Internet Explorer 9 Compatibility Cookbook

.NET Framework 3.0

Purpose

The *Internet Explorer 9 Compatibility Cookbook* is designed to help you understand changes in Windows Internet Explorer 9 that might impact applications that you developed for earlier versions of Windows Internet Explorer. Many changes help Internet Explorer comply with broader industry standards, and other changes improve performance and reliability.

The *Internet Explorer 9 Compatibility Cookbook* includes information about changes to features, identifies features that are deprecated or removed, and describes general tools and guidance. New topics will be added to this section as features are modified and as you identify areas where you need more information.

Developer Audience

The *Internet Explorer 9 Compatibility Cookbook* is intended for anyone who develops or maintains Internet Explorer applications.

Run-Time Requirements

The *Internet Explorer 9 Compatibility Cookbook* applies to applications that specify a browser version that is earlier than Internet Explorer 9 and that can run on Internet Explorer 9.

Send comments about this topic to Microsoft

Build date: 4/13/2011

Angle Brackets Are Not Allowed in the createElement Method

.NET Framework 3.0

Affected Internet Explorer Document Modes

IE9 Standards

Feature Impact

Severity: High

Probability: High

Description

Windows Internet Explorer 9 does not recognize angle brackets (< >) within the **createElement** method. If you use angle brackets in Internet Explorer 9 webpages that are not set to previous Windows Internet Explorer Standards document modes, an exception occurs. For example, the following code example will cause an exception in IE9 mode.

Copy

```
var elm = document.createElement("<div id='myDiv'>");
```

Affected Areas

If you use angle brackets (< >) within the **createElement** method, the webpage or webpage component fails, depending on where the error occurs. A user might notice the impact, depending on where it occurs.

Guidelines

There are two ways to accommodate this change in Internet Explorer 9:

- Create the element and add the attributes individually by using the **setAttribute** API, as the following code example shows.

  Copy

  ```
  var elm = document.createElement("div");
  elm.setAttribute("id","myDiv");
  ```

- Create the element inside a parent element by using the **innerHTML** API, as the following code example shows.

  Copy

```
var parent=document.createElement("div");

parent.innerHTML="<div id='myDiv'></div>";

var elm=parent.firstChild;
```

Alternatively, you can set the webpage to a document mode that is earlier than Internet Explorer 9.

Related Topics

Discovering Internet Explorer Developer Tools

Specifying Document Compatibility Modes

Send comments about this topic to Microsoft

Build date: 4/13/2011

Mixing Native XML and MSXML Objects

.NET Framework 3.0

[This documentation is preliminary and is subject to change.]

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: High
Probability of Impact: Medium

Description

Internet Explorer 9 introduces the concept of native XML objects. Native XML objects can be rendered within a page and used with the same DOM APIs supported for HTML objects. Prior versions of IE always managed XML via MSXML objects, which still exist in IE9. However, native XML objects are not compatible with MSXML objects. This means that MSXML objects cannot be used with native DOM APIs and native XML objects cannot be used with MSXML DOM APIs.

When the code in a site attempts to mix the two, this generally results in a JavaScript exception being thrown, which can cause compatibility problems.

Affected Areas

Passing a node obtained from **responseXML** (which returns an MSXML object) into a native DOM API is the most likely scenario in which this happens.

Copy

```
var doc = xhr.responseXML;
if(document.adoptNode) document.adoptNode(doc.documentElement);
```

Most sites already have fallback code to correctly handle MSXML objects, but the problem occurs when certain feature detects cause sites to attempt to use native DOM APIs. The most common detects that cause problems are:

Copy

```
if(window.XMLSerializer)
if(document.adoptNode)
if(document.implementation.createDocument)
```

Guidelines

- Update site code to work entirely with native objects and APIs OR MSXML objects and APIs

- In general, try to migrate to native objects and APIs unless you need features like XPath/XSLT; this can be done by passing **responseText** to DOMParser, instead of using **responseXML var**:

  Copy

```
parser = new DOMParser();
parser.parseFromString(xhr.responseText, "text/xml");
```

- If MSXML APIs are still required, feature checks can be updated to verify the type of node received in order to select the correct API:

  Copy

```
if(!window.Node || !(node instanceof Node)) {
     // This is an MSXML node
}
```

Related Topics

- MSXML
- responseXML
- DOMParser and XMLSerializer

Send comments about this topic to Microsoft

Build date: 4/13/2011

OBJECT Fallback Is Included in DOM and Matched by window[name]

.NET Framework 3.0

Affected Internet Explorer Document Mode

Internet Explorer 9 Standards

Feature Impact

Severity: High
Probability of Impact: Medium

Description

When an OBJECT element has fallback content (typically, an EMBED element), Windows Internet Explorer 9 now parses this content and includes it in the Document Object Model (DOM), whereas previous versions of Internet Explorer did not. This behavior change makes Internet Explorer 9's handling of such elements more standards-compliant and interoperable.

As a result, if an OBJECT element has the same name attribute as any of its fallback elements, then window["myName"] will now return a collection of all elements with the name "myName". Pages that assume Internet Explorer will return a single element (typically, the OBJECT element) as opposed to a collection can cause an exception to occur when accessing methods and properties on the returned value.

Affected Areas

If you use the same name attribute for an OBJECT element as you do for the fallback content and attempt to retrieve a reference to the instantiated plugin using window[name], more than one element will be returned, where only the OBJECT element was returned in previous versions of Internet Explorer.

