# Bundling and Minification

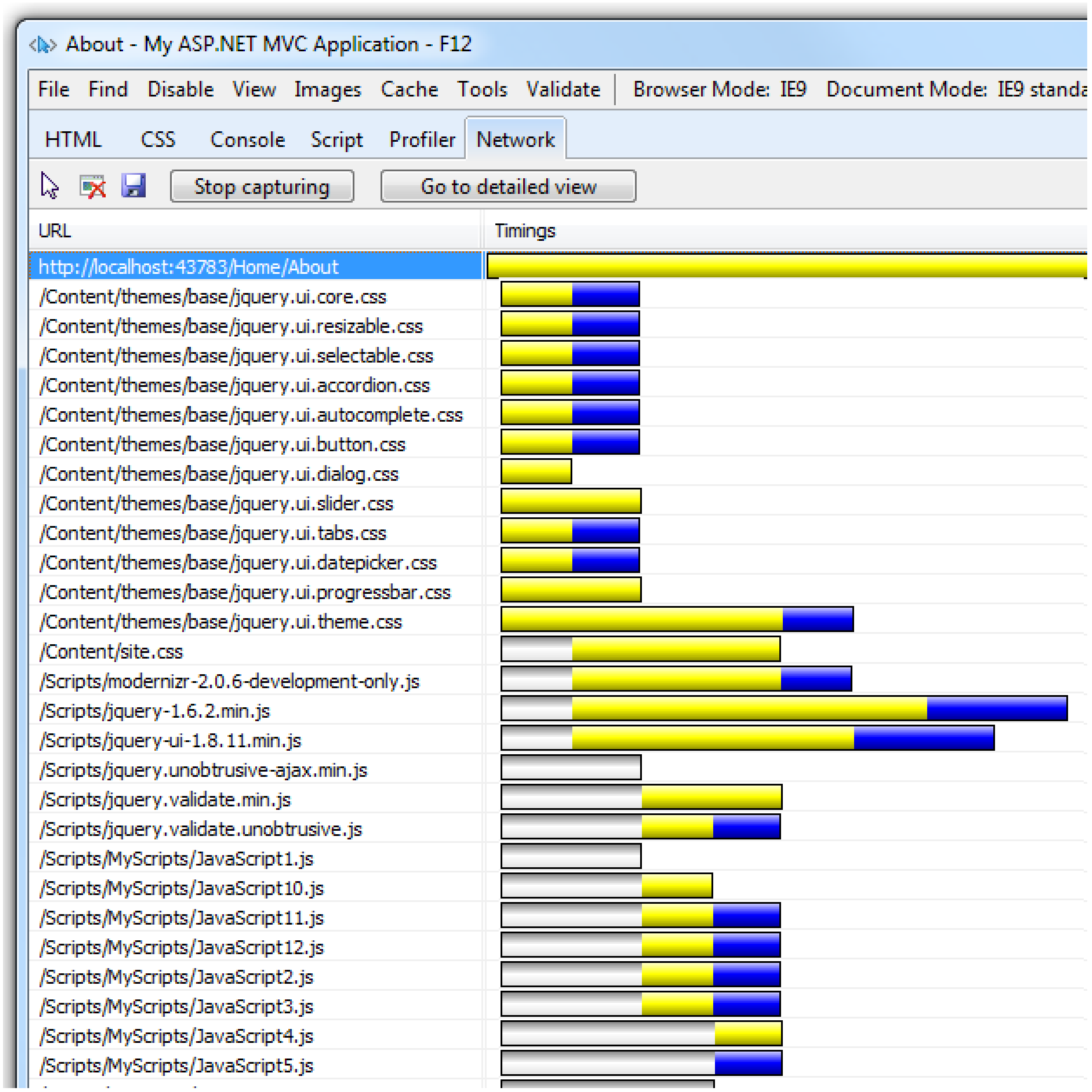
08/23/2012 • 13 minutes to read • +4

**In this article**

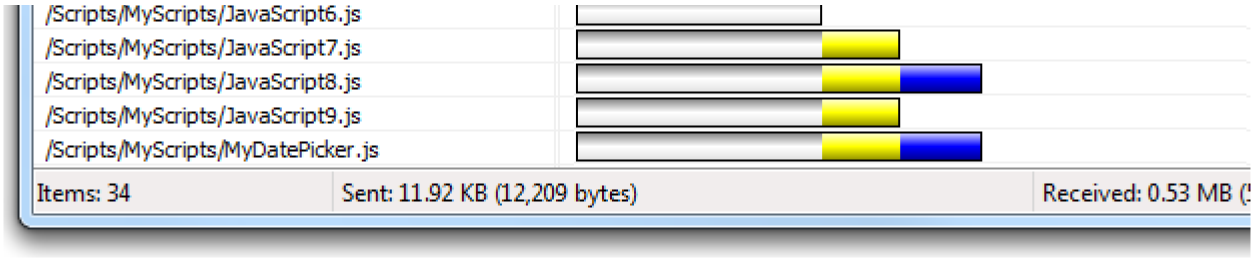
by Rick Anderson

> Bundling and minification are two techniques you can use in ASP.NET 4.5 to improve request load time. Bundling and minification improves load time by reducing the number of requests to the server and reducing the size of requested assets (such as CSS and JavaScript.)

Most of the current major browsers limit the number of simultaneous connections per each hostname to six. That means that while six requests are being processed, additional requests for assets on a host will be queued by the browser. In the image below, the IE F12 developer tools network tabs shows the timing for assets required by the About view of a sample application.


