



# Web Author Workshop B: Manual Accessibility Checks

Web Accessibility Working Group  
CSULA Accessible Technology Initiative  
Fall 2011, Version 2.0

## Table of Contents

<b>Introduction.....</b>	<b>2</b>
<b>Downloading the Data Files .....</b>	<b>2</b>
<b>Manual Checks.....</b>	<b>2</b>
Visual Check .....	2
Blinking Text and Marquees .....	4
Applets .....	5
Tables .....	6
Table Summary Attribute.....	6
Styles .....	6
Disabled Styles and Style Sheets .....	6
Headings .....	7
Linearizing the Page.....	8
Displaying Link Details .....	9
JavaScript and Accessibility.....	11
Page Navigation .....	12
Keyboard Navigation .....	12
Skip to Main Content .....	13
Visibility Issues .....	13
Checking a Web Page for Proper Color Contrast .....	13
Low Vision User Test .....	16
Code Validity and Doctypes .....	17
What Does Valid Code Mean?.....	17
Adding a Doctype.....	17
Validating HTML.....	17
Validating CSS .....	18
Deprecated Code .....	20
Legacy Web Design.....	20
Frames .....	20
Image Maps .....	21
<b>Additional Help.....</b>	<b>21</b>
<b>Additional Resources.....</b>	<b>21</b>

# Introduction

This handout is designed for web authors and University personnel who create or maintain CSULA websites. It is a continuation of the **Web Author Workshop A: Accessibility Tools and Reports** handout. It is meant to be used as one of the tools to assist in developing and maintaining web pages that conform to web accessibility requirements. Topics covered in this handout include repairing common navigation issues, ensuring proper color contrast, and checking for valid HTML and CSS code.

## Downloading the Data Files

This handout includes sample data files that can be used for hands-on practice. The data files are stored in a self-extracting archive. The archive must be downloaded and executed in order to extract the data files.

- The data files used with this handout are available for download at <http://www.calstatela.edu/its/training/datafiles/manualchecks.exe>.
- Instructions on how to download and extract the data files are available at <http://www.calstatela.edu/its/docs/download.php>.

## Manual Checks

### Visual Check

Visually impaired users often increase a web page's text size by zooming in. Zooming in should increase the text size without causing additional problems such as overlapping text. When laying out pages, ensure that columns, page elements, or text lines do not overlap each other when enlarged. Text should wrap within columns.

To zoom in on the web page:

1. Click the **Start** button, point to **All Programs**, point to **Mozilla Firefox**, and select **Mozilla Firefox**. The **Mozilla Firefox** window opens.
2. Click the **File** menu and select **Open File**. The **Open File** dialog box opens.
3. Navigate to the folder where you extracted the data files, select the **wrkshpbsample.htm** file, and then click the **Open** button.
4. Review the page to confirm that there are no overlapping elements.
5. To increase the text size, click the **View** menu, point to **Zoom**, and select **Zoom Text Only**.
6. Click the **View** menu, point to **Zoom**, and select **Zoom In** or press **Ctrl++** (see Figure 1).
7. Repeat step 6 three to four times to increase the text size. Notice that the **JavaScript Links** heading overlaps the paragraph text (see Figure 2).

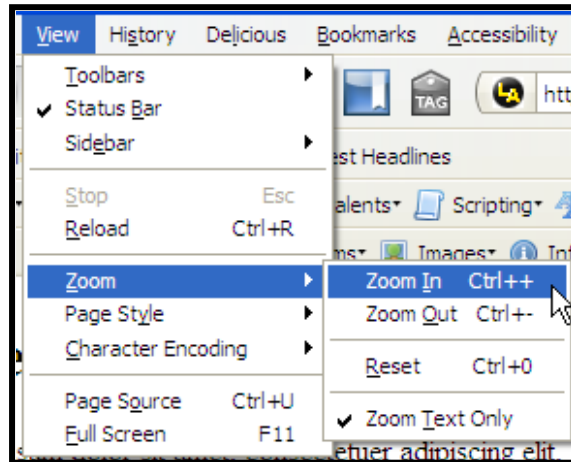


Figure 1 – Firefox View Menu

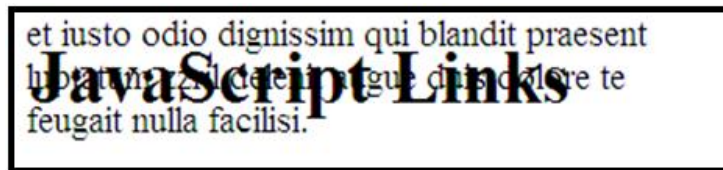


Figure 2 – Heading Overlapping Paragraph after Increasing Text Size

In this example, the CSS code's absolute positioning causes page items to overlap when text size is increased. The two columns under the *Nicely Aligned Layout* heading were created using divisions (divs) and both positioned in fixed locations – a certain number of pixel distance from the web page's upper left corner. When the user increases the text size, the neighboring space (the division) below the columns does not adjust to the new text size. As a result, overlap occurs. One possible solution to fix the problem in this example is to float the text instead of positioning it absolutely.

**NOTE:** A full explanation of absolute vs. relative CSS positioning is outside the scope of this handout. For more information, refer to the resources listed in the [Additional Resources](#) section of this handout.

To repair the overlapping text problem:

1. Click the **Start** button, point to **All Programs**, point to **Accessories**, and select **Notepad**. The **Notepad** window opens.
2. Click the **File** menu and select **Open**. The **Open** dialog box opens.
3. Click the **Files of type** arrow and select **All Files** from the list.
4. Navigate to the folder where you extracted the data files, select the **wrkshpbsample.htm** file, and then click the **Open** button. The HTML code of the web page is displayed.
5. In the body section of the web page, locate the first div containing absolute positioning `<div style="position: absolute; left: 370px; top: 70px; width: 340px">`.
6. Change the absolute positioning to float by replacing the code with `<div style="float: left; width: 50%">`.
7. Locate the second div containing absolute positioning `<div style="position: absolute; top: 300px; left: 10px">`.
8. Change the absolute positioning to float by replacing the code with `<div style="float: left; width: 50%">`.
9. Click the **File** menu and select **Save** to save the changes.
10. Open the **wrkshpbsample.htm** file in **Firefox** and increase the text size several times. Notice that the **JavaScript Links** heading no longer overlaps the paragraph text.

