



Web Site Troubleshooting Guidance

Learning to identify and resolve compatibility issues like the Internet Explorer® team

Originally Published September 2009

For the latest information, see <http://www.microsoft.com/ie8>

Version 1.0

Abstract

This document provides an understanding of the process Microsoft's Internet Explorer team uses to troubleshoot and resolve application compatibility issues, both for customer facing Web sites and internal line of business applications.

Contents

Learn to Troubleshoot Internet Explorer 8 Compatibility Issues from the Internet Explorer Team	5
Tools Overview	5
Developer Tools	5
Web SuperPreview & Windows Virtual PCs	5
Fiddler	5
The Debugging Process	6
Identify Problems Using Web SuperPreview or Windows Virtual PCs	6
Isolating Problems Using Developer Tools	6
Determining and Setting the Document Compatibility Mode	6
Digging into the Code Using the Developer Tools	7
Reducing the Web Page	7
More Debugging Resources	8
Solving the Problem	8
Common Problems and How to Fix Them... ..	9
Lists or tables appear broken in Internet Explorer 8 but display properly in Internet Explorer 7	9
My menu works in Internet Explorer 7 but breaks in Internet Explorer 8	10
My JavaScript works properly in Internet Explorer 7 but breaks in Internet Explorer 8 (Case 1)	12
My JavaScript works properly in Internet Explorer 7 but breaks in Internet Explorer 8 (Case 2)	14
My pop up window no longer works as expected	16
My site is centered in Internet Explorer 7 but aligned left in Internet Explorer 8	17
When I view my site I'm informed I should update my browser even though I'm using Internet Explorer 8	19
Conclusion	20

The information contained in this document represents the current view of Microsoft Corp. on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This guide is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form, by any means (electronic, mechanical, photocopying, recording or otherwise), or for any purpose, without the express written permission of Microsoft.

Microsoft may have patents, patent applications, trademarks, copyrights or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred.

Microsoft, Internet Explorer, the Internet Explorer logo, SmartScreen, Windows, Windows Server, and Windows Vista are trademarks of the Microsoft group of companies. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

© 2009 Microsoft Corp. All rights reserved.

Learn to Troubleshoot Internet Explorer 8 Compatibility Issues from the Internet Explorer Team

Internet Explorer 8 introduces many new features and capabilities, such as Accelerators and Web Slices for Web developers to build powerful customer experiences. Along with those new features, Internet Explorer 8 includes an entirely new standards based rendering and layout engine. In addition, a new, faster JScript engine was added to Internet Explorer 8. While this new platform provides Web developers the ability to ‘write to standards’ and have their code work in all major browsers, the changes can cause issues with previous Internet Explorer versions that are still in use today.

The focus of this document is to provide a few ‘real world’ examples (without site or company attribution) of issues the Internet Explorer team has run into helping customers fix issues with their Web sites. Instead of simply providing best practice guidance (code to standards!) or a list of issues to avoid, this document is designed to give you an understanding of the process the Internet Explorer team themselves use internally at Microsoft to troubleshoot and resolve application compatibility issues, both for customer Web sites and internet line of business applications.

Tools Overview

Throughout this document, we will use a set of tools to help simplify the debugging process. Below is a quick overview of some of the tools.

Developer Tools

Internet Explorer 8 comes equipped with a set of developer tools which empower you to quickly inspect your site’s HTML, Cascading Style Sheets (CSS), JavaScript, and Document Object Model (DOM) tree as they exist in Internet Explorer. This is especially helpful when debugging dynamic sites such as those which utilize frameworks.

Accessing the developer tools is simple: press F12 while viewing a page or select **Developer Tools** from the **Tools** menu.

Web SuperPreview & Windows Virtual PCs

Web SuperPreview is a visual debugging tool which helps you migrate web sites from earlier versions of Internet Explorer. Through the use of side by side comparisons or an onion-skin overlay, you can quickly identify differences in a layout rendered in various versions of Internet Explorer by using rulers, guides, and zoom/pan tools. You can obtain Web SuperPreview at the [Microsoft Download Center](#).

Alternatively, Windows Virtual PCs can be used to emulate different environments within Windows. Using multiple Windows Virtual PCs, you can view side by side how a website is rendered within Windows XP, Vista, or other installations running Internet Explorer 6, 7, and 8 without the use of multiple machines. Visit the [Windows Virtual PC website](#) to obtain a copy.

Fiddler

Fiddler is a useful tool often used by the Internet Explorer team which captures traffic between the Internet and your computer. It allows for inspection of both incoming and outgoing data and can be used to monitor and “fiddle” with requests and responses before they are received by the browser. Built by a Microsoft developer, Fiddler includes a powerful event-based scripting subsystem and can be extended using any .NET language.

Fiddler is available for free as an [Internet Explorer add-on](#).

The Debugging Process

Identify Problems Using Web SuperPreview or Windows Virtual PCs

One of the quickest ways you can identify potential problems in a web page is using Web SuperPreview or Windows Virtual PCs. By opening any site and viewing side by side comparisons or overlaying one page over the other in Web SuperPreview, you can visually isolate any problem areas which exist. This is helpful when you are creating new web pages and would like to ensure they are compatible in all versions of Internet Explorer or when testing your current web pages for compatibility with Internet Explorer 8.

Isolating Problems Using Developer Tools

Once a problem has been identified on a page the Developer Tools can be used to gather more information and reduce the problem size.

Determining and Setting the Document Compatibility Mode

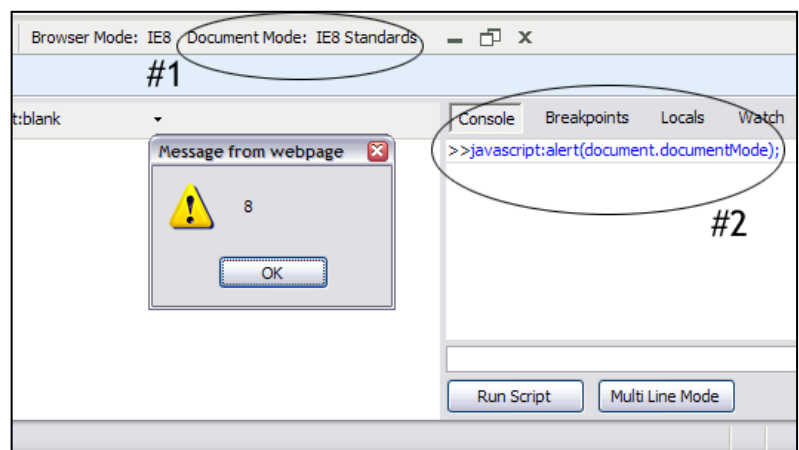
The first step when debugging any site in Internet Explorer 8 is identifying which document compatibility mode the page is rendered in. Document compatibility modes define how Internet Explorer should render the page and can be toggled by developers using an X-UA-Compatible header. Knowing the document compatibility mode informs us what set of rules Internet Explorer is abiding by and helps us to identify the cause of certain problems.