About - My ASP.NET MVC Application - F12
File Find Disable View Images Cache Tools Validate Browser Mode: IE9 Document Mode: IE9 standa
HTML CSS Console Script Profiler Network
Stop capturing
Go to detailed view

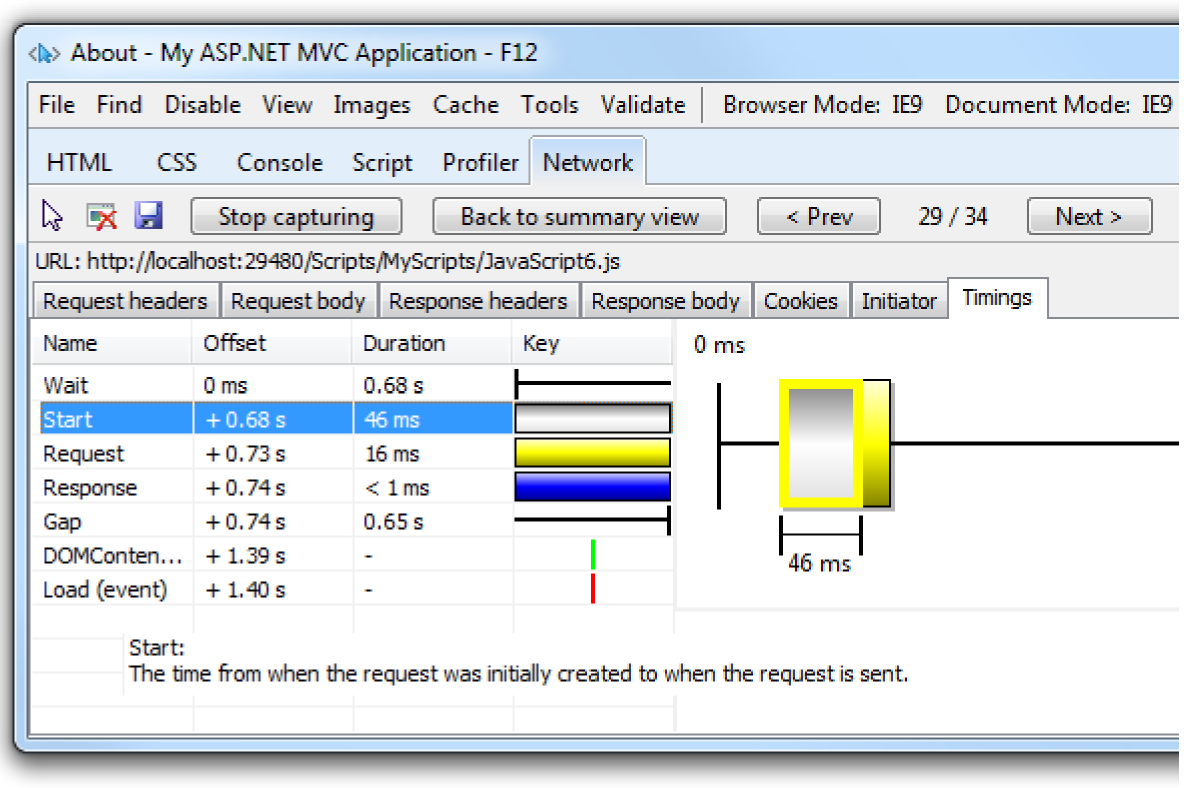
| URL | Timings |
|---|---|
| http://localhost:43783/Home/About | |
| /Content/themes/base/jquery.ui.core.css | |
| /Content/themes/base/jquery.ui.resizable.css | |
| /Content/themes/base/jquery.ui.selectable.css | |
| /Content/themes/base/jquery.ui.accordion.css | |
| /Content/themes/base/jquery.ui.autocomplete.css | |
| /Content/themes/base/jquery.ui.button.css | |
| /Content/themes/base/jquery.ui.dialog.css | |
| /Content/themes/base/jquery.ui.slider.css | |
| /Content/themes/base/jquery.ui.tabs.css | |
| /Content/themes/base/jquery.ui.datepicker.css | |
| /Content/themes/base/jquery.ui.progressbar.css | |
| /Content/themes/base/jquery.ui.theme.css | |
| /Content/site.css | |
| /Scripts/modernizr-2.0.6-development-only.js | |
| /Scripts/jquery-1.6.2.min.js | |
| /Scripts/jquery-ui-1.8.11.min.js | |
| /Scripts/jquery.unobtrusive-ajax.min.js | |
| /Scripts/jquery.validate.min.js | |
| /Scripts/jquery.validate.unobtrusive.js | |
| /Scripts/MyScripts/JavaScript1.js | |
| /Scripts/MyScripts/JavaScript10.js | |
| /Scripts/MyScripts/JavaScript11.js | |
| /Scripts/MyScripts/JavaScript12.js | |
| /Scripts/MyScripts/JavaScript2.js | |
| /Scripts/MyScripts/JavaScript3.js | |
| /Scripts/MyScripts/JavaScript4.js | |
| /Scripts/MyScripts/JavaScript5.js | |



| | |
|---|---|
| /Scripts/MyScripts/JavaScript6.js | |
| /Scripts/MyScripts/JavaScript7.js | |
| /Scripts/MyScripts/JavaScript8.js | |
| /Scripts/MyScripts/JavaScript9.js | |
| /Scripts/MyScripts/MyDatePicker.js | |

Items: 34 | Sent: 11.92 KB (12,209 bytes) | Received: 0.53 MB (

The gray bars show the time the request is queued by the browser waiting on the six connection limit. The yellow bar is the request time to first byte, that is, the time taken to send the request and receive the first response from the server. The blue bars show the time taken to receive the response data from the server. You can double-click on an asset to get detailed timing information. For example, the following image shows the timing details for loading the */Scripts/MyScripts/JavaScript6.js* file.



About - My ASP.NET MVC Application - F12
File Find Disable View Images Cache Tools Validate Browser Mode: IE9 Document Mode: IE9
HTML CSS Console Script Profiler Network
Stop capturing
Back to summary view
< Prev
29 / 34
Next >
Request headers Request body Response headers Response body Cookies Initiator Timings
0 ms
46 ms

URL: http://localhost:29480/Scripts/MyScripts/JavaScript6.js

| Name | Offset | Duration | Key |
|---|---|---|---|
| Wait | 0 ms | 0.68 s | |
| Start | + 0.68 s | 46 ms | |
| Request | + 0.73 s | 16 ms | |
| Response | + 0.74 s | < 1 ms | |
| Gap | + 0.74 s | 0.65 s | |
| DOMConten… | + 1.39 s | - | |
| Load (event) | + 1.40 s | - | |

