Advanced Web Accessibility

High Tech Center Training Unit

Of the California Community Colleges at the
Foothill-De Anza Community College District

21050 McClellan Road
Cupertino, CA 95014
(408) 996-4636

www.htctu.fhda.edu

3/8/2007

Table of Contents

Cascading Style Sheets, Level 1

Overview

Cascading Style Sheets (CSS) are a method of web design that formats web page content according to a presentation style specified by the web page author. There are several advantages to using CSS to format the presentation elements of a web page. CSS essentially separates document content from the manner in which it is presented, thus allowing for more fluid transitions between various browser platforms. CSS also provides for more precise control for spacing, alignment, and positioning of content without relying on the need for layout tables or frames. Font style, color, and font size can all be manipulated using CSS as well.

Unfortunately, not all browsers render style sheets in a similar manner. Additionally, earlier browsers may try to interpret CSS code as part of the HTML, thus rendering the document unreadable. This chapter will discuss the benefits and limitations of using CSS for web design.

Accessibility Benefits

CSS offers the potential for users to customize their web browser such that the content displayed in their preferred Internet browser will present the information in the manner the user chooses. For a low-vision individual, this provides the opportunity to enlarge web content displayed in the internet browser at a resolution appropriate to the individual. Using CSS provides the user with the option of manipulating the way in which the web content is presented in their specific Internet browser.

Types of CSS

There are essentially four different types of style sheets that can impact the manner in which content is displayed.

Inline

Inline style sheets apply the formatting elements locally – that is, at the level of the text within the web page document. An inline style sheet may contain the following:

```
<h1 style="font-family:sans-serif; font-size:20pt;
color:#996633">
```

Internal

Internal style sheets reside within the **<head>** section of an HTML document and affect the presentation of content throughout the document. An internal style sheet may contain the following:

```
<head>
<style>
<!—
body {font-family:Helvetica, sans-serif; font-size:extra-
large; color:orange}
-->
</ style>
</ head>
```

External

External style sheets exist outside of the document structure and are referenced when the document (or web page) is loaded into the browser. External style sheets have the advantage of being easy to change in order to affect a large number of web pages simultaneously. An external style sheet may contain the following:

```
h2 {font-family: Helvetica, sans-serif;
     font-size: extra-large;
     color: wheat
}

p {font-family: serif;
     font-size: 14;
     color: black;
}
```

This specifies all level 2 headings will be Helvetica font (or another sans-serif font), extra-large size, and a wheat color. All paragraph content will be a serif font (the default on the computer), 14 point, and a black color.

Imported

Imported style sheets are similar to **External** style sheets in that the attributed styles are *imported* into the document within the **`<Style>`** tag. An imported style sheet could include the following:

```
STYLE TYPE="text/css" MEDIA="screen, projection">
<!--
  @import url(http://www.htmlhelp.com/style.css);
  @import url(/stylesheets/punk.css);
  DT { background: yellow; color: black }
-->
</STYLE>
```

Style Sheet Formatting

The syntax for a linked style sheet is contained in a text document with an extension of **.css**. The **.css** files remain on the web server and are linked to their respective web page documents. The syntax for style sheets is the following:

```
h1  {property1: value1;
        property2: value1, value2;
        property3: value1;
}

h2  {property1: value1;
        property2: value1;
}

P  {property1: value1;
        property2: value1;
        property3: value1;
}
```

Elements

The elements are those for which you are setting a style. For instance, in the previous example, the Heading 1, Heading 2, and Paragraph functions would receive specific formatting as set by the style sheet.

Element Name	**Element Code**
Headings:	H1, H2, H3, H4, H5, H6
Body Text:	P, BLOCKQUOTE
Hyperlinks:	A
Lists:	UL, LI, OL

Properties

Properties represent the category of the desired presentation appearance. Properties include descriptions for the font-family, font-style, color, etc. These properties are then followed by the appropriate value for this category. A list of the different properties and the associated values is in the next section.

