Hands-On Lab

What's New in ASP.NET and Web Development in Visual Studio 11

Lab version: 1.2.0

Last updated: 3/9/2012

* + 1. 

Contents

[Overview 3](#_Toc319078024)

[Exercise 1: What's New in the CSS Editor 4](#_Toc319078025)

[Task 1 – New Editor Features 4](#_Toc319078026)

[Task 2 – The Color Picker 7](#_Toc319078027)

[Task 3 – CSS-Compatible Code Snippets 10](#_Toc319078028)

[Exercise 2: What’s New in the HTML Editor 14](#_Toc319078029)

[Task 1 – Improved DOCTYPE Validation 14](#_Toc319078030)

[Task 2 – Start/End Tags Automatic Update 16](#_Toc319078031)

[Task 3 – New HTML5 Code Snippets 17](#_Toc319078032)

[Task 4 – HTML Editor Smart Tags 19](#_Toc319078033)

[Task 5 – Smart Indentation 19](#_Toc319078034)

[Task 6 – Extract to User Control 21](#_Toc319078035)

[Exercise 3: What's New in the JavaScript Editor 24](#_Toc319078036)

[Task 1 – JavaScript Editor New Features 24](#_Toc319078037)

[Task 2 – Validating JavaScript 26](#_Toc319078038)

[Task 3 – XML Documentation for JavaScript 28](#_Toc319078039)

[Exercise 4: Bundling and Minification 30](#_Toc319078040)

[Task 1 – Installing the Bundling and Minification Package from NuGet 31](#_Toc319078041)

[Task 2 – Default Bundles 32](#_Toc319078042)

[Task 3 – Static Bundles 37](#_Toc319078043)

[Task 4 – Dynamic Folder Bundles 39](#_Toc319078044)

1. To give feedback please write to [VSKitFdbk@Microsoft.com](mailto:VSKitFdbk@Microsoft.com).
2. Copyright © 2012 by Microsoft Corporation. All rights reserved.

Overview

* 1. The new version of Visual Studio introduces a number of enhancements focused on improving the experience and performance when working with Web technologies. Visual Studio Editors for CSS, JavaScript and HTML have been completely revamped to include many of the most in-demand code aids, such as IntelliSense and automatic indentation. Regarding performance, bundling and minification are now integrated as built-in features to easily reduce page load time.

1. Visual Studio enables you to work with the latest website technologies. You can use cross-browser CSS3 Snippets to make sure your site works regardless of the client platform while also taking advantage of the new HTML5 elements and features.
2. Writing and profiling JavaScript code should be easier with this Visual Studio version. IntelliSense lists, integrated XML documentation and navigation features are now available for JavaScript code. You now have the JavaScript catalog at your fingertips. Additionally, you can check ECMAScript5 compliance with your scripts and detect syntax errors at an early stage.

Last but not least, this Visual Studio version implements built-in bundling and minification. Your script files and style sheets will be packed and compressed so that the site performs faster.

1. This lab walks you through the enhancements and new features previously described by applying minor changes to a sample Web application provided in the Source folder.

# Objectives

* 1. In this hands-on lab, you will learn how to:
  + Use the new features and improvements in the CSS editor
  + Use the new features and improvements in the HTML editor
  + Use the new features and improvements in the JavaScript editor
  + Configure and use bundling and minification

# Prerequisites

* + [Microsoft Visual Studio 11 Ultimate Beta](http://go.microsoft.com/fwlink/?LinkId=240160)
  + [Windows PowerShell](http://support.microsoft.com/kb/968930/) (for setup scripts – already installed on Windows 7 and Windows Server 2008 R2)
  + [Internet Explorer 9](http://windows.microsoft.com/en-US/internet-explorer/products/ie/home) – or an HTML5 compliant browser

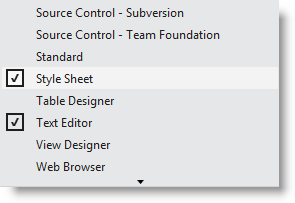
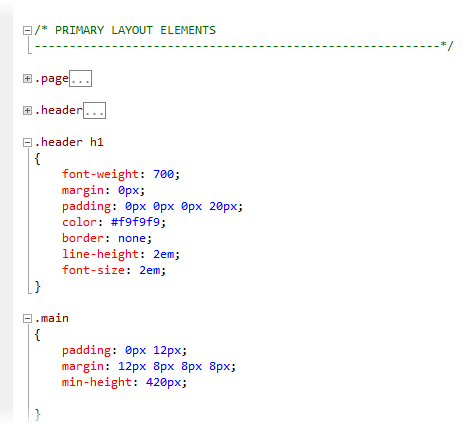
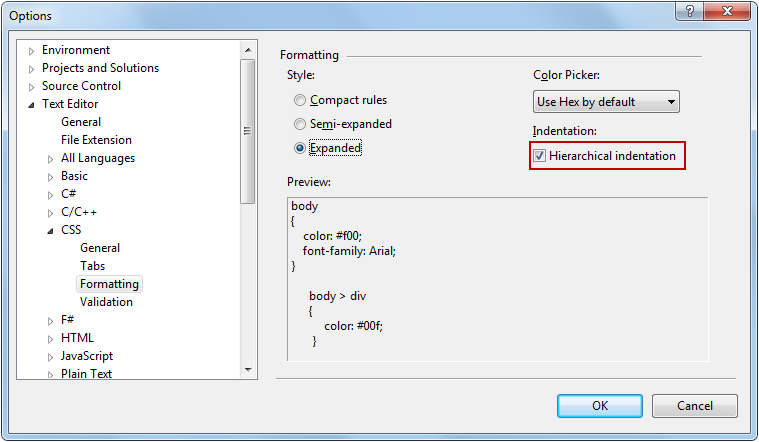
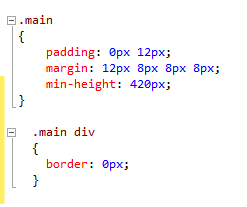
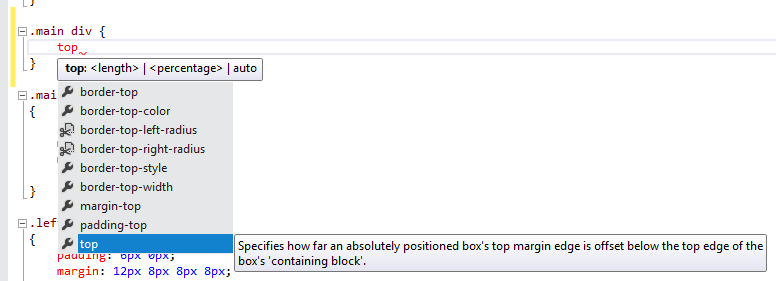
# Exercises

* 1. This hands-on lab includes the following exercises:
  2. What’s New in the CSS Editor
  3. What’s New in the HTML Editor
  4. What’s New in the JavaScript Editor
  5. Bundling and Minification
  6. Estimated time to complete this lab: **60 minutes**.

