



# Accessible Design: Best Practices

A comprehensive guide to the best design practices for accessible and usable web pages.

## Introduction

This guide is intended as an introduction to both good accessible and good usability web design for those in the OSU community planning to build a first site or considering a revision of an existing web site. It seeks to bring together the best techniques and standards for accessible design, typically found in a variety of resources and explain, in plain language, how web designers can implement these guidelines, in most cases, using simple HTML solutions. Whenever possible, guidelines are followed with real-life examples and include code-snippets for easier understanding.

The guidelines and suggestions presented here are adapted from several sources and cover OSU requirements for accessible web design as established by the [OSU Web Accessibility Policies and Standards](#) and based on [Section 508 of the Federal Rehabilitation Act](#), as well as recommendations based on the [World Wide Web Consortium's Web Content Accessibility Guidelines](#), and usability standards supported by industry experts [Jacob Nielsen](#) and [Vincent Flanders](#).

For more information about the guidelines referenced, including a key to the guideline symbols, see the [About the Guidelines](#) section at the end of this tutorial.

## About the WAC

The OSU Web Accessibility Center (WAC) analyzes web pages for their accessibility to people with disabilities. The WAC offers this as a free service to OSU faculty and staff in order to further its mission to expand opportunities for people with disabilities through the innovative uses of computer technology. The WAC's goal is to ensure that all distance education and online courses at OSU are fully accessible to persons with disabilities.

After you have reviewed this tutorial, if you would like to have your web site analyzed, e-mail the URL of the home page of the site that you want the WAC staff to examine to [webaccess@osu.edu](mailto:webaccess@osu.edu). The WAC will report any accessibility and/or browser compatibility errors found on the page. Once your site receives WAC approval, you are entitled to display a WAC approved icon on your site.

For more help with accessibility issues, visit the WAC online at <http://www.wac.ohio-state.edu>.

To receive a copy of this tutorial in an alternate format, please contact the WAC at (614) 292-1760 or email us: [webaccess@osu.edu](mailto:webaccess@osu.edu).

## Contents

- [Organizing and Naming Your Site.](#)
  - [Site Structure.](#)
  - [Organizing Files and Folders.](#)
  - [Default and Index Pages.](#)
  - [Naming Conventions.](#)
- [Layout.](#)
  - [Organization.](#)
  - [Using Style Sheets for Consistency.](#)
  - [Standard Page Elements.](#)
- [Header Information.](#)
  - [DOCTYPE Declaration.](#)
  - [Language Declaration.](#)
  - [Title.](#)
  - [META-description.](#)
  - [META-keywords.](#)
- [Navigation.](#)
  - [Understanding the Function of Navigation.](#)
  - [Skip Navigation Link.](#)
  - [Using Image Navigation.](#)
  - [Repeating Link Text.](#)
  - [Plug-in Links](#)

- [Mystery Meat Navigation.](#)
- [Link Order.](#)
- [Keyboard Shortcuts.](#)
- [Scripts](#)
  - [Script Content](#)
  - [Script Links](#)
  - [Time-Out Scripts](#)
- [Color.](#)
  - [Color used for layout.](#)
  - [Contrast Colors.](#)
- [Cascading Style Sheets.](#)
  - [Using Style Sheets for Consistency.](#)
  - [Linked v Embedded Styles.](#)
  - [Guidelines for Using Style Sheets.](#)
  - [Using Relative Sizes.](#)
  - [Color and Links.](#)
  - [Spacing and Positioning.](#)
  - [Creating User-Friendly Style Sheets.](#)
  - [CSS Guides and Tutorials.](#)
- [Lists.](#)
- [Images and Multimedia.](#)
  - [Text Equivalents for Images, Media, and Scripts.](#)
  - [Spacer Images.](#)
  - [Testing for ALT tags.](#)
  - [Captioning Multimedia.](#)
  - [Flicker.](#)
  - [Providing Equivalent Versions.](#)
- [Imagemaps](#)
  - [Client-Side](#)
  - [Server-Side](#)
- [Layout and Data Tables.](#)
  - [Types of Tables.](#)

