Designing Web Sites for Phone Browsers

April 2010

Microsoft Corporation
Rev. 2.0

Contents

Designing Web Sites for Phone Browsers

These guidelines are provided to assist developers in optimizing Web content for mobile browsers, particularly for Internet Explorer® Mobile. Whether or not you are experienced in the art of Web content design and application development, you should read this document to understand what users expect of mobile content and how to design and build Web sites that meet or exceed their expectations.

Understanding the Mobile Context

You should assume that at some point your content will go mobile. With the dramatic increase in mobile Web traffic and consumer expectations of available high-quality content, developers are presented with a new set of opportunities and challenges. Device form factors and capabilities will vary, for example between laptop, netbook, tablet computer, and smart phone. It is important to note that we cannot simply categorize mobile devices by a single characteristic such as small screen size. This paper will later describe how to target Internet Explorer Mobile, where characteristics such as screen size and orientation come into play.

We will first examine the broad context which draws a line between stationary and mobile Internet browsing. Stationary browsing occurs in a fixed location, with or without privacy, single-tasking or multi-tasking, and with expectations of consistent availability, network speed, and environmental conditions such as noise and lighting. The mobile browsing experience does not offer those same guarantees. That said, best practices are in place for considering, adapting, and targeting content for mobile devices. Thus, the remaining bulk of this paper is divided into four sections.

- **Considering Mobile Browsers**: This section shares best practices for being considerate of the mobile browsing experience without dependencies on specific browsers or devices.
- **Detecting Mobile Browsers**: Whether you are reusing content and adapting existing Web pages or creating new content targeted for the phone browser, the techniques described in this section may be relevant and useful.
- **Adapting for Mobile Browsers**: This section explains how something as simple as adding a single HTML tag to an existing Web page can enhance the mobile browsing experience. Also, by customizing the browsing experience with CSS and JavaScript, developers can deliver the same content with progressive enhancements to both desktop and mobile browsers.
- **Targeting Mobile Browsers**: This approach may involve repurposing or retailoring Web pages, but focuses more on how to create new content specifically for viewing in mobile browsers such as Internet Explorer Mobile.

The approaches described in these sections are not mutually exclusive, in that for any given Web site each page or sub-element may or may not be designed to target mobile devices on a case by case basis.

Considering Mobile Browsers

From a pure content perspective, Web developers have traditionally focused on who, what, when, and why information is accessed. Mobile browsing raises the stakes on prior considerations, and brings renewed awareness to where and how content is browsed.

- **Who**: Mobile people who are away from their home and office. Use cookies to store and reuse common session data and save the user time during subsequent visits.
- **Where**: Consider that network data connections are inconsistent, limited, and potentially expensive. Also consider environmental conditions: loud or quiet; bright or dark; sitting or walking; with or without privacy.
- **What**: Yours, or someone else's content. Remember that you are competing for the attention of someone on the go. Be crisp, clear, and succinct. Avoid small fonts, subtle colors, complicated splash or introductory screens, large image files, and unsupported third party plugins. You should also design intelligently to avoid sending unused or irrelevant style sheets and script files.
- **When**: Given accessibility to their devices, mobile users could browse your site at any time of the day. Depending on the scope of your content, consider optimizing for the date and time.
- **Why**: Consider that mobile users are likely either in need of a solution, or bored and casually browsing.
- **How**: Consider that mobile users are often focused on single-tasking and want quick access to perform targeted actions. Don't assume access to a keyboard or mouse. Consider that mobile users will interact with a stylus or finger gestures, so try to minimize the required input.

Design for One Web

In the simplest case, your site may be simple or fluid enough not to need mobile adaptation. If so, Internet Explorer Mobile will render the entire Web page zoomed-out and allow the user to zoom in on an area of interest. Internet Explorer Mobile assumes a default page width of 1024 and scales accordingly so that the entire page is in view. It is not expected that mobile users will be granted the same presentation or interactivity through a given site, but at a minimum all content should be visible even without applying style sheets or scripting. Here are some best practices for general Web design with a focus on the mobile context.

- Use well defined standards such as HTML4.0, XHTML1.0, XHTML-MP, ECMAScript 3, AJAX (i.e. XMLHttpRequest), DOM Level 1, and Cascading Style Sheets (CSS) Level 2.1.
- Use widely supported static content such as JPEG, GIF, and PNG, as well as rich media formats such as WMV, AVI, 3GP, MP4, and MP3.