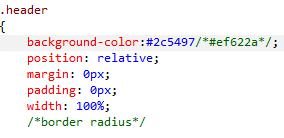
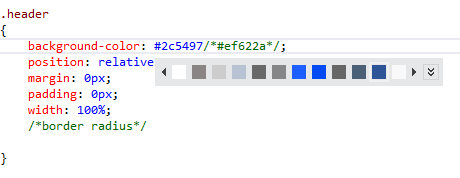
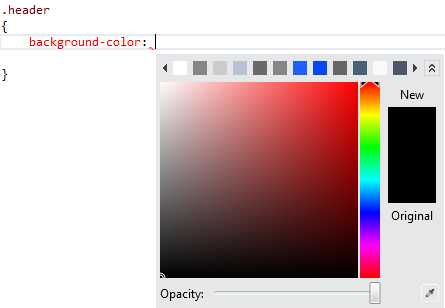
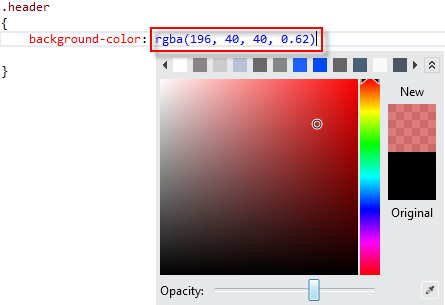
Exercise 1: What's New in the CSS Editor

* 1. Web developers should be familiar with many of the difficulties that related to CSS editing. One of the biggest issues of CSS styling is cross-browser compatibility. It often happens that, after applying styles to your site, you notice that it looks different if you open it in another browser or device. Therefore, you may spend a considerable time on fixing those visual issues to realize that, when you finally make it work in one browser, it is broken in the others.
  2. Visual Studio now includes features that help developers access, work and organize CSS style sheets effectively. Throughout this exercise, you will meet the new features for an effective organization and edition, as well as the CSS3 Code Snippets for cross-browser compatibility.

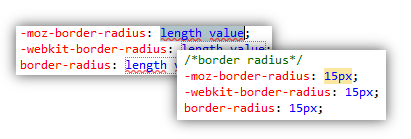
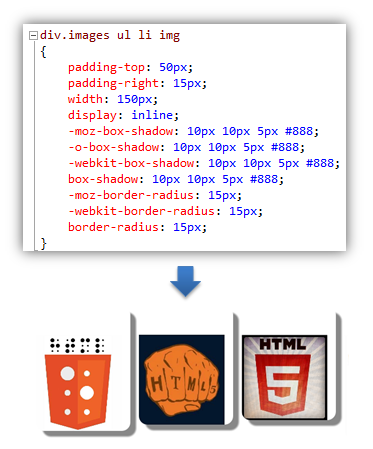
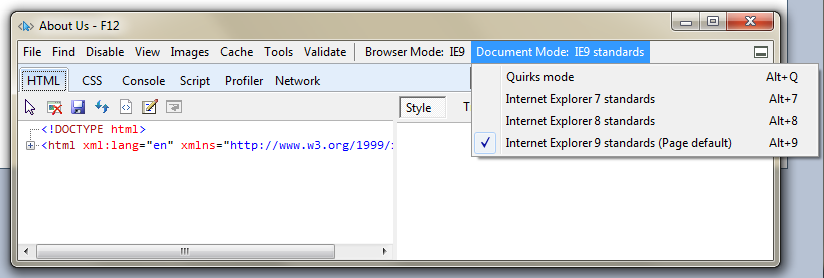
Task 1 – New Editor Features

* 1. In this task, you will discover the new features of the CSS Editor. This new editor will help you increase your productivity by taking advantage of the new smart indentation, the improved code comments and the enhanced IntelliSense list.
  2. Start **Visual Studio** and open the **WhatsNewASPNET.sln** solution located in the **Source\WhatsNewASPNET** folder of this lab.
  3. In Solution Explorer, open the **Site.css** file located under the **Styles** folder. Make sure the **Text Editor** tools are visible on the toolbar. To do that, select the **View** |**Toolbars** menu option, and check the **Text Editor** options. You will notice that, since this new version, the **Comment** button () and the **Uncomment** button () are also enabled for the CSS editor.
     1. 
     2. Figure 1
     3. Enabling Editor and CCS Tools
  4. Scroll the code and select any CSS class definition. Click the **Comment (****)** button to comment the selected lines. Then, click the **Uncomment** **(****)** button to undo the changes.
  5. Click the **Collapse** () and **Expand** () buttons located on the left margin of the text. Notice that you can now hide the styles you don’t use to have a cleaner view.
     1. 
     2. Figure 2
     3. Collapsing CSS classes
  6. Make sure that the smart indentation feature is enabled. Select the **Tools** | **Options** menu option, and then select the **Text Editor** | **CSS** | **Formatting** page in the left pane of the screen. Check the **Hierarchical indentation** option.
     1. 
     2. Figure 3
     3. Enabling hierarchical indentation
  7. Locate the main class definition (.main) and append a style to the div elements. You will notice that the code aligns automatically, helping users to find the parent classes at a glance.
     1. CSS
     2. .main
     3. {
     4. padding: 0px 12px;
     5. margin: 12px 8px 8px 8px;
     6. min-height: 420px;
     7. }
     8. **.main div**
     9. **{**
     10. **border: 0px;**
     11. **}**
     12. 
     13. Figure 4
     14. Hierarchical alignment in CSS
  8. Inside**.main div** class, locate the cursor at the end of **border: 0px;** and press **Enter** to display the IntelliSense list. Start typing **top** and notice how the list is filtered as you type. The list will display the elements that contain **top** at any part of the word (In prior versions of Visual Studio, the list is filtered by the items that *begin* with the term).
     1. 
     2. Figure 5
     3. IntelliSense enhancements in CSS

Task 2 – The Color Picker

* 1. In this task, you will discover the new CSS Color Picker integrated into Visual Studio IntelliSense.
  2. In **Site.css,** locate the header class definition (.header) and place the cursor next to **background-color** attribute, between the “:” and “#” characters on that line of code**.**
     1. 
     2. Figure 6
     3. Locating the cursor
  3. Press the **spacebar** to display the color picker. Notice that the first colors you will see are the most frequent colors of your site. If you click the white color, its HTML color code (#fff) will replace the current color code in the stylesheet.
     1. 
     2. Figure 7
     3. Color picker
  4. Press the **Expand** () button on the color picker to display the color gradient, and then drag the gradient cursor to select a different color. After that, click the **Eyedropper** button and select any color from the screen. Notice that background color value changes dynamically while you move the cursor.
     1. 
     2. Figure 8
     3. Color picker gradient
  5. In the **Opacity** slider, move the selector to the center of the bar to reduce the opacity. Notice that background-color value now changes its scale to RGBA.
     1. 
     2. Figure 9
     3. Color picker Opacity
     4. **Note:** The RGBA (Red, Green, Blue, Alpha) color definition in CSS3 enables you to define the color opacity value for a single item. Unlike **opacity –** a similar CSS attribute **–** RGBA colors are also compatible with the latest browsers.