Internet Explorer 8 supports the following document compatibility modes: **Emulate IE8**, **Emulate IE7**, **IE5**, **IE7**, **IE8**, and **Edge**. The two “Emulate” modes use a page’s DOCTYPE directive to determine how a page should be rendered. If a standard mode DOCTYPE is present, the page will be rendered using the mode specified (**IE8** mode for **Emulate IE8** or **IE7** mode for **Emulate IE7**). If a quirks mode DOCTYPE directive is present, the page will be rendered in **IE5** mode for both “Emulate” modes. **IE5**, **IE7**, and **IE8** modes, however, will force the browser to render the specified mode regardless of the DOCTYPE directive. Lastly, **Edge** mode will force Internet Explorer to use the highest support for standards available. In most cases, users are strongly encouraged to use an “Emulate” document compatibility mode such as **Emulate IE8** and **Emulate IE7** as these modes will not force pages normally rendered in quirks mode (such as framesets) into standards mode providing more comprehensive compatibility.

For more information regarding how document compatibility modes may affect your site and how you can implement them using an X-UA-Compatible header, you can visit MSDN’s [Defining Document Compatibility](#) page or view available videos on the [“How Do I” Videos —Internet Explorer 8](#) page.

Using the Developer Tools, we can quickly determine the document compatibility mode of a page using one of the following two methods:


1. Inspect the **Document Mode** option in the Developer Tools toolbar
2. Click the **Script** tab and in the **Console** pane type `javascript:alert(document.documentMode)` in the script text box and click **Run Script**. An alert will appear notifying what document mode the page is in.



While the **Document Mode** option informs us what document compatibility mode the page is finally rendered in, the **Browser Mode** details which browser version Internet Explorer is emulating. Setting the **Browser Mode** to **IE7** would define the user agent string (as Internet Explorer 7 in this case) to both the server and within the local script. Declaring the **Emulate IE7** document compatibility mode through an X-UA-Compatible header only affects the document mode, so the user agent string remains unchanged, meaning any local scripts would be presented the Internet Explorer 8 UA value. Using the Developer Tools, developers can temporarily toggle different Browser and Document Modes and view their effects on their pages.

Digging into the Code Using the Developer Tools

Some problems can be found and solved using the Developer Tools without the need of extensive updates to the original document.

You can use the Developer Tool's **Select Element by Click** (Ctrl + B within the dev tools)  option to help locate problems within the HTML markup and then dynamically update the page within the browser. By inspecting the HTML markup within the Developer Tools you can see how it exists within Internet Explorer which often gives insight into why a part of the page does not render or behave correctly. The Developer Tools allow real time updates to text, styles, and attributes letting you view the changes without leaving the browser.

The Developer Tools can also be used to debug scripts by enabling you to insert breakpoints, step through code, view locals, and add watch variables.

For a detailed guide on all the features within the Developer Tools and how to utilize them, visit [Discovering Internet Explorer Developer Tools](#).

Reducing the Web Page

If the problem cannot be solved strictly using the Developer Tools it is often useful to reduce the web page. Reducing a web page removes as much unnecessary code as possible while still reproducing the problem. This process helps isolate where the problem occurs within the code, thus making the debugging process simpler. The steps below summarize the actions needed to reduce a web page:

1. Create a local copy of the web page
2. Strip away any parts of the web page that do not affect the problem
3. Continue Step 2 until the web page contains minimal lines of code and no (or few) linked files while still reproducing the original problem.

If you do not have access to the original files, you can save a local copy of the web page using Internet Explorer 8:


1. Navigate to the page and wait for it to load completely
2. Click **Page > Save As...**
3. Change the **Save as type** to **Webpage, complete (*.htm;*.html)**
4. Choose a location and file name
5. Click the **Save** button

Sometimes when saving a copy of a web page all the necessary resources are not saved correctly. In this case and others when you'd like to minimize the amount of local resources saved, you can use Fiddler to create a local copy of the web page and replace the remote request for the page with your local copy. This allows you to only save a copy of the files you need to edit without worrying about dependent resources.

To replace a remote request using Fiddler:

1. Start Fiddler and Internet Explorer 8. Once both programs open, navigate to the page you want to reduce.
2. Fiddler will log all requests and responses, so within the Fiddler window find and highlight the webpage URL that you want to reduce or use **Find Sessions...** (Ctrl + F).
3. Right click on the correct row, select **Save > Response > Response Body ...**, and save the file locally.
4. In order for Fiddler to replace the requested copy with the local copy when the URL is accessed from Internet Explorer 8, you must create a rule. Click on the **AutoResponder** tab to access the current rules set for the URL row that you selected.
5. Click the **Add...** button and in the **Rule Editor** section at the bottom, click on the second dropdown menu and select **Find a file....**

6. Select your local copy of the page and click the **Save** button to save the rule.

Once a local copy of the web page has been created, you can again use the Developer Tool's **Select Element by Click** (Ctrl + B)  option to help locate the problem in the code. Then, using your favorite HTML editor begin to remove unnecessary HTML from the **<body>** section as well as any extraneous **<script>**, **<meta>**, and **<link>** tags in the **<head>** section. It is often best to start at the top most parent node of the element in question and remove code above and below it. Be sure to constantly save after each reduction step and verify that the problem still exists. If at any time the problem disappears, undo the last change and either remove a smaller section of code or begin to reduce a different part of the page.

Below are some helpful guidelines while reducing a web page:

- Many HTML editors such as [Expression Web](#) and [Visual Studio](#) have features such as HTML tidying which reformat the code, making it easier to read.
- Always remove matching start and end tags, ensuring that the page's hierarchy remains intact.
- Be sure to remove any unneeded properties within tags by reducing them to their base element. For example, reduce `<tr style="padding: 25px;" id="elemID" >` to just `<tr>`.
- If any `<script>` or `<link>` tags have source files which are needed to reproduce the problem, copy their source into the web page and continue the reduction process. Many bugs are caused by CSS or scripting and reducing these files as far as possible can help isolate the root cause of the problem.