## Blinking Text and Marquees

Section 508 standard J states that pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hertz (Hz) and lower than 55 Hz. Avoid using flashing, blinking, flickering, or staggered movement from a control like a marquee or animated GIF.

This provision is necessary because some individuals with photosensitive epilepsy could suffer a seizure triggered by displays that flicker, flash, or blink, particularly if the flash has a high intensity and is within certain frequency ranges.

To view the blinking marquee on the web page:

1. Open the **bad\_page\_fail.htm** file in **Firefox**.
2. Scroll down to the bottom of the web page to see the blinking marquee.

Running the AccVerify verification on the web page will result in the following failure (see Figure 3).

<a href="#">J. 508 Standards, Section 1194.22, (j)</a> Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz. <ul style="list-style-type: none"><li>• Rule: 7.1.1 - Documents are required not to contain the BLINK element.<ul style="list-style-type: none"><li>○ <b>Failure</b> - BLINK Element at Line: 104, Column: 15</li></ul></li><li>• Rule: 7.1.2 - Documents are required not to contain the MARQUEE element.<ul style="list-style-type: none"><li>○ <b>Failure</b> - MARQUEE Element at Line: 104, Column: 6</li></ul></li></ul>	No
---	----

Figure 3 – Section 508 Standard J Violation

To repair the blinking marquee text:

1. Open the **bad\_page\_fail.htm** file in **Notepad**.
2. Locate the HTML code for the blinking marquee text `<marquee><blink>This is hard to read. This is hard to read. This is hard to read.</blink></marquee>`.

NOTE: To quickly find the code, press **Ctrl+F** to open the **Find** dialog box, type `<marquee>` in the **Find what** box, and then click the **Find Next** button.

3. Drag to select the code from the opening `<marquee>` tag to the closing `</marquee>` tag, and then press the **Delete** key.
4. Type `<p>Possible Alternative: Use CSS to draw attention to specific text.</p>`, and then press the **Enter** key.
5. Type `<p style="background: #ffffdd; border: 1px dotted #333; padding: 10px; margin: 0 30px; font-weight: bold">Can you read me?</p>` to style the text using inline CSS rules.
6. Click the **File** menu and select **Save** to save the changes.
7. Open the **bad\_page\_fail.htm** file in **Firefox** to review the changes (see Figure 4).

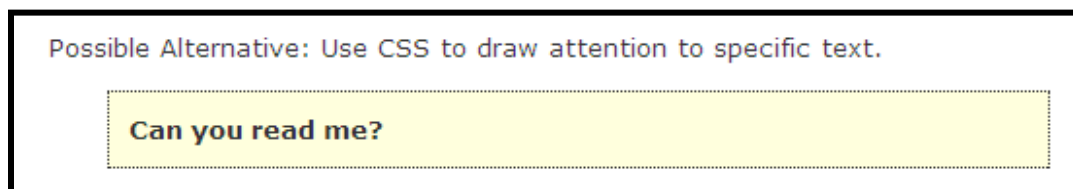


Figure 4 – CSS Used to Emphasize Text

NOTE: This is just one alternative for drawing attention to a web page's text. The page author can choose another method so long as all other accessibility requirements are met.

## Applets

Per Section 508 standard 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 standards A through L. An applet is a program needed by a user's computer to view page content. One of the most common applets is the Adobe Reader program for reading Portable Document Format (PDF) files.

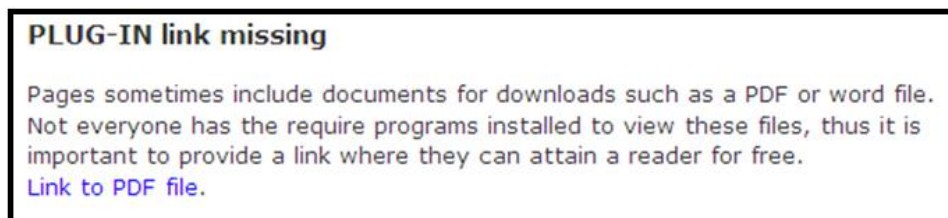


Figure 5 – Web Page with a Missing Plug-in Link

The web page in Figure 5 links to a PDF file, but there is no link to the Adobe Reader program. Running the AccVerify verification on the web page will result in the following failure (see Figure 6). To resolve the issue, simply include a link to the Adobe Reader program.

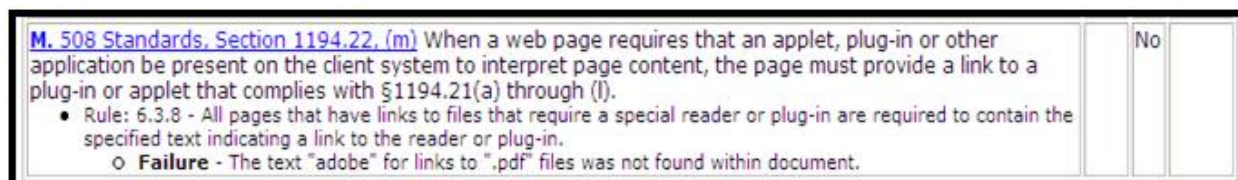


Figure 6 – Section 508 Standard M Violation

To include a link to the Adobe Reader program:

1. Open the **bad\_page\_fail.htm** file in **Notepad**.
2. Locate the HTML code for the link to the PDF file `<a href="sample.pdf">Link to PDF file</a>.</p>`, place the cursor at the end of the line, and then press the **Enter** key to create a new line.
3. Type `<a href="http://get.adobe.com/reader/">Click to get the Adobe Reader program</a>` to add a link to the Adobe Reader program.
4. Click the **File** menu and select **Save** to save the changes.
5. Open the **bad\_page\_fail.htm** file in a web browser to review the changes (see Figure 7).

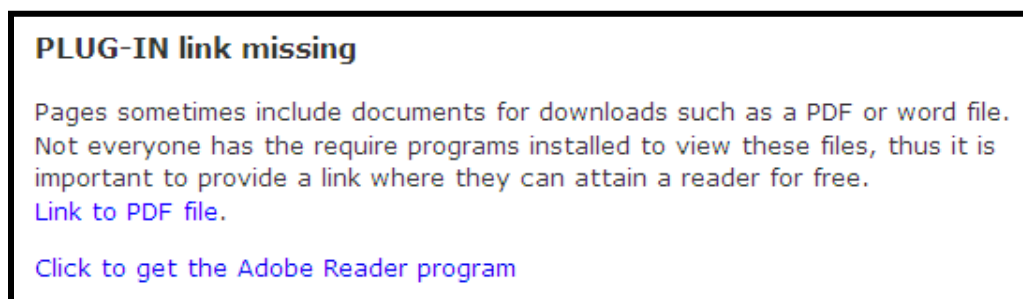


Figure 7 – Web Page with a Plug-in Link

These are the steps necessary to ensure that web pages with links to PDF files also have a link to the Adobe Reader program. For questions regarding other file types commonly found online (Flash, etc.), please visit <http://www.calstatela.edu/accessibility/tutorials.php>. For questions regarding any program/file not included on this website, please contact the Web Administrator at [wwwadmin@calstatela.edu](mailto:wwwadmin@calstatela.edu).