Task 3 – CSS-Compatible Code Snippets

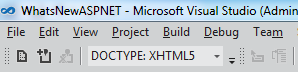
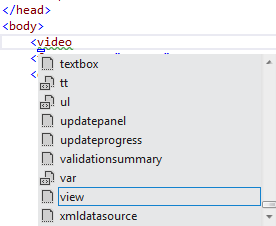
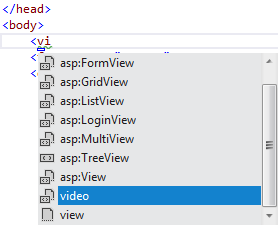
* 1. In this task, you will learn how to use cross-browser compatible CSS3 snippets in order to implement some features in your website.
  2. In the **Site.css** file, locate the **header** CSS class definition (.header) and place the cursor below the **/\*border radius\*/** placeholder to add a new snippet. Press **Enter** to display the IntelliSense list and type **radius** to filter the list. Select the **border-radius** option from the list with a single-click, and then press the TAB key twice to insert the snippet. Then, type a radius size in pixels and press **enter**. For instance, type **15px.**
     1. The CSS3 attributes added by the snippet will render rounded borders in most HTML5 compliance browsers, including Mozilla and WebKit-based browsers.
     2. 
     3. Figure 10
     4. Using a border-radius snippet
  3. Apply the same **border** snippets in the page style (.page).
     1. CSS
     2. .page
     3. {
     4. width: 960px;
     5. background-color: #fff;
     6. margin: 20px auto 0px auto;
     7. border: 1px solid #496077;
     8. /\*border radius\*/
     9. **-moz-border-radius: 15px;**
     10. **-webkit-border-radius: 15px;**
     11. **border-radius: 15px;**
     12. }
  4. Press **F5** to run the solution. Notice that each page now has rounded borders.
     1. 
     2. Figure 11
     3. Rounded corners
  5. Close the browser and return to Visual Studio.
  6. Open the **Custom.css** file located under the **Styles** folder and place the cursor inside **div.images ul li img** class definition.
  7. Press enter to display the IntelliSense list, type **box-shadow** and press the TAB key twice to insert the default shadow code snippet inside the class definition. Then, type **border-radius** and insert the code snippet. Type **15px** to set radius size and press **ENTER**.
     1. 
     2. Figure 12
     3. Rounded corners with shadow
     4. **Note:** At this moment, the shadow attribute is inserted with the corresponding prefix (moz, webkit, o) to support Mozilla and Webkit (Chrome, Safari, Konkeror) browsers.
  8. Create a new class **div.images ul li img:hover** below the **div.images ul li img** class definition and place the cursor inside the brackets**.**
     1. CSS
     2. div.images ul li img
     3. {
     4. ...
     5. }
     6. **div.images ul li img:hover**
     7. **{**
     9. **}**
  9. Type **transform** and press the TABkey twice in order to insert the transform snippet. Then, enter **15** to change the rotation angle value when images are hovered.
     1. CSS
     2. div.images ul li img
     3. {
     4. padding-top: 50px;
     5. padding-right: 15px;
     6. width: 150px;
     7. ...
     8. }
     9. div.images ul li img:hover
     10. {
     11. **-ms-transform: rotate(-15deg);**
     12. **-moz-transform: rotate(-15deg);**
     13. **-o-transform: rotate(-15deg);**
     14. **-webkit-transform: rotate(-15deg);**
     15. **transform: rotate(-15deg);**
     16. }
  10. Press **F5** to run the solution and browse to the CSS3 page. Notice that the images have rounded corners and box shadows. Hover the mouse over the images and watch them rotate.
      1. 
      2. Figure 13
      3. Transform snippet rotating an image
      4. **Note:** If you cannot see the shadows, make sure the document mode is set to IE9 standards. Press **F12** to open Internet Explorer developer tools and click **Document Mode** to change to IE9 standards.
      5. 

Exercise 2: What’s New in the HTML Editor

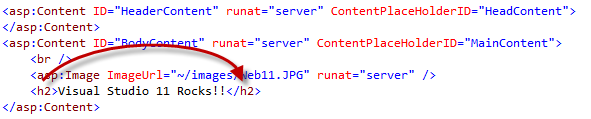
Visual Studio has an improved HTML editor. Some of the enhancements included in this version are smart indentation in HTML documents, HTML5 snippets, HTML start and end tag matching, and HTML validation. Throughout this exercise, you will see how these changes improve your fluency when working in the website markup.

Like the CSS editor, the HTML editor was also improved. Most of these improvements are small ones that make the Web developer’s life easier. Things like more code snippets for HTML5, smart indentation, matching start and end tags when editing and validation targeting the HTML document DOCTYPE are some of these improvements.

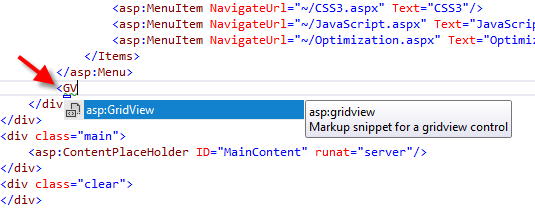
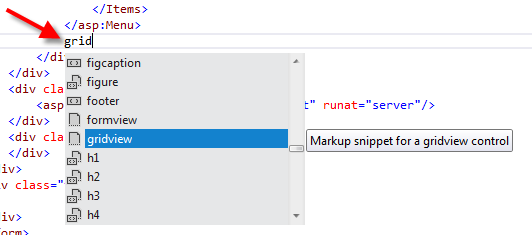
Task 1 – Improved DOCTYPE Validation

* 1. The HTML editor now has the ability to check the DOCTYPE of your page, even though the definition might be in the master page. Depending on the DOCTYPE of your page, the HTML editor will validate with the correct set of rules and will filter the IntelliSense list considering the DOCTYPE elements.
  2. In this task, you will change the DOCTYPE of a page to see how the HTML editor behavior changes accordingly.
  3. If not already opened, start **Visual Studio** and open the **WhatsNewASPNET.sln** solution located in the **Source\WhatsNewASPNET** folder of this lab.
  4. Open the **Site.Master** page.
  5. Notice the Target Schema for Validation Toolbar. The way the HTML editor behaves (Validation, IntelliSense, etc.) will properly change to fit the Doctype selected.
     1. 
     2. Figure 14
     3. Use Doctype in HTML Source Editing toolbar
  6. Place the cursor under the **body** element, and start typing the name of an HTML5 element (for example, **video**). Notice that the element is not available in the IntelliSense list.
     1. 
     2. Figure 15
     3. HTML5 elements not listed
  7. Undo the changes to the DOCTYPE definition, setting the value to HTML5.
     1. HTML
     2. <%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="Web11.SiteMaster" %>
     3. **<!DOCTYPE html>**
     4. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
     5. <head runat="server">
  8. Place the cursor under the **body** element and start typing an HTML5 element again (For example, like **video**). Notice that the HTML5 elements are now available in the IntelliSense list.
     1. 
     2. Figure 16
     3. HTML5 elements being listed

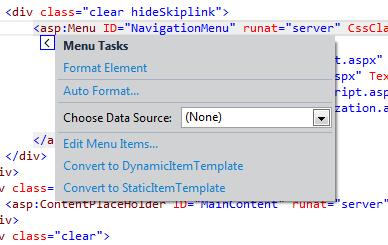
Task 2 – Start/End Tags Automatic Update

* 1. Visual Studio now updates the HTML opening or closing tags of the element that you are editing to match each other. This new feature will improve your productivity when editing HTML tags.
  2. On the **Default.aspx** page, add an **H3** element with a title (for example, Web11 Rocks!).
     1. HTML
     2. <asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
     3. <br />
     4. <asp:Image ImageUrl="~/images/Web11.JPG" runat="server" />
     5. **<h3>Visual Studio 11 Rocks!!</h3>**
     6. </asp:Content>
  3. Change the **H3** tag and type **H2** or **H1.**
     1. Notice that the end tag automatically updates. You can also modify the end tag to see that the start tag updates accordingly too.
     2. 
     3. Figure 17
     4. Automatic update of the end tag