- [Summaries for Tables.](#)
- [Relative Sizing.](#)
- [Guidelines for Layout Tables.](#)
- [Simple Tables.](#)
- [Complex Tables](#)
- [Abbreviated Headers](#)
- [Forms.](#)
  - [General Guidelines for Forms.](#)
  - [Form Control Labels.](#)
  - [Positioning Form Control Labels.](#)
  - [Creating a Logical Read Order.](#)
  - [Keyboard Shortcuts.](#)
  - [Place-holding Characters.](#)
- [Frames](#)
  - [Frame Titles](#)
- [About the Guidelines Referenced in this Tutorial.](#)
  - [Section 508.](#)
  - [Web Content Accessibility Guidelines.](#)

---

## Organizing and Naming Your Site

[Site Structure](#) || [Organizing Files and Folders](#) ||  
[Default and Index Pages](#) || [Naming Conventions](#)

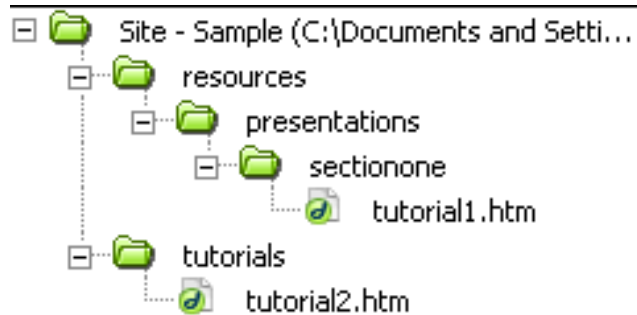
### Site Structure

Web sites consist of a collection of files. These files may or may not be organized within a directory structure of folders. The location of files in the web site corresponds with the URL.



It is a good idea to organize your site files into folders, but you should avoid making too many "levels" in your site, as each level adds a slash (/) to the URL needed to access that file.

## Consider this Sample site:



Within the site, there are two files (web pages): `tutorial1.htm` and `tutorial2.htm`. Let's say that this web site is stored on a server located at `http://www.sample.edu`. In order to reach the `tutorial1` page, we would have to go to:

`http://www.sample.edu/resources/presentations/sectionone/tutorial1.htm`.

Yet, if we wanted to visit the `tutorial2` page, we would only need to go to:

`http://www.sample.edu/tutorials/tutorial2.htm`.

This is because each folder within a site is represented by the folder name and a slash (/) in the URL of the corresponding web address.









The address to the `tutorial1` page is also called a "deep link" because it directs the visitor several levels into the web site. Deep links can be useful for pages you don't intend visitor's to arrive at directly (when you want to control the order in which visitor's view pages). However, because they result in long URLs that are difficult to remember, they should be used sparingly for information that will be regularly accessed or expect to be a high-traffic page. Shorter URLs are easier to advertise (better for you) and easier to remember (better for your users).

[return to contents](#)
































## Organizing Files and Folders.

While it is not a good idea to have too many levels (embedded folders) within your site, it is also not a good idea, unless you have a very simple site, to have no folders.

Here is an example of a site with no organizing structure:

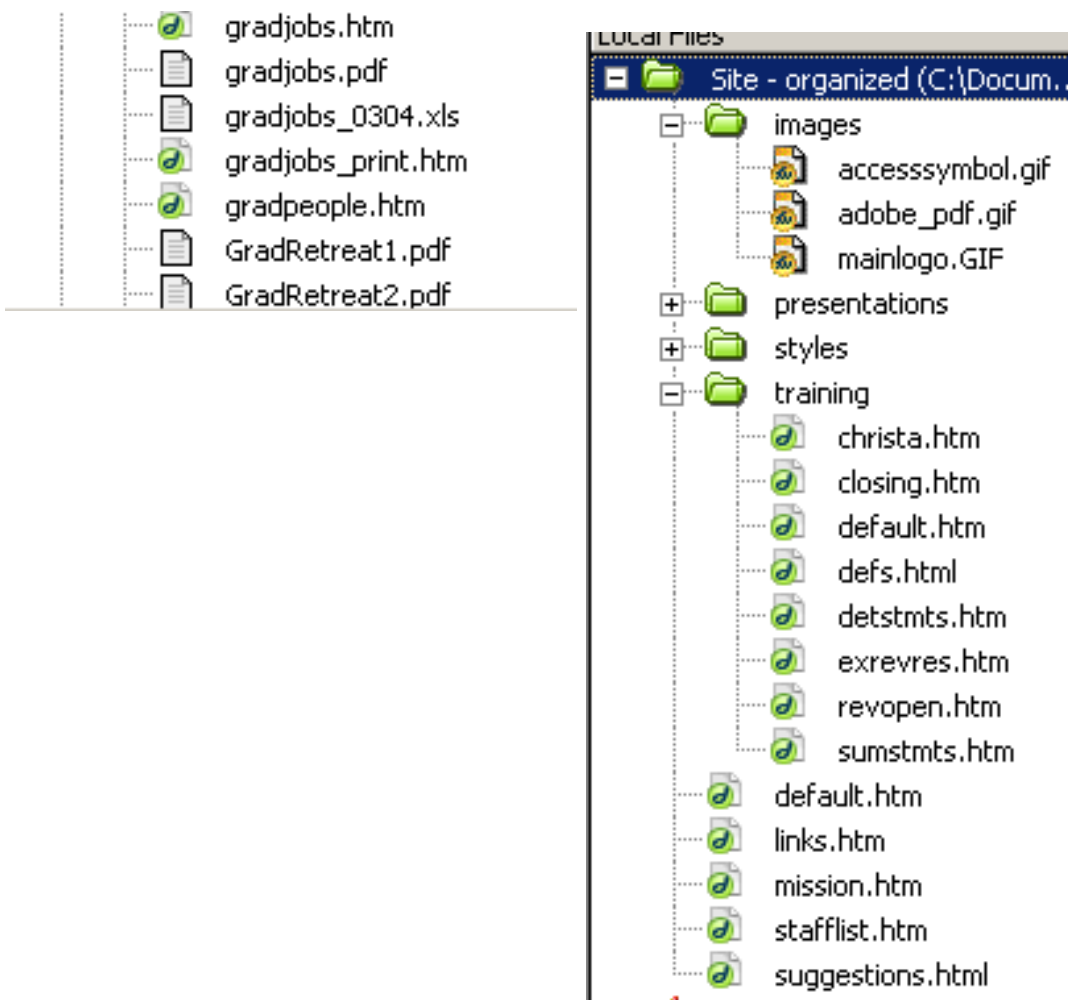
Local Files	Notes	Size	Type
Site - disarray (C:\Doc...)			
Folder			
	accesssymbol.gif	6KB	GIF Image
	adobe_pdf.gif	3KB	GIF Image
	index.htm	16KB	HTML Document
	links.htm	17KB	HTML Document
	mainlogo.GIF	6KB	GIF Image
	mission.htm	14KB	HTML Document
	stafflist.htm	15KB	HTML Document
	suggestions.html	13KB	HTML Document

Because the site is small, it is not difficult to distinguish between the pages (.htm) and the images (.gif). But sites can grow quickly:

	English903.doc
	examform.doc
	examtips.htm
	examtips_print.htm
	excover.htm
	excover_print.htm
	exit.htm
	exit_test.cfm
	exit_test.htm
	exreqfrm.htm
	external_fellow.htm
	external_print.htm
	faculty.htm
	faculty_print.htm
	fellow.htm
	fellow_print.htm
	fictexam.htm
	filmexam.htm
	folkexam.htm
	geliot.gif
	govern.htm
	govern_print.htm
	grad_pro.gif
	grad_program_13.gif
	gradarea.htm
	gradarea_print.htm
	gradbook.pdf
	gradcal.htm
	gradcal_print.htm
	gradjobs.htm
	gradjobs.pdf

The image on the left is only a partial view of a site that has all its files in a single folder. Can you see how difficult it might be to manage a site like this? Which page leads to which other pages? What documents are linked to what html files? What images belong on what pages?