For example, consider the following script and markup:

Markup

Copy

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" name="myPlugin"
codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.cab">
     <param name="movie" value="myflash.swf" />
     <embed src="myflash.swf" name="myPlugin" type="application/x-shockwave-flash"
     pluginspage="http://www.adobe.com/go/getflashplayer"></embed>
</object>
```

JavaScript

Copy

```
plugin = window["myPlugin"]; //the OBJECT element in IE8, a collection of the OBJE
CT and
EMBED elements in IE9
plugin.style.display = "none";  //Causes an exception in IE9: “SCRIPT5007: Unable
to set
value of the property ‘display’: object is null or undefined”
```

In Internet Explorer 8, window[name] would return just the OBJECT element (because Internet Explorer8 did not include the fallback content in the DOM).

In Internet Explorer 9, the EMBED element is included in the DOM and thus window[name] returns a collection of both the OBJECT and EMBED elements. This behavior is interoperable with other browsers and is standards compliant.

Guidelines

When expecting the instantiated plugin, use document[name] instead of window[name]. document[name] will not match fallback elements when the parent is instantiated.

Copy

```
plugin = document["myPlugin"]; //the instantiated plugin: OBJECT element in IE,
EMBED element in browsers which use the Netscape plugin model
```

Additional Details:

In Internet Explorer 9 Beta, document[“myPlugin”] returned the EMBED element and window[“myPlugin”] returned a HTML Collection of both the OBJECT and the EMBED elements in the above example.

However, the following common coding pattern was observed on numerous sites:

Copy

```
if(document[“myPlugin”]) {
     plugin = document[“myPlugin”]; //expected to be the object element in Interne
t Explorer and embed element in other browsers
}
```

To help site compatibility, Internet Explorer 9 Release Candidate has an improved behavior for document[name]. Provided the plugin loads successfully, it will return the instantiated plugin (in the above example, the OBJECT element). This effectively matches what Internet Explorer 8 returned while still allowing for standards-compliant parsing of the fallback content. This behavior allows for the above coding pattern to be interoperable in all major browsers.

window[name] remains unchanged from Internet Explorer 9 Beta and should only be used when expecting to match more than just instantiated elements. The following table describes how these APIs work in Internet Explorer 9.

window["foo"] returns	document["foo"] returns
• a, applet, area, embed, form, frame, frameset, iframe, img, or object elements that have a name "foo", **or** • HTML elements that have an id "foo	• embed or object elements that are showing and have a name or id of “foo”, **or** • embed or fallback-free object elements, which have no object ancestor that is showing, and have a name or id of “foo”, **or** • object elements that are either showing or have no ancestor that is showing, and have an id of “foo”, **or** • applet, form, iframe or img elements that have a name “foo”, **or** • applet elements that have an id “foo”, **or** • img elements that have an id “foo”, and that also have a name content attribute present

An object element is said to be fallback-free if it has no object or embed descendants.

Related Topics

- Discovering Internet Explorer Developer Tools
- Specifying Document Compatibility Modes

Send comments about this topic to Microsoft

Build date: 4/13/2011

JavaScript Protocols That Return Null

.NET Framework 3.0

[This documentation is preliminary and is subject to change.]

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: High

Probability of Impact: Low

Description

IE9 now follows HTML5 guidelines when handling JavaScript protocols that return null.

Guidelines

HTML5 states that if a script running within a JavaScript protocol returns null, the browser must treat the URL as if it has returned **HTTP 204 No Content**, which must not contain a response body.

Consider this example:

Copy

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
   <div id=”ad_content”>
      <iframe src=“javascript:document.write(“…”); return null;” />
   // document.write is meant to create the contents of the iframe
   </div>
</body>
</html>
```

Since the JavaScript executed as part of the JavaScript protocol (javascript:) returns null, Internet Explorer 9 and other browsers that comply with this part of HTML5 treat the URL as if it returned **HTTP 204 No Content** so the iframe is empty, regardless of what other JavaScript ran in the JavaScript protocol.

Send comments about this topic to Microsoft

Build date: 4/13/2011

MIME-Handling Change: text/css

.NET Framework 3.0

Affected Internet Explorer Doucment Mode

IE9 Standards

Feature Impact

Severity: High
Probability of Impact: Low

Description

Web servers send a HTTP response header named “Content-Type” that specifies the MIME-type of the file that is being sent. For security and standards-compliance reasons, style sheets should be delivered with the text/css MIME-type.

- In Internet Explorer 9’s Standards Mode, style sheets will be ignored (not applied) unless they are delivered with a **text/css** MIME type.
- In all document modes, style sheets will be ignored if the style sheet is delivered with the **X-Content-Type-Options: nosniff** header and the style sheet is not delivered with the **text/css** MIME type.
- In all document modes (and in legacy browser versions) style sheets delivered from a cross-origin context (for example, example.com uses a style sheet from Microsoft.com) will be ignored unless the style sheet is delivered with the **text/css** MIME type.

Affected Areas

If a style sheet is ignored due to an incorrect MIME-type, your site may fail to render as expected. Text, images, or other features may lack the desired styling.

If a style sheet is ignored because it does not bear the correct MIME-type, a notification will be logged in the IE9 F12 Developer Tools console.

A concise test case displays red text if a browser applies styles from a style sheet with an incorrect MIME-type; it renders the text in green if the invalid style sheet is correctly ignored.

Guidelines

Ensure that all style sheets are delivered with the proper HTTP response header: **Content-Type: text/css**.