## Tables

### Table Summary Attribute

Each data table needs its own unique table summary attribute to describe its primary purpose and explain its overall structure. Most assistive output technologies will read the summary first to help the user interpret and use the table. The summary's contents are not displayed on the screen by graphic browsers, but can be outputted by screen readers and Braille displays to assist users of these devices. The decision about what to include in the summary is the page author's. A good rule of thumb is to include what someone who is not able to see the table should understand about it.

To add a table summary:

1. Open the **bad\_page\_fail.htm** file in **Notepad**.
2. Locate the opening **<table border="1">** tag, place the cursor to the left of the right angle bracket (>), press the **Spacebar**, and then type **summary=** followed by the table summary enclosed in quotation marks.

For example:

```
<table border="1" summary="This table displays the biology courses and sections  
taught by professors Dr. Chan and Dr. Smith.">
```

3. Click the **File** menu and select **Save** to save the changes.

NOTE: The summary could also be included in a paragraph before the table in case a user's browser does not support the summary attribute.

## Styles

### Disabled Styles and Style Sheets

Section 508 standard D states that documents shall be organized so they are readable without requiring an associated style sheet. Typically, CSS styles are used to define the look and feel of a web page. If users choose to disable styles, the web page should remain fully usable with headings, paragraphs, hyperlinks, and lists being sensible and obvious to the viewer.

To disable style sheets:

1. Open the **disabledstylesheets.htm** file in **Firefox**. Notice the use of headings to structure the document's content, the visual separation of paragraphs, and a list to display items sequentially.
2. On the **Web Developer** toolbar, click the **CSS** button, point to **Disable Styles**, and select **All Styles** (see Figure 8). Notice the difference in the page after styles are disabled. The headings have disappeared. Paragraphs that were separated spatially and by a dashed border now run together. The paragraph above **Other items to watch for** has disappeared entirely. The list with red bullets (created in Word and saved as a web page) will appear to screen readers as three separate paragraphs.

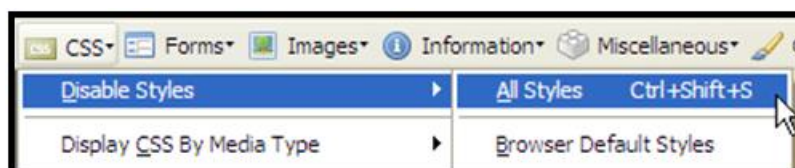


Figure 8 – Disabling All Styles Using the Web Developer Toolbar

To maintain page readability if styles are disabled:

1. Open the **disabledstylesheets.htm** file in **Notepad**.

NOTE: To make the code easier to read on the screen, click the **Format** menu and select **Word Wrap**.

2. In the body section of the web page, locate the text **Disabled Style Sheets** that is intended to be a heading. Replace the opening `<p class="style3">` tag with `<h1>` and the closing `</p>` tag with `</h1>`. This correctly establishes **Disabled Style Sheets** as a heading instead of regular text marked up to look like a heading.
3. Correctly revise the remaining headings.
4. To repair the paragraphs, locate the div that contains the first paragraph `<div id="para1">`. Delete `id="para1"` from within the opening `<div>` tag. Add an opening `<p>` tag after the opening `<div>` tag and a closing `</p>` tag before the closing `</div>` tag. This removes the paragraph from the absolutely positioned div.
5. Repeat step 4 for the second paragraph to enclose it within `<p>` and `</p>` tags.
6. To repair the disappearing paragraph, locate the code `<p style="background: black"><font color=white>If the page styles are disabled, this paragraph disappears!</font></p>`. Delete the opening `<font color=white>` tag and the closing `</font>` tag. Another background color can be applied, such as `<p style="background: red">` instead of black. Even if styles are disabled, the paragraph will still be readable.
7. Due to the complicated code generated by Word when saving the list as HTML, recreating the list will be easier. Locate the div containing the list which is located below `<p>Things to check for accessibility when styles are disabled:</p>`. Select all the code from the opening `<div class=Section1>` tag to the closing `</div>` tag and delete it.
8. Enter the following HTML code for the list:

```
<ul>
  <li>Headings</li>
  <li>Lists</li>
  <li>Paragraphs</li>
  <li>Links</li>
</ul>
```

9. Click the **File** menu and select **Save** to save the changes.

NOTE: To re-enable styles, click the **CSS** button on the **Web Developer** toolbar, point to **Disable Styles**, and select **All Styles** to uncheck the selection.

## Headings

Proper use of heading tags makes it easier for visitors who rely on screen readers to navigate web page content. Headings should be organized according to the content presented. Headings should be ordered consecutively (H1 before H2, H2 before H3, etc.). Every page should use at least one H1 heading. One way to see how headings are being used is by viewing a web page as an outline.

To view a web page as an outline:

1. Open the **bad\_page\_pass.htm** file in **Firefox**.
2. On the **Web Developer** toolbar, click the **Information** button and select **View Document Outline** (see Figure 9). A new tab opens displaying the web page in outline view with only the headings displayed (no graphics or pictures, etc.) (see Figure 10).



Figure 9 – Information Menu



Figure 10 – Web Page in Document Outline View

3. Confirm that the web page has at least one H1 heading.
4. Confirm that all headings advance in a logical hierarchical order (H1 before H2, H2 before H3, etc.).
5. If necessary, rearrange headings into the correct order.

## Linearizing the Page

Tables are a useful tool for arranging and presenting data. They are not necessarily inaccessible, but they can present problems to screen readers. It is important to realize that screen readers read content in the literal order it appears in the web page's HTML code which may or may not coincide with the order of items in a table. If the literal order of the content in the code is logical, then the linearized reading order will be logical.

