Whitepaper

Client Side Compatibility issues on Macintosh

Introduction

From the very start of World Wide Web, compatibility among different Browsers and Platforms has been withdrawing great attention of Web Developers. Every Browser has some special way of implementing the tasks apart from the Standard set by W3C for Document Object Model.

History.go(-1)

When developing public Web sites these days, it's potentially hazardous to do so within the confines of a single browser and operating system platform, regardless of what the installed base numbers tell you. Unless your site supports products aimed at a very narrow audience, you risk alienating a sizable number of potential visitors by demanding that everyone run your browser and your OS (this warning applies to Mac-centric developers, too).

Macintosh Platform is widely used for processing Print and Graphics related tasks. Now a very healthy number of users prefer Mac to surf the web and making the code compatible on Mac platform could be a head scratching task. A Web developer who has mastered the massive Microsoft Internet Explorer Document Object Model (DOM) for the PC may have too much learning when it comes to writing scripts that are compatible with Macintosh versions of Internet Explorer (IE). I did face the same scenario recently when I was asked to make one portal compatible with Macintosh Platform. This was quite challenging and interesting. At the end of the job I found that my knowledge repository got manifold.

Initializing the Fact array[]

It could be surprising for many that separate Microsoft teams produce Macintosh and Windows versions of Internet Explorer. This is quite different from, for example, the way Netscape Navigator 6 shares the same core browser engine in all operating systems. One big outcome of this Microsoft's approach is that the two browsers are not always on the same release schedule and therefore do not necessarily contain the same DOM features. Moreover, if you compare the resulting DOM implementations in IE5 for Windows and Macintosh, you might conclude that the Mac development team exercises its own discretion when it comes to implementing DOM specifics. The Windows version of the DOM may contain both Microsoft proprietary and W3C DOM syntax for a particular property or method of an HTML element object. In contrast, the Macintosh version may omit the Microsoft proprietary syntax in favor of the W3C DOM version. Overall, IE 5.0 for Macintosh provides more direct support for W3C DOM syntax than even the later IE5.5 for Windows does.

If(…….)

You can still count on a lot of basic DOM functionality and syntax implemented in versions of IE for Mac and Windows. For example, in addition to support for the document.all collection, both platforms support the popular set of element object properties that facilitate direct manipulation of page content: innerHTML, innerText, outerHTML, and outerText etc. Scripts in both browsers can control style sheet properties by way of an element object's style property. The fundamental IE4 event model is also consistent across both platforms.

In the Version 5 implementations of IE, both the Macintosh and Windows versions support enough of the W3C DOM node object model to be functional. Since the W3C DOM provides syntactically identical access to style sheet properties as the IE4 DOM does, you don't have to learn any new tricks to be compatible in that department. Perhaps the largest omission in the IE5 family is the W3C DOM event listener model, but IE5 for both Mac and Windows continue to support the IE4 event model.

It appears as though a lot of code written for IE/Windows would run smoothly under IE/Macintosh -- and …….it does(). But if you let your script writing be governed by all of the syntax prepared for the Windows version, you may find yourself in trouble when you get down to the details.

Document.Topics.Focus()

Here we will examine some features, which may differ from Windows to Mac. Internet Explorer for Windows bundles numerous ActiveX controls, some of which the browser requires for normal operations. The browser also has instant communication with other ActiveX controls on the client machine, whether the controls are part of the Windows operating system or are downloaded or installed from third parties. What most other browsers treat as "plug-ins" are implemented as ActiveX controls for IE/Windows. Although IE5/Macintosh offers a modicum of support for ActiveX, the controls devised for Windows do not run under other operating systems, nor have IE/Windows built-in controls (such as the basic Web Browser Control, on which many IE/Windows features are built) been implemented as such in IE/Mac.