Consider, instead, this organized site:



Note that all the training-related pages are in the "training" folder and only five pages reside in the top-level of the site: default, links, mission, stafflist, and suggestions.



In the organized site, images are located in one central folder, and all html pages are organized using a directory structure that is easy to understand.

[return to contents](#)

## Default and Index Pages



Each folder within your site should have a default or index page within it. If a folder contains a default or index page, when a user types the URL using the folder name, this page will be displayed. Without a default or index page, users must specify the exact file name they want to visit.

For example, let's consider again the sample site from "Site Structure" above. There is no default or index page in the "sectionone" folder. This URL will work:

<http://www.sample.edu/resources/presentations/sectionone/tutorial1.htm>.

This URL will not:

<http://www.sample.edu/resources/presentations/sectionone>

Depending on the server settings, the user who types the second URL will see a "File Not Found" message or a plain-text display of an alphabetical list of all the files in that folder -- without indication of which page the user should look at first. However, if we copy and rename tutorial1.htm to default.htm or index.htm, both URLs will work.

Thus, it is good design practice to name one file in each folder within the site either default or index. Note: if you have a folder with both a default and an index page, the browser will display the default page.

[return to contents](#)

## Naming Conventions

For best results across various servers and with different browsers, when naming files and folders within your web site:

- Use only lower-case characters.
- Do not use spaces: the underscore ( \_ ) or dash ( - ) can be used instead.
- Do not use special characters such as punctuation marks, dollar signs, or other characters like: @, #, ^, \*, [, (
- Shorter names are better -- remember it will be included in the URL of the page (e.g., webtutorialone.htm v tutorial1.htm). However . . .
- Avoid cryptic or non-descript names (e.g., t1s5c7.htm v tutorial1sec5chp7.htm)-- users will not be able to remember them and they can be confusing when managing the site.

[return to contents](#)

## Layout

[Organization](#) | [Using Style Sheets for Consistency](#) |

[Standard Page Elements: header information, link to home page, navigation to major sections, search option](#), and a [link to OSU home page](#).

## Organization.



Create a style of presentation that is consistent across pages.  
([WCAG 14.3](#))

To optimize the usability of your site: pick a place for stable and repeating elements, such as logos, navigation bars, and even the bulk of the content, and keep them there.

For instance, the WAC site uses the same basic layout for each page: search box at the top, home page link at the top-left, navigation on the left, and content on the right. Here's a sample (from the [WAC Links page](#)):

Search WAC site:   [Search Help](#)



[Accessibility: Why Bother?  
Overview of Accessibility  
Guidelines,  
Links and Resources,](#)  
Tutorial

### OSU Web Accessibility Center

Universal Web Design Assistance for Faculty, Staff and TA's

#### Links and Resources

OSU Links:

[Fast Facts for Faculty](#)

[WAC's Top Ten Tips for Designing Accessible Web Pages](#)

[Web Accessibility: How and Why?](#)

[Web Accessibility Standards \(Working Draft\)](#)

[OERI Info on Web Accessibility](#)

However, certain pages in our site, like those in this tutorial, move the logo and navigation to the top of the page. This design is used for pages that contain a lot of information and we expect users might want to print out and save for later. The top-heavy design allows more room for text on the page and reduces the number of printed pages. Here is an example of this layout (from the [How and Why Guide](#)):



[WAC How and Why?](#)  
[Links and Resources,](#)  
[Tutorials,](#)  
[Staff,](#)  
[About the WAC.](#)

1760 Neil Avenue  
150 Pomerene Hall  
Columbus, OH 43210-1297  
Phone: 292-3307  
Email: [webaccess@osu.edu](mailto:webaccess@osu.edu)

Search WAC site:   [Search Help](#)

