# ACCESSIBLE JAVASCRIPT

## 1. Creating Accessible JavaScript - http://www.webaim.org/techniques/javascript/

### JavaScript Accessibility Issues

JavaScript allows developers to add increased interaction, information processing, and control in web-based content. However, JavaScript can also introduce accessibility issues. These issues include:

- **Navigation.** Inability or difficulty navigating using a keyboard or assistive technology.
- **Hidden content.** Presentation of content or functionality that is not accessible to assistive technologies.
- **User control.** Lack of user control over automated content changes.
- **Confusion/Disorientation.** Altering or disabling the normal functionality of the user agent (browser) or triggering events that the user may not be aware of.

A web page containing JavaScript will typically be fully accessible if the functionality of the script is device independent (does not require only a mouse or only a keyboard) and the information (content) is available to assistive technologies. Unfortunately, there is no easy fix that can be applied to solve all accessibility problems associated with JavaScript. The only way to ensure JavaScript accessibility is by evaluating each individual script and devising a unique solution to the accessibility problem it poses. Developers must be familiar with the issues surrounding JavaScript accessibility and apply techniques to do one or both of the following:

1. Ensure the JavaScript is directly accessible
2. Provide an accessible, non-JavaScript alternative

### JavaScript that does not impact accessibility

Just because JavaScript is utilized on a page does not mean that the page is inaccessible. In many cases, JavaScript can be used to increase accessibility. Additional information, warnings, or instructions can be given to users through JavaScript prompts. For instance, under the Section 508 guidelines of United States law, a user must be notified when a timed response is required and given

sufficient time to indicate more time is required. Such functionality would be difficult with HTML alone.

JavaScript is sometimes used to create visual interface elements that do not affect accessibility. JavaScript is commonly used for image rollovers, where one image is replaced with another when the mouse is placed above it; for example, when a navigation item changes to display a shadow, glow, or highlight when the user mouses over it.

Place your mouse over the following image to see a JavaScript example that does not impact accessibility.


## Problems

None. In this example, there is no important content or functionality introduced by the JavaScript. The swapping of images is purely cosmetic.

## Solution

No additional accessibility techniques are required. Remember, the image itself must have alternative text (i.e., <img alt="home"...>). You can also accomplish image rollovers without using JavaScript - external link.

Such uses of JavaScript do not need additional accessibility features incorporated because important content is not displayed or functionality introduced by such scripting.

### Disabling JavaScript

Follow the directions to disable or enable JavaScript in your browser. You can also determine if JavaScript is disabled. Test a JavaScript enabled web page and see if the content and functionality are accessible. Be sure to re-enable JavaScript when you're done.

**Internet Explorer 6.X**

1. Open Internet Explorer.
2. Select **Tools > Internet Options**.
3. In **Internet Options** dialog box select the **Security** tab.
4. Click **Custom Level** button at bottom. The **Security Settings** dialog box will pop up.

5. Under the **Scripting** category, enable/disable **Active Scripting, Allow paste options via script** and **Scripting of Java applets**.
6. Click **OK** twice to close out.
7. Click **Refresh**.

**Netscape 7.X**

1. Open Netscape.
2. Select **Edit > Preferences**.
3. Click the arrow next to **Advanced**.
4. Click **Scripts & Plugins**.
5. Check/uncheck **Navigator** beneath "Enable Javascript for".
6. Click **OK**.
7. Click **Reload**.

**Opera 7.X**

1. Open Opera.
2. Select **File > Quick Preferences**.
3. Check/uncheck **Enable Javascript**.
4. Click **Reload**.

## 2. Java Accessibility and Usability Work - http://trace.wisc.edu/world/java/java.htm

## Java Application Accessibility Examples

Please download the instructions (readme.txt) on how to setup your environment for running these examples. Also for a short description of what each example is about, read the text file with the same name as the Java source file.

1. Test.java (download)
   - Test.htm
2. SimpleExample.java (download)
   - SimpleEx.htm
3. SimpleExample2.java (download)
   - SimplEx2.htm
4. SimpleExample3.java (download)

    - SimplEx3.htm
5.
6. Dial.java (download)
    - dial.htm
    7.
    - Additional notes: joptionp.htm

### Java Applet Accessibility Examples

1. Freezing marquee text

## 3. Creating Accessible JavaScript (Alternatives) - http://www.webaim.org/techniques/javascript/alternatives.php

## Using the <noscript> Element

Making JavaScript natively accessible is very important. However, in some cases, the end user may not have JavaScript enabled or may be using technologies that do not support JavaScript (e.g., cell phone, PDA, etc.). In such cases, you can provide non-JavaScript alternatives to user's who cannot or choose not to view JavaScript content.