To test the linear reading order of a web page:

1. Open the **linearized.html** file in **Firefox**.
2. On the **Web Developer** toolbar, click the **Miscellaneous** button and select **Linearize Page**. The web page displays in a linear fashion.
3. Examine the reading order of the elements on the page. Does the reading order make sense?

Keep tables simple when using them. See Figure 11 for an example of a simple table that has a logical linear order. The items appear in this same order in the HTML code.

<b>Step 1 :</b>
Buy ingredients.
Add ingredients to mixer. Blend.
<b>Next:</b>
Pour batter into cake pan.
Put in oven for 35 mins. at 350 degrees.
<b>Last:</b>
Remove from oven and place cake on cooling rack.
Wait one hour before applying the frosting.

Figure 11 – Simple Table with Logical Linear Order

The table in Figure 11 is one way to solve the linear order problem. Individual author’s choices may vary so long as they are accessible.

Another useful tool for checking linearization is the WAVE toolbar available at <http://wave.webaim.org/toolbar>.

To use the WAVE toolbar in Firefox:

1. Open the web page you want to test in **Firefox**.
2. On the **WAVE** toolbar, click the **Structure/Order** button. Blue boxes with numbers appear on each form element, table, or div, displaying the reading order (see Figure 12). Refer to the boxes to assist in rearranging the elements in a logical linear order.



Figure 12 – Web Page Displaying Linear Order

## Displaying Link Details

Sometimes web authors use CSS code to redefine how links appear on a web page and inadvertently make them inaccessible in the process. Typically, a link is recognizable by its blue color and underline. Sometimes links are bold or italicized as well. Links that are not distinguishable from regular text could cause visually impaired users to be unable to navigate the page. One way to ensure that links are distinguishable is by viewing the link details.

To display link details:

1. Open the **wrkshpbsample.htm** file in **Firefox**.
2. On the **Web Developer** toolbar, click the **Information** button and select **Display Link Details** (see Figure 13). On the web page, yellow boxes appear to the left of each hyperlink, displaying the link's URL (see Figure 14).

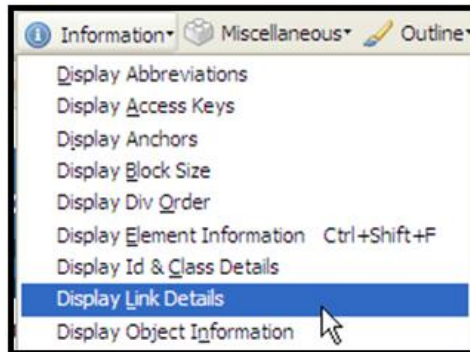


Figure 13 – Information Menu

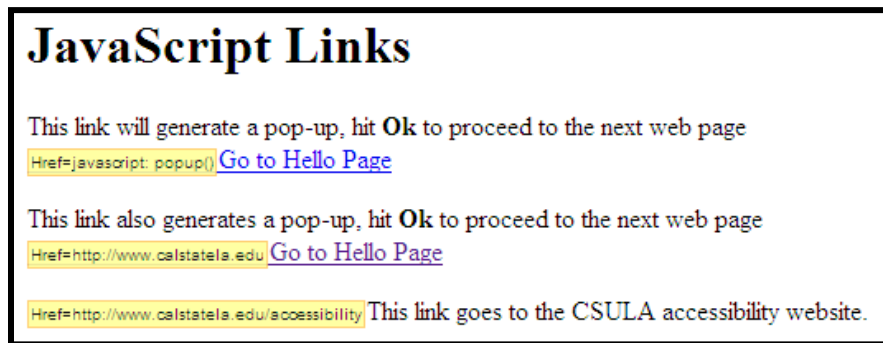


Figure 14 – Web Page Displaying Link Details

3. Check for yellow boxes next to text that is not immediately recognizable as a hyperlink. Any links that are not distinguishable from regular text need to have their associated HTML and/or CSS code examined.

**NOTE:** While link information is displayed, check if any link contains the word *javascript*. If it does, it needs to be repaired. For more information, refer to the [JavaScript and Accessibility](#) section of this handout.

Under the **JavaScript Links** heading, the last link's text-decoration property has been set to none (instead of underline) and its font color set to black (instead of the more common blue) (see Figure 14). The link should be either bolded, colored, or underlined to differentiate it from regular page text.


To restore the link's default attributes:

1. Open the **wrkshpbsample.htm** file in **Notepad**.
2. Locate the HTML code `<p><a style="text-decoration: none; color: black" href="http://www.calstatela.edu/accessibility">This link goes to the CSULA accessibility website.</a></p>`.
3. Drag to select `style="text-decoration: none; color: black"` within the opening `<a>` tag, and then press the **Delete** key. This removes the style, returning the link back to its default state – blue underlined text.
4. Click the **File** menu and select **Save** to save the changes.

## JavaScript and Accessibility

JavaScript is a scripting language used primarily by the client (a computer making requests from a server) to make websites dynamic. JavaScript can check if a user has filled out a form correctly (validation), generate pop-up windows, and leave a trail of breadcrumbs (used to display the path to the current page), among other things. Although JavaScript has its uses, some users will disable it. Therefore, page authors should ensure that their pages are still accessible without the use of JavaScript.

To view link behaviors:

1. Open the **wrkshpbsample.htm** file in **Firefox**.
2. Under the **JavaScript Links** heading, click to follow the first link. A pop-up window opens.
3. Click the **OK** button to continue. The Cal State L.A. home page displays.
4. Click the **Back** button  to go back one page.
5. Repeat steps 2 through 4 for the second link.

NOTE: The links to the Cal State L.A. home page are followed only after clicking the **OK** button. These links should work with JavaScript disabled.

6. To disable JavaScript in **Firefox**, click the **Tools** menu and select **Options**. The **Options** dialog box opens.
7. On the **Content** tab, deselect the **Enable JavaScript** check box (see Figure 15).