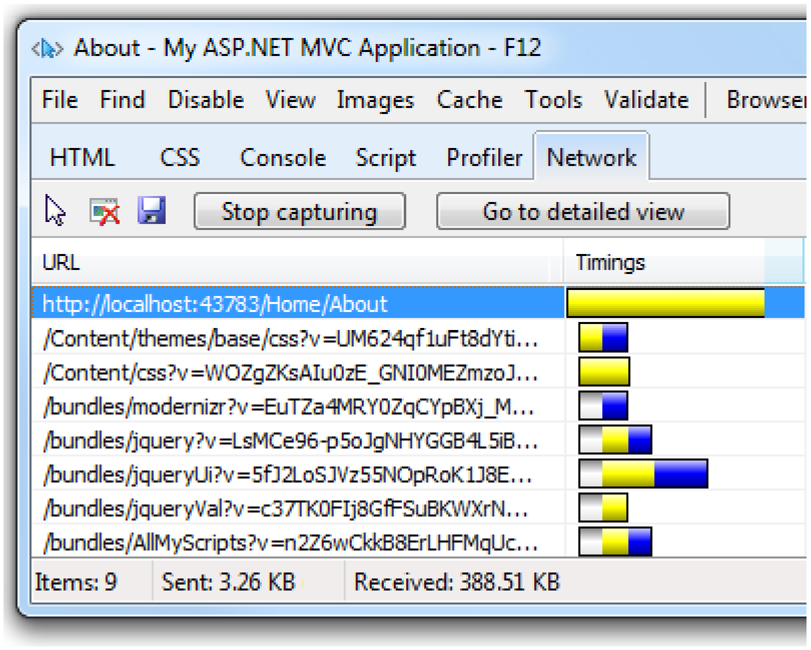
Start:
The time from when the request was initially created to when the request is sent.

The preceding image shows the **Start** event, which gives the time the request was queued because of the browser limit the number of simultaneous connections. In this case, the request was queued for 46 milliseconds waiting for another request to complete.

## Bundling

Bundling is a new feature in ASP.NET 4.5 that makes it easy to combine or bundle multiple files into a single file. You can create CSS, JavaScript and other bundles. Fewer files means fewer HTTP requests and that can improve first page load performance.

The following image shows the same timing view of the About view shown previously, but this time with bundling and minification enabled.


About - My ASP.NET MVC Application - F12
File Find Disable View Images Cache Tools Validate Browser
HTML CSS Console Script Profiler Network
Stop capturing
Go to detailed view

| URL | Timings |
|---|---|
| http://localhost:43783/Home/About | |
| /Content/themes/base/css?v=UM624qf1uFt8dYti... | |
| /Content/css?v=WOZgZKsAIu0zE_GNI0MEZmzoJ... | |
| /bundles/modernizr?v=EuTZa4MRY0ZqCYpBXj_M... | |
| /bundles/jquery?v=LsMCe96-p5oJgNHYGGB4L5iB... | |
| /bundles/jqueryUi?v=5fJ2LoSJVz55NOpRoK1J8E... | |
| /bundles/jqueryVal?v=c37TK0FIj8GfFSuBKWXrN... | |
| /bundles/AllMyScripts?v=n2Z6wCkkB8ErLHFMqUc... | |

Items: 9 Sent: 3.26 KB Received: 388.51 KB

## Minification

Minification performs a variety of different code optimizations to scripts or css, such as removing unnecessary white space and comments and shortening variable names to one character. Consider the following JavaScript function.

JavaScript Copy

```
AddAltToImg = function (imageTagAndImageID, imageContext) {
    ///<signature>
    ///<summary> Adds an alt tab to the image
    // </summary>
    //<param name="imgElement" type="String">The image selector.</param>
    //<param name="ContextForImage" type="String">The image context.</param>
    ///</signature>
    var imageElement = $(imageTagAndImageID, imageContext);
    imageElement.attr('alt', imageElement.attr('id').replace(/ID/, ''));
}
```

After minification, the function is reduced to the following:

JavaScript Copy

```
AddAltToImg = function (n, t) { var i = $(n, t); i.attr("alt", i.attr("id").replace(/ID/, "")) }
```

In addition to removing the comments and unnecessary whitespace, the following parameters and variable names were renamed (shortened) as follows:

| **Original** | **Renamed** |
|---|---|
| imageTagAndImageID | n |
| imageContext | t |
| imageElement | i |

## Impact of Bundling and Minification

The following table shows several important differences between listing all the assets individually and using bundling and minification (B/M) in the sample program.

| | Using B/M | Without B/M | Change |
|---|---|---|---|
| **File Requests** | 9 | 34 | 256% |
| **KB Sent** | 3.26 | 11.92 | 266% |
| **KB Received** | 388.51 | 530 | 36% |
| **Load Time** | 510 MS | 780 MS | 53% |