Values

Values are what specify how the property is to be displayed within the style sheet. Each property has several values from which to choose in order to present the content according to the web designers intent. A list of the different values with their associated properties is available in the next section.

Font Properties:

Syntax:	Allowed Values:
font-family	serif (e.g., Times) sans-serif (e.g., Arial) cursive (e.g., *Zapf-Chancery*) fantasy monospace (e.g., `Courier`) or specific font name

Example code:

```
P { font-family: "New Century Schoolbook", Times, serif }
```

Syntax:	Allowed Values:
font-style	normal *italic* *oblique*

Example code:

```
H1 { font-style: oblique }
P  { font-style: normal }
```

Syntax:	Allowed Values:
font-variant	normal small-caps

Example code:

```
SPAN { font-variant: small-caps }
```

Syntax:	Allowed Values:
font-weight	normal **bold** **bolder** lighter 100-500 (lighter) **600-900 (darker)**

Example code:

```
H1 { font-weight: 800 }
```

Syntax:	Allowed Values:
font-size	absolute-size (xx-small, x-small, small, medium, large, x-large, xx-large) relative-size (larger, smaller) length (pt) percentage (%)

Example code:

```
H1 { font-size: large }
P  { font-size: 12pt }
LI { font-size: 90% }
```

Color and Background Properties

Syntax:	Allowed Values:
color	The color values can be defined using a RGB color chart or a verbal representation of a particular color: (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow, wheat)

Example code:

```
H1 { color: blue }
H2 { color: #000080 }
```

Syntax:	Allowed Values:
background-color	color-value transparent

Example code:

```
BODY { background-color: white }
H1   { background-color: #000080 }
```

Syntax:	Allowed Values:
background-image	<url> none

Example code:

```
BODY { background-image: url(/images/foo.gif) }
P    { background-image: url(http://www.htmlhelp.com/bg.png) }
```

Syntax:	Allowed Values:
background-repeat	repeat repeat-x repeat-y no-repeat

Example code:

```
BODY { background: white url(candybar.gif);
       background-repeat: repeat-x }
```

The background-repeat property determines how a specified background image is repeated. The repeat-x value will repeat the image horizontally while the repeat-y value will repeat the image vertically

Text Properties

Syntax:	Allowed Values:
word-spacing	normal length format (e.g., relative or absolute)

Example code:

```
P EM   { word-spacing: 0.4em }

P.NOTE { WORD-SPACING: -0.2EM }
```

Syntax:	Allowed Values:
letter-spacing	normal length format (e.g., relative or absolute)

Example code:

```
H1     { letter-spacing: 0.1em }
P.note { letter-spacing: -0.1em }
```

Syntax:	Allowed Values:
text-decoration	none underline overline ~~line-through~~ blink

Example code:

```
A:link, A:visited, A:active { text-decoration: none }
```

Syntax:	Allowed Values:
vertical-align	baseline sub super top text-top middle bottom text-bottom percentage (%)

Percentage will move the element the specified percentage from the baseline (negative values can be used for subscripts).

Example code:

```
IMG.middle { vertical-align: middle }
.exponent  { vertical-align: super }
```

Syntax:	Allowed Values:
text-transform	none Capitalize (capitalize first letter in each word) UPPERCASE (capitalize all the words) lowercase (uses small letters for each word)

Example code:

```
H1 { text-transform: uppercase }
H2 { text-transform: capitalize }
```

Syntax:	Allowed Values:
text-align	left right center justify

Example code:

```
H1          { text-align: center }
P.newspaper { text-align: justify }
```

Syntax:	Allowed Values:
text-indent	length (relative or absolute) percentage (%)

Example code:

```
P { text-indent: 5em }
```

Syntax:	Allowed Values:
line-height	normal number (value) length (relative or absolute) percentage (%)

The line-height property will accept a value to control the spacing between baselines of text. When the value is a number, the line height is calculated by multiplying the

element's font size by the number. Percentage values are relative to the element's font size. Negative values are not permitted.