Caution:

Internet Explorer Mobile currently supports streaming media via HTTP, but not through the Real-Time Streaming Protocol (RTSP) or Microsoft Media Server (MMS) protocol. Also note that rich media files are currently recognized and handled by their file type extension, and not by their Multipurpose Internet Mail Extension (MIME) types, so make sure that the file extension is the last text within the URL string for such links.

- In consideration of small mobile browsers, use short and descriptive page titles.
- Always use a DOCTYPE to ensure that your content is displayed as intended.
- Separate content, presentation, and behavior to ease the path towards mobile adoption.
- Clearly identify the file type or resulting action of links that would take the user out of the browser or otherwise break the user browsing experience.
- Use CSS instead of tables for layout positioning. Using tables to position elements results in overly complicated code that may not abide by the principle of separating content from presentation. It is also more difficult to create table-based layouts that respond well to the varying screen sizes of mobile devices, particularly ones that are able to change their dimensions.
- When you must use tables to display tabular data, avoid nested tables and limit the number of columns in consideration of small screen sizes.
- Although supported within Internet Explorer Mobile, avoid using image maps since interactions within small mobile browsers can be limited and challenging.
- Avoid using frames and framesets since they would occupy too much of the display area on small mobile browsers. Internet Explorer Mobile expands each frame by default to eliminate scroll bars and optimize the browsing experience.

Detecting Mobile Browsers

Whether you are reusing content and adapting existing Web pages or creating new content targeted for the phone browser, device detection can help to ensure a rich mobile Web browsing experience on a range of mobile devices. Browser detection can be implemented both on the client and the server. Client-side script is used to detect device and browser characteristics and capabilities, while script on the Web server examines the incoming user agent string from the browser. Additional techniques, including tiered browser support and third party databases will be discussed later in this section.

Client-Side Browser Detection

A best practice for client-side browser detection is to test for the presence of particular criteria. For example, to query whether or not the browser supports DOM Level 1 use a script such as the following:

```
function hasBasicDOM() {
     // Check for support of DOM Level 1 functions
     if (document.getElementById && document.getElementsByTagName)
          return true;
     return false;
}
```

A series of similarly themed tests could be run which focus purely on what the device and browser can or cannot do.

Server-Side Browser Detection

The most common server-side detection practice is to examine the user agent. For example, the user agent string for Internet Explorer Mobile is the following:

```
Mozilla/4.0 (compatible; MSIE 7.0; Windows Phone OS 7.0; Trident/3.1;
IEMobile/7.0) <DeviceManufacturer>;<DeviceModel>
```

Thus, one way of detecting Internet Explorer Mobile would be to look for the “IEMobile/7” sub-string. In practice, this kind of string search would be part of a larger detection scheme that segments incoming user agents into up-level and down-level tiers, and delivering the HTML content along with the corresponding style sheets and script resources to give the user the richest experience as appropriate. However, there are some pitfalls to consider when relying on the user agent string:

- Complex logic could mistakenly exclude rich media capable browsers and devices.
- Maintenance of a user agent string list as new browsers and devices come to market.
- Devices could modify the user agent string and resemble another device or become unrecognizable.

Tiered Browser Support

Since the user agent string cannot be relied on independently, it is recommended to segment your intended supported browsers into discrete levels or tiers. Some browsers may receive a richer set of content and presentation elements, while less capable browsers display a down-level experience. For more information about such progressive enhancements, read the **Adapting for Mobile Browsers** section later in this paper. There are various thresholds which draw the line between browser tiers, but a general rule of thumb is that rich media browsers meet the following criteria:

- ECMAScript 3
- W3C DOM Level 1
- W3C standard box model support
- CSS2 rendering
- Client-side cookies support
- AJAX XMLHttpRequest object support

The ASP.NET Mobile Device Browser File

The Mobile Device Browser File (MDBF) is a freely available third-party database file that can be downloaded from CodePlex at http://mdbf.codeplex.com that contains data on more than 1,500 mobile devices. Once this file is installed on your ASP.NET server, you can use the Request.Browser object to detect capabilities through dozens of browser properties. Here is a simple example:

```
/* Query whether or not this is a mobile device */
if (Request.Browser.IsMobileDevice)
     /* This is a mobile device */
else
     /* This is not a mobile device */
```

Most requests take the form of Request.Browser[“PropName”], where PropName corresponds to the queried property. Here is an example to identify the input system that the device supports:

```
// Returns “keyboard”, “telephoneKeypad”, or “virtualKeyboard”
string sInputType = Request.Browser[“InputType”];
```

The MDBF file is continuously maintained, so be sure to check back for updates. A full list of supported capabilities is documented at the MDBF page on CodePlex.