If you find any sites that are sending improper MIME types and behave incorrectly in Internet Explorer, please file a bug on Connect

Related Topics

- MIME-Handling Changes in Internet Explorer
- Connect

Send comments about this topic to Microsoft

Build date: 4/13/2011

MIME-Handling Change: text/plain

.NET Framework 3.0

Affected Internet Explorer Document Mode

All document modes

Affected Internet Explorer Browser Mode

IE9 Browser mode (all Document modes)

Feature Impact

Severity: High
Probability of Impact: Low

Description

In Internet Explorer 9's Browser Mode, documents delivered with a **text/plain** MIME type will not be MIME-sniffed to another type. Documents will render or download as plain text only.

Affected Areas

If IE9 encounters a HTML document delivered with a text/plain content-type, the document will be rendered as plain text unless the site is rendering in Compatibility View.

This is useful for web developer scenarios because it allows easier sharing of HTML source code snippets. It's also a welcome change from a security point-of-view, because IE9 will be less susceptible to script injection attacks in files delivered with a **text/plain** Content Type.

Guidelines

- Use the content type **text/plain** to ensure a plain text rendering in IE9
- Configure your server to send proper Content-Type headers for all documents that your server will deliver; for instance, if your server offers PDF files for download, be sure that these files are delivered with the **application/pdf** MIME type

If you find any sites that are sending improper MIME types and behave incorrectly in Internet Explorer, please file a bug on Connect.

- MIME-Handling Changes in Internet Explorer
- Content Type Logic in Internet Explorer
- Handling MIME types in Internet Explorer
- MIME-type Detection in Internet Explorer
- Connect

Send comments about this topic to Microsoft

Build date: 4/13/2011

MIME-Handling Change: X-Content-Type-Options: nosniff

.NET Framework 3.0

Affected Internet Explorer Document Mode

All

Feature Impact

Severity: High

Probability of Impact: Low

Description

SCRIPT and STYLESHEET elements will reject responses with incorrect MIME types if the server sends the response header **X-Content-Type-Options: nosniff**. This is a security feature that helps prevent attacks based on MIME-type confusion.

Affected Areas

This change impacts the browser's behavior when the server sends the X-Content-Type-Options: nosniff header on its responses.

If the nosniff directive is received on a response received by a STYLESHEET reference, Internet Explorer will not load the "stylesheet" file unless the MIME type matches **text/css**.

If the nosniff directive is received on a response retrieved by a SCRIPT reference, Internet Explorer will not load the "script" file unless the MIME type matches one of the following values:

- application/ecmascript
- application/javascript
- application/x-javascript
- text/ecmascript
- text/javascript
- text/jscript
- text/x-javascript
- text/vbs
- text/vbscript

When such content is blocked, the F12 developer tools show the following message:

> SEC7112: Script from http://www.debugtheweb.com/test/mime/textplainnosniff.asp was blocked due to mime type mismatch script.asp

Guidelines

Ensure that in any response received with the nosniff directive has a MIME type that matches one of the values listed above.

If you find any sites that are sending improper MIME types and behave incorrectly in Internet Explorer, please file a bug on Connect.

Related Topics

- MIME-Handling Changes in Internet Explorer
- X-Content-Type-Options: nosniff
- X-Content-Type-Options enhancements in IE9 Beta
- Connect

Send comments about this topic to Microsoft

Build date: 4/13/2011

XSLT Compatibility

.NET Framework 3.0

[This documentation is preliminary and is subject to change.]

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: High

Probability of Impact: Low

Description

In Internet Explorer 9, the processing of XML and XSLT files has been modified for improved standards compliance and interoperability with other browsers. In particular, certain non-standard behaviors relating to the processing of XSLT files have changed.

Affected Areas

Scenario 1: The legacy XSL namespace is no longer supported for XSLT files. If this namespace is applied to elements in an XSLT file, those elements will not be interpreted as XSLT elements and the document will not process as expected:

Copy

```
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/TR/WD-xsl">
```

Scenario 2: Processing instructions with the name “xml:stylesheet” (note the colon) no longer cause XSLT to be processed:

Copy

```
<?xml:stylesheet type="text/xsl" href="my.xslt"?>
```

Scenario 3: The “xsl:output” element can now be used to specify XML parsing for XSLT output, meaning output will no longer always parse as HTML.

Copy

```
<xsl:output method="xml">
```

Guidelines

Scenario 1: Migrate to the standardized XSLT namespace:

Copy

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Scenario 2: Use the standardized “xml-stylesheet” processing instruction for loading XSLT:

Copy

```
<?xml-stylesheet type="text/xsl" href="my.xslt"?>
```

Scenario 3: Explicitly opt-in to HTML parsing for XSLT if your output depends on HTML parsing rules:

Copy

```
<xsl:output method="html">
```

Related Topics

XSLT

Send comments about this topic to Microsoft

Build date: 4/13/2011

IE9 Standards Mode Does Not Support the arguments.caller Property

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impacts

Severity: Mixed (dependent upon how the page uses the feature)
Probability of Impact: Medium

Description

The arguments.caller property is not supported in IE9 Standards mode of Internet Explorer 9.

Affected Areas

When argument objects are created, in all modes of Internet Explorer 8 and Quirks, IE7 Standards, and IE8 Standards modes of Internet Explorer 9, a property with the name “caller” is created. This caller property stores the reference to the argument object of the function that called it.

In the following example, all document modes in IE8 and Quirks, Internet Explorer 7 standards, and Internet Explorer 8 standards document modes in IE9 return “1”. Internet Explorer 9 standards mode issues the script error “object is null or undefined”.