Example code:
```
P { line-height: 200% }
```

List Properties

Syntax:	Allowed Values:
list-style-type	none disc ○ circle ● square ■ decimal 1 2 3 etc. lower-roman i ii iii etc. upper-roman I II III etc. lower-alpha a b c etc. upper-alpha A B C etc.

The list-style-type property specifies the type of list-item marker, and is used if list-style-image is none or if image loading is turned off.

Examples:
```
LI.square { list-style-type: square }
UL.plain  { list-style-type: none }
OL        { list-style-type: upper-alpha } /* A B C */
```

Syntax:	Allowed Values:
list-style-image	none <url of image>

The list-style-image property specifies the image that will be used as list-item marker when image loading is turned on, replacing the marker specified in the **list-style-type** property.

Examples:
```
UL.check { list-style-image: url(/LI- markers/checkmark.gif) }
UL LI.x  { list-style-image: url(x.png) }
```

Syntax:	Allowed Values:
list-style-position	inside outside

The list-style-position property takes the value inside or outside, with outside being the default. This property determines where the marker is placed in regard to the list item. If the value inside is used, the lines will wrap under the marker instead of being indented.
Examples:
Outside rendering:

* **List item 1**
  second line of list item

Inside rendering:

* **List item 1**
second line of list item

Syntax:	Allowed Values:
list-style	<list-style-type> <list-style-position> <url>

The list-style property is a shorthand for the list-style-type, list-style-position, and list-style-image properties.

Examples:

```
LI.square { list-style: square inside }
UL.plain  { list-style: none }
UL.check  { list-style: url(/LI-markers/checkmark.gif) circle }
OL        { list-style: upper-alpha }
OL OL     { list-style: lower-roman inside }
```

Length and Percentage Units

(from www.w3c.org)

Length units
The format of a length value is an optional sign character ('+' or '-', with '+' being the default) immediately followed by a number (with or without a decimal point) immediately followed by a unit identifier (a two-letter abbreviation). After a '0' number, the unit identifier is optional.

Some properties allow negative length units, but this may complicate the formatting model and there may be implementation-specific limits. If a negative length value cannot be supported, it should be clipped to the nearest value that can be supported. There are two types of length units: relative and absolute. Relative units specify a length relative to another length property. Style sheets that use relative units will more easily scale from one medium to another (e.g. from a computer display to a laser printer).

These relative units are supported:

```
H1 { margin: 0.5em }      /* ems, the height of the element's font */
H1 { margin: 1ex }        /* x-height, ~ the height of the letter 'x' */
P  { font-size: 12px }    /* pixels, relative to canvas */
```

The relative units 'em' and 'ex' are relative to the font size of the element itself. The only exception to this rule in CSS1 is the 'font-size' property where 'em' and 'ex' values refer to the font size of the parent element.

Pixel units, as used in the last rule, are relative to the resolution of the canvas, i.e. most often a computer display. If the pixel density of the output device is very different from that of a typical computer display, the UA should rescale pixel values. The suggested *reference pixel* is the visual angle of one pixel on a device with a pixel density of 90dpi and a distance from the reader of an arm's length. For a nominal arm's length of 28 inches, the visual angle is about 0.0227 degrees.

Child elements inherit the computed value, not the relative value:

```
BODY {
  font-size: 12pt;
  text-indent: 3em;  /* i.e. 36pt */
}

H1 { font-size: 15pt }
```

In the example above, the 'text-indent' value of 'H1' elements will be 36pt, not 45pt.