More Debugging Resources

For more information regarding designing, testing, and debugging for Internet Explorer 8 including the use of Developer Tools and web page reduction see the [Internet Explorer 8 Compatibility Test Guide](#).

Solving the Problem

Once you have located the source of the problem using the Developer Tools and by reducing the page, inspect the code and try to determine the cause of the problem. Many resources exist such as [Site Compatibility and Internet Explorer 8](#) which detail common compatibility problems and how to resolve them. Often times you'll find the source of your problem can be traced to one of these common issues.

Common Problems and How to Fix Them...

Now that we have touched on debugging and some of the useful tools available to help throughout the process, we can begin to look at some common problems and the steps you can take to determine their root cause using these debugging methods.

Lists or tables appear broken in Internet Explorer 8 but display properly in Internet Explorer 7

The Problem

You are maintaining a web page which contains an unordered list that appears broken in Internet Explorer 8 but renders perfectly in Internet Explorer 7.




Internet Explorer 7 List displays correctly



Internet Explorer 8 List displays incorrectly

Debugging Process

1. Open the web page in Internet Explorer 8.
2. Open the **Developer Tools** by pressing **F12** or select **Developer Tools** from the **Tools** menu to begin checking your code.
3. Switch the **Document Mode** to **Internet Explorer 7 Standards** and ensure the list renders properly. If it does, you can force the browser into Internet Explorer 7 Standards mode as a temporary fix while updating your site (see **Determining and Setting the Document Compatibility Mode** for details)
4. Improperly displayed lists and tables are often caused by extraneous or missing tags, leading to malformed HTML. Using the **Developer Tools**, you can quickly view how the code exists within Internet Explorer and check for any malformed HTML. Inside the **HTML** tab click the **Select Element by Click** button (Ctrl + B)  and click the list item which is rendered improperly.
5. After you have clicked on the broken list item, you will be able to view it as it appears in the Document Object Model (DOM). Ensure it and all elements around it have matching closing/ending tags and no extraneous tags are found.
6. If you are unable to find the offending piece of code, create a copy of the code and begin to reduce the page (see **Reducing the Web Page** for details) in order to help isolate the problem. Since the problem resides in an unordered list, your focus should be on removing all elements outside the list while still being able to reproduce the problem. Begin by eliminating large chunks of unneeded code within the **<body>** section such as headers, footers, and extraneous content around the list.
7. Next, remove all inline and linked JavaScript. If at any point you are unable to reproduce the problem, copy all JavaScript into the page and begin to reduce the scripts to isolate any scripting which affects the unordered list.

8. Lastly, copy the styles contained within the all linked CSS files into the page.
Eliminate all style declarations which do not affect the unordered list.

Once the page has been fully reduced, it should be easier to spot any malformed HTML or other cause for the broken list. Looking at the code snippet to the right, you will notice the extraneous tag causing the list to appear improperly in Internet Explorer 8. By simply removing the extra tag, the list renders properly.

```
<ul>
  <li>1.1
    <ul>
      <li>1.1.1</li>
    </ul>
  </li><!-- Unnecessary Tag -->
  <li>1.1.2</li>
</ul>
```

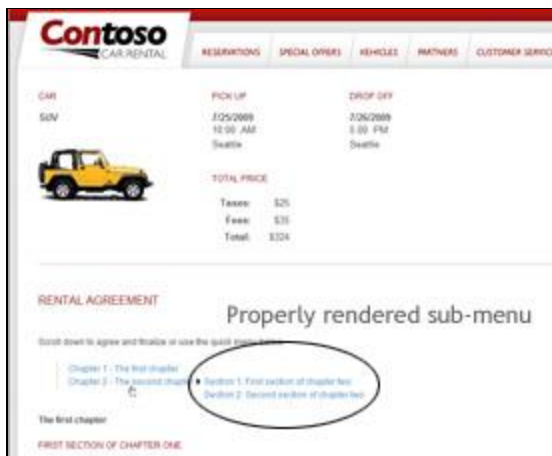
More Information

Strict interpretation of markup in Internet Explorer 8 makes the browser more standards compliant ([Site Compatibility and Internet Explorer 8](#)). If you have code (such as an unordered list or table) that is not rendering properly you will want to make sure that your code is valid and marked up properly. This includes checking for extra or missing tags that might cause rendering issues within your page.

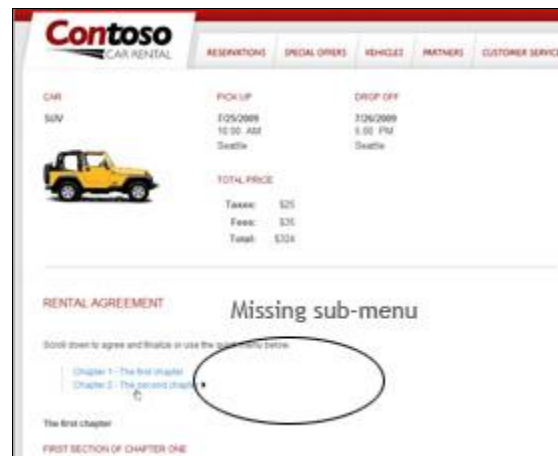
My menu works in Internet Explorer 7 but breaks in Internet Explorer 8

The Problem

A rental agreement page implements an [ASP.net menu control](#) to provide quick navigation to each chapter and section of the agreement document. The menu works perfectly fine in Internet Explorer 7, but when viewed in Internet Explorer 8 the secondary menu items do not display.




Internet Explorer 7 Working Menu



Internet Explorer 8 Non-Working Menu

Debugging Process

1. Open the web page in Internet Explorer 8.
2. Open the **Developer Tools** by pressing **F12** or select **Developer Tools** from the **Tools** menu to begin checking your code.
3. Switch the **Document Mode** to **Internet Explorer 7 Standards** and ensure the menu works properly. If it does, you can force the browser into Internet Explorer 7 Standards mode as a temporary fix while updating your site (see **Determining and Setting the Document Compatibility Mode** for details)
4. Because the secondary menu simply does not display in Internet Explorer 8, the cause of the problem can be due to HTML, CSS, JavaScript or any combination thereof. To begin, it is always easiest to inspect the code as it rendered in Internet Explorer 8. Using the **Select Element by Click** (Ctrl + B)  option inside the **HTML** tab of the **Developer Tools**, click the menu to view it within the Document Object Model (DOM).