When JavaScript is used within a Web page, the most straightforward way to provide an alternative for the JavaScript-generated content is to provide content within the <noscript> element. The <noscript> element can be used within your page to display content in browsers that do not support or have disabled JavaScript. However, if JavaScript IS enabled the <noscript> element is ignored.

Providing an accessible alternative within the `<noscript>` element for an inaccessible script will not make the page accessible. The `<noscript>` content will only display if JavaScript is disabled. Most screen reader users have JavaScript enabled, and will thus encounter your inaccessible script and not the `<noscript>` content.

<noscript> should be used anytime alternative or non-javascript content or functionality is required.

```
<script type="text/javascript">
<!-- document.write("The current time is " + currenttime) -->
</script>
<noscript>
<!-- link to page that displays time from server-side script -->
```

```
<a href="time.htm">View the current time</a>
</noscript>
```

## 4. Creating Accessible JavaScript (Other Issues) - http://www.webaim.org/techniques/javascript/other.php

### Pop-up Windows

Pop-up windows provide a unique accessibility problem. First of all, most usability experts would argue against the use of pop-up windows except in the most extreme of cases. If you must use pop-up windows, know that they introduce several very unique accessibility issues. For a visual user, it may be difficult to notice and navigate to the new window. For someone who is using assistive technologies, the new window may be annoying and confusing because the default behavior for the link has been modified. JavaScript implementation may make the new window difficult or impossible to resize or scale for someone using a screen enlarger. For someone who is blind, there is typically no indication that they are presently navigating in a new window. When the screen reader user attempts to return to the previous page by selecting the back button, it may be confusing to find that this does not work.

#### Pop-up example

Select this link to open a new window

```
<a href="popup.htm"
onclick="window.open(this.href); return false;">Select this...</a>
```

## 5. Creating Accessible JavaScript (JavaScript Event Handlers) - http://www.webaim.org/techniques/javascript/eventhandlers.php

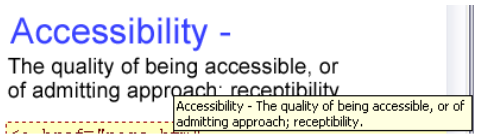
### onMouseOver and onMouseOut

The onMouseOver event handler is triggered when the mouse cursor is placed over an item. As its name implies, onMouseOver requires the use of a mouse, thus it is a device dependent event handler and may cause accessibility issues. onMouseOver, and its companion, onMouseOut, can be used, as long as any important content or functionality is also available without using the mouse.

**Example 1**

Place your mouse over the following image to see an example of onMouseOver. When the mouse is placed over the image of the word "Accessibility", another image appears in its place which presents the definition of the word "Accessibility".

Accessibility

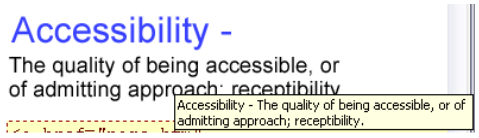
to mouse over 


```
<a href="page.htm" onmouseover="document.images['myimage'].src='imageoff.gif';"
onmouseout="document.images['myimage'].src='imageon.gif';"> <img
src="media/imageoff.gif" width="289" height="79" id="myimage" alt="Accessibility - The quality of
being accessible, or of admitting approach; receptibility." /> </a>
```

## Solutions

In addition to onMouseOver and onMouseOut, use onFocus and onBlur. These actions will be triggered when the keyboard is used to navigate to and from a link that surrounds the <img> element.

Accessibility

to mouse over 


```
<a href="page.htm" onmouseover="document.images['myimage'].src='imageon.gif';"
onfocus="document.images['myimage'].src='imageoff.gif';"
onmouseout="document.images['myimage'].src='imageoff.gif';"
onblur="document.images['myimage'].src = 'imageoff.gif';"> <img src="imageoff.gif" width="289"
height="79" id="myimage" alt="Accessibility – The quality of being accessible, or of admitting
approach; receptibility." /></a>
```

You must still provide the descriptive alternative text for non-visual users, but onFocus and onBlur will provide the visual image swap for users that are using a keyboard only.