Absolute length units are only useful when the physical properties of the output medium are known. These absolute units are supported:

```
H1 { margin: 0.5in }      /* inches, 1in = 2.54cm */
H2 { line-height: 3cm }   /* centimeters */
H3 { word-spacing: 4mm }  /* millimeters */
H4 { font-size: 12pt }    /* points, 1pt = 1/72 in */
H4 { font-size: 1pc }     /* picas, 1pc = 12pt */
```

Percentage units
The format of a percentage value is an optional sign character ('+' or '-', with '+' being the default) immediately followed by a number (with or without a decimal point) immediately followed by "%".

Percentage values are always relative to another value, for example a length unit. Each property that allows percentage units also defines what value the percentage value refer to. Most often this is the font size of the element itself:

P { line-height: 120% } /* 120% of the element's 'font-size' */

Color

(from www.w3c.org)

A color is either a keyword or a numerical RGB specification. The suggested list of keyword color names is:

aqua	black
blue	fuchsia
gray	green
lime	maroon
navy	olive
purple	red
silver	teal
white	yellow

These 16 colors are taken from the Windows VGA palette, and their RGB values are not defined in this specification.

```
BODY {color: black; background: white }
H1 { color: maroon }
H2 { color: olive }
```

The RGB color model is being used in numerical color specifications. These examples all specify the same color:

```
EM { color: #f00 }              /* #rgb */
EM { color: #ff0000 }           /* #rrggbb */
EM { color: rgb(255,0,0) }      /* integer range 0 - 255 */
EM { color: rgb(100%, 0%, 0%) } /* float range 0.0% - 100.0% */
```

The format of an RGB value in hexadecimal notation is a '#' immediately followed by either three or six hexadecimal characters. The three-digit RGB notation (#rgb) is converted into six-digit form (#rrggbb) by replicating digits, not by adding zeros. For example, #fb0 expands to #ffbb00. This makes sure that white (#ffffff) can be specified

with the short notation (#fff) and removes any dependencies on the color depth of the display.

The format of an RGB value in the functional notation is 'rgb(' followed by a comma-separated list of three numerical values (either three integer values in the range of 0-255, or three percentage values in the range of 0.0% to 100.0%) followed by ')'. Whitespace characters are allowed around the numerical values.

Values outside the numerical ranges should be clipped.

```
EM { color: rgb(255,0,0) }       /* integer range 0 - 255 */
EM { color: rgb(300,0,0) }       /* clipped to 255 */
EM { color: rgb(110%, 0%, 0%) }  /* clipped to 100% */
```

Classes of Style

An advantage of using CSS for web pages is the ability to globally affect the presentation of specific throughout a website. However, there is always the situation in which certain elements require different formatting. This can be accomplished by using the class element in order to specify the final presentation of select elements.

The following code would apply a large, sans-serif, italicized font to all first-level headings.

```
H1      {font-family: Arial, sans-serif;
         font-style: italic;
         font-size: large;
}
```

However, if you wanted to have some of the first-level heading to appear in a blue, serif font, it would be necessary to create a class the specifies the desired appearance.

```
.specialh1      {font-family: Times, serif;
                 color: blue;
}
```

In the HTML code, it would be necessary to call in this special class in order to display this specific presentation style.

```
<h1>This is an example of the regular Heading 1 style</h1>

<h1 class="specialh1">This is an example of the special Heading 1
style</h1>
```

Using the `<div>` tag, a designer can also apply specific styles to whole blocks of content on a web page. Content within the `<div>` tag would change the appearance to that specified by the class element.

Example:

```
<div class=”specialh1”>
     <p>Paragraph content would go here…..</p>
     <table>
          Table data would be formatted as well….
     </table>
</div>
```

Inheritance

Another advantage to using CSS in web page design is the ability of different styles to *inherit* appearance values from previous levels. For instance, in the previous example discussing the Class property, the Heading 1 style was a large, sans-serif, italicized font.

```
H1      {font-family: Arial, sans-serif;
         font-style: italic;
         font-size: large;
}
```

We also created a special class that changed the font-family to Times and the color to blue.

```
.specialh1      {font-family: Times, serif;
                 color: blue;
}
```

However, because we did not specify differences to the font-style and the font-size, the .specialh1 class will retain those property values when displaying the web page. This is due to inheritance.