Adapting for Mobile Browsers

This section explains the techniques that you can use to adapt existing desktop Web content to work in the mobile context. At a basic level, Internet Explorer Mobile enables users to zoom in on an area of interest. Simple steps can be taken to provide an even better experience. One of the easiest practices is through the use of mobile ready tags. HTML `<meta>` tags can be inserted within the `<head>` content section of a Web page and provide specific information about display capabilities. From oldest to newest, the mobile ready tags are `HandheldFriendly`, `MobileOptimized`, and `Viewport`.

Tip:

The HandheldFriendly and MobileOptimized tags are not required to target content for Internet Explorer Mobile, for which the Viewport tag is preferred. To be compatible with a wider range of browsers, it is recommended to use all three tags in the order listed above.

HandheldFriendly

The HandheldFriendly tag indicates whether or not the content is designed for a mobile browser. Setting this tag value as `true` signals that the content is mobile or handheld friendly. This tag also tells mobile browsers that the content should be displayed without applying scaling.

```
<META name=”HandheldFriendly” content=”true” />
```

MobileOptimized

The MobileOptimized tag provides the browser with an integer that corresponds to the intended display width of the screen. This tag is also used by search engines to determine if the page is mobile-optimized. When this tag is set, the browser forces the page into a single-column layout at the width that is specified by the tag, and prevents the layout engine from attempting to fit the content on the screen.

```
<META name=”MobileOptimized” content=”240” />
```

Viewport

The viewport is a rectangular region that controls how the document’s content is laid out and where text will wrap on the page. The default display width for Internet Explorer Mobile is 1024 pixels, so if your Web page width is less than 1024 pixels you should set the viewport width accordingly.

```
<meta name=”viewport” content=”width=320”/>
```

Internet Explorer Mobile supports the following viewport property and value paired settings.

Property name	Value
`width`	Sets the width of the viewport. Can be any integer number or `device-width`. The range is from 320 to 10,000. The default value is 320.
`height`	Sets the height of the viewport. Can be any integer number or `device-height`. The range is from 480 to 10,000.
`user-scalable`	Indicates whether or not the user can scale the viewport, or in other words whether or not they can zoom in and out within content. `yes` or `no` The default and recommended value is ‘`yes`’.

Note:

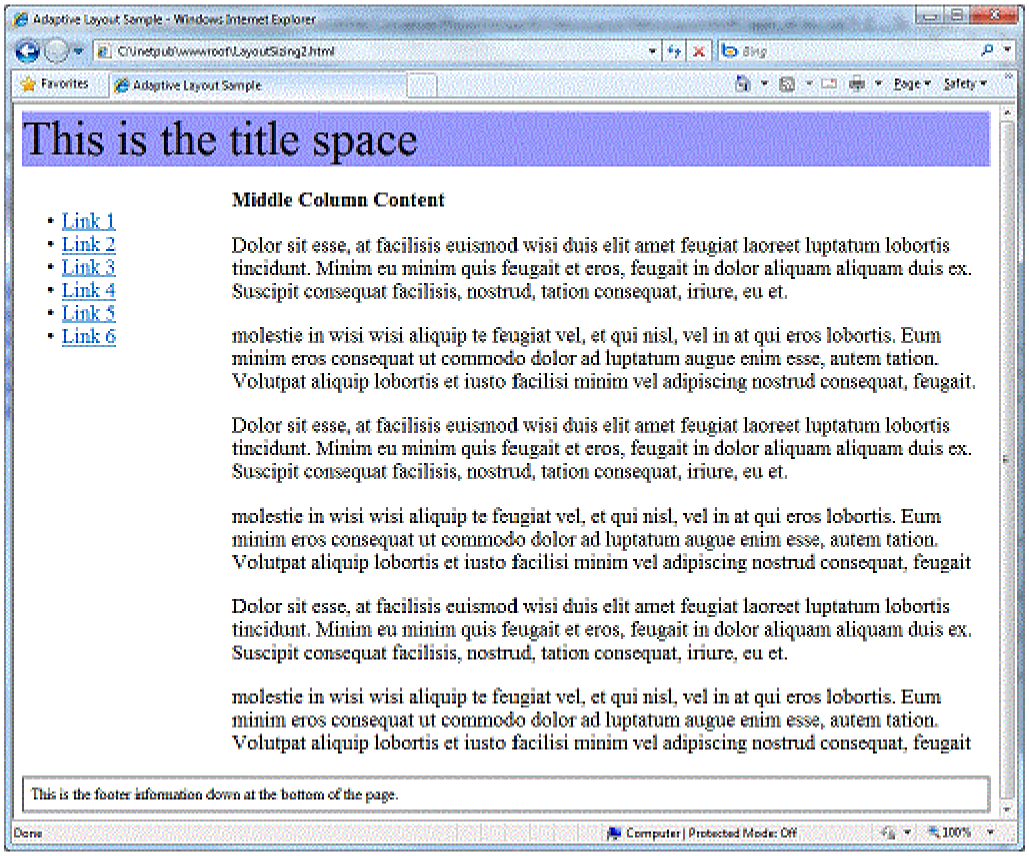
While available in some browsers the `minimum-scale`, `maximum-scale`, and `initial-scale` properties are currently unsupported for Internet Explorer Mobile.

Progressive Enhancement with Script and CSS

By using a combination of separate Cascading Style Sheets (CSS) for each medium and writing some JavaScript, you can serve the same pages to different browsers and have each one display properly.

Consider a common Web layout that uses an unordered list to provide navigation links in the left-hand column with the main Web content in the middle column, and accompanied by a page header and footer. Displaying such a page within a mobile browser would not result in an optimal layout, since the left-hand navigation column takes up a lot of wasted space, and the font size for the content and header is rather large. The page would fit within a mobile browser more efficiently if the navigation links were across the top between the header and the main content column.

The following image shows the example layout on a desktop browser.


Adaptive Layout Sample - Windows Internet Explorer
C:\inetpub\wwwroot\LayoutSizing2.html
Bing
Favorites
Adaptive Layout Sample
Page
Safety
Done
Computer | Protected Mode: Off
100%

This is the title space

- Link 1
- Link 2
- Link 3
- Link 4
- Link 5
- Link 6

Middle Column Content

Dolor sit esse, at facilisis euismod wisi duis elit amet feugiat laoreet luptatum lobortis tincidunt. Minim eu minim quis feugait et eros, feugait in dolor aliquam aliquam duis ex. Suscipit consequat facilisis, nostrud, tation consequat, iriure, eu et.

molestie in wisi wisi aliquip te feugiat vel, et qui nisl, vel in at qui eros lobortis. Eum minim eros consequat ut commodo dolor ad luptatum augue enim esse, autem tation. Volutpat aliquip lobortis et iusto facilisi minim vel adipiscing nostrud consequat, feugait.

Dolor sit esse, at facilisis euismod wisi duis elit amet feugiat laoreet luptatum lobortis tincidunt. Minim eu minim quis feugait et eros, feugait in dolor aliquam aliquam duis ex. Suscipit consequat facilisis, nostrud, tation consequat, iriure, eu et.