Copy

```
function alertCallerLength() {
        alert(arguments.caller.length);
}

function callingFunction() {
        alertCallerLength();
}
callingFunction(1)
```

Related Topics

- Caller Property (JavaScript)

Send comments about this topic to Microsoft

Build date: 4/13/2011

Automatic Binding of Binary Element Behaviors Does Not Work

.NET Framework 3.0

Affected Internet Explorer Document Modes

IE9 Standards

Feature Impact

Severity: Mixed (depends on how a webpage uses the feature)

Probability: Low

Description

Windows Internet Explorer 8 does not include the automatic binding capability to make Windows Internet Explorer more consistent with other browsers. For example, the following code activates an SVG plug-in in Internet Explorer 8 if that plug-in registered for the SVG namespace.

Copy

```
<html>
 <head>
  <title>SVG embedded inline in XHTML</title>
 </head>
 <body>
  <svg width="600" height="300" xmlns="http://www.w3.org/2000/svg">
   <linearGradient id="gradient">
    <stop style="stop-color:yellow" offset="0%" />
    <stop style="stop-color:green" offset="100%" />
   </linearGradient>
   <rect x="0" y="0" width="100" height="100" style="fill:url(#gradient)" />
   <circle cx="50" cy="50" r="30" style="fill:url(#gradient)" />
   <circle cx="150" cy="100" r="50" />
  </svg>
 </body>
</html>
```

But in IE9 mode, this code example uses the native SVG handling in Windows Internet Explorer 9.

Affected Areas

The removal of this capability affects only pages that you create for Internet Explorer 8. A user would see a generic element instead of the specific control.

Guidelines

You must manually instantiate the binding control when you create webpages for Internet Explorer 9, as the following code example shows.

Copy

```
<html xmlns:svg>
 <head>
  <title>SVG embedded inline in XHTML</title>
  <!-- The following is the "hookup code"  that Internet Explorer requires. -->
  <object id="AdobeSVG" classid="clsid:78156a80-c6a1-4bbf-8e6a-3cd390eeb4e2">
  </object>
  <?import namespace="svg" implementation="#AdobeSVG"?>
 </head>
 <body>
  <svg:svg width="600" height="300">
   <svg:linearGradient id="gradient">
    <svg:stop style="stop-color:yellow" offset="0%" />
    <svg:stop style="stop-color:green" offset="100%" />
   </svg:linearGradient>
   <svg:rect x="0" y="0" width="100" height="100" style="fill:url(#gradient)" />
   <svg:circle cx="50" cy="50" r="30" style="fill:url(#gradient)" />
   <svg:circle cx="150" cy="100" r="50" />
  </svg:svg>
 </body>
</html>
```

However, if you set the webpage to run in IE8 mode, the previous code syntax works correctly.

Related Topics

Internet Explorer Namespace Support

-ms-behavior Attribute | behavior Property

Binary Behaviors

Discovering Internet Explorer Developer Tools

Specifying Document Compatibility Modes

Send comments about this topic to Microsoft

Build date: 4/13/2011

Calling a Method with a Function Pointer without ".call" or ".bind"

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Mixed (dependent upon how the page uses the feature)

Probability of Impact: Medium

Description

Previous versions of Internet Explorer supported caching a pointer to a method and then using the cached pointer to call the method. This support was removed in Internet Explorer 9 in order to increase interoperability with other browsers.

A common practice on webpages targeting older versions of IE was to save common methods to a variable and use that variable as a proxy for the method in order to make JavaScript code more compact:

Copy

```
var d = document.writeln;

d("<script language=VBScript>");
```

In IE9, an object is required in order to invoke the method. By default the "window" object is provided in global scope (such as in the previous example). However, the "window" object does not have a method "writeln" and so the JavaScript error message "Invalid calling object" is reported.

Affected Area

Script error: "Invalid calling object"

Guidance

Now you must specify the target for the method call just as you do in all other browsers. So while this code works in IE8 and earlier:

Copy

```
var d = document.writeln;
```

```
d("<script language=VBScript>");
```

Now it fails in IE9 just as it fails in all other browsers. An easy fix for this issue is to use the "call" method (a property of all functions) to explicitly provide the appropriate calling object:

Copy

```
d.call(document, "<script language="VBScript">”);
```

The long term fix for this is to use JavaScript's "bind" API to associate an implicit calling object with the method. This is done as follows (again, drawing on the previous example):

Copy

```
var d = document.writeln.bind(document);

d("<script language=VBScript>"); // Now this is OK.
```

Send comments about this topic to Microsoft

Build date: 4/13/2011

Content Attributes and DOM Expandos Are No Longer Connected

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Mixed (dependent upon how the page uses the feature)
Probability of Impact: Medium

Description

In past versions of Internet Explorer, content attributes were represented on JavaScript objects as DOM expandos. In Internet Explorer 9, this link between content attributes and DOM expandos has been severed in order to increase interoperability between IE and other browsers.

A **content attribute** is an attribute that is specified in the HTML source, for example, <element attribute1="value" attribute2="value">. Many content attributes are predefined as part of HTML; HTML also supports additional user-defined content attributes.

DOM expandos are the values that can be retrieved from an object in JavaScript, for example, document.all["myelement"].domExpando. Most objects have a predefined set of properties that typically represent the value of their same-named content attribute. JavaScript supports additional user-defined properties as well.

In the following example, 'id' and 'class' are predefined content attributes in HTML and 'myAttr' is a user-defined content attribute:

Copy

```
<div id="myElement" class="b" myAttr="custom"></div>
```

In the following script sample, 'id' and 'className' are predefined properties:

Copy

```
var div = document.getElementById("myElement");
var divId = div.id; // Gets the value of the id content attribute
var divClass = div.className; // Gets the value of the class content attribute
```

In IE8 and before (including IE8 document mode and previous modes in IE9), the presence of the 'myAttr' content attribute would imply the presence of a 'myAttr' DOM expando:

Copy

```
var divExpando = div.myAttr; // divExpando would get the value "custom" in IE8
```

The breaking change in IE9 is that DOM expandos will no longer be implied by the presence of user-defined content attributes:

Copy

```
var divExpando = div.myAttr; // divExpando would get an undefined value
```

Affected Areas

This problem can manifest as a script error, but rarely will the problem be seen directly at the point of failure in the JavaScript program. Rather the failure will typically be seen later when the value [supposedly retrieved] from the DOM expando is used in another part of the code.