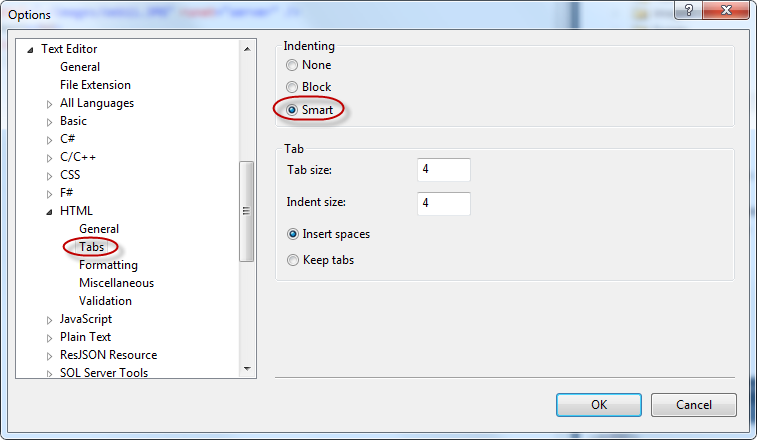
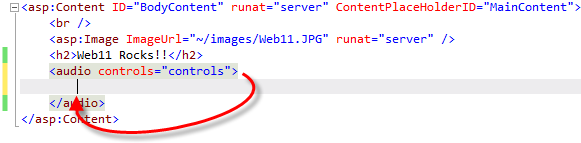
Task 3 – New HTML5 Code Snippets

* 1. Visual Studio now includes several HTML5code snippets. In this task, you will use some of these snippets.
  2. Add a new folder named **audio** to the root of the web site folder. Open Windows Explorer and copy any audio file into the **audio** folder of the WhatsNewASPNET solution.
  3. In the **Default.aspx** page, locate the cursor under the Web11 Rocks!! Header. Type **audio** and press the TAB key.
     1. The new HTML editor includes code snippets for HTML5 content. Remember to use the proper DOCTYPE definition to enable the HTML5 snippets.
     2. 
     3. Figure 18
     4. Inserting HTML5 Code Snippets
  4. Update the audio source to point to an existing audio file.
     1. HTML
     2. <asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
     3. <br />
     4. <asp:Image ImageUrl="~/images/Web11.JPG" runat="server" />
     5. <h2>Web11 Rocks!!</h2>
     6. <audio controls="controls">
     7. **<source src="audio/Kalimba.mp3" />**
     8. </audio>
     9. </asp:Content>
  5. Press **F5** to run the site and play the audio.
     1. 
     2. Figure 19
     3. Running the audio control
     4. **Note:** You can also try more snippets included in Visual Studio, such as video, figure, etc.
  6. Now, try to insert a control in some part of the page. For example, try to insert a **GridView** control, but instead of typing **Gri,** start typing **GV.** Notice that the IntelliSense list shows the **asp:GridView** control.
     1. IntelliSense in the HTML Editor now provides title-casing search, as well as partial matching (retrieving all elements that contains the term).
     2. 
     3. Figure 20
     4. Inserting a GridView with IntelliSense lists
     5. If you type **grid** you will get the all the items that match the term, but Visual Studio will suggest the **gridview** control:
     6. 
     7. Figure 21
     8. Inserting a GridView with IntelliSense lists and partial matching

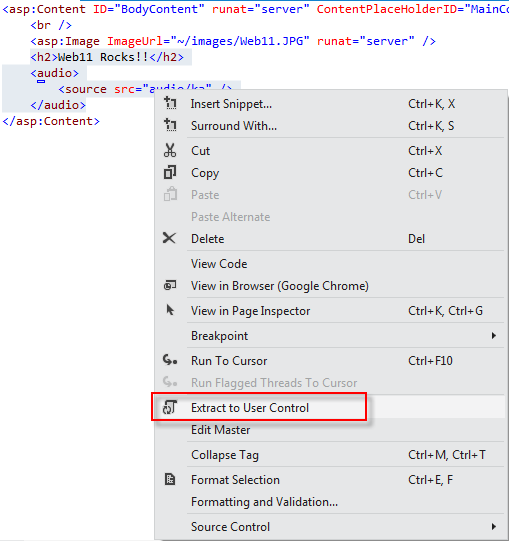
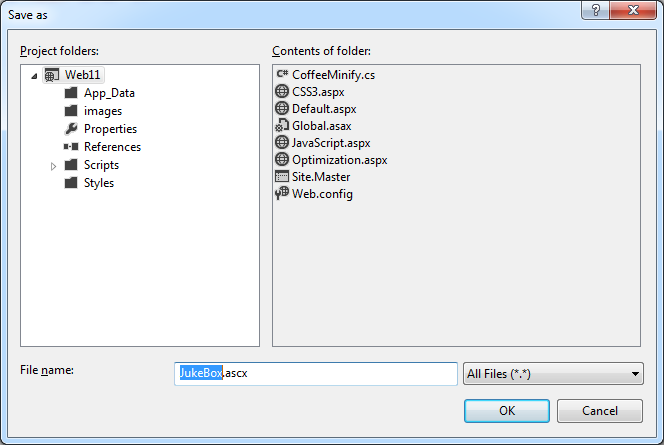
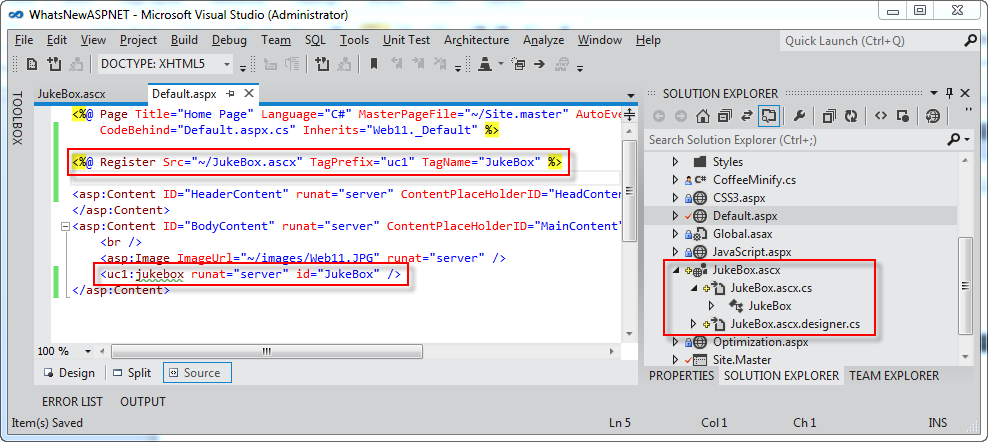
Task 4 – HTML Editor Smart Tags

* 1. Another improvement in the HTML Editor is the Smart Tags feature. Smart tags make it easy to perform common or repetitive development tasks on a per-control basis. This feature was already available in the HTML Designer, but not in the HTML Editor.
  2. Open **Site.Master** and locate the **asp:Menu** element. Place the cursor on the start tag and notice that the small glyph displayed at the bottom of the element – click it to open the smart tasks menu. Notice that you have quick access to some tasks related to the Menu control.
     1. 
     2. Figure 22
     3. Smart tasks for the Menu control

Task 5 – Smart Indentation

* 1. One of the best practices in HTML is indenting the nested elements to keep the code readable. In Visual Studio 11, you will notice that the editor automatically indents the elements while you are writing the code.
  2. **Note:** In previous version of Visual Studio, smart indentation was available in the XML editor but not in the HTML editor.
  3. Make sure that the Indenting configuration on the HTML Editor is set to Smart Indentation. To do that, select the **Tools | Options** menu option and then select the **Text Editor | HTML | Tabs** page in the left pane of the screen. Select the Smart indentation option.
     1. 
     2. Figure 23
     3. HTML Editor settings
  4. On the **Default.aspx** page, remove all the content under the audio element.
  5. Place the cursor at the end of the opening **audio** element and hit **ENTER**.
     1. Notice that the new position of cursor has an additional indentation level.
     2. 
     3. Figure 24
     4. Smart indentation in the HTML Editor
  6. Restore the audio tag with the content you have removed, or close Default.aspx without saving the changes.