5. At this point, you would normally inspect the code to ensure the problem is not caused by malformed HTML or any other obvious problem (see **Lists or tables appear broken in Internet Explorer 8 but display properly in Internet Explorer 7** for details). This cause is less likely, however, with an ASP.net menu control since its HTML, CSS, and JavaScript is automatically generated for you.
6. Unable to locate any obvious problem with the code, the next step is to create a copy of the code and begin to reduce the page (see **Reducing the Web Page** for details). This will limit the scope of the problem and help simplify the debugging process. Begin reducing the page by removing as much code around the top node of the menu control as possible. This involves removing headers, footers, and any other content around menu control which does not prevent you from reproducing the problem. Along the way, switch between **Internet Explorer 7 Standards** and **Internet Explorer 8 Standards** document modes to ensure the menu still displays properly in Internet Explorer 7 which will help prevent the introduction of a new bug while fixing the current problem.
7. Next, copy any linked JavaScript code into the page. Begin to eliminate any JavaScript which does not affect the menu. Again, be careful to ensure at each step the menu still renders properly in **Internet Explorer 7 Standards** mode as JavaScript is often used to control the behavior of menus.
8. Finally copy all CSS styles contained within linked CSS files into the page. Eliminate all styles which do not affect the menu. Be sure to watch for any styles which set the z-index of elements, as incorrectly set z-index values can cause elements to be hidden behind other elements on a page.

After reducing the web page, you will be left with an ASP.NET page containing only the menu control with no additional scripts or styles except those generated by .NET. If no immediate problems are apparent, it is advised that you thoroughly research specific ASP.NET menu control compatibility issues before attempting to reduce the code generated by the menu control.



Reduced Web Page

Reduced Source Code

More Information

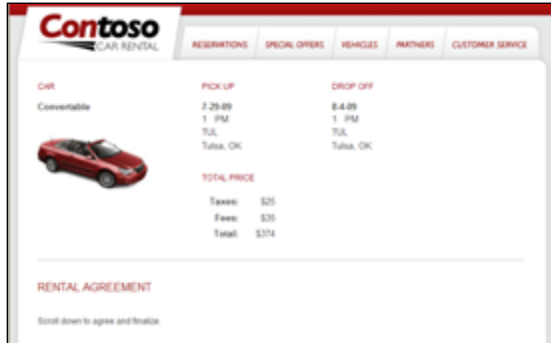
A quick online search will show that this is a known compatibility issue caused by how Internet Explorer 8's currentStyle object returns unset properties (see [Site Compatibility and Internet Explorer 8](#)). For detailed instructions on how to remedy the problem see Giorgio Sardo's blog entry [ASP.NET Menu and Internet Explorer 8 rendering white issue](#).

My JavaScript works properly in Internet Explorer 7 but breaks in Internet Explorer 8 (Case 1)

The Problem

You are maintaining a summary page for a car rental site which displays order information and an image which is updated via JavaScript based on the user's preferences. The image is updated properly in Internet Explorer 7 but it is blank in Internet Explorer 8 and generates the following JavaScript error:

Error: 'null' is null or not an object




Internet Explorer 7 Properly Updates the Image

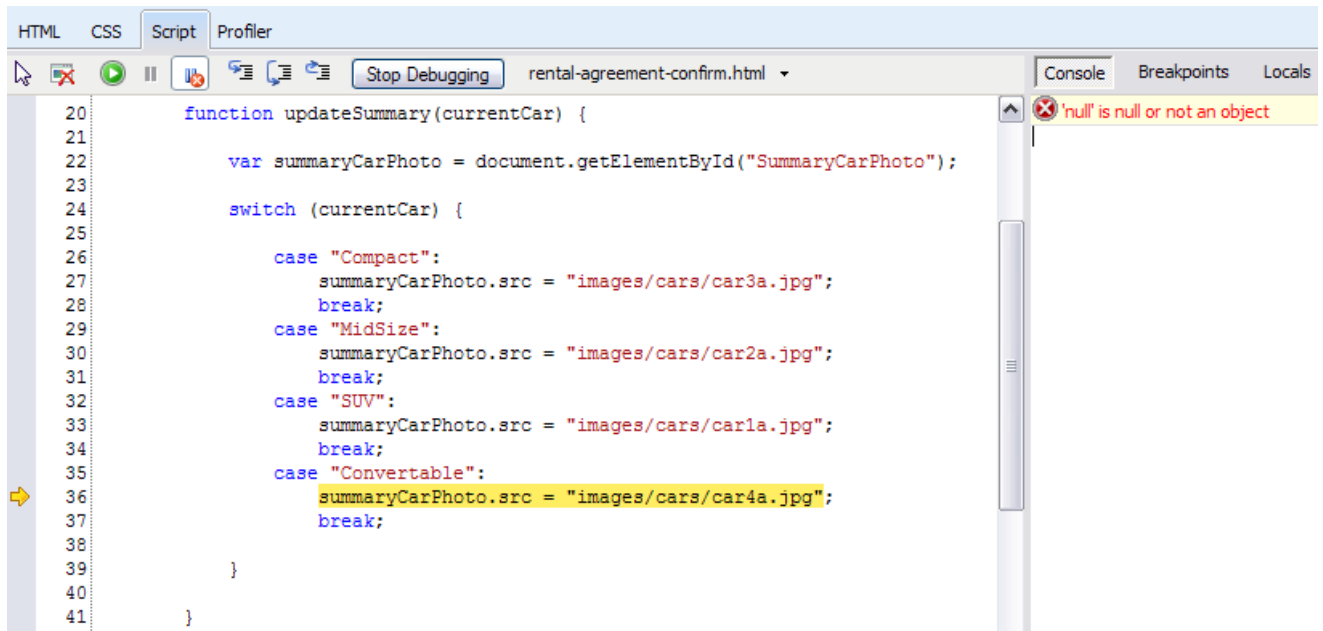


Internet Explorer 8 Displays a Blank Image & JavaScript Error

Debugging Process

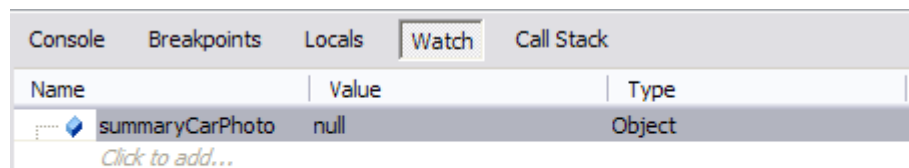
At times JavaScript errors can be difficult to debug. To help ease the process, you can use the Developer Tools to track down the source of the JavaScript error. Once you locate where the code breaks, you can begin to reduce the source code to determine the root cause of the error.