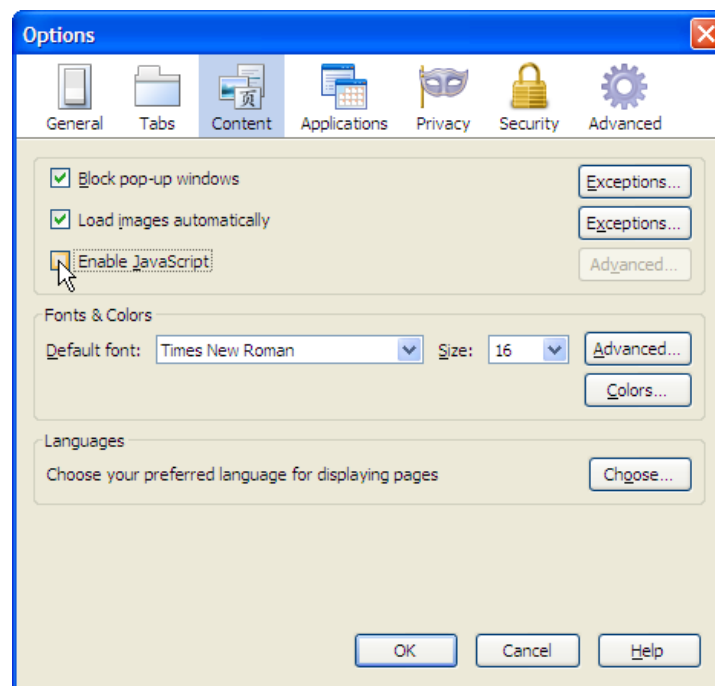



Figure 15 – Content Tab of the Options Dialog Box

8. Click the **OK** button.
9. Under the **JavaScript Links** heading, click to follow the second link again. Notice that although the pop-up window does not open, the link to the home page still works.
10. Click the **Back** button  to go back one page.
11. Click to follow the first link again. With JavaScript disabled, the link does not work because the JavaScript code associated with the link cannot be executed. This link will need to be repaired.

To make the links accessible:

1. Open the **wrkshpbsample.htm** file in **Notepad**.
2. Drag to select the HTML code `<a href="http://www.calstatela.edu" onclick="popup()" >`. This action selects the JavaScript code that allows the link to function even if a user has JavaScript disabled.
3. Click the **Edit** menu and select **Copy** or press **Ctrl+C** to copy the code.
4. Drag to select the code for the link that does not work when JavaScript is disabled `<a href="javascript: popup()">`.
5. Click the **Edit** menu and select **Paste** or press **Ctrl+V** to paste the previously copied code.
6. Click the **File** menu and select **Save** to save the changes.
7. Open the **wrkshpbsample.htm** file in **Firefox** and test the two links to see if they work.

## Page Navigation

### Keyboard Navigation

In some instances, it may be necessary for web authors to manually arrange form elements due to layout constraints. Divs (short for divisions) have their own rules regarding navigation. Once inside a div, the tabbing order proceeds from top to bottom and then left to right before moving to anything outside the div (see Figure 16).

**Organized logically (left to right, top to bottom)**

However, since the elements are in divs, the tabbing order is not logical.

First name <input type="text"/>	Last name <input type="text"/>
Extension <input type="text"/>	Building <input type="text"/>
Title <input type="text"/>	Supervisor <input type="text"/>

Figure 16 – Form Elements Organized by Divs

To override the usual order of tabbing down the left div and then over and down the right one, the tabbing order must be set manually using the `tabindex` attribute.

To manually set the page's tab order:

1. Open the **tabindexorder.html** file in **Notepad**.
2. Locate the HTML code for the last name form field `<p>Last name <input type="text" name="textfield10" tabindex="4" /></p>`, and then change the `tabindex` value to **2**.
3. Repeat step 2 to establish the correct `tabindex` order for the form elements.
  - Change the `tabindex` value for **Extension** to **3**.
  - Change the `tabindex` value for **Title** to **5**.
  - Change the `tabindex` value for **Building** to **4**.
4. Click the **File** menu and select **Save** to save the changes.

NOTE: The order of form elements can be checked using the **WAVE** toolbar.

## Skip to Main Content

Section 508 standard O mandates that a method shall be provided that permits users to skip repetitive navigation links. This link, commonly referred to as a skipper, should immediately move to the body of the page and bypass common navigation to assist screen readers or users navigating by keystroke. The University web templates include a skipper.

To add a skipper link to a web page:

1. Open the **bad\_page\_pass.htm** file in **Notepad**.
2. Locate the opening **<body class="column2">** tag, place the cursor at the end of the line, and then press the **Enter** key to create a new line.
3. Type **<div><a href="#mainContent" id="skipper">Skip to the content</a></div>**. This code establishes a link allowing users to jump to the element with the **id="mainContent"** attribute.
4. Locate the opening **<div class="column2">** tag, place the cursor to the left of the right angle bracket (**>**), press the **Spacebar**, and then type **id="mainContent"**.
5. Click the **File** menu and select **Save** to save the changes.

## Visibility Issues

### Checking a Web Page for Proper Color Contrast

It is important for web pages to provide a sufficient visual contrast ratio of at least 5:1 between text and background colors to ensure that visually impaired users can still use them. The Colour Contrast Analyser, an add-in for the Internet Explorer web browser, can be used to test color contrast.

To check a web page for appropriate color contrast:

1. Click the **Start** button, point to **All Programs**, and select **Internet Explorer**. The **Internet Explorer** window opens.
2. Click the **File** menu and select **Open**. The **Open** dialog box opens.
3. Click the **Browse** button, navigate to the folder where you extracted the data files, select the **bad\_page\_pass.htm** file, click the **Open** button, and then click the **OK** button.
4. Scroll down until you see the yellow, orange, and red text on the web page.
5. On the **AIS Web Accessibility** toolbar, click the **Colour** button and select **Contrast Analyser [application]** (see Figure 17). The **Colour Contrast Analyser** dialog box opens.

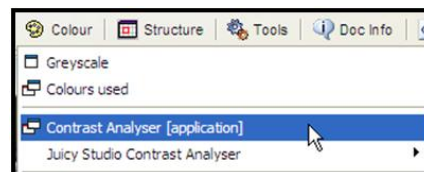



Figure 17 – Colour Menu

6. Select the foreground color to be tested. Click the color picker button  in the **Foreground** section of the **Colour Contrast Analyser** dialog box. The arrow changes to a square and displays a zoomed in window.
7. Move the pointer over to the web page and position it directly over one of the characters in the red text **Even red does not provide enough contrast. Ratio is 4:1**. Click to select the red color. The selected color appears in the **Colour select** box in the **Foreground** section and its hexadecimal (or RGB) value appears next to it (see Figure 18).