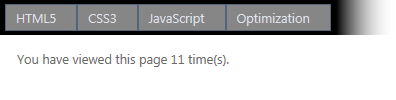
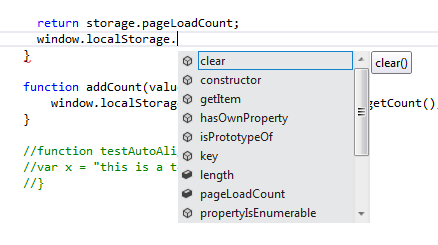
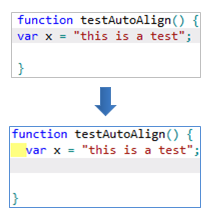
Task 6 – Extract to User Control

* 1. The Refactoring tools included in Visual Studio, such as extracting a portion of code to a function, are great features that facilitate the improvement and the refactoring the existing code. The counterpart for ASP.NET pages would be the extraction of HTML code to a User Control. Doing it manually would involve several steps, like creating a new User Control, moving the code section to the User Control, registering a tag prefix for the User Control, and, finally, instantiating the User Control on the pages. Now, the new *Extract to User Control* tool automatically performs all those steps for you.
  2. In this task, you will use the new Extract to User Control contextual operation to generate a new user control from the selected code.
  3. On the **Default.aspx** page, select the **H2** and **audio** elements.
  4. Right-click and select **Extract to User Control**.
     1. 
     2. Figure 25
     3. Extract to User Control menu option
  5. Type a name for the new user control. For instance, **Jukebox.ascx**, and then click **OK**.
     1. 
     2. Figure 26
     3. Saving the extracted user control
  6. Notice that the selected code was extracted to a user control and the original location of the selected code was replaced with an instance of the new user control.
     1. 
     2. Figure 27
     3. Page automatically updated to use the new user control
  7. Press **F5** to run the page and verify that the control works.

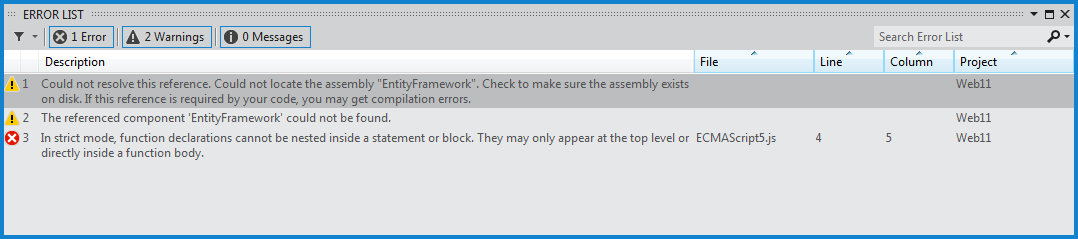
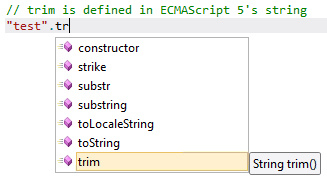
Exercise 3: What's New in the JavaScript Editor

* 1. Writing or editing JavaScript code is not an easy task, especially when your application starts to grow in size and you find yourself dealing with long files and hundreds of functions. Script developers usually have to do some extra work to maintain code legibility and navigate across files. With the inclusion of JavaScript libraries like jQuery, script navigation has become a challenge itself because of the code length.
  2. Visual Studio has renewed the JavaScript editor with the promise to make the code mode accessible and organized. Many Visual Studio features that already existed in C# or VB editors are now implemented in the JavaScript editor: Go To Definition, automatic indentation, documentation and validation when you are writing. With the renewed IntelliSense list you will have the JavaScript function catalog at your fingertips.
  3. In this exercise, you will learn some of the new features and improvements of JavaScript editor. You will browse sample files and discover each of the new characteristics that will make your JavaScript programming more efficient within Visual Studio 11.

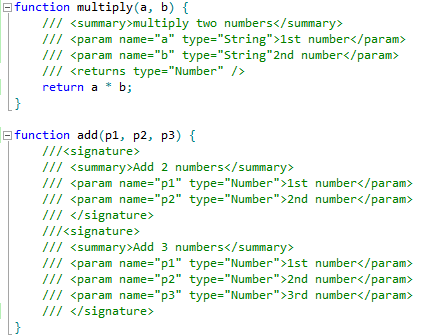
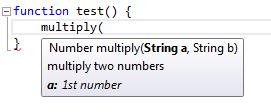
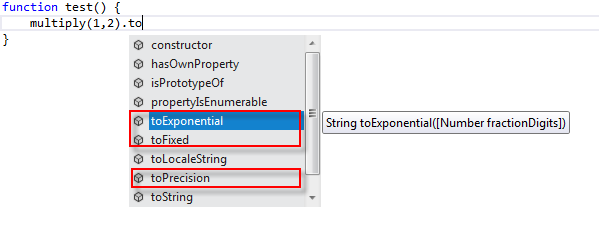
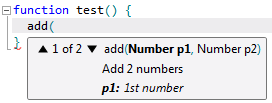
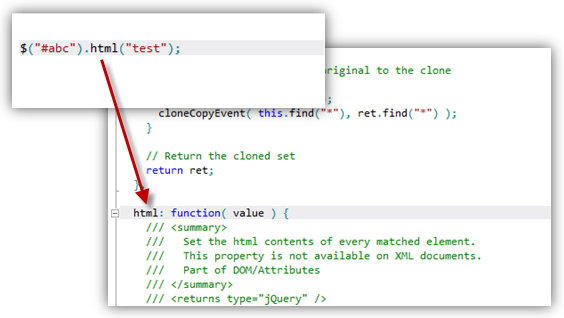
Task 1 – JavaScript Editor New Features

* 1. This task will introduce you to some of the new JavaScript editor features, which focus on organizing your code and bringing a better user experience.
  2. If not already opened, start **Visual Studio** and open the **WhatsNewASPNET.sln** solution located in the **Source\WhatsNewASPNET** folder of this lab.
  3. Press **F5** to run the application, then click the JavaScript link in the navigation bar. Refresh the page several times and check how the counter increments.
     1. 
     2. Figure 28
     3. Page counter
  4. Close the browser and go back to Visual Studio.
  5. Open the JavaScript.aspx page and locate the **<script>** block (shown below).
     1. The following code uses HTML5 local storage to store a *pageLoadCount* variable that stores the number of times the page has been visited by the current user. Local Storage is a client-side key-value database introduced with the HTML5 standard. The data is saved on the local machine, inside the user’s browser.
     2. HTML
     3. <script>
     4. addCount(1);
     5. document.getElementById('count').innerHTML = getCount();
     7. function getCount() {
     8. var storage = window.localStorage;
     9. if (!storage.pageLoadCount)
     10. storage.pageLoadCount = 0;
     11. return storage.pageLoadCount;
     12. }
     14. function addCount(value) {
     15. window.localStorage.pageLoadCount = parseInt(getCount(), 10) + value;
     16. }
     17. ...
     18. </script>
  6. Edit the code and notice that IntelliSense for JavaScript includes HTML5 features, like local storage, and their inner methods.
     1. 
     2. Figure 29
     3. HTML5 JavaScript features in JavaScript
  7. Click any opening bracket (**{**) from the scripting code and notice that the brackets are highlighted.
     1. 
     2. Figure 30
     3. Brackets are highlighted
  8. Uncomment the function **testAutoAlign()** (you can use CTRL+K;CTRL+U) and locate the cursor inside the function code. Press enter to append a second line. Notice that the code is now **aligned** and **auto-indented**.
     1. 
     2. Figure 31
     3. JavaScript code is auto aligned