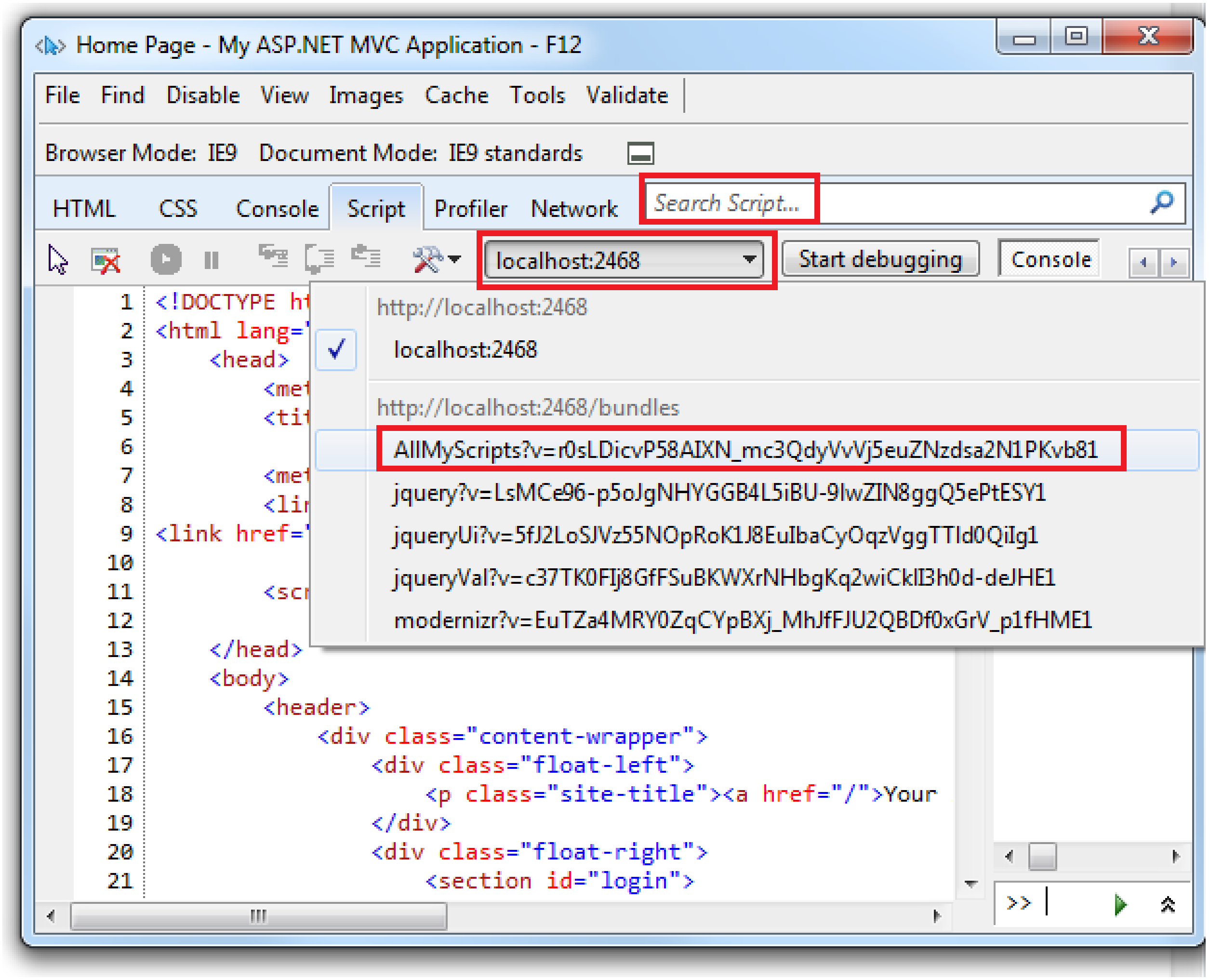
The bytes sent had a significant reduction with bundling as browsers are fairly verbose with the HTTP headers they apply on requests. The received bytes reduction is not as large because the largest files (*Scripts\jquery-ui-1.8.11.min.js* and *Scripts\jquery-1.7.1.min.js*) are already minified. Note: The timings on the sample program used the Fiddler tool to simulate a slow network. (From the Fiddler **Rules** menu, select **Performance** then **Simulate Modem Speeds**.)

## Debugging Bundled and Minified JavaScript

It's easy to debug your JavaScript in a development environment (where the compilation Element in the *Web.config* file is set to `debug="true"` ) because the JavaScript files are not bundled or minified. You can also debug a release build where your JavaScript files are bundled and minified. Using the IE F12 developer tools, you debug a JavaScript function included in a minified bundle using the following approach:

1. Select the **Script** tab and then select the **Start debugging** button.

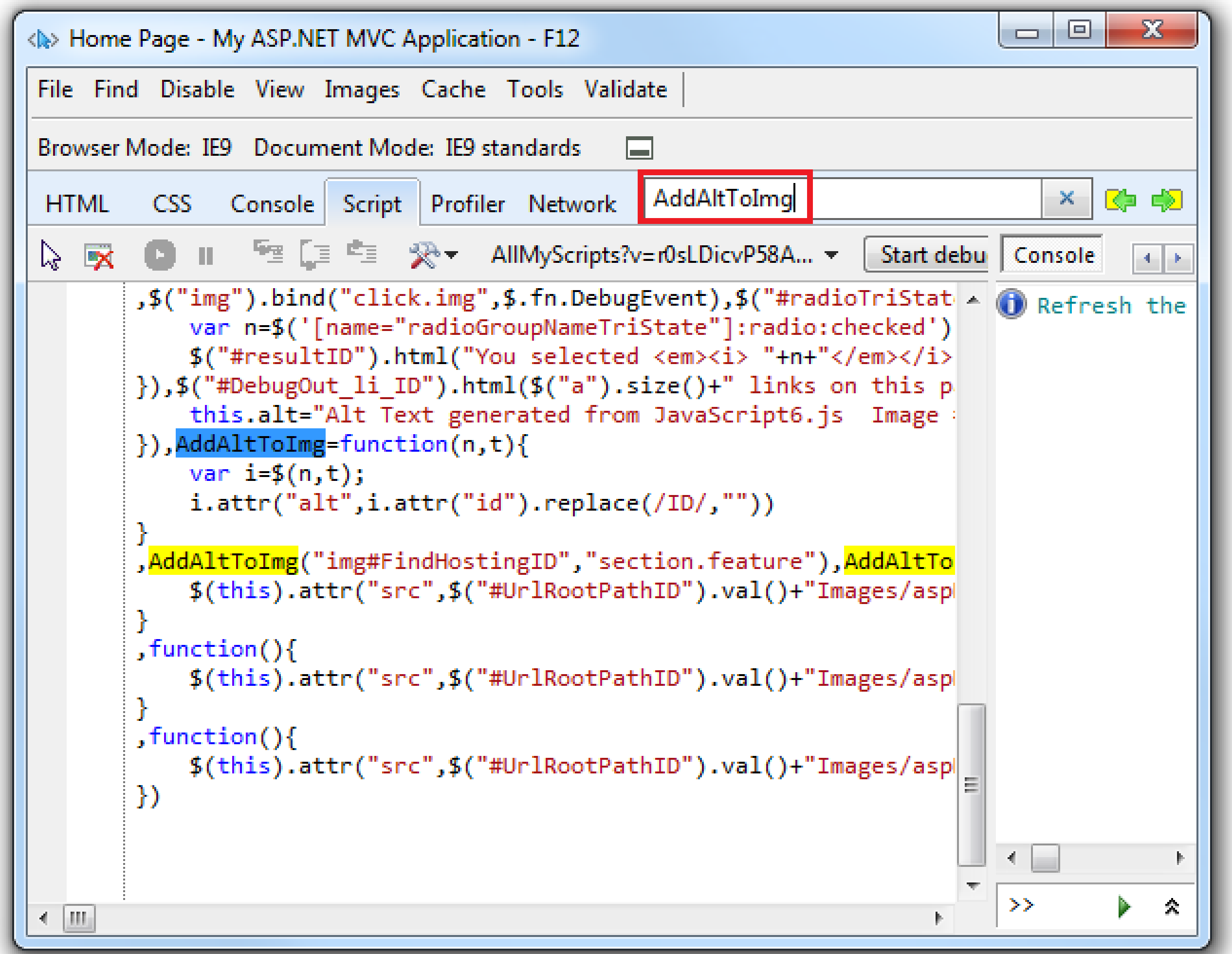
2. Select the bundle containing the JavaScript function you want to debug using the assets button.