Guidance

To fix this issue, please use the 'getAttribute' API to retrieve the value of user-defined content attributes. This workaround is recommended for all versions of IE and does not require special-casing for new versions compared to older versions of IE.

For example (based on the previous example):

Rather than:

Copy

```
var divExpando = div.myAttr;
```

Use:

Copy

```
var divExpando = div.getAttribute("myAttr");
```

Related Topics

- Attribute Differences in IE8

Send comments about this topic to Microsoft

Build date: 4/13/2011

Internet Explorer 9 Compatibility with Popular JavaScript

.NET Framework 3.0

[This documentation is preliminary and is subject to change.]

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Mixed (dependent upon how the page uses the feature)

Probability of Impact: Medium

Description

Many Internet Explorer 9 features were added or modified for improved standards compliance and interoperability with other web browsers. However, many existing JavaScript frameworks contained functionality that was dependent upon existing IE-specific behavior or quirks. Consequently, the changes made in IE9 caused parts of many popular JavaScript frameworks to stop working correctly in IE9 Standards Mode.

Most of these frameworks have already been updated to work correctly in IE9. However, many sites are still using older versions of JavaScript frameworks that are not compatible with IE9.

Guidelines

If your site relies on a JavaScript framework and is encountering issues in IE9, try updating to the latest version of the framework. The below table lists versions of popular JavaScript frameworks that claim to be compatible with IE9:

Framework	IE-Compatible Versions	Get the Latest Version From
Cufon	1.09i+	http://cufon.shoqolate.com/generate/
jQuery	1.5.1+	http://jquery.com/
jQuery UI	1.6.8+	http://jqueryui.com/
MooTools	1.3+	http://mootools.net/
Prototype	1.7+	http://prototypejs.org/

NOTE: Frameworks that are not listed above may have a newer version that is compatible with IE9, so upgrading to the latest version of a framework remains the correct initial course of action to resolve

compatibility issues. Please share knowledge of the compatibility status of other frameworks in the comments below.

Send comments about this topic to Microsoft

Build date: 4/13/2011

Dynamic VML Patterns May Not Work

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Mixed (dependent upon how the page uses the feature)

Probability of Impact: Low

Description

To support dynamic VML in Internet Explorer 9 standards mode, the VML behavior must be attached to an element before any VML properties are assigned. If the order-of-operations is reversed, the VML engine cannot obtain the value from the DOM and the VML may not render correctly or at all. In the following example, VML is loaded statically without a problem:

Copy

```
<style>oval {behavior: url(#default#VML);position:absolute}</style>

<body>

 <oval style="left:0;top:0;width:100px;height:50px" fillcolor="blue" stroked="f"/>

</body>
```

Problems can occur if, for example, the 'oval' element is accessed before adding the style sheet and a 'fillcolor' property is set on that object. In that case, VML will be unable to retrieve the fillcolor value and may not render correctly.

This problem is commonly encountered by older versions of the Cufon.js library that use VML for advanced font rendering (or the Canvas tag, if available).

Affected Areas

Missing text on webpages, upside-down text, missing VML graphics

Guidance

If the page is using the Cufon library, update to the latest version of the library that is compatible with IE9.

Send comments about this topic to Microsoft

Build date: 4/13/2011

Global Object's Properties Cleared When Window Is Orphaned

.NET Framework 3.0

[This documentation is preliminary and is subject to change.]

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Mixed (dependent upon how the page uses the feature)
Probability of Impact: Low

Description

Properties on the global object (window) are cleared when a window is orphaned. The properties are cleared to allow garbage collection (GC) of the orphaned window when no additional references to it are found. Additionally, timers stop firing and event propagation (inside the orphaned window) stops immediately. The following definitions will help in understanding this topic:

Transient iFrame: An iFrame that is parsed or temporarily created, added and then removed from the DOM tree and not reused is known as a "transient" iFrame.

Orphaned window: When a transient iFrame is removed from the DOM tree, its window is considered "orphaned" (alive, but not directly reachable) until it is closed.

Affected Areas

When iFrame elements are removed from the DOM tree, several things happen:

- Their orphaned window object is "cleaned" to allow the orphaned window to GC. A cleaned window object (that is, global object) will not have any properties visible on it. The cleaning only involves property removal on the window object—it does not actually delete the objects formerly referenced by those properties. This provides the ability for any built-in or user-defined objects present on the orphaned window to be reachable if, and only if, there is another external reference to the object.
- Existing timers (for example, setTimeout) are canceled and no further timers are allowed to run
- Event propagation that was in progress within the window that becomes orphaned is stopped (for example, as if the stopImmediatePropagation API was invoked on the related event)

These are changes from Internet Explorer 8 behavior where none of the above steps were taken when the transient iframe's window was orphaned. These behavior changes apply only to IE9 standards mode. Legacy modes behave as they did in Internet Explorer 8.