---

## Accessible Web Site Design -- Why and How

- [Part II., Guidelines for an Accessible Site](#)
  - [What are the guidelines that identify an accessible site?](#)
  - [Understanding Section 508 standards](#)
  - [What is the difference between Section 508 and the WCAG guidelines?](#)
  - [What are the WCAG Priority One issues and how are they determined?](#)
  - [10 basic considerations when designing a web site](#)
- Go to: [Part I.](#) or [Part III.](#)

### What are the Guidelines that identify an accessible site?

OSU requires web developers to meet or exceed the [OSU Web Accessibility Policies and Standards](#), which are based on [Section 508 of the Rehabilitation Act of 1973](#). Section 508 requires that certain guidelines or priorities are met to insure those with visual, auditory, physical, mental, or learning disabilities are able to access electronic information.

---

[return to contents](#)

## Using Style Sheets for Consistency

If used properly, Cascading Style Sheets (CSS) can help you create a uniform "look" for your site by defining font type, colors, and backgrounds site-wide.

For more information, see the [Style Sheets](#) section in this guide.

[return to contents](#)

## Standard Page Elements.

As the Web has developed, users have come to expect certain elements on every web page they visit. Including these elements in the expected locations helps users quickly get the "lay of the land" whether they arrive at your stunning home page or via a "deep link" somewhere deep within the structure of your site.

[return to contents](#)

## Include on every page:

- **Complete header information, including [doctype](#), [language declaration](#), [title](#), [meta description](#), [meta keywords](#), and [style sheet link \(if used\)](#).** Header information helps users find the pages they want and identify the page they are on. It also helps browsers correctly interpret your pages. For more information on header information, see the [Headers](#) section in this guide.
- **A link to your home page**, (usually located in the upper-left corner of the page. (But, as long as your consistent across the site, can be moved to a different location). The link should either be in a text link or represented by an easily identified graphic link, such as an image of a home or a program logo.
  - The [Center for the Study and Teaching of Writing](#) site shows a good example of a home page link using a logo.
  - The [College of Biological Sciences](#) uses a image-for-text logo. In most cases, designers should avoid using images where text would work just as well. However, logos are the exception to the rule.
  - The [Department of Computer and Information Science](#) uses a text link, which appears in the gray navigation bar on the top-right. This is a good example of changing the expected location (top-left), but keeping consistent, so users can quickly orient themselves to the site.
- **Navigation to the major sections of your site.** For more information on creating user-friendly and accessible navigation, see the [Navigation](#) section in this guide.
- **A search option.** In general, the search should appear as close to the top of the page as possible, and preferably before navigation. OSU has an easy and useful search engine that you can use on your site. To see complete information on including the OSU search on your page, visit OSU's [HTML resources](#) page.

Here is an example of a simple search form using the OSU search engine:

Search site:

[Search Help](#)

Here is the HTML code used to create this search box. Simply change the URL information to your own site location:

```
<form action="http://search.uts.ohio-state.edu/query.html" method="GET">
<p><input type="hidden" name="qs" value="+url:yoursite.osu.edu">
<label for="search">Search site:</label>
```

```
<input name="qt" type="text" value=" " size="30" maxlength="30" id="search">
<input type="submit" value=" SEARCH " width="75" alt="Seek" border=0 name="submit"
id="search" accesskey="s">
<a accesskey="h" href="http://search.uts.ohio-state.edu/help/">Search Help</a></p>
</form>
```

- **A link to the OSU home page.** If your site is hosted by a college, department, or affiliated program of The Ohio State University, you should include a link to the OSU home page, both as a convenience to visitors, as well as an identifier that makes your association clear. In addition, many colleges require departments and programs to display a link back to the college home page.
  - To find out more about when and how to link back to the OSU home page, visit the [IT Polices](#) page from the [Office of the Chief Information Officer](#).
  - To download a copy of the OSU masthead, visit the [Approved Ohio State Web Graphics](#) page.

For more information on navigation guidelines, see the WAI's [Web Content Accessibility Guidelines 1.0](#).

[return to contents](#)

## Setting Header Information

Every web page in your site should contain standard header information that identifies the [version of HTML used to create the page](#) and the [primary language](#), as well as gives a [page title](#), [a summary of the page](#) and provides [key words](#) for easier searching.