Contextual selectors

(from www.w3c.org)

Inheritance saves CSS designers typing. Instead of setting all style properties, one can create defaults and then list the exceptions. To give 'EM' elements within 'H1' a different color, one may specify:

```
H1 { color: blue }
EM { color: red }
```

When this style sheet is in effect, all emphasized sections within or outside 'H1' will turn red. Probably, one wanted only 'EM' elements within 'H1' to turn red and this can be specified with:

```
H1 EM { color: red }
```

The selector is now a search pattern on the stack of open elements, and this type of selector is referred to as a *contextual selector*. Contextual selectors consist of several simple selectors separated by whitespace (all selectors described up to now have been simple selectors). Only elements that match the last simple selector (in this case the 'EM' element) are addressed, and only if the search pattern matches. Contextual selectors in CSS1 look for ancestor relationships, but other relationships (e.g. parent-child) may be introduced in later revisions. In the example above, the search pattern matches if 'EM' is a descendant of 'H1', i.e. if 'EM' is inside an 'H1' element.

```
UL LI    { font-size: small }
UL UL LI { font-size: x-small }
```

Here, the first selector matches 'LI' elements with at least one 'UL' ancestor. The second selector matches a subset of the first, i.e. 'LI' elements with at least two 'UL' ancestors. The conflict is resolved by the second selector being more specific because of the longer search pattern. Contextual selectors can look for element types, CLASS attributes, ID attributes or combinations of these:

```
DIV P          { font: small sans-serif }
.reddish H1    { color: red }
#x78y CODE     { background: blue }
DIV.sidenote H1 { font-size: large }
```

The first selector matches all 'P' elements that have a 'DIV' among the ancestors. The second selector matches all 'H1' elements that have an ancestor of class 'reddish'. The third selector matches all 'CODE' elements that are descendants of the element with 'ID=x78y'. The fourth selector matches all 'H1' elements that have a 'DIV' ancestor with class 'sidenote'.

Several contextual selectors can also be grouped together:

```
H1 B, H2 B, H1 EM, H2 EM { color: red }
```

Which is equivalent to:

```
H1 B { color: red }
H2 B { color: red }
H1 EM { color: red }
H2 EM { color: red }
```

The ! Important Property

Using the **! important** property, web designers can force specific styles to appear on a web page. The **! important** declaration will overide all other CSS rules for the specified property and value. Thus, the web page authors can create style sheet rules that will overide those as set by users.

Example:

```
H1 { color: black ! important; background: white ! important }
P  { font-size: 12pt ! important; font-style: italic }
```

This will format the heading 1 styles to have a black text color with a white background. Additionally, all paragraphs will have a 12pt font-size, but may or may not have a font-style of italics.

It is not recommended that web designers use the **! important** property to format web page content. One advantage to using CSS is that there is some flexibility in the manner in which web content is rendered. A low-vision user could create their own style sheet in order to make the on-line content more usable for their needs. The following style sheet is just one example that may be useful to a low-vision user.

```
H1      { font-family: Verdana ! important;
          font-style: normal ! important;
          font-size: xx-large ! important;
          color: black ! important;
}

P       { font-family: Times ! important;
          font-style: normal !important;
          font-size: large ! important;
          color: yellow ! important;
          background: black ! important;
}
```

InFocus 4.1.2

Publisher

SSB Technologies
645 Harrison Street, Suite 204
San Francisco, CA 94107

Phone: (415) 975 8000
Fax: (415) 975 8038

Email: contact@ssbtechnologies.com

Retail Cost: $899 (go through FCCC)

System Requirements:

The InFocus software from SSB Technologies provides for several different operating system platforms. The necessary information is listed below:

Windows - Recommended Setup

- Pentium II - 300 MHZ+
- 96-128 Megabytes of RAM

Linux - Intel - System Requirements

- Pentium II - 300 MHZ+
- 128 Megabytes of RAM
- JRE for Java 2 - JDK version 1.3 or greater
- XFree86 4.0 or current X Server

Solaris - System Requirements

- 300 MHZ or greater processor speed
- 128 Megabytes of RAM
- JRE version 1.3

Description:

InFocus 4.1.2 is a web evaluation, reporting, and fixing tool designed to make the necessary changes to HTML structure to insure accessibility. InFocus compares the

underlying HTML code of a web page with the standards set forth by the Access Board. These standards, the US Section 508 Standards for Electronic and Information Technology, identify the methods necessary to increase access to web-based content.

To diagnose and fix a web page:

1. Open the **File** menu from the Menu Bar.

2. Select **Open URL** to check a live web page or **Open File** to check a local page.

3. The selected page or file will appear under the **Diagnosed Pages** in the left frame.

4. Choose the accessibility issue from the list under **Diagnosed Pages**.

5. Follow the directions under the **Fix Violation** tab in the bottom right frame to change the HTML code to improve the accessibility.

6. The **Fix Information** tab provides information on how to make the necessary changes to the web page to increase its accessibility.

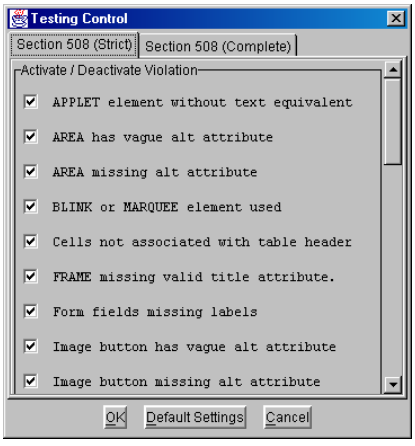
7. The **Violation Information** tab provides context-specific information on the US Section 508 Standards for Electronic and Information Technology and why the HTML code was identified as a potential accessibility error.

8. Save the file to before exiting the program.

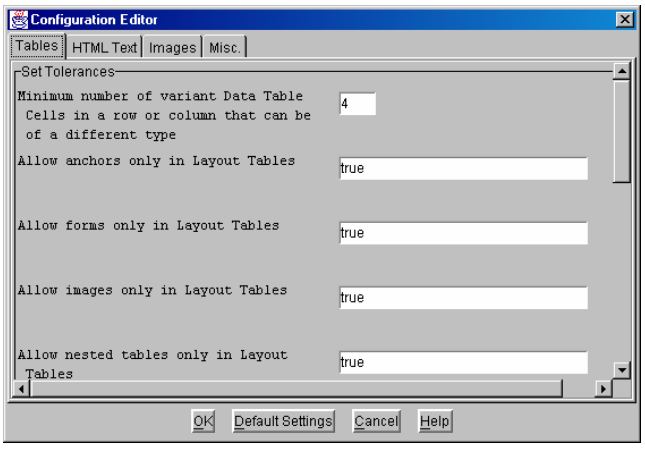
Configuring the Evaluation

InFocus 4.1.1 provides the capability of customizing the evaluation checklist in order to check for specific accessibility errors. There are several methods that one can use to evaluate web content.

Use the **Testing Control Panel** (**Ctrl-T**) to vary the web page evaluation. This provides the user with the flexibility to search for and fix specific accessibility errors.



Use the **Configuration Editor** (**Ctrl-G**) to set evaluation test standards for Tables, HTML Text, Images, Applets and Color accessibility issues.



Report Types
SSB software creates reports in HTML, XML, and Excel to provide users with an array of options. HTML is best suited for easy viewing of reports and understanding the overall compliance level of a set of pages. In addition, the reports can easily be posted to an Intranet or secure web site to track a compliance effort. XML and Excel offers flexibility for exporting information to a database. This allows users effective project management options.