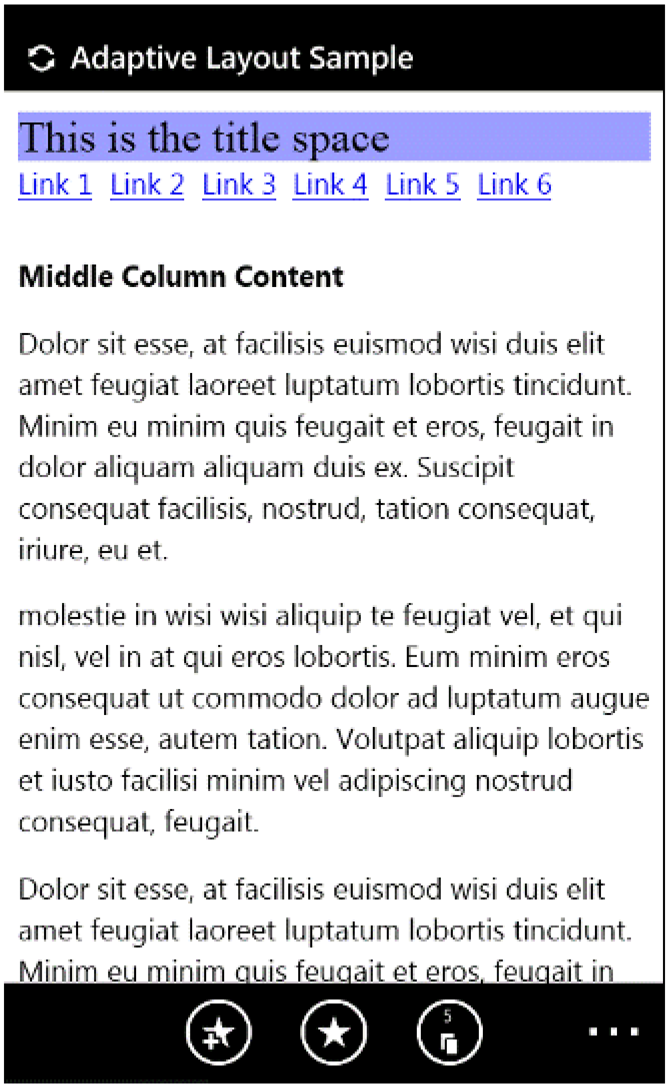
molestie in wisi wisi aliquip te feugiat vel, et qui nisl, vel in at qui eros lobortis. Eum minim eros consequat ut commodo dolor ad luptatum augue enim esse, autem tation. Volutpat aliquip lobortis et iusto facilisi minim vel adipiscing nostrud consequat, feugait

Dolor sit esse, at facilisis euismod wisi duis elit amet feugiat laoreet luptatum lobortis tincidunt. Minim eu minim quis feugait et eros, feugait in dolor aliquam aliquam duis ex. Suscipit consequat facilisis, nostrud, tation consequat, iriure, eu et.

molestie in wisi wisi aliquip te feugiat vel, et qui nisl, vel in at qui eros lobortis. Eum minim eros consequat ut commodo dolor ad luptatum augue enim esse, autem tation. Volutpat aliquip lobortis et iusto facilisi minim vel adipiscing nostrud consequat, feugait

This is the footer information down at the bottom of the page.

The following image shows the example layout on Internet Explorer Mobile.


Adaptive Layout Sample

This is the title space

Link 1 Link 2 Link 3 Link 4 Link 5 Link 6

Middle Column Content

Dolor sit esse, at facilisis euismod wisi duis elit amet feugiat laoreet luptatum lobortis tincidunt. Minim eu minim quis feugait et eros, feugait in dolor aliquam aliquam duis ex. Suscipit consequat facilisis, nostrud, tation consequat, iriure, eu et.

molestie in wisi wisi aliquip te feugiat vel, et qui nisl, vel in at qui eros lobortis. Eum minim eros consequat ut commodo dolor ad luptatum augue enim esse, autem tation. Volutpat aliquip lobortis et iusto facilisi minim vel adipiscing nostrud consequat, feugait.

Dolor sit esse, at facilisis euismod wisi duis elit amet feugiat laoreet luptatum lobortis tincidunt. Minim eu minim quis feugait et eros, feugait in

You can design such a layout without having to build and maintain two separate pages of Web content by creating two distinct CSS style sheets – one for the desktop and one for the mobile browser. The following style sheet defines the desktop layout:

```
.bodyNormal #masthead {
     font-size:36pt;
     font-family:"Times New Roman", serif;
     background-color:#9999FF
}
.bodyNormal #container {
     font-size: 16pt;
     position: relative;
     width: 100%;
}
.bodyNormal #left_col {
     width: 200px;
     height: 100%;
     float:left;
}
.bodyNormal #page_content {
     margin-left: 210px;
}
.bodyNormal #footer {
     border: 2px gray solid;
     padding: 5pt;
     margin-top: 5pt;
}
```