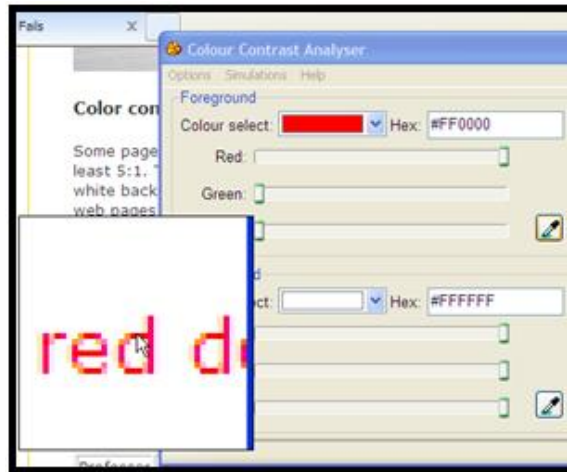



Figure 18 – Selecting a Foreground Color with the Colour Contrast Analyser

8. Select a background color. Click the color picker button  in the **Background** section.
9. Move the pointer over the web page and position it directly over the background color to be tested. Click to select the white background. The selected color appears in the **Colour select** box in the **Background** section and its hexadecimal (or RGB) value appears next to it (see Figure 19).

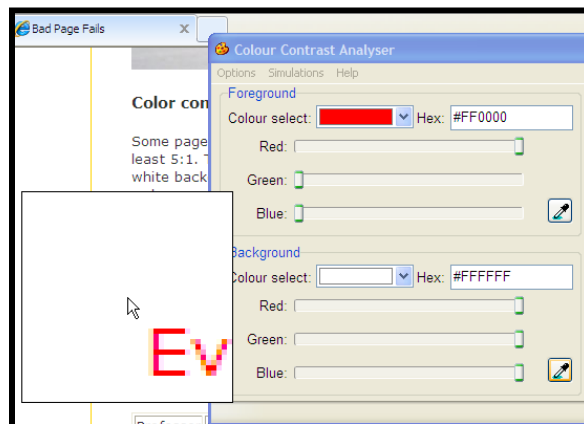


Figure 19 – Selecting a Background Color with the Colour Contrast Analyser

10. Select the **Show contrast result for colour blindness** check box (see Figure 20).

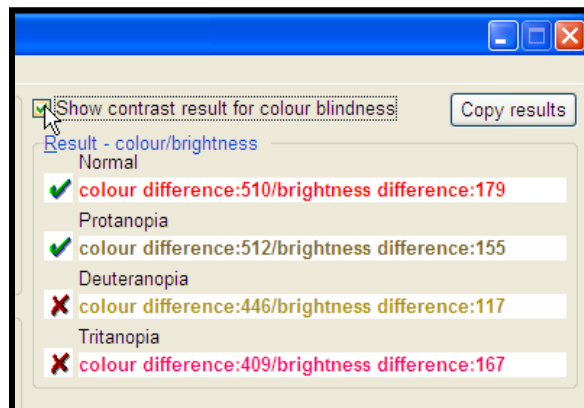


Figure 20 – Show Contrast Result for Colour Blindness Check Box

11. In the **Result** section, check to see if the color passed all of the various color and brightness tests. If there is a red X next to one of the tests, the color combination cannot be used due to insufficient color contrast. Selecting appropriate colors is a trial and error process.
12. To find a value of red that will pass the test, drag the **Red** slider in the **Foreground** section slowly to the left. When a passing color is found, the red X's in the **Result** section will change to green check marks. The hexadecimal value **#BC0000** passes the tests (see Figure 21).

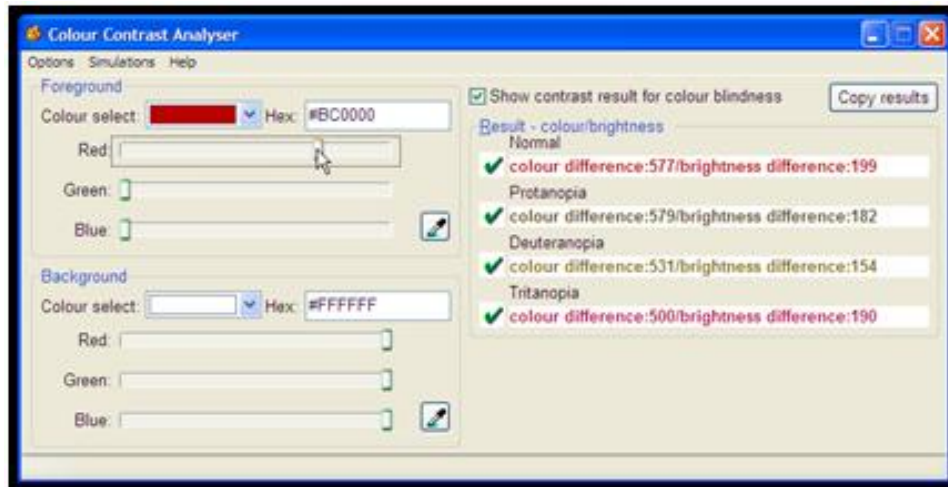


Figure 21 – Selecting a Color That Passes All Color and Brightness Tests

13. To find the RGB value of the selected color, click the **Options** menu, point to **Displayed color value**, and select **RGB** (see Figure 22). Either the hexadecimal value **#BC0000** or the RGB value **188,0,0** for this shade of red can be used against a white background (see Figure 23).

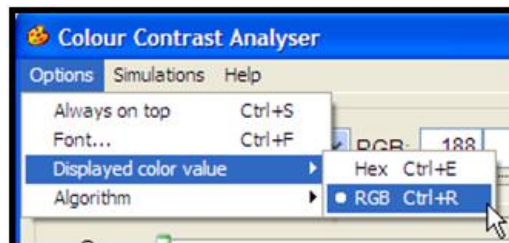


Figure 22 – Options Menu

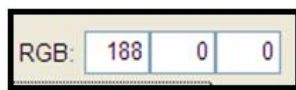


Figure 23 – RGB Value

**NOTE:** The *Colour Contrast Analyser* does not repair web page files. It is to be used as a tool for testing color contrast and choosing appropriate color contrast schemes. The RGB or hexadecimal values found in the *Foreground* and *Background* sections can be either hand-coded or selected via the various color pickers in Dreamweaver, FrontPage, etc.

**NOTE:** The *Colour select* boxes can be used independently (without selecting colors on a pre-existing web page) to find colors that will provide sufficient color contrast.

## Low Vision User Test

Some users will apply their own custom style sheet to a web page to make page viewing easier. The page's author should not use code in a way that will prevent a user from personalizing the page to fit their needs. Applying the lowvis.css style sheet is a simple way to test if a page can be personalized.