Home Page - My ASP.NET MVC Application - F12
File Find Disable View Images Cache Tools Validate
Browser Mode: IE9 Document Mode: IE9 standards
HTML CSS Console Script Profiler Network
Search Script...
localhost:2468
Start debugging
Console

```
<!DOCTYPE ht
<html lang='
    <head>
        <met
        <tit

        <met
        <lin
<link href='

        <scr

    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p class="site-title"><a href="/">Your
                </div>
                <div class="float-right">
                    <section id="login">
```

```
http://localhost:2468
✓ localhost:2468
http://localhost:2468/bundles
AllMyScripts?v=r0sLDicvP58AIXN_mc3QdyVvVj5euZNzdsa2N1PKvb81
jquery?v=LsMCe96-p5oJgNHYGGB4L5iBU-9lwZIN8ggQ5ePtESY1
jqueryUi?v=5fJ2LoSJVz55NOpRoK1J8EuIbaCyOqzVggTTId0QiIg1
jqueryVal?v=c37TK0FIj8GfFSuBKWXrNHbgKq2wiCkII3h0d-deJHE1
modernizr?v=EuTZa4MRY0ZqCYpBXj_MhJfFJU2QBDf0xGrV_p1fHME1
```

3. Format the minified JavaScript by selecting the **Configuration button**, and then selecting **Format JavaScript**.
4. In the **Search Script** input box, select the name of the function you want to debug. In the following image, **AddAltToImg** was entered in the **Search Script** input box.


Home Page - My ASP.NET MVC Application - F12
File Find Disable View Images Cache Tools Validate
Browser Mode: IE9 Document Mode: IE9 standards
HTML CSS Console Script Profiler Network
AddAltToImg
AllMyScripts?v=r0sLDicvP58A...
Start debu
Console
Refresh the

```
,$("img").bind("click.img",$.fn.DebugEvent),$("#radioTriStat
    var n=$('[name="radioGroupNameTriState"]:radio:checked')
    $("#resultID").html("You selected <em><i> "+n+"</em></i>
}),$("#DebugOut_li_ID").html($("a").size()+" links on this p
    this.alt="Alt Text generated from JavaScript6.js  Image :
}),AddAltToImg=function(n,t){
    var i=$(n,t);
    i.attr("alt",i.attr("id").replace(/ID/,""))
}
,AddAltToImg("img#FindHostingID","section.feature"),AddAltTo
    $(this).attr("src",$("#UrlRootPathID").val()+"Images/aspl
}
,function(){
    $(this).attr("src",$("#UrlRootPathID").val()+"Images/aspl
}
,function(){
    $(this).attr("src",$("#UrlRootPathID").val()+"Images/aspl
})
```

For more information on debugging with the F12 developer tools, see the MSDN article Using the F12 Developer Tools to Debug JavaScript Errors.

## Controlling Bundling and Minification

Bundling and minification is enabled or disabled by setting the value of the debug attribute in the compilation Element in the *Web.config* file. In the following XML, `debug` is set to true so bundling and minification is disabled.

XML

Copy

```xml
<system.web>
    <compilation debug="true" />
    <!-- Lines removed for clarity. -->
</system.web>
```

To enable bundling and minification, set the `debug` value to "false". You can override the *Web.config* setting with the `EnableOptimizations` property on the `BundleTable` class. The following code enables bundling and minification and overrides any setting in the *Web.config* file.

C#

Copy

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

    // Code removed for clarity.
    BundleTable.EnableOptimizations = true;
}
```

ⓘ **Note**

Unless `EnableOptimizations` is `true` or the debug attribute in the compilation Element in the *Web.config* file is set to `false`, files will not be bundled or minified. Additionally, the .min version of files will not be used, the full debug versions will be selected. `EnableOptimizations` overrides the debug attribute in the compilation Element in the *Web.config* file

## Using Bundling and Minification with ASP.NET Web Forms and Web Pages

- For Web Pages, see the blog entry Adding Web Optimization to a Web Pages Site.
- For Web Forms, see the blog entry Adding Bundling and Minification to Web Forms.

## Using Bundling and Minification with ASP.NET MVC

In this section we will create an ASP.NET MVC project to examine bundling and minification. First, create a new ASP.NET MVC internet project named **MvcBM** without changing any of the defaults.

Open the *App\_Start\BundleConfig.cs* file and examine the `RegisterBundles` method which is used to create, register and configure bundles. The following code shows a portion of the `RegisterBundles` method.

C# Copy

```
public static void RegisterBundles(BundleCollection bundles)
{
     bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                  "~/Scripts/jquery-{version}.js"));
         // Code removed for clarity.
}
```

The preceding code creates a new JavaScript bundle named *~/bundles/jquery* that includes all the appropriate (that is debug or minified but not *.vsdoc*) files in the *Scripts* folder that match the wild card string "~/Scripts/jquery-{version}.js". For ASP.NET MVC 4, this means with a debug configuration, the file *jquery-1.7.1.js* will be added to the bundle. In a release configuration, *jquery-1.7.1.min.js* will be added. The bundling framework follows several common conventions such as:

- Selecting ".min" file for release when *FileX.min.js* and *FileX.js* exist.
- Selecting the non ".min" version for debug.

- Ignoring "-vsdoc" files (such as *jquery-1.7.1-vsdoc.js*), which are used only by IntelliSense.

The `{version}` wild card matching shown above is used to automatically create a jQuery bundle with the appropriate version of jQuery in your *Scripts* folder. In this example, using a wild card provides the following benefits:

- Allows you to use NuGet to update to a newer jQuery version without changing the preceding bundling code or jQuery references in your view pages.
- Automatically selects the full version for debug configurations and the ".min" version for release builds.

## Using a CDN

The follow code replaces the local jQuery bundle with a CDN jQuery bundle.

C# Copy

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

    bundles.UseCdn = true;   //enable CDN support

    //add link to jquery on the CDN
    var jqueryCdnPath = "https://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.1.min.js";

    bundles.Add(new ScriptBundle("~/bundles/jquery",
                jqueryCdnPath).Include(
                "~/Scripts/jquery-{version}.js"));

    // Code removed for clarity.
}
```