[DOCTYPE Declaration](#) || [Language Declaration](#) ||  
[Title](#) || [META-Description](#) || [META-Keywords](#)

[return to contents](#)

### DOCTYPE Declaration.



Create documents that validate to published formal grammars.  
[\(WCAG 3.2\)](#)

Example: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">`

Doctype is short for "document type declaration" (or DTD). Any (X)HTML Web page needs to define a Doctype : it is used to declare which version of (X)HTML the Web page conforms to. It appears as the first line of code on a web page, before the `<html>` tag.

To find your correct DOCTYPE declaration, see the [WAI's list of valid DTDs you can use in your document](#). Some web editors, like Dreamweaver, allow you to define what DTD you want to use to validate your code and then automatically includes the appropriate DOCTYPE declaration in each new file.

[return to contents](#)

## Why Declare a DOCTYPE?

In general the overall accessibility of any web page will be improved if you adhere to the standards for the web-presentation language you are using. The major reason for this is that most developers of assistive technology or accessible web technology have based their accessibility features on the premise that content authors are following the standards.

In addition, most web browsers will more or less follow the published standards. Writing the "cleanest" code possible should ensure your pages will look best across the widest range of web display technology.

If your coding meets the requirements of a particular grammar, use the DOCTYPE statement as the first line of your HTML file. For example, the following DOCTYPE statement would indicate to servers, browsers and validators that you expect your code to conform to the HTML 4.0 Transitional standard:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

For more information on this guideline, see the WAI's [Web Content Accessibility Guidelines 1.0](#). Adapted from: WAI -- "[Example for Checkpoint 3.2 - Create documents that validate to published formal grammars](#)".

[return to contents](#)

## Language Declaration.



Clearly identify changes in the natural language of a document's text and any text equivalents. ([WCAG 4.1](#))

Example: `<HTML lang="en">` or `<HTML lang="en-US">`

lang = language-code [CI]

Two-letter primary codes are reserved for [ISO639] language abbreviations. Two-letter codes include fr (French), de (German), it (Italian), nl (Dutch), el (Greek), es (Spanish), pt (Portuguese), ar (Arabic), he (Hebrew), ru (Russian), zh (Chinese), ja (Japanese), hi (Hindi), ur (Urdu), and sa (Sanskrit). For a complete list of available language codes, see the [WAI's ISO 639 Language Codes](#) or the [OASIS CoverPages Code for the Representation of the Names of Languages. From ISO 639, revised 1989.](#)

[return to contents](#)

## Why Declare a Primary Language?

The language declaration is an attribute that defines the HTML tag. Language information specified via the lang attribute may be used by a browser to control display in a variety of ways. Some situations where author-supplied language information may be helpful include:

- Assisting search engines.
- Assisting speech synthesizers.
- Helping a user agent select glyph variants for high quality typography.
- Helping a user agent choose a set of quotation marks.
- Helping a user agent make decisions about hyphenation, ligatures, and spacing.
- Assisting spell checkers and grammar checkers.

For more information on language declaration, see the WAI's [Web Content Accessibility Guidelines 1.0](#).

[return to contents](#)

## Title.



## Provide metadata to add semantic information to pages and sites. ([WCAG 1.1](#))

Example: `<title> WAC Tutorial: Before You Build </title>`

The TITLE tag appears within the `<head></head>` tag in your document.

There may only be one title in any document. It should identify the content of the document in a fairly wide context.

The title is not part of the text of the document, but is a property of the whole document. It may not contain anchors, paragraph marks, or highlighting. It is not normally displayed in the text of a document itself.

The title should ideally be less than 64 characters in length. That is, many applications will display document titles in window titles, menus, etc where there is only limited room. Whilst there is no limit on the length of a title (as it may be automatically generated from other data), it may be truncated if long.

[return to contents](#)

## Why Use a Title?

Titles are used in:

- window title bars,
- bookmark lists,
- result lists from search services.