Task 2 – Validating JavaScript

* 1. In this task, you will discover the new JavaScript validation for the ECMAScript5 standard. This feature will help you to write compliant JavaScript code, while preventing scripting issues before site deployment.
  2. **Note:** Visual Studio 2010 implemented ECMAStript3 compliance, while the Visual Studio 11 provides ECMAScript5 compliance.
  3. Open **ECMA5script5.js** located under the **Scripts\custom** project folder. You will now test validation for ECMAScript5 standard.
     1. JavaScript
     2. "use strict";
     3. if (true) {
     4. function StrictModeError()
     5. {
     6. }
     7. }
     8. You can check out the “**use strict”** direction in the first line of the file, which enables ECMAScript5 **strict mode**. This mode consists in a subset of the language that clarifies ambiguities from the past edition, and adds some new features, such as getters and setters, library support for JSON, and more complete reflection on object properties.
  4. Open the **Error List** if not already opened (View menu | **Error List**)**.** Notice the **function** declaration is underlined. This is because in ECMA5 standard functions cannot be nested inside language structures. In the error list below you will see the warning details.
     1. 
     2. Figure 32
     3. JavaScript validation error message
  5. Comment out the **“use strict”** direction and notice that errors disappear, but the warnings remain.
  6. In the last line of the file, write any string like **“test”** (include the quotation marks to indicate it is as string). Write a period next to the string to display the IntelliSense list, and select the “trim” option.
     1. In ECMAScript5 standard, string values and variables also have string methods defined, like trim, uppercase, search and replace.
     2. 
     3. Figure 33
     4. IntelliSense list in JavaScript

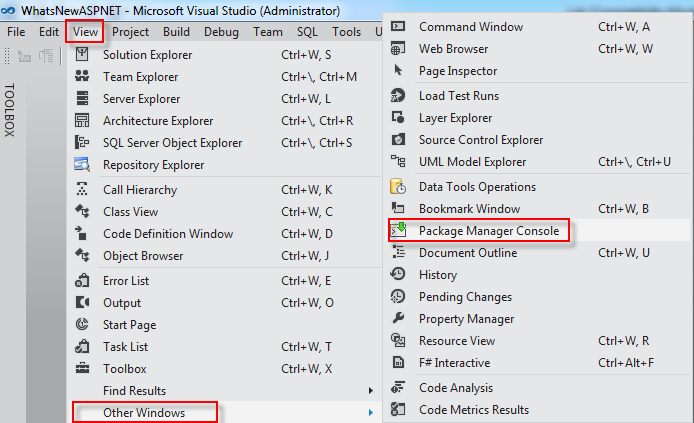
Task 3 – XML Documentation for JavaScript

* 1. In this task, you will explore Visual Studio features for XML documentation in JavaScript. You will see the JavaScript IntelliSense list now shows the XML documentation of each function. Additionally, you will discover the navigation feature in JavaScript.
  2. Open **XMLDoc.js** file located in **Scripts/custom** project folder. This file contains XML documentation on each of the JavaScript functions.
     1. 
     2. Figure 34
     3. JavaScript XML documentation integrated to IntelliSense
  3. Below **add** function in **XMLDoc.js** file, create a new function named **test**.
  4. In the **test** function, call the **multiply** function that receives two parameters. Notice the tooltip box is showing the **multiply** functiondocumentation.
     1. JavaScript
     2. **function test() {**
     3. multiply(
     4. **}**
     5. 
     6. Figure 35
     7. XML documentation for JavaScript functions
  5. Complete the function call statement and type a *dot* to open the IntelliSense list on the returned value. Notice that Visual Studio is detecting the **return** value in the documentation, treating the value as a number.
     1. 
     2. Figure 36
     3. XML documentation for return types
  6. Now, insert a call to add function. Notice that the JavaScript editor now supports function overloads. When you write a function name, you will be able to select any of the available overloads specified in the documentation.
     1. 
     2. Figure 37
     3. XML documentation for overloads
  7. Open **GotoDefinition.js** file and locate the **$().html()** function call. Locate the cursor on **html**.
  8. Press **F12** and navigate to the definition. Notice you can now access and browse your JavaScript code without using the **Find** tool.
  9. Locate the cursor on the jQuery instruction prior to the signature block at the bottom of the code file. Press **F12**. You will navigate to the jQuery library file. Notice you can also navigate across the jQuery files using **F12**.
     1. 
     2. Figure 38
     3. Navigating to jQuery definitions
     4. **Note:** Make sure that GotoDefinition.js has no syntax errors before saving the file.

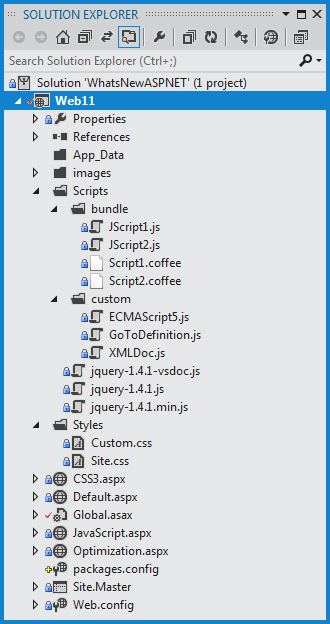
Exercise 4: Bundling and Minification

* 1. How many times do your websites include more than one JavaScript or CSS file? This is a very common scenario where bundling and minification can help to reduce the file size and make the site perform faster. The new bundling feature in ASP.NET 4.5 packs a set of JS or CSS files into a single element, and reduces its size by minifying the content (i.e. removing not required blank spaces, removing comments, reducing identifiers).
  2. Bundling and minification in ASP.NET 4.5 is performed at runtime, so that the process can identify the user agent (for example IE, Mozilla, etc.), and thus, improve the compression by targeting the user browser (for instance, removing stuff that is Mozilla specific when the request comes from IE).
  3. In this exercise, you will learn how to enable and use the different types of bundling and minification in ASP.NET 4.5.

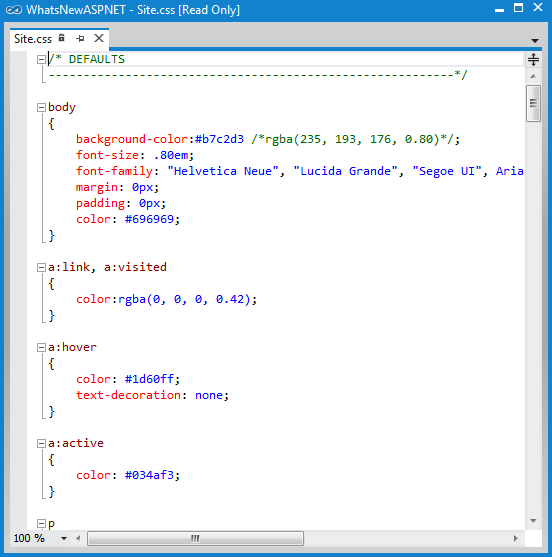
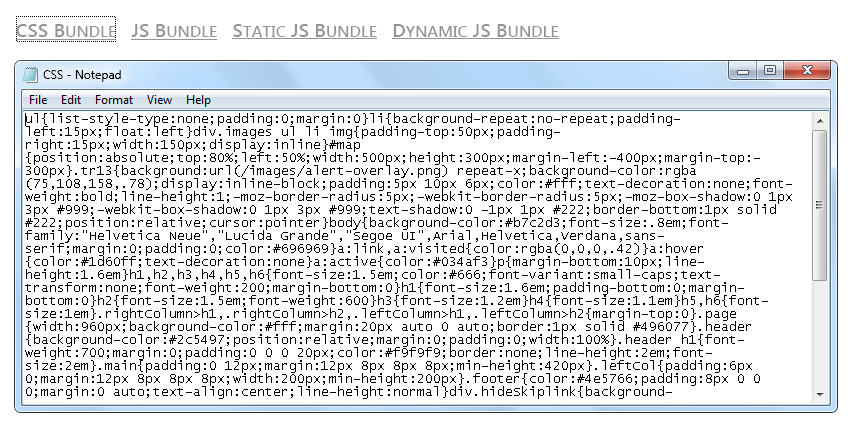
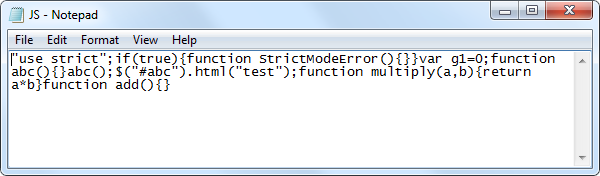
Task 1 – Installing the Bundling and Minification Package from NuGet