In the code above, jQuery will be requested from the CDN while in release mode and the debug version of jQuery will be fetched locally in debug mode. When using a CDN, you should have a fallback mechanism in case the CDN request fails. The following markup fragment from the end of the layout file shows script added to request jQuery should the CDN fail.

CSHTML Copy

```
</footer>

        @Scripts.Render("~/bundles/jquery")

        <script type="text/javascript">
            if (typeof jQuery == 'undefined') {
                var e = document.createElement('script');
                e.src = '@Url.Content("~/Scripts/jquery-1.7.1.js")';
                e.type = 'text/javascript';
                document.getElementsByTagName("head")[0].appendChild(e);

            }
        </script>

        @RenderSection("scripts", required: false)
    </body>
</html>
```

## Creating a Bundle

The Bundle class `Include` method takes an array of strings, where each string is a virtual path to resource. The following code from the `RegisterBundles` method in the *App_Start\BundleConfig.cs* file shows how multiple files are added to a bundle:

C# Copy

```
bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
    "~/Content/themes/base/jquery.ui.core.css",
    "~/Content/themes/base/jquery.ui.resizable.css",
    "~/Content/themes/base/jquery.ui.selectable.css",
    "~/Content/themes/base/jquery.ui.accordion.css",
    "~/Content/themes/base/jquery.ui.autocomplete.css",
    "~/Content/themes/base/jquery.ui.button.css",
    "~/Content/themes/base/jquery.ui.dialog.css",
    "~/Content/themes/base/jquery.ui.slider.css",
    "~/Content/themes/base/jquery.ui.tabs.css",
    "~/Content/themes/base/jquery.ui.datepicker.css",
    "~/Content/themes/base/jquery.ui.progressbar.css",
    "~/Content/themes/base/jquery.ui.theme.css"));
```

The Bundle class `IncludeDirectory` method is provided to add all the files in a directory (and optionally all subdirectories) which match a search pattern. The Bundle class `IncludeDirectory` API is shown below:

C# Copy

```csharp
public Bundle IncludeDirectory(
    string directoryVirtualPath,  // The Virtual Path for the directory.
    string searchPattern)         // The search pattern.

public Bundle IncludeDirectory(
    string directoryVirtualPath,  // The Virtual Path for the directory.
    string searchPattern,         // The search pattern.
    bool searchSubdirectories)    // true to search subdirectories.
```

Bundles are referenced in views using the Render method, (`Styles.Render` for CSS and `Scripts.Render` for JavaScript). The following markup from the *Views\Shared_Layout.cshtml* file shows how the default ASP.NET internet project views reference CSS and JavaScript bundles.

CSHTML Copy

```html
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    @* Markup removed for clarity.*@
    @Styles.Render("~/Content/themes/base/css", "~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    @* Markup removed for clarity.*@

   @Scripts.Render("~/bundles/jquery")
   @RenderSection("scripts", required: false)
</body>
</html>
```

Notice the Render methods takes an array of strings, so you can add multiple bundles in one line of code. You will generally want to use the Render methods which create the necessary HTML to reference the asset. You can use the `Url` method to generate the URL to the asset without the markup needed to reference the asset. Suppose you wanted to use the new HTML5 async attribute. The following code shows how to reference modernizr using the `Url` method.

CSHTML Copy

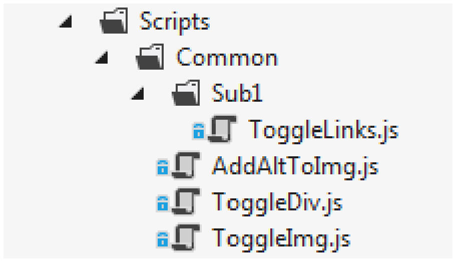
```
<head>
    @*Markup removed for clarity*@
    <meta charset="utf-8" />
    <title>@ViewBag.Title - MVC 4 B/M</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width" />
    @Styles.Render("~/Content/css")

   @* @Scripts.Render("~/bundles/modernizr")*@

    <script src='@Scripts.Url("~/bundles/modernizr")' async> </script>
</head>
```

## Using the "*" Wildcard Character to Select Files

The virtual path specified in the `Include` method and the search pattern in the `IncludeDirectory` method can accept one "*" wildcard character as a prefix or suffix to in the last path segment. The search string is case insensitive. The `IncludeDirectory` method has the option of searching subdirectories.

Consider a project with the following JavaScript files:

- *Scripts\Common\AddAltToImg.js*
- *Scripts\Common\ToggleDiv.js*
- *Scripts\Common\ToggleImg.js*
- *Scripts\Common\Sub1\ToggleLinks.js*



The following table shows the files added to a bundle using the wildcard as shown:

| **Call** | **Files Added or Exception Raised** |
|---|---|
| Include("~/Scripts/Common/*.js") | *AddAltToImg.js*, *ToggleDiv.js*, *ToggleImg.js* |
| Include("~/Scripts/Common/T*.js") | Invalid pattern exception. The wildcard character is only allowed on the prefix or suffix. |
| Include("~/Scripts/Common/*og.*") | Invalid pattern exception. Only one wildcard character is allowed. |
| Include("~/Scripts/Common/T*") | *ToggleDiv.js*, *ToggleImg.js* |
| Include("~/Scripts/Common/*") | Invalid pattern exception. A pure wildcard segment is not valid. |

| Call | Files Added or Exception Raised |
|---|---|
| IncludeDirectory("~/Scripts/Common", "T*") | *ToggleDiv.js*, *ToggleImg.js* |
| IncludeDirectory("~/Scripts/Common", "T*", true) | *ToggleDiv.js*, *ToggleImg.js*, *ToggleLinks.js* |

Explicitly adding each file to a bundle is generally the preferred over wildcard loading of files for the following reasons:

- Adding scripts by wildcard defaults to loading them in alphabetical order, which is typically not what you want. CSS and JavaScript files frequently need to be added in a specific (non-alphabetic) order. You can mitigate this risk by adding a custom IBundleOrderer implementation, but explicitly adding each file is less error prone. For example, you might add new assets to a folder in the future which might require you to modify your IBundleOrderer implementation.