Creating a Report
Users can create a report on either a single page or set of pages that have already been diagnosed. To create the reports, select **Reporting** and **Generate Reports** (**Ctrl-R**) from the menu bar. This will allow you to choose the type of report (HTML, XML, Excel) and the pages on which you want to run the report. To select multiple pages, choose the first page, then hold down the **Shift** key and scroll up or down with your arrow keys.

It is also possible to generate charts using the InFocus software. Select **Reporting** from the menu bar and then choose **Generate Chart**. Identify the web page (or pages) and select the type of chart you wish to display; Pie Chart or Bar Graph. The chart will then identify the accessibility errors contained on the page as well as any "fixes" that have been applied. These charts can be saved in either a .JPEG or .PNG graphic file format.

Saving a Report
You can choose to have your report saved to your hard drive. By default, the report will be saved to a temporary file, and deleted when you are finished. However, if you would like to save your report, enter the location into the area below the history tree in the Generate Report dialog. You can select **Choose File** to navigate your hard drive or type in the file location directly. The report will then be saved so that you can access it later.

Previewing Page Content
InFocus 4.1.2 allows the web page designer to preview a web page with select elements removed in order to evaluate the potential usability and accessibility issues that can arise when using assistive computer technology or different browsers. In order to use this function, select the web page to evaluate and then chose one of the options from **Preview** on the menu bar. These options are listed below:

Style Sheets Off (Ctrl+F5)
This will preview the web page as if style sheets are not enabled or supported by the user's browser. Requiring style sheets can limit the ability of an individual to gain access to page content (e.g., older browser will not render style sheets or will render information incorrectly). The web page content should still be readable without requiring a style sheet according to Part D of the US Section 508 Standard.

Linearized (Ctrl+F6)
Selecting linearization simulates how a screen-reader may render the web page content to the user. Linearization removes any table formatting (both layout table formatting and data table formatting) and left-justifies the page content. While this is a simulation as to how a screen-reader may render a web page, it should not necessarily be substituted for a through evaluation.

Grayscale (Ctrl+F7)
(Windows operating system only) This feature is useful for checking content that may be described in color alone. Web pages should be able to be displayed such that information can be conveyed regardless of color requirements. This is required under Part C of the US Section 508 Standard.

Javascript Off (Ctrl+F8)
It is necessary to evaluate a page to determine if the page can still be read with scripts disabled. If it is not possible to access the page content with javascript disabled, it is important to include a <NOSCRIPT> function, which can provide an alternate means of accessing the web page content. This is required under Part L of the US Section 508 Standards.

Media Preview (Ctrl+F9)
Media preview allows the user to view the page to identify multimedia content and if this information requires an alternative method to provide content. According to Part B of the US Section 508 Standards, multimedia content on a web page is required to have a synchronized alternative.

Blinking Elements (Ctrl+F10)
Blinking elements on a web page can pose difficulty to individuals sensitive to visual stimuli. It is recommended to avoid blinking elements with a frequency between 2Hz and 55Hz. This is required under Part J of the US Section 508 Standard.

Federal Accessibility Standards for Web-based Intranet and Internet Information and Applications

(Provided by www.usability.gov)
This page contains excerpts from Electronic and Information Technology Accessibility Standards issued by the ARCHITECTURAL AND TRANSPORTATION BARRIERS COMPLIANCE BOARD.
Shown below are:
Paragraphs from the Overview of the Standards
Subpart B — Technical Standards: Sec. 1194.22 Web-based intranet and internet information and applications.
This page only contains an excerpt of the summary and standards that directly relate to Web sites. Other parts of the standard may apply to your situation. See the complete, officially posted standards at http://www.access-board.gov/news/508-final.htm.