The following style sheet defines the mobile layout:

```
.bodySmall #masthead {
     font-size:24pt;
     font-family:"Times New Roman", serif;
     background-color:#9999FF
}
.bodySmall #container {
     font-size: 16pt;
     position:inherit;
     width: 100%;
}
.bodySmall #left_col {
     display:block;
     width:100%;
}
.bodySmall #left_col ul {
     margin-left:0pt;
}
.bodySmall #left_col li {
     display:inline;
     margin-right:5pt;
     list-style-type:none;
}
.bodySmall #page_content {
     display:block;
}

.bodySmall #footer {
     border: 1px black solid;
     padding: 5pt;
}
```

To serve each corresponding style sheet to the appropriate device browser, this short script can detect the screen width of the display:

```
function setStyleForSize() {
     if (screen.width <= 480) {
          document.body.className = "bodySmall";
     }
     else {
          document.body.className = "bodyNormal";
     }
}

function addEventHandler(oNode, evt, oFunc) {
     if (typeof(window.event) != "undefined")
          oNode.attachEvent("on"+evt, oFunc);
     else
          oNode.addEventListener(evt, oFunc, true);
}

addEventHandler(window, "load", function() { setStyleForSize(); } );
addEventHandler(window, "resize", function() { setStyleForSize(); } );
```

When the screen width is less than 480, the mobile styles are applied to the document's body content. Otherwise, the regular styles are applied, rendering the full desktop experience.

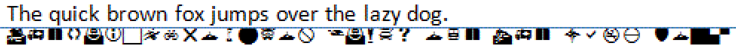
Targeting Mobile Browsers

This approach may involve repurposing or retailoring Web pages, but focuses more on how to create new content specifically for viewing in mobile browsers such as Internet Explorer Mobile. The method to most directly bring your content to the mobile Web is to intentionally create pages that are specifically targeted for the mobile Web and served from a separate place on your site, such as a mobile-designated subfolder (http://www.example.com/m/) or from a mobile sub-domain (http://m.example.com). Typically, you would choose this option when the page that you want to serve to the user is too complex to adapt using only style sheets or simple client-side script.

The following sections of this document describe some best practices for developing Web site content specifically for a mobile browser and assume that you do not intend to repurpose the content for display on a desktop browser. Specific best practices for displaying content in Internet Explorer Mobile are discussed where applicable.

Default to Available Fonts

Since Internet Explorer Mobile does not support downloadable fonts, a font will be chosen from within Internet Explorer Mobile that attempts to match the desired font and eventually fall back to either Serif or Sans-serif. Thus, you should create mobile Web pages that use the fonts that are provided by default within Internet Explorer Mobile and are illustrated in the following table.

Font	Appearance
Arial	The quick brown fox jumps over the lazy dog.
Arial Black	**The quick brown fox jumps over the lazy dog.**
Comic Sans MS	The quick brown fox jumps over the lazy dog.
Courier New	`The quick brown fox jumps over the lazy dog.`
Georgia	The quick brown fox jumps over the lazy dog.
Lucida Sans Unicode	The quick brown fox jumps over the lazy dog.
Times New Roman	The quick brown fox jumps over the lazy dog.
Trebuchet MS	The quick brown fox jumps over the lazy dog.
Verdana	The quick brown fox jumps over the lazy dog.
Tahoma	The quick brown fox jumps over the lazy dog.
Calibri	The quick brown fox jumps over the lazy dog.
Webdings	

Be Aware of CSS Properties in the Mobile Context

If not used carefully, some CSS properties could cause layout problems for mobile Web pages. The following are some properties and values that should be used with care:

- `float`: Although supported by many mobile browsers, content that uses float may not look good on a small screen.
- `padding` and `margin`: These are often used to add space within and around layout elements. Use these properties sparingly since screen space is limited and elements are already competing for display area.
- `background-image`: This assigns a background image to a layout element such as a page, which can particularly reduce readability on small screens. When multi-serving content to both desktops and mobile devices, use CSS to assign an appropriately sized image to the corresponding browser.
- Use measurements such as points, ems, or percentages instead of absolutes: Using absolute values for measurements such as pixels will cause content to display incorrectly when the user scales the content or rotates the device.