- View specific files added to a directory using wild card loading can be included in all views referencing that bundle. If the view specific script is added to a bundle, you may get a JavaScript error on other views that reference the bundle.

- CSS files that import other files result in the imported files loaded twice. For example, the following code creates a bundle with most of the jQuery UI theme CSS files loaded twice.

  C# Copy

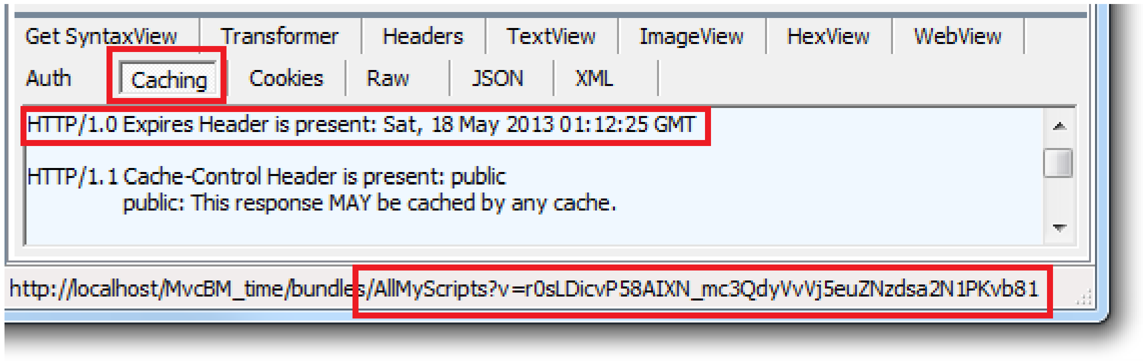
  ```
  bundles.Add(new StyleBundle("~/jQueryUI/themes/baseAll")
      .IncludeDirectory("~/Content/themes/base", "*.css"));
  ```

  The wild card selector "*.css" brings in each CSS file in the folder, including the *Content\themes\base\jquery.ui.all.css* file. The *jquery.ui.all.css* file imports other CSS files.

## Bundle Caching

Bundles set the HTTP Expires Header one year from when the bundle is created. If you navigate to a previously viewed page, Fiddler shows IE does not make a conditional request for the bundle, that is, there are no HTTP GET requests from IE for the bundles and no HTTP 304 responses from the server. You can force IE to make a conditional request for each bundle with the F5 key (resulting in a HTTP 304 response for each bundle). You can force a full refresh by using ^F5 (resulting in a HTTP 200 response for each bundle.)

The following image shows the **Caching** tab of the Fiddler response pane:



```
HTTP/1.0 Expires Header is present: Sat, 18 May 2013 01:12:25 GMT

HTTP/1.1 Cache-Control Header is present: public
        public: This response MAY be cached by any cache.
```

http://localhost/MvcBM_time/bundles/AllMyScripts?v=r0sLDicvP58AIXN_mc3QdyVvVj5euZNzdsa2N1PKvb81

The request

`http://localhost/MvcBM_time/bundles/AllMyScripts?v=r0sLDicvP58AIXN_mc3QdyVvVj5euZNzdsa2N1PKvb81`

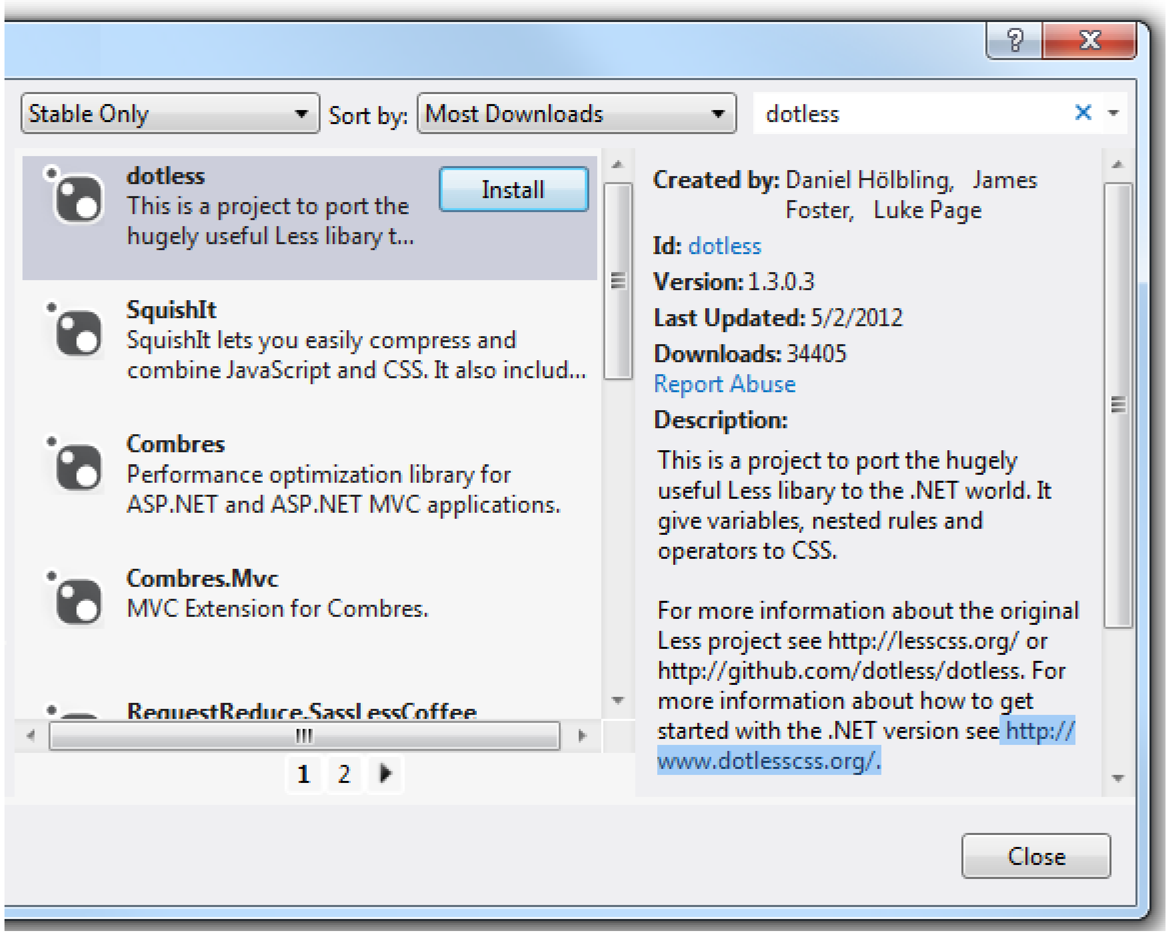
is for the bundle **AllMyScripts** and contains a query string pair **v=r0sLDicvP58AIXN_mc3QdyVvVj5euZNzdsa2N1PKvb81**. The query string **v** has a value token that is a unique identifier used for caching. As long as the bundle doesn't change, the ASP.NET application will request the **AllMyScripts** bundle using this token. If any file in the bundle changes, the ASP.NET optimization framework will generate a new token, guaranteeing that browser requests for the bundle will get the latest bundle.