Guidelines

In previous versions of Internet Explorer (including legacy Internet Explorer 9 document modes), when an iFrame was removed from the DOM tree, the memory for the orphaned window was reclaimed only at page navigation time. Prior to page navigation, the orphaned window's memory would persist without being garbage collected. In rare cases when sites use multiple transient iFrames to download and/or run content without intervening page navigations, the amount of memory per orphaned window would compound until

it negatively impacted a user's experience on the page. Pre-existing references to objects within the orphaned window remain available; however, re-requesting these properties via the global object will fail. Failures will manifest as script errors where "myRequestedProperty" does not exist or is undefined.

We recommend that:

- Objects required from an orphaned window have pre-existing references (from outside the orphaned window) in order to be reachable, and that functions executed in the context of the orphaned window avoid references to properties from the global object (for example, Array, Object)
- Code that expects timers to fire should be made resilient to the possibility of their window becoming orphaned, resulting in the cancellation of existing timers
- Event handlers that have a side effect of removing their containing iFrame should be made resilient to the possibility of their window becoming orphaned, resulting in a "stopImmediatePropagation" of the currently propagating event

Related Topics

Details regarding the stopImmediatePropagation API

Send comments about this topic to Microsoft

Build date: 4/13/2011

JavaScript Property Enumeration Differs in Internet Explorer 9

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards | IE8 Standards | IE7 Standards | Quirks

Feature Impact

Severity: Mixed (dependent upon how the page uses the feature)
Probability of Impact: Low

Description

Due to the changes made in the JavaScript object model of Internet Explorer 9, JavaScript properties may be enumerated differently from how they are enumerated in Internet Explorer 8.

Affected Areas

When using the for…in Statement, in any document mode, the order of property enumerations may be different from the order returned by Internet Explorer 8. For example, numeric properties now enumerate before non-numeric properties. The following sample illustrates the difference in enumeration order between Internet Explorer 8 and Internet Explorer 9:

Copy

```
var obj = {first : "prop1", second: "prop2", 3: "prop3"};

var s = "";
for (var key in obj) {
    s += key + ": " + obj[key] + " ";
}
document.write (s);
```

Running this sample in Internet Explorer 8 and Internet Explorer 9 would print the following results:

All modes of Internet Explorer 8:
first: prop1 second: prop2 3: prop3
All modes of Internet Explorer 9:
3: prop3 first: prop1 second: prop2

Internet Explorer 8 does not include enumerations of properties that have the same name as built-in properties of a prototype object. All document modes in Internet Explorer 9 include these properties in the enumeration. The following sample illustrates this difference:

Copy

```
var obj = { first: "prop1", toString : "Hello" }
var s = "";
for (var key in obj) {
    s += key + ": " + obj[key] + " ";
```

```
}
document.write (s);
```

Running this sample in Internet Explorer 8 and Internet Explorer 9 would print the following results:

All modes of Internet Explorer 8:
first: prop1
All modes of Internet Explorer 9:
first: prop1 toString: Hello

Related Topics

- for…in statement (JavaScript)

Send comments about this topic to Microsoft

Build date: 4/13/2011

Math Precision Differs in Internet Explorer 9

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards | IE8 Standards | IE7 Standards | Quirks

Feature Impact

Severity: Mixed (dependent upon how the page uses the feature)
Probability of Impact: Low

Description

Math precision differs from Internet Explorer 8 in certain edge cases.

Affected Areas

The Chakra engine uses Streaming SIMD Extensions 2 (SSE2) if the platform supports them, which results in faster mathematical operations but also yields a difference in precision from the JScript engine of Internet Explorer 8. The following example demonstrates the difference:

Copy

```
function test() {
    var x = 6.28318530717958620000;
    var val = Math.sin(x);
    document.write(Math.abs(val))
}
test();
```

In all modes of Internet Explorer 8, this will print “2.4492127076447545e-16”. When the platform supports SSE2, in all modes of Internet Explorer 9, this will print “2.4492935982947064e-16”.

Related Topics

- Math Object (JavaScript)
- SSE2

Send comments about this topic to Microsoft

Build date: 4/13/2011

Thai and East Asian Text and Font Sizing

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Low
Probability of Impact: High

Description

Thai and East Asian text may look smaller in IE9 than in IE8 and earlier releases. In IE8, Thai and East Asian text could be rendered at a larger font size than specified when:

- The specified font size was 9pt or smaller
- The specified font family did not support Thai or East Asian characters such as Arial

Thus, a Thai paragraph of 8pt Arial would be rendered using the fallback font specified in Internet Options-Fonts and the latter would then be scaled up to match Arial metrics. As a result, the real size is larger than it would have been if the web author had requested that font at 8pt.

In IE9, the specified font size is always respected. Thus, as the fallback font is no longer scaled up, text may appear smaller.

Affefcted Areas

Thai and East Asian content related to font sizes.

Guidelines

When possible, make sure the first value of the CSS font-family property supports your language.

For instance, instead of asking for Arial, use MS PGothic. Instead of Times New Roman, use Mincho. Instead of Verdana, use Meiryo. You can use the Internet Options – Fonts dialog to check the default fallback mappings used by IE.

This will ensure your text renders using the specified font size in all versions and modes of IE.

Build date: 4/13/2011

Default User-Agent (UA) String Changed

.NET Framework 3.0

Affected Internet Explorer Document Modes

IE9 Standards

Feature Impact

Severity: Low

Probability: High

Description

The user agent (UA) string has changed in Windows Internet Explorer 9 in several ways.

Shortened User Agent String

By default, Internet Explorer 9 sends a new short UA string to improve the overall performance, interoperability, and compatibility. Internet Explorer 9 does not send additions to the UA string that are made by other software that is installed on the computer, such as the .NET Framework and many other programs.