To apply a user defined style sheet to a web page:

1. Open the web page you want to test in **Firefox**.
2. On the **Web Developer** toolbar, click the **CSS** button and select **Add User Style Sheet** (see Figure 24). The **Add User Style Sheet** dialog box opens.

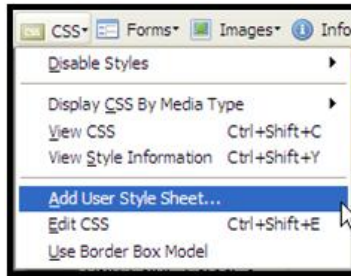


Figure 24 – CSS Menu

3. Navigate to the folder where you extracted the data files, select the **lowvis.css** file, and then click the **Open** button. The following changes should occur (see Figure 25):
  - All text sizes increase dramatically.
  - The background and text colors change to a brown hue.
  - Paragraphs are displayed with a surrounding border.



Figure 25 – Sample Web Page with the Lowvis Style Sheet Applied

4. To return the page back to normal, click the **CSS** button on the **Web Developer** toolbar and deselect **Add User Style Sheet**.

If any of the changes do not occur, the web page will need to be repaired so that a user's style sheet can override the page's style settings.

To troubleshoot when lowvis.css does not work:

- Look for the text *!important* in the CSS code. If it appears to the right of any declaration, the lowvis.css style sheet will not override it.
- Look for paragraphs separated by `<br>` tags instead of `<p>` or `<div>` tags.

## Code Validity and Doctypes

### What Does Valid Code Mean?

The World Wide Web Consortium (W3C), an international consortium dedicated to developing web standards, defines HTML as “the coded format language used for creating hypertext documents on the World Wide Web and controlling how web pages appear.” There are several different versions of the language (e.g., HTML 4.01 Strict, HTML 4.01 Transitional, XHTML 1.0 Strict). Just as there are different versions of HTML, there are different versions of web browsers (Firefox, Internet Explorer, Opera, etc.). Each browser needs to know which version of HTML code is being used so it can display the page accordingly. A web page declares the version of HTML it is using in a Document Type Definition (DTD), also known as a *doctype* (see Figure 26).

A screenshot of a Mozilla Firefox browser window. The address bar shows the source URL: http://www.calstatela.edu/its/. The browser's menu bar (File, Edit, View, Help) is visible. The source code displayed in the main window area is: 

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

Figure 26 – Code for XHTML 1.0 Transitional Doctype

When a web page includes a doctype, a validator can assist in ensuring that the rules for that particular version of HTML are being followed. If no doctype is declared, or there are errors in the declaration, browsers will *guess* how to display the code used to create the page. Proper use of the doctype helps ensure that pages are rendered correctly to the benefit of users, including those using assistive technologies such as screen readers.

### Adding a Doctype

To check whether or not a web page includes a doctype or to see the particular doctype in use, view the page’s source code using a web browser such as Firefox (click the *View* menu and select *Page Source*) or Internet Explorer (click the *View* menu and select *Source*), or open the page in a text editor such as Notepad or TextPad. The code for the doctype is located at the top of the file.

If a web page is missing a doctype, the Web Services group recommends that the HTML 4.01 Transitional doctype be added.

To add the HTML 4.01 Transitional doctype:

1. Open the **missingdoctype.html** file in **Notepad**.
2. Locate the opening **<html>** tag, place the cursor to the left of the tag, and then press the **Enter** key to create a new line.
3. On the new line, enter the code **<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">**.
4. Click the **File** menu and select **Save** to save the changes.

### Validating HTML

To test whether or not a web page adheres to the rules for the version of HTML it was written in, the code must be validated. The validators on the Web Developer toolbar can test the page for validity.

NOTE: The HTML validation test cannot be run locally. The web page must be online.

To validate the HTML code:

1. Open the web page you want to validate in **Firefox**.
2. On the **Web Developer** toolbar, click the **Tools** button and select **Validate HTML**. **Firefox** displays the results in a new tab.
3. If the results indicate that the document passed validation, no further action is necessary (see Figure 27).

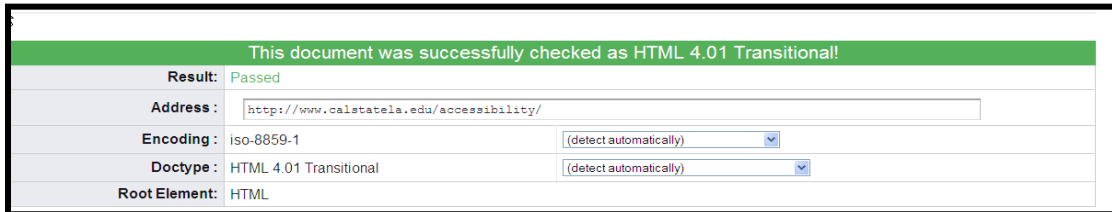


Figure 27 – Results for a Web Page That Passed HTML Validation

4. If errors are found, read and interpret the error messages and make the appropriate corrections (see Figure 28 and Figure 29).



Figure 28 – Results for a Web Page That Failed HTML Validation

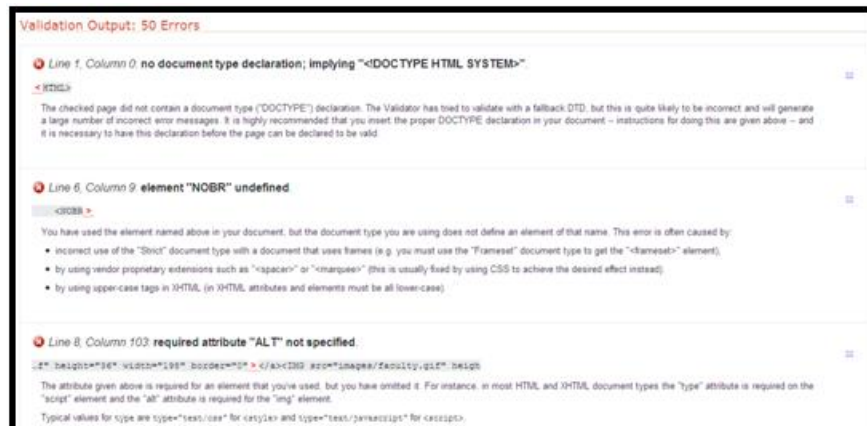


Figure 29 – Validation Output Explaining Errors

5. After correcting the HTML code, save the changes and revalidate the web page.

**NOTE:** Any user who is unable to successfully repair an HTML file that does not pass validation should contact the Web Administrator at [wwwadmin@calstatela.edu](mailto:wwwadmin@calstatela.edu) for assistance.