1. Open the web page in Internet Explorer 8.
2. Open the **Developer Tools** by pressing **F12** or select **Developer Tools** from the **Tools** menu to begin checking your code.
3. Switch the **Document Mode** to **Internet Explorer 7 Standards** and check to see if the image is displayed properly. If it is, you can force the browser into **Internet Explorer 7 Standards** mode as a temporary fix while updating your site (see **Determining and Setting the Document Compatibility Mode** for details)
4. Click the **Script** tab and ensure the **Break on Error** (Ctrl + Shift + E)  button is pressed.
5. Press the **Start Debugging** button to begin debugging your JavaScript. By enabling **Break on Error** and entering **Debugging** mode, the page will now halt processing of the page when a JavaScript error is encountered. The Developer Tools will also highlight the line of code producing the error while giving you access to the current state of the script. From here, you can implement breakpoints and use locals and watch variables to further inspect the script.
6. After pressing the **Start Debugging** button, the page will refresh and halt on the line of code producing the JavaScript error as seen below.



The line `summaryCarPhoto.src = "images/cars/car4a.jpg"` is producing the JavaScript error.

- You can further investigate the offending line of code by inspecting the **summaryCarPhoto** object which is being updated. Click the **Watch** button in the right hand side of the Developer Tools and type in `summaryCarPhoto` where it says **Click to add...** Its



value is null so it appears the car image element is never found on the page. Reducing the page will help simplify the process of determining why the element is not found.

- Create a copy of the code and begin to reduce the page (see **Reducing the Web Page** for details) by isolating the elements which the JavaScript interacts with and removing any extraneous JavaScript, CSS, and HTML. In this case, only the JavaScript which updates the car photo and the image itself are necessary to reproduce the problem. At each step of the reduction process, verify the page still works in **Internet Explorer 7 Standards** mode ensuring no additional errors have been introduced.

Once a page has been reduced it's easier to locate potential problems such as the one shown in the code to the right. By inspection, you'll notice the JavaScript is using a method which searches for the car image by a given id but the image element does not contain an id.

```

17 function updateSummary(currentCar) {
18
19     var summaryCarPhoto = document.getElementById("SummaryCarPhoto");
20
21     switch (currentCar) {
22
23         case "Compact":
24             summaryCarPhoto.src = "images/cars/car3a.jpg";
25             break;
26         case "MidSize":
27             summaryCarPhoto.src = "images/cars/car2a.jpg";
28             break;
29         case "SUV":
30             summaryCarPhoto.src = "images/cars/car1a.jpg";
31             break;
32         case "Convertible":
33             summaryCarPhoto.src = "images/cars/car4a.jpg";
34             break;
35     }
36 }
37
38 </script>
39 </head>
40
41 <body>
42 
43 </body>
44 </html>

```

More Information

After looking through common differences between Internet Explorer 8 and Internet Explorer 7 on the [Site Compatibility and Internet Explorer 8](#) page, you will notice the `getElementById()` method searched through name attributes as well as Ids in Internet Explorer 7 but only searches by Id in Internet Explorer 8. By adding an Id attribute to the image element the page will become compatible with Internet Explorer 8 and be more standards compliant.

My JavaScript works properly in Internet Explorer 7 but breaks in Internet Explorer 8 (Case 2)

The Problem


You maintain a car rental site which retrieves available vehicle information based on user entered preferences. It retrieves the vehicle information by using a remote service which accepts the user preferences as a JSON encoded string. When viewed in Internet Explorer 7 the form works as expected but when viewed in Internet Explorer 8 it encounters the following obscure JavaScript error:

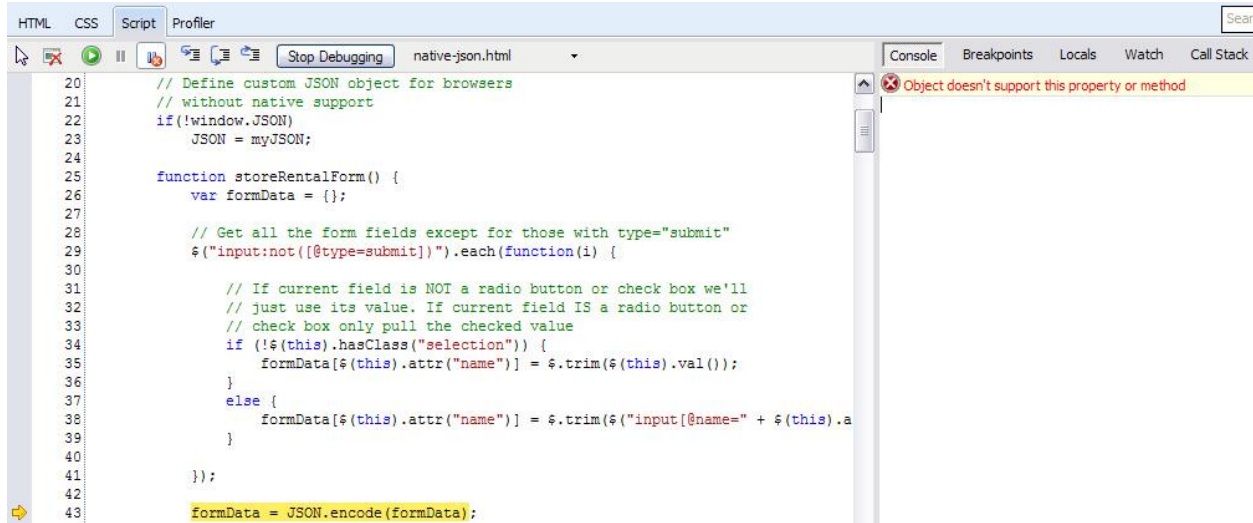
Error: Object doesn't support this property or method

Debugging Process

As in the previous example, you can use the Developer Tools to track down the source of the JavaScript error. Once you find where the code breaks, you can begin to reduce the source code to determine the root cause of the error.



1. Open the web page in Internet Explorer 8.
2. Open the **Developer Tools** by pressing **F12** or select **Developer Tools** from the **Tools** menu to begin checking your code.
3. Click the **Script** tab and ensure the **Break on Error** (Ctrl + Shift + E)  button is pressed.
4. Press the **Start Debugging** button to begin debugging your JavaScript. By enabling **Break on Error** and entering **Debugging** mode, the page will now halt processing of the page when a JavaScript error is encountered. The Developer Tools will also highlight the line of code producing the error while giving you access to the current state of the script. From here, you can implement breakpoints and use locals and watch variables to further inspect the script.
5. After you press the **Start Debugging** button, the page will refresh and halt on the line of code producing the JavaScript error as seen above below.