There are four primary changes to UA string from Internet Explorer 8:

- The application version is incremented from "Mozilla/4.0" to "Mozilla/5.0", to match other browsers and make Internet Explorer 9 an interoperable browser.
- The version token is incremented from "MSIE 8.0" to "MSIE 9.0".
- The Trident token is incremented from "Trident/4.0" to "Trident/5.0".
- Internet Explorer 9 sends the following short UA string without additions from other software that is installed on the computer.

Copy

```
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
```

Because previous long, extended UA strings cause compatibility issues, Internet Explorer 9 sends the short UA string that is shown earlier without Pre-Platform and Post-platform registry value tokens.

Applications and platforms can continue to add to the UA string through the Pre-Platform and Post-Platform registry keys as they did in previous Windows Internet Explorer releases. Internet Explorer 9 does not change existing registry values.

Your websites can still get the extended UA string with the Pre-Platform and Post-Platform tokens through the navigator.userAgent property.

UA String in Compatibility View

Similar to Windows Internet Explorer 8, the Internet Explorer 9 Compatibility View maps to Windows Internet Explorer 7 Standards Mode. The UA string for Internet Explorer 9 in Compatibility View takes the following form.

Copy

```
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; Trident/5.0)
```

In Compatibility View, Internet Explorer 9 reports itself as Windows Internet Explorer 7 through the application version number (for example, "Mozilla/4.0") and version token (for example, "MSIE 7.0") for compatibility. The incremented Trident token (from "Trident/4.0" to "Trident/5.0") enables websites to distinguish between when Internet Explorer 9 is running in Compatibility View and when Internet Explorer 8 is running in Compatibility View.

Guidelines

Test how your website responds to the Internet Explorer 9 UA string by checking and changing the UA string through the registry. If your site does not already respond with Internet Explorer-compatible content, update it to recognize Internet Explorer 9 and be future-proof.

Related Topics

Internet Explorer 8 - User-Agent String

The Internet Explorer 8 User-Agent String (Updated Edition)

Understanding User Agent Strings

navigator.userAgent Property

Send comments about this topic to Microsoft

Build date: 4/13/2011

Indirect 'eval' Function Calls Behave Differently in Internet Explorer 9

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Low
Probability of Impact: Low

Description

Calling eval methods indirectly (that is, other than by the explicit use of its name) inside a function produces different results in Internet Explorer 9 than it does in Internet Explorer 8.

Affected Areas

In all modes of Internet Explorer 8 and Quirks, IE 7 standards, and IE8 standards modes of Internet Explorer 9, the string passed to the indirect eval is evaluated in the local function scope. In IE9 Standards mode of Internet Explorer 9, it is evaluated in the global scope as per the ECMAScript standard (5th edition).

In the following example, the eval function has been called indirectly by assigning it to a variable and calling the variable as if it were eval.

Copy

```
function test() {
   var dateFn = "Date(1971,3,8)";
   var myDate;
   var indirectEval = eval;
   indirectEval("myDate = new " + dateFn + ";");
   document.write(myDate);
}
test();
```

This sample will return “Thu Apr 8 00:00:00 PDT 1971” in all document modes of Internet Explorer 8 and Quirks, IE7 standards, and IE8 standards document modes in Internet Explorer 9. However in IE9 standards mode of Internet Explorer 9, this would print “undefined”.

Related Topics

- eval Function (JavaScript)

Send comments about this topic to Microsoft

Build date: 4/13/2011

Internet Explorer 9 Handles Array Elements with a Large Index Differently

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards | IE8 Standards | IE7 Standards | Quirks

Feature Impact

Severity: Low
Probability of Impact: Low

Description

Array elements with large indices are handled differently than in Internet Explorer 8.

Affected Areas

Internet Explorer 8 does not conform to the ECMAScript (3rd edition) specification in situations where the initial value of the array's length property is greater than 2E+31-1 (that is, 2147483647). When an array element is created with an index greater than 2147483647, the index of new element created will be a negative integer.

Internet Explorer 9 correctly handles array elements that use indices between 2E+31-1 and 2E+32-2. The Internet Explorer 8 behavior is not replicated in any of the Internet Explorer 9 document modes.

This can be observed when an element is pushed to an array of length 2E+31-1.

The following sample prints "true" in Internet Explorer 8, but prints "false" in Internet Explorer 9.

Copy

```
function test() {
    var arr = new Array();
    arr[2147483650] = 10000;
    arr.push(10);
    document.write(arr["-2147483645"] == 10);
}
test();
```

Related Topics

- Array Object (JavaScript)
- push Method (JavaScript)

Send comments about this topic to Microsoft

Build date: 4/13/2011

Overlapping Elements Are Cloned

.NET Framework 3.0

Affected Internet Explorer Document Modes

IE9 Standards

Feature Impact

Severity: Low

Probability: Low

Description

Overlapping formatting elements are cloned in Windows Internet Explorer 9 to reduce ambiguity in the Document Object Model (DOM).

Affected Areas

Typically, a site looks the same with or without this feature.

However, if you perform DOM operations on elements that are overlapped in markup, the operations might not work the same way in Internet Explorer 9. For example, if you make DOM calls such as **firstChild** or **nextSibling** on an overlapped element, these calls might not work the same way.

Guidelines

Test your code by using the Internet Explorer Developer Tools to find and fix any overlapping elements.

Related Topics

Discovering Internet Explorer Developer Tools

Send comments about this topic to Microsoft

Build date: 4/13/2011

Some Behavior-Connecting Methods Do Not Work in XML

.NET Framework 3.0

Affected Internet Explorer Document Modes

IE9 Standards

Feature Impact

Severity: Mixed (depends on how a webpage uses the feature)

Probability: Low

Description

The markup-based form for connecting behaviors works in Windows Internet Explorer 9 modes, but it does not work in xml mode. A behavior can be specified at the top of a webpage, as the following code example shows.