## Validating CSS

To test whether or not the CSS file associated with a web page adheres to the CSS specifications, the CSS code must be validated. The validators on the Web Developer toolbar can test the page for validity.

To validate the CSS code:

1. Open the web page you want to validate in **Firefox**.
2. On the **Web Developer** toolbar, click the **Tools** button and select **Validate CSS**. **Firefox** displays the results in a new tab.
3. If the results indicate that the document passed validation, no further action is necessary (see Figure 30).



Figure 30 – Results for a Web Page That Passed CSS Validation

4. If errors are found, read the output of the error report and make the appropriate corrections (see Figure 31).



Figure 31 – Results for a Web Page That Failed CSS Validation

5. After correcting the CSS code, save the changes and revalidate the web page.

NOTE: Any user who is unable to successfully repair a CSS file that does not pass validation should contact the Web Administrator at [wwwadmin@calstatela.edu](mailto:wwwadmin@calstatela.edu) for assistance.

To validate the CSS code by file upload:

1. Open a web browser and go to <http://jigsaw.w3.org/css-validator/>.
2. Click the **By file upload** tab (see Figure 32).
3. Click the **Browse** button. The **Choose File to Upload** dialog box opens.
4. Navigate to the folder that contains the website files, select the CSS file to be tested, and then click the **Open** button.
5. Click the **Check** button to submit the file for validation.
6. If the results indicate that the document passed validation, no further action is necessary (see Figure 30).
7. If errors are found, read the output of the error report and make the appropriate corrections (see Figure 31).

NOTE: Any user who is unable to successfully repair a CSS file that does not pass validation should contact the Web Administrator at [wwwadmin@calstatela.edu](mailto:wwwadmin@calstatela.edu) for assistance.



Figure 32 – Validate CSS by File Upload

## Deprecated Code

Originally, HTML was meant to be a structural language used mostly by researchers. There was no need for graphics or different fonts or colors on a page. Eventually, browsers added new features which led to more users and a need for visually pleasing pages. HTML struggled to support structure and layout.

The W3C, the governing body that sets HTML and other technical standards, decided to separate presentation from structure by replacing formatting tags with Cascading Style Sheets (CSS).

Pages using deprecated codes such as `<font>`, `<b>`, or `<i>` will need to be revised. A list of deprecated codes and their alternatives can be found at <http://www.calstatela.edu/accessibility/deprecated.php>.

## Legacy Web Design

### Frames

Per Section 508 standard I, frames shall be titled with text that facilitates frame identification and navigation. The use of frames is not recommended due to the difficulty screen readers have interpreting them. If frames must be used, frames need names to inform users of the purpose they serve. Label frames inside the `<frame>` tag itself using the title and name attributes of the `<frame>` tag.

To add a title and name to frames:

1. Open the **frames.html** file in **Notepad**.
2. Locate the frame code which starts with the **<frameset>** tag.
3. For each **<frame>** tag, add a title attribute and a name attribute.

For example:

```
<frameset cols="15%, 85%">
  <frame src="navmenu.html" title="Navigation menu" name="navmenu">
  <frame src="content.html" title="Main content" name="content">
</frameset>
```

4. Click the **File** menu and select **Save** to save the changes.

## Image Maps

Image maps are acceptable and can be accessible provided that hot spots have alt text.

To add alt text to image map hot spots:

1. Open the web page with the image map in **Notepad**.
2. Locate the image map code which starts with the `<map>` tag.
3. For each `<area>` tag, add an alt attribute.

For example:

```
<map name="Map" id="Map">
  <area shape="rect" coords="197,8,227,33" href="http://www.calstatela.edu"
  alt="Visit the Cal State L.A. home page" />
  <area shape="rect" coords="266,35,334,102"
  href="http://www.calstatela.edu/accessibility" alt="Visit the Cal State L.A.
  Accessibility website" />
</map>
```

4. Click the **File** menu and select **Save** to save the changes.

## Additional Help

Below is a list of contact information for getting additional help with web accessibility.

- For general questions regarding web accessibility reports and assistance with making websites compliant, contact the Web Administrator at [wwwadmin@calstatela.edu](mailto:wwwadmin@calstatela.edu).
- For general questions regarding the University web templates, contact Public Affairs at [paffairs@cslanet.calstatela.edu](mailto:paffairs@cslanet.calstatela.edu).
- For general assistance with campus software and hardware installations, contact your Information Technology Consultant (ITC). A list of ITCs is available at <http://www.calstatela.edu/itc>.

## Additional Resources

Below is a list of additional resources related to web accessibility.

- Cal State L.A. web accessibility reports: <http://accmonitor.calstatela.edu/>
- Accessibility websites:
  - Cal State L.A. accessibility website: <http://www.calstatela.edu/accessibility>
  - CSU Accessible Technology Initiative: <http://www.calstate.edu/accessibility>
  - World Wide Web Consortium accessibility website: <http://www.w3.org/WAI/intro/accessibility.php>
  - Adobe accessibility website: <http://www.adobe.com/accessibility>
  - Apple accessibility website: <http://www.apple.com/accessibility>
  - Microsoft accessibility website: <http://www.microsoft.com/enable>
- Common accessibility errors and solutions: <http://www.calstatela.edu/accessibility/errors.php>
- Tools for testing web accessibility: <http://www.calstatela.edu/accessibility/tools.php>
- YouTube video demonstrating the importance of proper heading use: [http://www.youtube.com/watch?gl=JP&hl=ja&v=AmUPhEVWu\\_E](http://www.youtube.com/watch?gl=JP&hl=ja&v=AmUPhEVWu_E)
- W3C quality assurance tips for webmasters: <http://www.w3.org/QA/Tips/Doctype>

- Most common web standards and accessibility errors:  
[http://www.personal.psu.edu/v23/presentations/accessibility/20errors/standards\\_errors.html](http://www.personal.psu.edu/v23/presentations/accessibility/20errors/standards_errors.html)
- How to achieve web standards and quality on your website:  
<http://www.w3.org/QA/2002/04/Web-Quality>
- Absolute vs. relative positioning with CSS:  
<http://www.yourhtmlsource.com/stylesheets/csslayout.html>