As a result, a few powerful Windows-only features are not available on the Mac version. These include Data Binding (direct connectivity to server data sources via the Data Source Object), CSS filters (for transition visual effects and text effects), DHTML Behaviors (the action-oriented equivalent to external style sheet definitions), IE 5.5 content editing, executing external commands (via the execCommand() method), and the TextRange object (allowing script control of arbitrary blocks of text independent of their HTML containers).

Admittedly, this is a long list of features not built into IE/Mac. Most of them are also missing from Netscape Navigator 6 on all OS platforms because (with the exception of the similar, but incompatible TextRange object) these features are not part of the W3C DOM standards yet. Therefore, if you want to deploy these features on a public Web site's pages, apply them as additive, rather than mission-critical, aspects of the page. For example, you can apply transitions between pages of your site, but only IE/Windows users will see the effect.

Speaking of plug-ins, you should be aware that IE/Mac manages plug-ins much the same way as Netscape Navigator. In other words, there is a one-to-one relationship between any given MIME type and a plug-in installed in the browser. Therefore, if you embed an audio element in your page, the browser will use whichever plug-in is currently enabled for the MIME type of the audio file that comes from the server. This differs from the IE5/Windows (and W3C DOM) approach, whereby an OBJECT element invokes a specific plug-in to play the file (assuming the desired plug-in is installed). One benefit of using the Netscape approach, however, is that IE5/Mac supports the same plug-in detection technique as Navigator, complete with navigator.plugins and navigator.mimeTypes arrays whose objects report the same property values as Netscape Navigator. For many scripters, this kind of plug-in detection is much easier than the roundabout way scripts have to validate IE/Windows media player facilities -- the latter sometimes requiring a few lines of Windows-only Visual Basic Script (VBScript) to complete the job.

Now.CaptureEvents();

Compared to Explorer 5, versions 5.5 and 6 for Windows nearly double the number of event types that HTML element objects recognize. For instance, several event handler types in IE5 signal user-dragging motions with respect to an element like,

onDrag,
onDragEnd,
onDragEnter,
onDragLeave,
onDragOver,
onDragStart and.
onDrop

Another event group provides script control over cutting, copying, and pasting content (onBeforeCopy, onBeforeCut, onBeforePaste, onCopy, onCut, onPaste). And still another event type (onContextMenu) makes it easy to interrupt the system-produced context (right-click on the Windows mouse) menu and create custom menus.

In contrast, the IE/Macintosh event types stays within the W3C DOM. Thus, you have the range of mouse, keyboard, and system-generated events. Speaking of events, the IE/Macintosh event object is a property of the window object, just as it is in IE/Windows. In IE5/Mac, however, the event object's properties include not only the IE4 properties, but also some W3C DOM event object properties, such as event.charCode (character and key codes are not necessarily the same values) and event.target.

Continue

Far more difficult to predict are differences in the ways the Windows and Macintosh versions of IE render HTML elements that are not absolute-positioned and explicitly sized (in pixels). For example, form buttons render very differently on each OS platform, affecting not only the look of the form control, but also the space it occupies on the page. Additional differences in the ways the browsers report page scrolling and event coordinates can yield unexpected results. Complicating the situation even further is that your page's layout or design scheme can influence these differences. This makes it difficult to offer simple workarounds or golden rules that apply to all designs. Even so, you should be on the lookout for key mismatches.

For example, if you employ element positioning in your page, you should experience little or no problems if you use the style object's positioning properties (style.left, style.posLeft, style.top, style.posTop) exclusively in your script manipulation of the elements. Avoid trying to mix and match regular element positions (as exposed by properties such as elemObject.clientLeft and elemObject.clientTop) with style object positioning properties. The offset-related properties of non-positioned elements report widely different values in the two OS browser versions, especially if the non-positioned element is a component of a TABLE element. For example, attempting to position a floating element atop a TD element that is nested inside a TABLE will be difficult to accomplish across browsers without a great deal of browser-specific hand tweaking of values.