Copy

```
<html xmlns:myNamespace>

        <?import namespace="myNamespace" implementation = "my.htc">

        …

        <myNamespace:calendar/>
```

Affected Areas

This problem affects only new content.

Guidelines

Instead of using HTML markup, use Cascading Style Sheets (CSS)-based registration through the **behavior** property, as the following code example shows.

Copy

```
<style>

.calendar {
```

```
-ms-behavior: url(my.htc);
}
</style>
…
<div class="calendar"></div>
```

You can use CSS-based registration through elements other than the **class** attribute.

Related Topics

-ms-behavior Attribute | behavior Property

Binary Behaviors

Send comments about this topic to Microsoft

Build date: 4/13/2011

styleSheet.title Is readonly in IE9 Mode

.NET Framework 3.0

Affected Internet Explorer Document Mode

IE9 Standards

Feature Impact

Severity: Low
Probability of Impact: Low

Description

In IE8 mode and below you can change the value of a styleSheet Object's title. In IE9 mode the write command will be ignored and the original value will remain.

Affected Areas

How this affects the page will differ depending on the reason the page is changing the styleSheet title.

Guidelines

If you have to change a styleSheet Objects title, you can do so by changing the title attribute on the link element or style element that contains the styleSheet.

Related Topics

- W3C DOM Style Sheets

Send comments about this topic to Microsoft

Build date: 4/13/2011

Table Object Model Is Now More Consistent with Other Browsers

.NET Framework 3.0

Affected Internet Explorer Document Modes

IE9 Standards

Feature Impact

Severity: Mixed (depends on how a webpage uses the feature)
Probability: Medium

Description

To improve consistency between Windows Internet Explorer and other browsers, the IE9 Standards mode includes the following changes to the table object model:

- Extra **thead** and **tfoot** elements do not appear in the **tBodies** collection.
- The rows collection has a different ordering. First, it includes any rows in the **thead** element, then all remaining rows that are not in the **tfoot** element, and then any rows in the **tfoot** element, regardless of their order in the document.
- A call for rows returns all rows at all depths within the table, including direct row children of the table.
- The **getElementsByTagName** and **HtmlElement.children** methods do not return comment nodes.

Affected Areas

If you do not consider these changes in your application, the application might encounter script errors that are minor, that keep pages from loading, or that create content that is not intended.

Guidelines

If your code works in other browsers, it should work in Windows Internet Explorer 9.

If you have not tested your code in other browsers, you might need to update your code to work with IE9 Standards mode. Test your code by using the Internet Explorer Developer Tools to find and fix problems.

Alternatively, you can force Internet Explorer to use IE8 mode and use the same behavior as Windows Internet Explorer 8.

Related Topics

Discovering Internet Explorer Developer Tools
Specifying Document Compatibility Modes

Send comments about this topic to Microsoft

Build date: 4/13/2011

Text Layout Uses Natural Metrics

.NET Framework 3.0

Affected Internet Document Modes

IE9 Standards

Feature Impact

Severity: Mixed (depends on how a webpage uses the feature)

Probability: Medium

Description

For text layout in IE9 Standards mode, Windows Internet Explorer 9 uses natural metrics instead of the graphical device interface (GDI) metrics that other Microsoft Windows browsers use. GDI metrics align text to pixel boundaries, but natural metrics use inter-pixel spacing to create more accurately rendered and readable text.

Other document modes for Windows Internet Explorer continue to use GDI metrics.

Affected Areas

Page layouts that are written for earlier document modes might display incorrectly in Internet Explorer 9. The most common error is unexpected text wrapping, which can cover elements that are below the wrapped text. This error is most likely when a text box does not include extra horizontal space or when the size of the text box is connected to another element on the page, such as a graphic.

Guidelines

Do not assume that the size of a particular font renders identically across browsers or within the same browser because users can choose a larger font display (for example, 125%).

Do

Use the following design guidelines to make sure that your webpages display text layout consistently:

- Set a text box's size to a specific number of pixels.
- Include extra space in your text boxes and avoid tight spaces.
- Use non-statically sized text boxes.
- Include extra space in bounding areas that depend on other page elements.
- Make sure your page can accommodate text wrapping if you permit users to change the page font size.
- Test your webpages in all common browsers. If you see a text wrapping issue, adjust the page to make sure the page appropriately renders the text.
- If you designed a webpage for a previous mode and you do not want to update it to use natural metrics, set the page to display in that mode even when users view it in Internet Explorer 9.

Do not

Avoid the following designs for text layout:

- Depend on font sizes to render the same way across browsers.
- Use static-sized text boxes.

Related Topics

Internet Explorer Compatibility

Send comments about this topic to Microsoft

Build date: 4/13/2011

White Spaces Are Preserved in the Document Object Model

.NET Framework 3.0

Affected Internet Explorer Document Modes

IE9 Standards

Feature Impact

Severity: Low

Probability: Low

Description

Any white space that you add to a webpage persists in the Document Object Model (DOM). Consider how the following code example appears in Windows Internet Explorer 9 and Windows Internet Explorer 8.

Copy

```
<div>
   Text </div>
```

Internet Explorer 9:

Copy

```
     div
              |->"\n   Text "
```

Internet Explorer 8:

Copy

```
     div
```

```
|->"Text"
```

Affected Areas

If you want behavior like Internet Explorer 8, use the Element Traversal API (for example, **firstElementChild**).

Related Topics

Element Traversal API

Send comments about this topic to Microsoft

Build date: 4/13/2011