minlength: 5 },
      confirm_password: { required: true,minlength: 5,equalTo: "#password" },
      email: { required: true, email: true },
      date_picker: { required: true },
      agree: "required"
      topic: { required: "#newsletter:checked", minlength: 2 },
    },
```

```
    messages: {
      firstname: { required: "Required Field",
        minlength: "Minimum 5 Characters",
        maxlength: "Maximum 10 Characters" },
      lastname: { required: "Required Field",
        minlength: "Minimum 5 Characters" },
      username: { required: "Required Field",
        minlength: "Minimum 8 Characters" },
      password: { required: "Required Field",
        minlength: "Minimum 5 characters" },
      confirm_password: { required: "Required Field",
        minlength: "Minimum 5 characters",
        equalTo: "Same Password as Above!" },
      email: { required: "Required Field",
        email: "Invalid Email" },
      date_picker: { required: "Required Field" },
      agree: "Please Accept Policy"
    }
  });

  // Propose username by combining first- and lastname
  $("#username").focus(function() {
    var firstname = $("#firstname").val();
    var lastname = $("#lastname").val();
    if(firstname && lastname && !this.value) {
      this.value = firstname + "_" + lastname;
    }
  });

  // Code to hide topic selection
  var newsletter = $("#newsletter");
  // Newsletter topics are optional, hide at first
  var inital = newsletter.is(":checked");
  var topics = $("#newsletter_topics")
    [inital ? "removeClass" : "addClass"]("gray");
  var topicInputs = topics.find("input").attr("disabled", !inital);
  // Show when newsletter is checked
  newsletter.click(function() {
    topics[this.checked ? "removeClass" : "addClass"]("gray");
    topicInputs.attr("disabled", !this.checked);
  });
});
```

There is a rules section, and a message section. If no messages are defined, the default messages for the rule is used (as defined in the JavaScript.

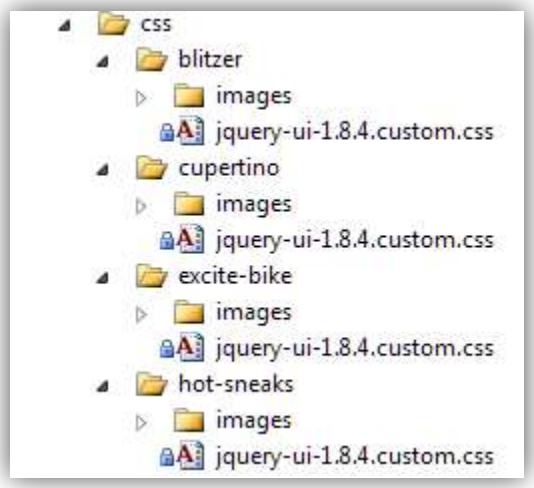
From the information above we can see that all fields are required. If the Submit button is clicked we see the following screen:



At this point every field will be validate while the user enters data. As an extra feature on this page, the user can click the 'I'd like to receive the nrewletter?' check box, after which the user has to select at least 2 topics. All this is done dynamically on the client side.

Change and Save Selected Theme

When retrieving code from the jQuery web site, you will be asked to select the options that you would like to include, as well as which theme you would like. When downloading the code, a customized zip file is created with all necessary code (JavaScripts, style sheets and images). The structure of the themes looks like this (shown for 4 different themes):



In order to implement a theme selector on the web site, we need the following:

- Theme selector
- A way to save the selected theme
- A way to update the theme

We need a named `<link>` to be available for update (should be included in the master page):

```
<link href="css/overcast/jquery-ui-1.8.4.custom.css" rel="stylesheet"
   type="text/css" id="theme" />
```

The HTML code for theme switcher page can be a dropdown menu, radio buttons or regular buttons. An example of buttons is shown below:

```
<button type="button" id="blitzer">Blitzer</button>
<button type="button" id="cupertino">Cupertino</button>
<button type="button" id="excite-bike">Excite-Bike</button>
<button type="button" id="hot-sneaks">Hot Sneaks</button>
```

JavaScript code for theme switcher page:

```
$(function () {
  $("#blitzer").click(function () { switchTheme("blitzer") });
  $("#cupertino").click(function () { switchTheme("cupertino") });
  $("#excite-bike").click(function () { switchTheme("excite-bike") });
  $("#hot-sneaks").click(function () { switchTheme("hot-sneaks") });

  // Set the theme variable based on cookie
  var theme = readCookie('theme');
  if (!theme) { theme = 'overcast'; }
});
```

JavaScript code to retrieve and set the theme (should be included in the master page):

```
function switchTheme(theme) {
  $("link#theme").attr("href", "css/" + theme +
    "/jquery-ui-1.8.4.custom.css");
  createCookie("theme", theme, 365);
}

var c = readCookie('theme');
if (c)
  switchTheme(c);
else
  switchTheme('overcast');
```

A few themes:

Blitzer

Cupertino

Excite-Bike

UI-Darkness

Themed Buttons

jQuery also comes with themed buttons. The way it works, is that you can add icons to a button using predefined graphics. This is all setup in the style sheet, and is very easy to use.

The jQuery code looks like this:

```
$('#iconLeft').button({ icons: { primary: 'ui-icon-circle-triangle-e'} });
$('#iconRight').button({ icons: {secondary: 'ui-icon-circle-triangle-w'} });
$('#iconBoth').button({ icons: { primary: 'ui-icon-circle-triangle-e',
                                 secondary: 'ui-icon-circle-triangle-w'} });
```

The hover-over colors and icons are all set automatically.

The individual icons are all part of one big image file, and each defined class is a 16 x 16 pixel snippet of the file. Again all this is taken care of by the loaded style sheet. The file shown below (ui-icons_70b2e1_256x240.png) is the one used for the icons shown above.

These are the icons that we have used:

```
.ui-icon-circle-triangle-e { background-position: -48px -192px; }
.ui-icon-circle-triangle-s { background-position: -64px -192px; }
.ui-icon-circle-triangle-w { background-position: -80px -192px; }
.ui-icon-circle-triangle-n { background-position: -96px -192px; }
```

This approach is very efficient, since one image is cached, and each icon is just a 16 x 16 section of the file.

The way it works is that the buttons widget converts the code from:

```
<button id="iconLeft" type="button">Left Icon</button>
```

To something like:

```
<button type="button" id="iconLeft"
  class="ui-button ui-widget ui-state-default ui-corner-all
  ui-button-text-icon-primary" role="button" aria-disabled="false">
  <span class="ui-button-icon-primary ui-icon
    ui-icon-circle-triangle-e"></span>
  <span class="ui-button-text">Left Icon</span>
</button>
```

Commonly used jQuery Plugins

It is easy to add jQuery plugins to a web page.

Below is a list of some of the plugins that we are using:

jquery.alphanumeric.js

Checks if an alphanumeric key has been pressed. If not, the entered character is ignored.

jquery.autocomplete.js

This is used to autocomplete entered information. Mainly used to select City, State and Country information. The matching data is retrieved from the database when at least 3 characters have been entered for the city.

jquery.cluetip.js

This plugin is used to popup a clue tip when the mouse is hovering over an item (such as a service). The information displayed is coming from the language file.

jquery.dynamicDropDown.js and jq_selectlist.js

These two jQuery script have pretty much the same functionality. We can use these to dynamically populate a drop down box based on a selection.

jquery.hrplus.validate.js

This jQuery plugin is used for validating input fields in a form it has been slightly modified to our needs. Most of the used validation scripts are created dynamically based on the settings for a particular page or partial view (mostly for the order page). Below is a list of various validation methods:

Method	Description
minlength	Minimum number of characters required
maxlength	Maximum number of characters required
rangelength	Value range required
min	Minimum value required
max	Maximum value required
range	Value range required
email	Makes the element require a valid email
url	Makes the element require a valid url
dateISO	Makes the element require an ISO date
number	Makes the element require a decimal number
digits	Makes the element require digits only
creditcard	Makes the element require a credit card number
accept	Makes a file upload accept only specified mime-types
equalTo	Requires the element to be the same as another one

jquery.jloupe.js or jquery.magnifier.js

This jQuery plugin will show you a magnified section that can be seen when the mouse hovers over an image (used in the widget selection screen)

jquery.maskedinput-1.2.2.js

This plugin allows us to add a mask to an input field, such as phone number as well as SSN formats.

jquery.signaturepad.js

This is used for the Signature box included in the Conformation and Disclosure pages allowing us to use jQuery to capture the information. This is using the canvas based signature pad in HTML5. The data is saved as JSON and then converted to a image.

jquery.uniform.js

This can be used to generate uniformly defined buttons, input fields and drop down boxes so they all look the same in various browsers.

HR Plus jQuery Examples

Meet the IT Team Widget on the test site

Html Code:

```
<div class="widget-block center team">
  <img src="img/HRPlus/IMG_2743.JPG" title="Beth (x376)" />
  <img src="img/HRPlus/IMG_4029.JPG" title="Charles (x306)" />
  <img src="img/HRPlus/IMG_4030.JPG" title="Database Bob (x619)" />
  <img src="img/HRPlus/IMG_5707.JPG" title="Frank (x313)" />
  <img src="img/HRPlus/IMG_4035.JPG" title="Harvey" />
  <img src="img/HRPlus/IMG_4031.JPG" title="Jiby (x374)" />
  <img src="img/HRPlus/IMG_4034.JPG" title="Johannes (x322)" /><br />
  <img src="img/HRPlus/IMG_4028.JPG" title="Joseph (x312)" />
  <img src="img/HRPlus/IMG_5706.JPG" title="Kudos Kris (x320)" />
  <img src="img/HRPlus/IMG_4020.JPG" title="Margaret (x627)" />
  <img src="img/HRPlus/IMG_5702.JPG" title="Mike" /><br />
  <img src="img/HRPlus/IMG_5704.JPG" title="Pierre (x631)" />
  <img src="img/HRPlus/IMG_5703.JPG" title="Venki (x389)" />
  <img src="img/HRPlus/IMG_5708.JPG" title="Zach (x310)" />
</div>
```

jQuery code:

```
$(function () {
  // Add CSS classes to the thumbnail images
  $('div.team img').addClass('ui-corner-all ui-widget-content');

  // Do this if any thumbnail image is clicked
  $('div.team img').click(function() {
    // Set html code for the team_member dialog
    $('#team_member').html('<img class="team-member ui-corner-all
    ui-widget-content" src="' + $(this).attr('src') + '" />');
    // Set the title for the team_member dialog
    $('#team_member').dialog({title: $(this).attr('title')});
    // Open the team_member dialog
    $('#team_member').dialog('open');
  });

  // Define settings for the team_member dialog
  $('#team_member').dialog({
    autoOpen: false,
    modal: false,
    hide: {effect: "blind"},
    show: {effect: "blind"},
    resizable: false,
    height: 470,
    width: 340
  });
});
```

CSS code:

```
.team img { height: 100px; width: 80px; margin: 1px; }
.team-member { width: 300px; height: 400px; margin-top: 5px; }
```

Result:

Update Custom Setting on the Web Store

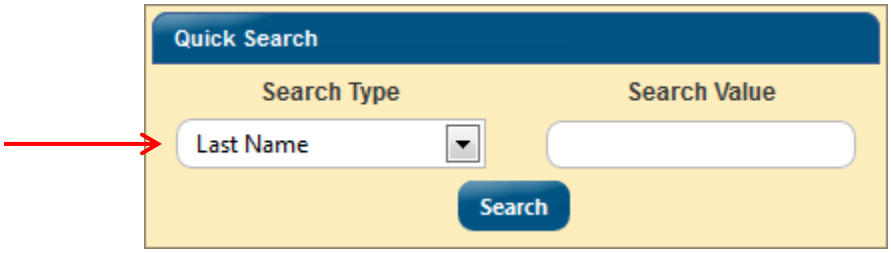
The Web Store has a number of settings that are saved in the database, as well as in cookies. These settings are automatically updated when the user changes certain settings. Below is a list of the various settings:

Key	Description
selected_search_criteria	Value of search dropdown in My Workplace widget as well as results page
selected_search_type	Value of search type in My Workplace widget as well as the results page
selected_scope	Status of 'Scope of Investigation Retrieval' accordion (show or hide)
selected_order_templates	Comma delimited List of order templates for used in the 'Start New Order' widget
selected_order_type	Candidate or Client order used in the 'Start New Order' widget
Language *)	Site language
Theme *)	Site theme
date_range	Performance Graph - date range
graph_type	Performance Graph - graph type
report_type	Performance Graph - report type
show_help	Show or collapse help sections
adv_use_calendar *)	Use calendar in advanced search
adv_start_date *)	Start date in advanced search
adv_end_date *)	End date in advanced search
adv_date_range *)	Date range for advanced search
investigation_page_size	Page size for investigation grid

*) Currently not used.

How does it work?

All updates are driven by change events. As an example, when the user changes the 'Search Type' selection in the 'My Workplace' widget:



Then a change event is triggered:

```
$('#quickSearchCriteria').change(function () {
  updateCustomSetting('selected_search_type',
  $('#quickSearchCriteria').val(), '0', '-1')
});
```

This event will get the value of the selected 'Search Type', which in this case is 'LastName', and call the JavaScript 'updateCustomSetting()' with the value retrieved.

'updateCustomSetting()' can be found in 'jq_misc.js' (which contains a number of handy and often used JavaScript functions):

updateCustomSetting (jq_misc.js)

```
// ---------------------------------------------------------------------------
// Update custom setting
// ---------------------------------------------------------------------------
function updateCustomSetting(Key, Value, Type, OrgId) {
  createCookie(Key, Value, 365);
  var formData = "Key=" + Key.replace(" ", "+") + "&Value=" +
  Value.replace(" ", "+") + "&Type=" + Type + "&OrgId=" + OrgId;
  $.ajax({
    url: '/widget/saveCustomSetting',
    type: 'POST',
    dataType: 'html',
    data: formData,
    contentType: 'application/x-www-form-urlencoded',
    success: function (result) { },
    error: function (result) { }
  });
}
```

This function will create a cookie as well executing an AJAX call passing a number of parameters, such as Key, Value, Type and OrgId.

saveCustomSetting (WidgetController)

```
///**********************************************************************
/// <summary>
/// saveCustomSetting
/// </summary>
/// <param name="FormInfo"></param>
/// <returns></returns>
///**********************************************************************
[HttpPost]
[ExcludeAntiForgeryToken]
public ActionResult saveCustomSetting(FormCollection FormInfo)
{
  CustomSettingEditModel CustomSetting = new CustomSettingEditModel();
  TryUpdateModel(CustomSetting, FormInfo.ToValueProvider());
  CustomSetting.OrgUserId = Convert.ToInt32(Session[SessionKey.OrgUserId]);
  if (CustomSetting.Misc == null)
    CustomSetting.Misc = "";

  if (CustomSetting.OrgId != -1)
    CustomSetting.OrgId = Convert.ToInt32(Session[SessionKey.OrgId]);
  else
    CustomSetting.OrgId = 0;

int Result = _widgetService.CustomValueUpsert(CustomSetting.Key,
  CustomSetting.OrgUserId, CustomSetting.OrgId, 0, "",
  CustomSetting.Value).resultData;