* 1. Open **Visual Studio 11** if not already opened.
  2. If not already opened, start **Visual Studio** and open the **WhatsNewASPNET.sln** solution located in the **Source\WhatsNewASPNET** folder of this lab.
  3. Open the NuGet Package Manager Console. To do this, use the menu **View** | **Other Windows** | **Package Manager Console**.
     1. 
     2. Figure 39
     3. Opening the package manager console
  4. In the **Package Manager Console,** type **Install-Package Microsoft.Web.Optimization**

Task 2 – Default Bundles

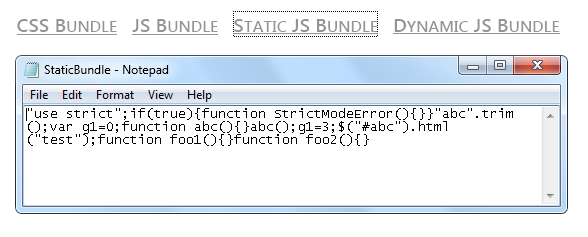
* 1. The simplest way to use bundling and minification is to enable the default bundles. This method uses conventions to let you reference the bundled and minified version for the JS and CSS files in a folder.
  2. In this task, you will learn how to enable and reference the bundled and minified JS and CSS files and view the resulting output.
  3. If not already opened, start **Visual Studio** and open the **WhatsNewASPNET.sln** solution located in the **Source\WhatsNewASPNET** folder of this lab.
  4. In the **Solution Explorer**, expand the **Styles**, **Scripts\custom** and **Scripts\bundle** folders.
     1. Notice that the application is using more than one CSS and JS file.
     2. 
     3. Figure 40
     4. Multiple Stylesheets and JavaScript files in the application
  5. Open the **Global.asax.cs** file.
     1. Notice that the new **Microsoft.Web.Optimization** namespace is commented out at the beginning of the file. Uncomment the using directive to include the bundling and minification features.
     2. C#
     3. using System;
     4. using System.Collections.Generic;
     5. using System.Linq;
     6. using System.Web;
     7. using System.Web.Security;
     8. using System.Web.SessionState;
     9. using Microsoft.Web.Optimization;
  6. Locate the **Application\_Start** method.
     1. In this method, you are enabling the default bundling. This enables us to reference a bundled collection of CSS files in a folder by using the path to that folder, plus the “CSS” or the “JS” suffix.
     2. C#

void Application\_Start(object sender, EventArgs e)

* + 1. {
    2. // Default behavior
    3. // Bundles all .js files in folders such as "scripts" if URL pointed to it: http://localhost:54716/scripts/custom/js
    4. BundleTable.Bundles.EnableDefaultBundles();
    5. ...
    6. }
  1. Open the **Optimization.aspx** file and locate the Content control for **HeadContent**.
     1. Notice the CSS files and the JS files are have a single referenced tag.
     2. HTML
     3. <asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" runat="server">
     4. <link href="**Styles/CSS**" rel="stylesheet" type="text/css" />
     5. <script src="**Scripts/custom/JS**"></script>
     6. </asp:Content>
     7. **Note:** This code is for demo purposes. Ideally, you will reference the bundles in the Site.Master file. In this sample code, you will find that some of the bundled files are also being referenced by the Site.Master file, making this last reference redundant.
  2. Notice that the links are using the bundling conventions in the **href** attribute to get all the CSS or JS files from the Styles and Scripts\custom folder respectively.
     1. You can use the path **Scripts/{custom}/JS** as shown below to bundle and minify all the JS files inside a **Scripts/{custom}** folder. This is the default behavior with the default bundles.
     2. HTML
     3. <asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
     4. <ul>
     5. <li>
     6. **<h2><a href="Styles/CSS">CSS Bundle</a></h2>**
     7. </li>
     8. <li>
     9. **<h2><a href="Scripts/custom/JS">JS Bundle</a></h2>**
     10. </li>
  3. Open the **Styles\Site.css** file.
     1. Notice that the original CSS file contains indented code, blank spaces and comments that enlarge the file. (Also the JavaScript file contains blank spaces and comments).
     2. 
     3. Figure 41
     4. One of the original CSS files in the Scripts folder
  4. Press **F5** to run the application and navigate to the **Optimization** page.
  5. Click on the **CSS Bundle** link to download and open the file.
     1. Check out the minified bundled file. You will notice that all the blank spaces, comments and indentation characters have been removed, generating a smaller file.
     2. 
     3. Figure 42
     4. Bundled CSS files
  6. Now click the **JS Bundle** link and open the downloaded file with Notepad. You can safely disregard the explorer warning. Notice the JavaScript files under the **custom** folder are also bundled and minified.
     1. 
     2. Figure 43
     3. Bundled JavaScript files
     4. Enabling compression for CSS or JS files was much more complicated in previous ASP.NET version. Now, as you have seen, you just need to add one line in the *Global.asax* file to enable bundling, and then reference the bundled files from your site.



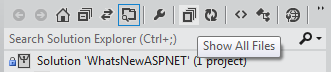
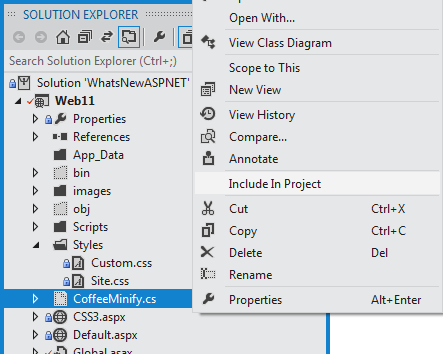
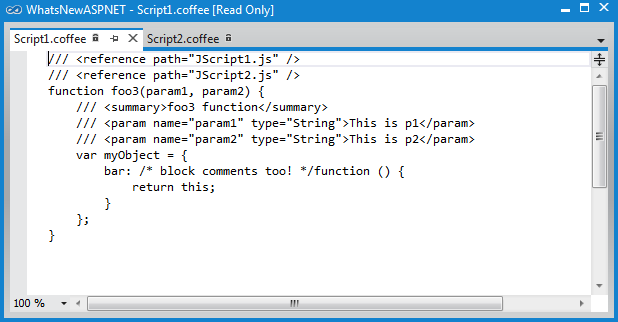
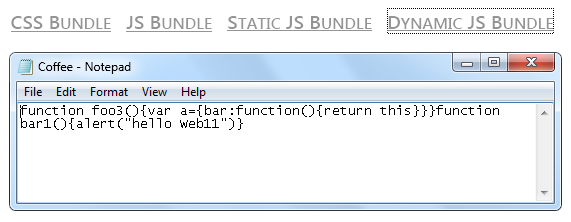
Task 3 – Static Bundles