The line `formData = JSON.encode(formData)` appears to be causing the problem. For some reason, the encode method for the JSON object is producing an error. This method is used to take a JavaScript object and convert it to a JSON string to be saved and then sent to the remote service.

6. You can further inspect the JSON object by clicking the **Watch** button in the right hand side of the Developer Tools and typing in `JSON.encode` where it says **Click to add...** When you try to inspect the `JSON.encode` method you will see that it is not defined, which would produce an error when the script tries to invoke the method.



7. To help determine why this method is not available when viewing the site in Internet Explorer 8, you can reduce the code (see **Reducing the Web Page** for details). When reducing a page to debug a JavaScript error it's often best to focus on removing extraneous JavaScript, keeping only the code which is used by the line causing the error. Since the code in question does not manipulate CSS or XHTML, you can safely disregard those sections of code and focus solely on the page's JavaScript.

8. After reducing the page you are left with the code to the right. Here you can see that the JSON object is being assigned a custom object called `myJSON`. This is often done when implementing JSON in browsers which do not natively support JSON (such as Internet Explorer 7). This code appears to be checking for native support and defining a custom JSON object if it is not found.

```

1<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
2<html xmlns="http://www.w3.org/1999/xhtml">
3<head>
4  <title>Contoso Car Rental</title>
5  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6
7  <script src="resources/js/myJSON.js" type="text/javascript"></script>
8  <script type="text/javascript">
9
10     // Define custom JSON object for browsers
11     // without native support
12     if(!window.JSON)
13         JSON = myJSON;
14
15     function storeRentalForm() {
16         var formData = {};
17
18         formData = JSON.encode(formData);
19     }
20
21 </script>
22</head>

```

- Using the Developer Tools, you can inspect the myJSON object using the Watch panel. As you may notice, this object does include an encode method. However, since Internet Explorer 8 natively supports JSON, the myJSON object is never assigned and the encode method is not available to the JSON object attempting to use it.

Console	Breakpoints	Locals	Watch	Call Stack
Name	Value	Type		
myJSON	{...}	Object		
[Methods]				
encode()	encode(value, replacer, spa...	Variant		
parse()	parse(text, reviver)	Variant		
Click to add...				

More Information

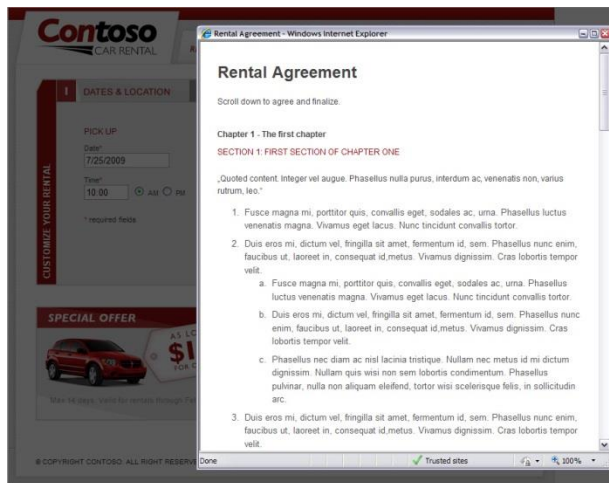
After further investigation on the [Site Compatibility and Internet Explorer 8](#) page, you will discover that this is a common problem encountered by developers migrating sites to Internet Explorer 8. The page contains further information on the root cause of the problem as well as a solution.

My pop up window no longer works as expected

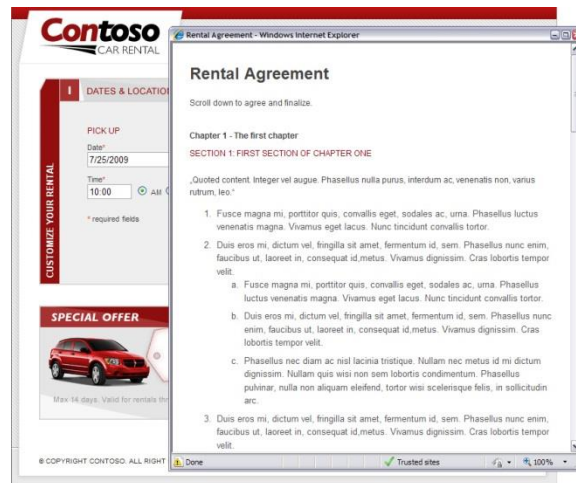
The Problem

You maintain a car rental site which pops up a rental agreement form that a user must read through before continuing on the site. The pop up window shades the main page until the user clicks an agreement button at the bottom of the form. After some recent updates and site maintenance the main page is no longer shaded when the form is opened and the following JavaScript error is reported:

Error: Access is denied.




The working page



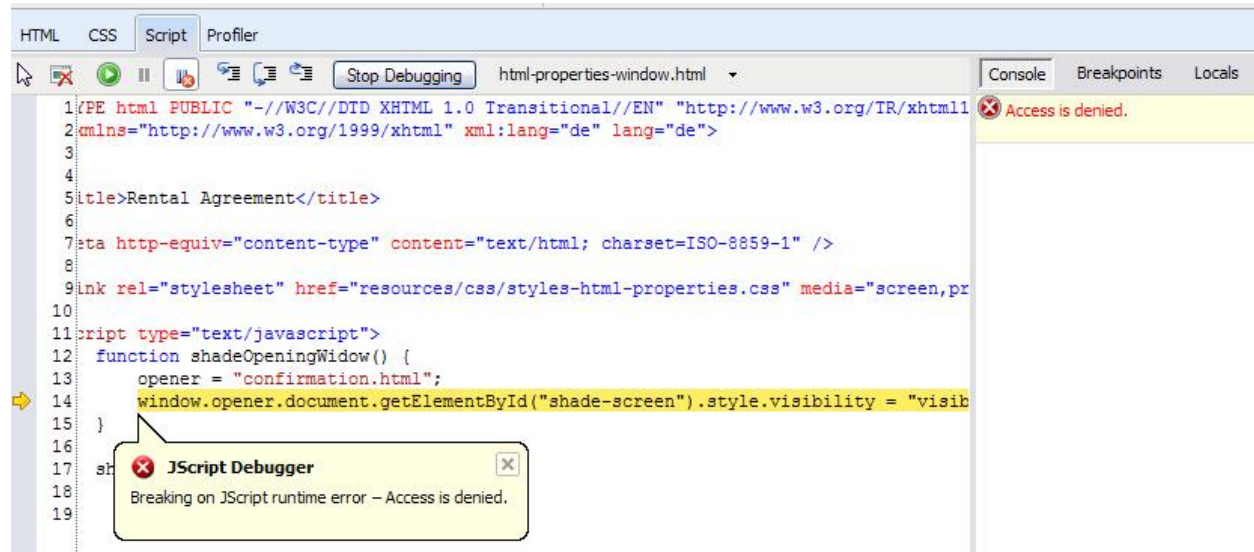
Main page is no longer shaded, error reported