If you run the IE9 F12 developer tools and navigate to a previously loaded page, IE incorrectly shows conditional GET requests made to each bundle and the server returning HTTP 304. You can read why IE9 has problems determining if a conditional request was made in the blog entry [Using CDNs and Expires to Improve Web Site Performance](https://docs.microsoft.com).

## LESS, CoffeeScript, SCSS, Sass Bundling.

The bundling and minification framework provides a mechanism to process intermediate languages such as SCSS, Sass, LESS or Coffeescript, and apply transforms such as minification to the resulting bundle. For example, to add .less files to your MVC 4 project:

1. Create a folder for your LESS content. The following example uses the *Content\MyLess* folder.
2. Add the .less NuGet package **dotless** to your project.



**dotless**
This is a project to port the hugely useful Less libary t...

**SquishIt**
SquishIt lets you easily compress and combine JavaScript and CSS. It also includ...

**Combres**
Performance optimization library for ASP.NET and ASP.NET MVC applications.

**Combres.Mvc**
MVC Extension for Combres.

**RequestReduce.SassLessCoffee**

**Created by:** Daniel Hölbling, James Foster, Luke Page
**Id:** dotless
**Version:** 1.3.0.3
**Last Updated:** 5/2/2012
**Downloads:** 34405
Report Abuse
**Description:**
This is a project to port the hugely useful Less libary to the .NET world. It give variables, nested rules and operators to CSS.

For more information about the original Less project see http://lesscss.org/ or http://github.com/dotless/dotless. For more information about how to get started with the .NET version see http://www.dotlesscss.org/.

3. Add a class that implements the IBundleTransform interface. For the .less transform, add the following code to your project.

C# Copy

```
using System.Web.Optimization;

public class LessTransform : IBundleTransform
{
    public void Process(BundleContext context, BundleResponse response)
    {
        response.Content = dotless.Core.Less.Parse(response.Content);
        response.ContentType = "text/css";
    }
}
```

4. Create a bundle of LESS files with the `LessTransform` and the [CssMinify]() transform. Add the following code to the `RegisterBundles` method in the *App_Start\BundleConfig.cs* file.

C# Copy

```
var lessBundle = new Bundle("~/My/Less").IncludeDirectory("~/My", "*.less");
lessBundle.Transforms.Add(new LessTransform());
lessBundle.Transforms.Add(new CssMinify());
bundles.Add(lessBundle);
```

5. Add the following code to any views which references the LESS bundle.

CSHTML Copy

```
@Styles.Render("~/My/Less");
```

## Bundle Considerations

A good convention to follow when creating bundles is to include "bundle" as a prefix in the bundle name. This will prevent a possible [routing conflict]().

Once you update one file in a bundle, a new token is generated for the bundle query string parameter and the full bundle must be downloaded the next time a client requests a page containing the bundle. In traditional markup where each asset is listed individually, only the changed file would be downloaded. Assets that change frequently may not be good candidates for bundling.

Bundling and minification primarily improve the first page request load time. Once a webpage has been requested, the browser caches the assets (JavaScript, CSS and images) so bundling and minification won't provide any performance boost when requesting the same page, or pages on the same site requesting the same assets. If you don't set the expires header correctly on your assets, and you don't use bundling and minification, the browsers freshness heuristics will mark the assets stale after a few days and the browser will require a validation request for each asset. In this case, bundling and minification provide a performance increase after the first page request. For details, see the blog Using CDNs and Expires to Improve Web Site Performance.

The browser limitation of six simultaneous connections per each hostname can be mitigated by using a CDN. Because the CDN will have a different hostname than your hosting site, asset requests from the CDN will not count against the six simultaneous connections limit to your hosting environment. A CDN can also provide common package caching and edge caching advantages.

Bundles should be partitioned by pages that need them. For example, the default ASP.NET MVC template for an internet application creates a jQuery Validation bundle separate from jQuery. Because the default views created have no input and do not post values, they don't include the validation bundle.

The `System.Web.Optimization` namespace is implemented in *System.Web.Optimization.dll*. It leverages the WebGrease library (*WebGrease.dll*) for minification capabilities, which in turn uses *Antlr3.Runtime.dll*.

*I use Twitter to make quick posts and share links. My Twitter handle is*: @RickAndMSFT

## Additional resources

- Video:Bundling and Optimizing by Howard Dierking

- Adding Web Optimization to a Web Pages Site.
- Adding Bundling and Minification to Web Forms.
- Performance Implications of Bundling and Minification on Web Browsing by Henrik F Nielsen @frystyk
- Using CDNs and Expires to Improve Web Site Performance by Rick Anderson @RickAndMSFT
- Minimize RTT (round-trip times)

## Contributors

- Hao Kung
- Howard Dierking
- Diana LaRose