Electronic and Information Technology Accessibility Standards
ARCHITECTURAL AND TRANSPORTATION BARRIERS COMPLIANCE BOARD
[Published in the Federal Register on December 21, 2000]

Summary:
Web-based Intranet and Internet Information and Applications (1194.22)
The criteria for web-based technology and information are based on access guidelines developed by the Web Accessibility Initiative of the World Wide Web Consortium. Many of these provisions ensure access for people with vision impairments who rely on various assistive products to access computer-based information, such as screen readers, which translate what's on a computer screen into automated audible output, and refreshable Braille displays. Certain conventions, such as verbal tags or identification of graphics and format devices, like frames, are necessary so that these devices can "read" them for the user in a sensible way. The standards do not prohibit the use of web site graphics or animation. Instead, the standards aim to ensure that such information is also available in an accessible format. Generally, this means use of text labels or descriptors for graphics and certain format elements. (HTML code already provides an "Alt Text" tag for graphics which can serve as a verbal descriptor for graphics). This section also addresses the usability of multimedia presentations, image maps, style sheets, scripting languages, applets and plug-ins, and electronic forms.
The standards apply to Federal web sites but not to private sector web sites (unless a site is provided under contract to a Federal agency, in which case only that web site or portion covered by the contract would have to comply). Accessible sites offer significant advantages that go beyond access. For example, those with "text-only" options provide a faster downloading alternative and can facilitate transmission of web-based data to cell phones and personal digital assistants.

§ 1194.22 Web-based intranet and internet information and applications.
(a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).

(b) Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

(c) Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.

(d) Documents shall be organized so they are readable without requiring an associated style sheet.

(e) Redundant text links shall be provided for each active region of a server-side image map.

(f) Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

(g) Row and column headers shall be identified for data tables.

(h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

(i) Frames shall be titled with text that facilitates frame identification and navigation.

(j) Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

(k) A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.

(l) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.

(m) When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).

(n) When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

(o) A method shall be provided that permits users to skip repetitive navigation links.

(p) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

Note to §1194.22: 1. The Board interprets paragraphs (a) through (k) of this section as consistent with the following priority 1 Checkpoints of the Web Content Accessibility Guidelines 1.0 (WCAG 1.0) (May 5, 1999) published by the Web Accessibility Initiative of the World Wide Web Consortium:

Section 1194.22 Paragraph	WCAG 1.0 Checkpoint
(a)	1.1
(b)	1.4
(c)	2.1
(d)	6.1
(e)	1.2
(f)	9.1
(g)	5.1
(h)	5.2
(i)	12.1
(j)	7.1
(k)	11.4

2. Paragraphs (l), (m), (n), (o), and (p) of this section are different from WCAG 1.0. Web pages that conform to WCAG 1.0, level A (i.e., all priority 1 checkpoints) must also meet paragraphs (l), (m), (n), (o), and (p) of this section to comply with this section. WCAG 1.0 is available at http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505.

Web Accessibility Resources

Cascading Style Sheets Information

Accessibility Features of Cascading Style Sheets
http://www.w3.org/TR/CSS-access

W3C Standard for Cascading Style Sheets, Level 1
http://www.w3.org/TR/REC-CSS1

Using Style Sheets in HTML documents
http://www.w3.org/TR/html4/present/styles.html

Cascading Style Sheets Tutorial
http://www.w3schools.com/css/default.asp

W3C HTML Validation Service
http://validator.w3.org/

W3C CSS Validation Service
http://jigsaw.w3.org/css-validator/

Style Sheet Reference Guide Master Grid
http://www.webreview.com/style/css1/charts/mastergrid.shtml

Evaluation and Repair Tools

SSB Technologies Web Evaluation and Repair Tool (InFocus 4.1.1)
http://www.ssbtechnologies.com/index.php

LIFT for Macromedia DreamWeaver Evaluation and Repair Tool
http://www.usablenet.com/lift_dw/lift_dw.html

XHTML Resources

W3C XHTML Recommendation
http://www.w3.org/TR/xhtml1/

XHTML Tutorial
http://www.w3schools.com/xhtml/default.asp