Debugging Process

- Open the web page in Internet Explorer 8.
- Open the **Developer Tools** by pressing **F12** or select **Developer Tools** from the **Tools** menu to begin checking your code.
- Click the **Script** tab and ensure the **Break on Error** (Ctrl + Shift + E)  button is pressed.
- Press the **Start Debugging** button to begin debugging your JavaScript. By enabling **Break on Error** and entering **Debugging** mode, the page will now halt processing of the page when a JavaScript error is encountered. The Developer Tools will also

highlight the line of code producing the error while giving you access to the current state of the script. From here, you can implement breakpoints and use locals and watch variables to further inspect the script.

- After you press the **Start Debugging** button, the pop up refreshes and now breaks at the line of code where the error is encountered as seen below. It appears that access is being denied on the line of code which sets the shade-screen element of the main page to visible. Although the bug is not immediately apparent, this gives you a good idea of what code should be isolated when reducing the page.



- Create a copy of the code and begin to reduce the page (see **Reducing the Web Page** for details). Your focus should be to isolate the section of code producing the JavaScript error as much as possible. Remove any extraneous JavaScript, CSS, and HTML which does not prevent you from reproducing the problem while still having a working page in **Internet Explorer 7 Standards** mode.

While reducing the page you'll notice if you remove `opener = "confirmation.html";` above the offending line of code the pop up works as expected again. At this point you know this line was the source of the "Access is denied" JavaScript error.

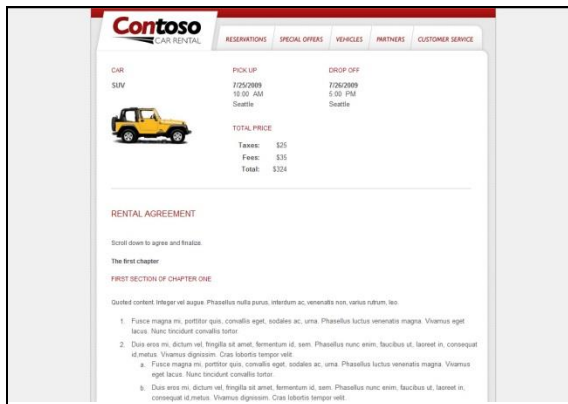
More Information

While searching online for "window.opener access is denied" you will discover that others have also experienced this issue. The `opener` variable can be used in the script but must be preceded by a `var` declaration to prevent Internet Explorer 8 from restricting access to the `window.opener` property.

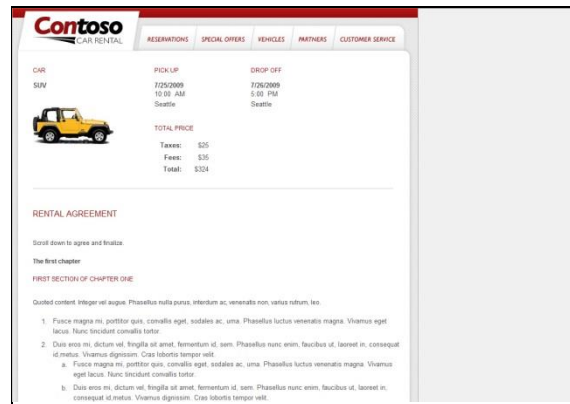
My site is centered in Internet Explorer 7 but aligned left in Internet Explorer 8

The Problem

You maintain a site that is properly centered in the page when viewed in Internet Explorer 7 but improperly aligns left when viewed in Internet Explorer 8.




Internet Explorer 7 Properly Centered Page

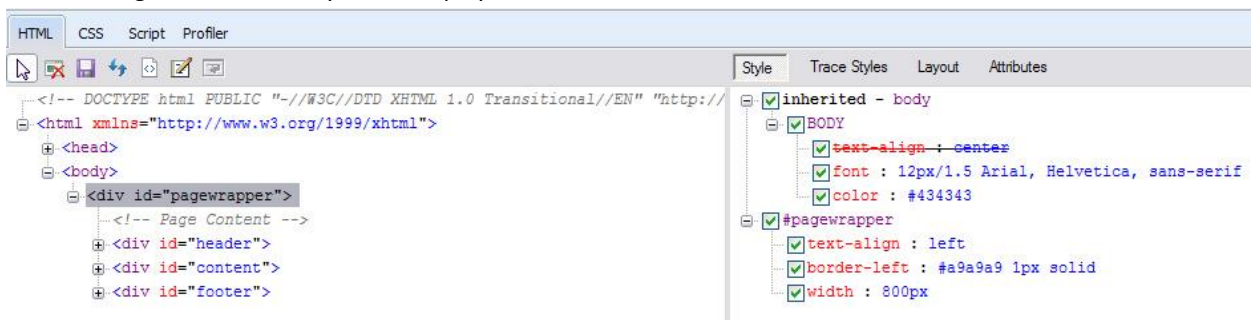


Internet Explorer 8 Improperly Aligned Left Page

Debugging Process

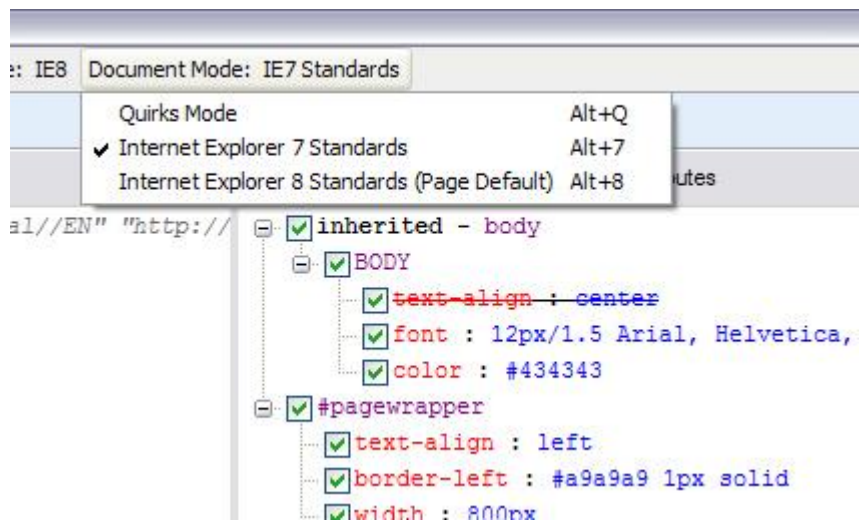
Layout issues are most commonly caused by a page's CSS or XHTML. Since the page renders properly in Internet Explorer 7, we can first force the browser into Internet Explorer 7 Standards mode as a temporary fix while updating the site (see **Determining and Setting the Document Compatibility Mode** for details). In order to find the root cause of the issue and produce a solution for Internet Explorer 8, we'll need to further inspect the page using the Developer Tools.