Reading mouse event coordinates in an element requires slightly different treatment in Mac and Windows versions of IE, especially if there is a chance that the page containing the element can be scrolled. At issue here is determining the coordinates of the event within the element. The event.offsetX and event.offsetY properties in IE/Windows accurately report the offsets within the element. IE/Mac does, too, provided the page holding the element hasn't scrolled. If the page has scrolled, however, then the scrolling values (as determined by the document.body.scrollTop and document.body.scrollLeft properties) are automatically subtracted from the event coordinates before they are reported. Thus, for the Mac version, you must add back the document.body scroll values to the event coordinates:

var leftInsideElem = event.offsetX + document.body.scrollLeft

var topInsideElem = event.offsetY + document.body.scrollTop

(!Properties && Methods())

In addition to the Windows-only features outlined earlier, a handful of Microsoft DOM properties and methods that are implemented in IE5/Windows (and some newer ones implemented in IE5.5/Windows) are not supported in IE5/Mac. If you have been developing pages for sites that must also accommodate IE4, then you are probably not using any of these IE5-only properties and methods. But if you have been limiting access to IE5 or later browsers, then see the table below for a list of the more popular properties and methods that you might be using but are not available in IE5/Mac.

Unsupported Properties and Methods in IE5/Mac

Object Type Property Method
HTML Elements canHaveChildren attachEvent()
RuntimeStyle componentFromPoint()
uniqueID detachEvent()
getExpression() replaceAdjacentText() setExpression()
document recalc()
form -------- autocomplete
table -------- moveRow()

Document.write("It is Intresting");

Contrary to IE/Win, IE/Mac's "iframe" (Dynamic Frame) element has the higher z-index value than "Div" element. What does it mean? It means that if you are using any dynamic pop up menu or Sliding Menu on a page where there is a iframe, the menu will go behind the iframe. Same, if you remember, like those annoying Select Boxes coming over all elements.

There is only one way to escape from this. Use iframe to contain the script or Layer and give the higher z-index value to this iframe.

Another very interesting fact I found was about Layer element hiding option. Suppose you are using a popup-menu or sliding menu. You define the function for Layer Visibility to Show or Hide. IE5/Mac does the hiding of the Layer but it doesn't terminate the Layer entirely from the page. Even after hiding the layer, it keeps that Layer's place occupied and in result it may not let you to click or select anything in that region. You might wonder, what is wrong that you are not able to click buttons or not able to insert value in text boxes etc. Look out for any layer, which may be invisibly existing on the page Like Mr.India (Mogambo are you listening?).

Macintosh.Test()

It may be tempting to pick up a used Mac to be your testbench. If you do so, be sure it has the minimum requirements to run IE5: a PowerPC CPU and MacOS 7.6.1 or later. You will also save yourself a ton of development time if you can have the test Mac on a local area network with your development PC. Third-party products let Wintel and Macintosh computers share files and directories across an Ethernet or LocalTalk network. This setup allows you to edit only one source code file (or set of related files) in your favorite OS and test the results quickly -- ideally with a Wintel and Macintosh computer side by side. Include the Macintosh in your testing matrix as early in the development cycle as possible, especially if your scripting touches on any of the technical areas discussed earlier.

About Raybiztech

Raybiztech is a leading Global Information Technology (IT) Services and Solutions, a CMMI Level 3, ISO 27001:2013 and ISO 9001:2015 Certified Company. We are a Member of NASSCOM, HYSEA, NJTC, and AIIA. Raybiztech offers comprehensive end-to-end IT Services for Business Application Development, Enterprise Solutions, Enterprise Collaboration Services, Testing and Quality Assurance Services, Cloud Computing and IT Infrastructure Management to organizations in the Banking & Finance, Insurance, Healthcare, Manufacturing, Retail, Media & Entertainment, Leisure & Travel, Telecom and Energy & Utilities verticals as well as Independent Software Vendors.