Avoid Popup Windows

Pop-up windows are commonly used in desktop Web pages to present help content, forms to be filled out, or advertising pitches. The same pop-ups can lead to bad experiences on mobile devices for the following reasons:

- Since mobile Web pages are full-screen by design, it is very jarring to the user to suddenly have the entire screen taken over by another window. It interrupts their flow of interaction, and since there are no windows with title bars, it’s difficult to switch among windows.
- Many mobile browsers place a limit on the number of windows that can be opened at any one time, and usually no more than 10. If multiple windows or tabs are already open when a script tries to open another one, the operation may fail without a proper explanation to the user.

For a better user experience, try to include as much relevant information on one page as you can. If the mobile device supports rich HTML and JavaScript, consider using DHTML to display a dialog instead of opening a new window. If you insist on navigating to another page, make it clear and easy for the user to get back to the current page.

Avoid Page Auto-Refresh

Pages that auto-refresh are not desired by end users who must pay for network data or access the Web over slow connections. Instead of using auto-refresh, use AJAX to asynchronously retrieve updated data from the server and display it in the browser. If you must use auto-refresh, consider using an IFRAME to minimize the amount of content that must be refreshed. If this approach is not feasible, then fall back to using a META tag that refreshes the page. If your page has an auto-refresh capability, you should consider providing the user with a feature that turns off the refreshing or allows them to control how often the data is refreshed.

Redirect from the Server

Users should be redirected to alternate pages from the server rather than from the client. Performing a redirect on the client increases response time since another request must be issued to the server. Network data usage also increases since the redirected page must be downloaded in addition to the original page.

Hover and Mouse-over Events

Many desktop-oriented pages use mouseover or the CSS :hover pseudo-class to trigger events such as popup menus. Touch-based devices do not use a mouse as their primary pointing device, and as such cannot easily respond to these kinds of events. Your mobile Web pages should avoid the use of hover-like events in order to provide primary access to the features of your site. Ensure that features can be reached using only a single-tap gesture.

Handling Key Events

Some Web sites examine key events to simulate an onchange event for edit fields or to attempt to mask the input of certain characters.

- With Internet Explorer Mobile, the order of key events differs slightly from the desktop because of the way that the virtual keyboard sends events. The desktop browser sends the events in the order of KeyDown, KeyPress, KeyUp, whereas within Internet Explorer Mobile the order is KeyDown, KeyUp, then KeyPress. If your site depends on this order, you should make sure that you are accounting for it on your mobile pages.
- If you are attempting to detect content changes, for example within an edit box to display a drop-down list of suggestions, it is recommended to use the onchange event or to set up a timer using setTimeout().

Handling Screen Orientation Changes

Most modern mobile devices allow the user to hold the device in landscape mode in order to automatically rotate the view to take advantage of the wider screen size. The browser responds to this event by notifying the Web page that the window has been resized. Some mobile browsers implement a custom event to handle display orientation changes. Internet Explorer Mobile sends an onresize event to the page when the orientation changes. The client script within the page can retrieve the page dimensions to determine whether it is in portrait or landscape mode. Here is an example:

```
function getScreenOrientation()
{
     // If available, use the orientation property.
     if (window.orientation) {
          if (window.orientation == 90 || window.orientation == -90)
               return “landscape”;
          else return “portrait”;
     }
     else {
          // Get the screen size and determine if it is in portrait
          // or landscape mode by comparing the width and height.
          var wid = screen.width;
          var ht = screen.height;
          if (wid > ht) return “landscape”;
          else return “portrait”;
     }
}
```

Expanded JavaScript

Internet Explorer Mobile implements the following native JavaScript functions, which are not applicable to Internet Explorer 7 for the desktop.

Function	**Description**
`getElementsByClassName(name)`	Given the class `name` parameter, returns all elements or sub-elements within scope of this function call.
`querySelector(selector)`	Given the query `selector` parameter, returns the first element within the sub-tree of the scope of this function call.
`querySelectorAll(selector)`	Given the query `selector` parameter, returns a list of all elements within the sub-tree of the scope of this function call.

XHTML Page Rendering

Internet Explorer Mobile enables end users to view your pages with .xhtml extensions. In Internet Explorer 7 for the desktop, users are prompted to save the file locally since the .xhtml file type is not recognized.

Adjusting Text Size with Custom CSS

Internet Explorer Mobile gives developers the option to control the Web page text size by setting the `–ms-text-size-adjust` CSS property. When the end user double-taps a page element, Internet Explorer Mobile scales the viewport to position the double-tapped element within the visible area of the screen. The corresponding text is also scaled to a legible size. Developers may choose to either control the adjusted text size or turn it off.