* 1. The static bundle approach allows you to customize the set of files to bundle, the reference and the minification method that will be used.
  2. In this task, you will configure a static bundle to define a specific set of files to bundle and minify.
  3. Close the browser.
  4. Open the **Global.asax.cs** file and locate the **Application\_Start** method.
  5. Uncomment the static bundle code as shown in the code below.
     1. You are defining a static bundle that will be referenced with the “**~/StaticBundle**” virtual path and use **JsMinify** for minification for all the specified files with the **AddFile** method. Finally, you are adding the static bundle to the **BundleTable** and enabling it.
     2. Notice that the files are not located in the same place; this is another advantage over the default bundling.
     3. C#
     4. void Application\_Start(object sender, EventArgs e)
     5. {
     6. // Default behavior
     7. // Bundles all .js files in folders such as "scripts" if URL pointed to it: http://localhost:54716/scripts/custom/js
     8. BundleTable.Bundles.EnableDefaultBundles();
     9. **// Static bundle.**
     10. **// Access on url http://localhosthost:54716/StaticBundle**
     11. **Bundle b = new Bundle("~/StaticBundle", typeof(JsMinify));**
     12. **b.AddFile("~/scripts/custom/ECMAScript5.js");**
     13. **b.AddFile("~/scripts/custom/GoToDefinition.js");**
     14. **b.AddFile("~/scripts/bundle/JScript1.js");**
     15. **b.AddFile("~/scripts/bundle/JScript2.js");**
     16. **BundleTable.Bundles.Add(b);**
     17. //// Dynamic bundle
     18. //// Bundles all .coffee files in folders such as "script" when "coffee" is appended to it: http://localhost:54716/scripts/coffee
     19. //DynamicFolderBundle fb = new DynamicFolderBundle("coffee", typeof(CoffeeMinify), "\*.coffee");
     20. //BundleTable.Bundles.Add(fb);
     21. }
  6. Open the **Optimization.aspx** file.
     1. Notice that the link to **Static JS Bundle** is using the path you have declared when you configured the static bundle in the Global.asax.cs file: **/StaticBundle**.
     2. HTML
     3. <asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
     4. <ul>
     5. <li>
     6. <h2><a href="Styles/CSS">CSS Bundle</a></h2>
     7. </li>
     8. <li>
     9. <h2><a href="Scripts/custom/JS">JS Bundle</a></h2>
     10. </li>
     11. <li>
     12. <h2><a href="**StaticBundle**">Static JS Bundle</a></h2>
     13. </li>
     14. <li>
     15. <h2><a href="Scripts/bundle/Coffee">Dynamic JS Bundle</a></h2>
     16. </li>
     17. </ul>
     18. </asp:Content>
  7. Press **F5** to run the application, and then navigate to the **Optimization** page.
  8. Click on the **Static JS Bundle** link to download and open the file in Notepad.
     1. Notice that the minified bundled JavaScript file is the output for all the JavaScript files configured in the static bundle file under the path “/StaticBundle”.
     2. 
     3. Figure 44
     4. Static JavaScript files bundle
  9. Close the browser and return to Visual Studio.

Task 4 – Dynamic Folder Bundles

* 1. In this task, you will learn how to configure dynamic folder bundles. The power of dynamic bundling is that you can include static JavaScript, as well as other files in languages that compiles into JavaScript, and thus, require some processing before the bundling is executed.
  2. In this example, you will learn how to use the **DynamicFolderBundle** class to create a dynamic bundle for files written in CofeeScript. CofeeScript is a programming language that compiles into JavaScript and provides a simpler syntax for writing JavaScript code, enhancing JavaScript's brevity and readability.
  3. Open the **Global.asax.cs** file and locate the **Application\_Start** method.
  4. Uncomment the dynamic bundle code as shown in the code below.

You are defining a dynamic folder bundle that will use the **CoffeeMinify** custom minification processor that will only apply to the files with the “**.coffee**” extension (CoffeeScript files). Notice that you can use a search pattern to select the files to bundle within a folder, like ‘\*.coffee’.

* + 1. C#
    2. void Application\_Start(object sender, EventArgs e)
    3. {
    4. // Default behavior
    5. // Bundles all .js files in folders such as "scripts" if URL pointed to it: http://localhost:54716/scripts/custom/js
    6. BundleTable.Bundles.EnableDefaultBundles();
    7. // Static bundle.
    8. // Access on url http://localhosthost:54716/StaticBundle
    9. ...
    10. **// Dynamic bundle**
    11. **// Bundles all .coffee files in folders such as "script" when "coffee" is appended to it: http://localhost:54716/scripts/coffee**
    12. **DynamicFolderBundle fb = new DynamicFolderBundle("coffee", typeof(CoffeeMinify), "\*.coffee");**
    13. **BundleTable.Bundles.Add(fb);**
    14. }
  1. Click the **Show All Files** button in the **Solution Explorer** window
     1. 
     2. Figure 45
     3. Showing all files
  2. Right-click the **CoffeeMinify.cs** file in the **Solution Explorer** and select **Include in Project**
     1. 
     2. Figure 46
     3. Include the CoffeeMinify.cs file in the project
  3. Open the **CoffeeMinify.cs** file.
     1. This class inherits from JsMinify to minify the JavaScript code. Notice that it is incomplete; ideally, you will call the CoffeeScript compiler to generate the JavaScript code first, and then you will send it to the JsMinify.Process method to minify the resulting code.
     2. C#
     3. public class CoffeeMinify : JsMinify
     4. {
     5. public CoffeeMinify() { }
     6. public override void Process(BundleResponse bundle)
     7. {
     8. //Write coffee compiler calls here
     9. //pass bundle.Files to it
     10. //replace bundle.Files with the output of Coffee Compiler
     11. //now pass it to JS Minify
     12. base.Process(bundle);
     13. }
     14. }
  4. Open the **Script1.coffee** and **Script2.coffee** files from the **Scripts/bundle** folder.
     1. These files will include the CoffeScript code to be compiled while performing the bundling with the CoffeeMinify class.
     2. For simplicity purposes, the CoffeeScript files provided are only including JavaScript and CoffeeScript code . The comments are excluded by the JsMinify process.
     3. 
     4. Figure 47
     5. CoffeeScript files
     6. **Note:** [CofeeScript](http://jashkenas.github.com/coffee-script/) provides a simpler syntax for writing JavaScript code, enhancing JavaScript's brevity and readability, as well as adding other features like array comprehension and pattern matching.
  5. Open the **Optimization.aspx** file and locate the bundle links.
     1. Notice that the link to **Dynamic JS Bundle** is referencing the **Scripts/bundle** folder by using the **/Coffee** suffix you configured for the dynamic folder bundle.
     2. HTML
     3. <asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
     4. <ul>
     5. <li>
     6. <h2><a href="Styles/CSS">CSS Bundle</a></h2>
     7. </li>
     8. <li>
     9. <h2><a href="Scripts/custom/JS">JS Bundle</a></h2>
     10. </li>
     11. <li>
     12. <h2><a href="**StaticBundle**">Static JS Bundle</a></h2>
     13. </li>
     14. <li>
     15. **<h2><a href="Scripts/bundle/Coffee">Dynamic JS Bundle</a></h2>**
     16. </li>
     17. </ul>
     18. </asp:Content>
  6. Press **F5** to run the application, and then navigate to the **Optimization** page.
  7. Click on the **Dynamic JS Bundle** link to download and open the file in Notepad.
     1. Notice that the JavaScript content that was included in this bundle only contains **.coffee** files. You can also see that the commented-out code has been removed.
     2. 
     3. Figure 48
     4. Dynamic JS files bundle