  return Json(Result);
}
```

If the OrgId is not '-1', then use the OrgId, else use '0'. Use the stored procedure 'CustomValueUpsert()' to update / insert a record in the 'CustomValue' table.

Including Text from the Language file in JavaScript

We have tried to design the web site to allow multiple languages. Currently only one language is used, but this could easily be extended to other languages (if everything could be translated properly). The selected language can also be saved in the 'CustomValue' table mentioned above.

However how do we include text from the language file into JavaScript? JavaScripts that are included on a page (as an external file), will not be able to use anything but hardcoded text since these files are static in nature.

There are a few ways to solve this problem.

1) Create an AJAX call to a controller, where we will pass the name that matches the name in the language file. The AJAX call will then return the actual text to be displayed. This has not been implemented.
2) We decided to embed the JavaScript in the ASPX or ASCX pages, and then use the following code to retrieve the text:

   ```
   <%= Language.ResourceManager.GetString("Name") %>
   ```

   At run time the code will be replaced with the appropriate text before it is sent to the client side browser. We can also use the text to validate text and code.

   A disadvantage is that we do not have the JavaScript included as an external file, and that will make page loading slower (JavaScripts are in general cached in the browser).

This brings back the issue that was discussed earlier – that the best optimization for a web page is to include JavaScripts and CSS files as external files.

Dynamic JavaScript

Using the same principle as above, we can actually create dynamic JavaScript based on selections as well as various values in the model that is passed to the view. Here is a typical example where we use Model information, enum definitions as well as language information:

```
$("select#quickSearchCriteria option:contains('<%=Language.ResourceManager.GetString("MyStatus_Search_Type_8")%>')").addClass('dto');
$("select#quickSearchCriteria option:contains('<%=Language.ResourceManager.GetString("MyStatus_Search_Type_10")%>')").addClass('dto');
$("select#quickSearchCriteria option:contains('<%=Language.ResourceManager.GetString("MyStatus_Search_Type_11")%>')").addClass('dto');
$("select#quickSearchCriteria option:contains('<%=Language.ResourceManager.GetString("MyStatus_Search_Type_12")%>')").addClass('dto');
$("select#quickSearchCriteria option:contains('<%=Language.ResourceManager.GetString("MyStatus_Search_Type_14")%>')").addClass('dto');
$("select#quickSearchCriteria option:contains('<%=Language.ResourceManager.GetString("MyStatus_Search_Type_13")%>')").hide();
$("button:contains('<%=Language.ResourceManager.GetString("MyStatus_Search_Type_13")%>')").addClass('dto');

<% if (Model.AvailableServiceLineType == Convert.ToInt32(DTServiceLineType.Investigation_Only)) { %>
  $('select#quickSearchCriteria option.dto, tr.dto, td.dto, button.dto').hide().attr('disabled', 'disabled');
<% } else if (Model.AvailableServiceLineType == Convert.ToInt32(DTServiceLineType.Investigation_And_Drug_Test)) { %>
  $('select#quickSearchCriteria option.dto, button.dto span').append(' (<%=Language.ResourceManager.GetString("Advanced_Search_Type_2") %>)');
  $('select#quickSearchCriteria').val('');
<% } %>
```

Could eventually be rendered like:

```
$("select#quickSearchCriteria option:contains('Tracking Info')").addClass('dto');
$("select#quickSearchCriteria option:contains('Result')").addClass('dto');
$("select#quickSearchCriteria option:contains('Status')").addClass('dto');
$("select#quickSearchCriteria option:contains('Ordered By')").addClass('dto');
$("select#quickSearchCriteria option:contains('All Incomplete')").hide();
$("button:contains('All Incomplete')").addClass('dto');

$('select#quickSearchCriteria option.dto, button.dto span').append(' (Drug Test Only)');
$('select#quickSearchCriteria').val('');
```

JavaScript Versioning

One problem we have had in the past, is JavaScript (and CSS) being cached by the browser. If a change was made to one of the external JavaScripts, it would not be reloaded unless the page was refreshed (by pressing [F5] or clicking the Reload button in the browser).

This issue can be resolved by adding a version to the JavaScript (or CSS) that needs to be reloaded the first time a user logs in after a deployment.

Here is an example:

```
<script type="text/javascript"
  src="<%= Url.Content ("~/Scripts/jq_selectlist.js?v1.0") %>">
</script>
```

The version added will fool the browser into thinking that this is a new file, and will reload it the first time.