1. Open the web page in Internet Explorer 8.
2. Often pages are "wrapped" by an element which is used to position the page in the browser. To find and inspect this element using the Developer Tools, we can use the **Select Element by Click** (Ctrl + B)  option. By hovering over the page we can find the element which encloses the entire page. Since the whole page is being shifted left and is enclosed by this element, it provides us with the best place to start the debugging process.
3. After clicking on the element you are displayed the information below.



In the left pane you can view the Document Object Model (DOM) as it is in Internet Explorer. On first inspection, there doesn't appear to be any malformed XHTML which would cause the layout to break in Internet Explorer 8. In the right pane we can view CSS style information which is inherited or set by the **pagewrapper** id. The **pagewrapper** id itself doesn't appear to implement any positioning styles.

4. Since the root cause of the problem was not found through inspection, the next step is to focus on reducing the page (see **Reducing the Web Page** for details) in order to track down the cause of the problem. Begin by using the Developer Tools to set the Document Mode to **Internet Explorer**



7 Standards mode so you can view the site as it is rendered by Internet Explorer 7.

5. Next, begin to reduce to the code as far as you can without causing the layout to break in **Internet Explorer 7 Standards** mode. This will allow you to identify the section of code which causes the layout to center in Internet Explorer 7.
6. Start by removing all the elements within the top node you are concerned with (in this case the **pagewrapper** div). This involves deleting the **header**, **content**, and **footer** divs and the elements contained within them. You can replace these elements with some dummy text in order to see whether or not the **pagewrapper** div is still centered on the page.
7. Next, remove all inline and linked JavaScript. Since the problem you are attempting to resolve appears to be a layout problem and not a behavioral problem, you can be fairly certain the problem is not caused by JavaScript. This step, however, will allow you to eliminate that possibility entirely.
8. Finally, copy the styles contained within the linked CSS files into the page itself. You can eliminate all style declarations which do not affect the page's centered layout. If at any point the page no longer centers, you know we have found the line of code we are concerned with.

After reducing the web page, you are left with the code featured to the right. Whenever you remove the `text-align: center;` statement in the body style the **pagewrapper** div no longer centers on the page. You'll need to determine why this line of code no longer works in Internet Explorer 8 in order to find a solution to the problem.

More Information

As it turns out, this is an example of improved standards support by Internet Explorer 8. An online search of "Internet Explorer 8 text align center" shows many people have run into the same issue. The long term solution would be to implement a more standard approach to centering the page, such as applying `margin: 0 auto;` on the **pagewrapper** div.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Contoso Car Rental - Rental Agreement</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=8" />
<style type="text/css">
body {
text-align: center;
}

#pagewrapper {
width: 800px;
border: 1px solid gray;
}
</style>
</head>
<body>
<div id="pagewrapper">
Content
</div>
</body>
</html>
```

When I view my site I'm informed I should update my browser even though I'm using Internet Explorer 8

The Problem

Your site contains logic to determine the version of Internet Explorer visitors are running. The logic should detect older versions of Internet Explorer and inform users they should consider upgrading, but it is mistakenly informing users running Internet Explorer 8 they should upgrade.

Debugging Process

1. Open the web page in Internet Explorer 8.
2. Open the **Developer Tools** by pressing **F12** or select **Developer Tools** from the **Tools** menu to begin checking your code.
3. Determine what **Document Mode** you are running in (see **Determining and Setting the Document Compatibility Mode**). Ensure you are running in **Internet Explorer 8 Standards** mode. Running in other document modes may cause problems with certain browser detection methods.

4. If you are running in the proper document mode, the next step is to isolate the browser detection code to simplify the debugging process. Create a copy of the code and begin to reduce the page (see **Reducing the Web Page** for details).
5. Detection logic usually resides in the **<head>** section of the page in the form of JavaScript or conditional comments. Be sure to remove as much HTML as possible and copy any linked JavaScript and CSS files into the page. As this is a behavioral problem, you should be able to remove all CSS styles. Reduce the JavaScript code as far as you can while still reproducing the problem.

After reducing the code, you come across the following conditional expression:

```
<!--[if !(IE 7)]>
<script type="javascript">
    alertUser("You are running an older version of Internet Explorer and should
        consider upgrading.");
</script>
-->
```

By looking at the detection logic above, it's clear only Internet Explorer 7 will be detected as a supported version of Internet Explorer. By revising this logic to check for all versions less than the supported version (Internet Explorer 7), compatibility with future versions of Internet Explorer is guaranteed.

It's important to note that the **Browser Mode** can affect how Internet Explorer 8 is detected as well. For instance, when running in **Compatibility Mode** or **IE7 Browser Mode**, Internet Explorer 8 will be detected as Internet Explorer 7 which can also lead to undesired behavior with proper conditional comments. Be sure to check that your script or server is not setting a document compatibility mode which forces Internet Explorer 8 into a different browser mode (see **Determining and Setting the Document Compatibility Mode**).

To detect users who have chosen to run in **Compatibility Mode**, developers can look for the "Trident/4.0" token in Internet Explorer 8's User-Agent string.

More Information

For more information on better browser detection methods visit [Detecting Internet Explorer More Effectively](#). More information about Conditional Comments can be found in this [MSDN document](#). Also, details about the Trident/4.0 User-Agent String can be found at the following [Internet Explorer Blog entry](#). For an example on how to detect the Trident/4.0 token using JavaScript, see Giorgio Sardo's blog entry titled [How to detect Internet Explorer 8 using JavaScript \(client side\)](#).

Conclusion

After reading through this document you should feel more comfortable debugging web pages using some of the same tools and methodologies used by the Internet Explorer team.

Although we could not cover all compatibility issues that you may come across, you now have the knowledge and experience to identify compatibility problems, isolate them by reducing the page, and remedy the issue through research of available documentation.