Note:

The `–ms-text-size-adjust` property will be ignored where the viewport tag is present.

The following example turns off text size adjustment for an HTML page:

```
html { -ms-text-size-adjust:none }
```

The following example turns on automatic text size adjustment for the body of an HTML page:

```
body { -ms-text-size-adjust:auto }
```

The following example adjusts the text size by 150 percent for a division of an HTML page:

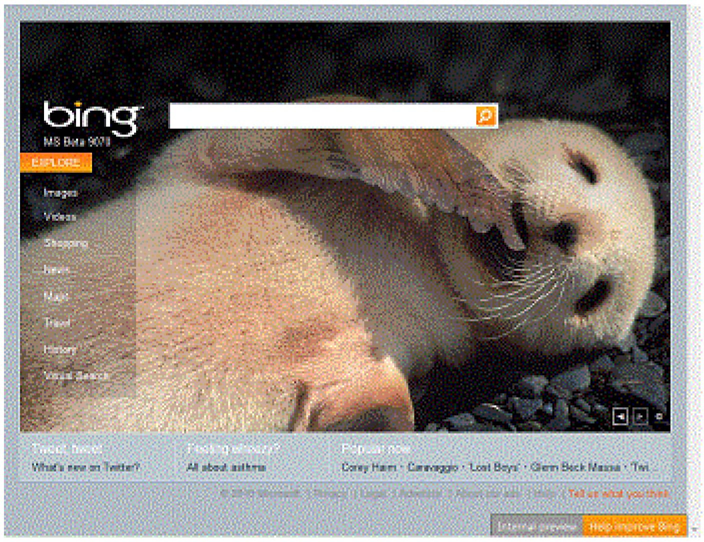
```
div { -ms-text-size-adjust:150% }
```

Note:

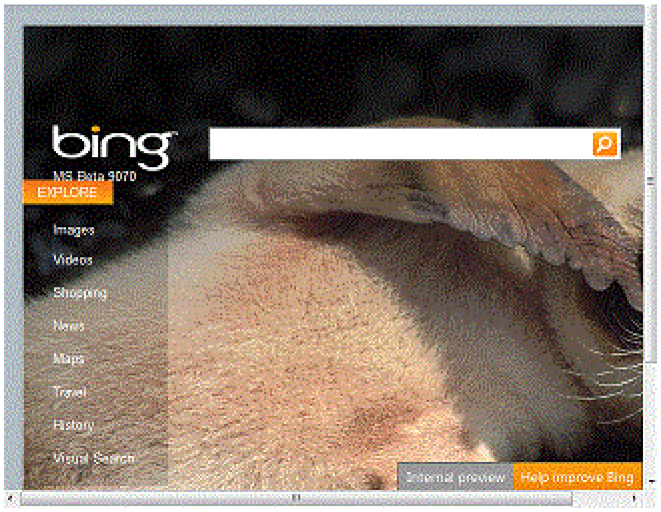
Internet Explorer Mobile also supports the `-webkit-text-size-adjust` property name in place of `-ms-text-size-adjust`.

Behavior of Fixed Positioning

Elements that are positioned using CSS fixed positioning behave differently on Internet Explorer Mobile compared to Internet Explorer 7. On the desktop, fixed elements are positioned relative to the client view of the browser window. This means that if an element is fixed in view at the bottom right-hand corner, it will always appear fixed in view at the bottom right-hand corner even when the end user scrolls the page up, down, left, or right. The following image shows a fixed element in the bottom right-hand corner of a desktop browser.



The following image demonstrates that on a desktop browser the fixed element remains in the bottom right-hand corner.



In Internet Explorer Mobile, fixed elements are positioned relative to the document itself and may not remain in view when the user scrolls the page. This means that if an element is fixed relative to the bottom right-hand corner of a document, the user will only see it when they navigate to the bottom right-hand corner of the Web page.

Note:

When a user zooms in or out of a document, fixed elements will scale accordingly.

Unsupported Plugins

Unlike the desktop browser, Internet Explorer Mobile does not allow end users to download or install third-party plugins such as ActiveX controls. Web pages should not prompt users to download plugins. Additionally Internet Explorer Mobile does not support DHTML or binary behaviors.