The title appears in the window of the browser and is also used in tracking the HISTORY (BACK button) of where the user has been. It can be particularly helpful when a user has multiple windows open, if it uniquely identifies each page. For example, here is a sample of a Windows status bar with multiple pages open from a site that does not use page titles:



Which of these “Untitled Documents” has the information you want to return to? Keep in mind, too, most screen-readers read the title as a means of orienting users

as they move through different windows.

**Quick Tip:** Be sure to use descriptive titles. "Section One" won't help much when it shows up in a search results list. Section one of what?. Better: `<title>Section One of the Modern Music Guide.</title>`

For more information on designing good titles, see the [W3C's Style Guide: Title](#).

[return to contents](#)

## META-description.



Example: `<meta name="Description" content="This is a summary of my page.">`

META tags appear within the `<head></head>` tags in your documents.

[return to contents](#)

## Why Use the META-description?

Some search engines pick up this description to show with the results of searches, instead of showing the first few of lines of the page as the summary. Thus, a description can help users identify the content of your pages when searching, especially when the first lines of your pages consist of repeated content or menus.

You can define your own search fields with HTML meta tags. These fields can be searched using Infoseek's field syntax.

Suppose your page contains:

```
<meta name="author" content="William Shakespeare">
```

Infoseek will index this page so the that it can be found with any of the queries:

author: Shakespeare

author: "William Shakespeare"

author: "william shakespeare"

[return to contents](#)

## META-keywords.



Example: `<META name="keywords" content="web, accessibility, access, design, center,">`

META tags appear within the `<head></head>` tags in your documents.

[return to contents](#)

## Why Use the META-keywords?

A common use for META is to specify keywords that a search engine may use to improve the quality of search results. When several META elements provide language-dependent information about a document, search engines may filter on the lang attribute to display search results using the language preferences of the user. For example,

```
<-- For speakers of US English -->
< META name="keywords" lang="en-us"
content="vacation, Greece, sunshine">
```

```
< -- For speakers of British English -->
< META name="keywords" lang="en"
content="holiday, Greece, sunshine">
```

```
< -- For speakers of French -->
< META name="keywords" lang="fr"
content="vacances, Gr&egrave;ce, soleil">
```

[return to contents](#)

## Navigation

[Function of Navigation](#) | [Skip Navigation Link](#) | [Using Pop-up Windows](#) | [Repeating Link Text](#) | [Plug-in Links](#) | [Using Image Navigation](#) | [Mystery Meat Navigation](#) | [Link Order](#) |

## Keyboard Shortcuts

### Provide clear navigation mechanisms:



- **14.3 Create a style of presentation that is consistent across pages. [Priority 3]**
- **13.4 Use navigation mechanisms in a consistent manner. [Priority 2]**
- **13.5 Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3]**

([WCAG 13](#))

[return to contents](#)

## Understanding the Function of Navigation

The Web offers users multiple entry points to sites, as well as multiple navigation paths within sites. Users may enter your site through the "front door" (home page), or a search engine may lead them straight to a page within the site. Users also bring with them different levels of Internet competency, different languages and cultural backgrounds, and different mental models of finding information (some may prefer to browse, some to search, while others like to go straight to an index). Consistent, predictable navigation will help every user make the most of your site.

Following is a list of things to keep in mind when establishing guidelines related to navigation:

- Include standard navigation information on all pages. As a rule, each page should provide a link to at least the main page, site map, search, and basic contact information. In addition, it is often useful to include links to the main sections of the web site.
- Provide at least some contextual information about where the user is in relation to the section of the web site and the web site as well.
- Give users an indication of where the next link will take them—for example, what type of information it leads to, and whether it takes them off site.
- Make navigation terms as descriptive and straightforward as possible —"clever" or unusual naming schemes are likely to confuse users.
- Decide on a consistent format for your site and stick to it.
- Provide the shortest possible routes to information.

- Ensure that your overall navigation structure will remain stable in the face of additions to the site. You don't want to have to revise the navigation structure on each page every time a new page is added.

Adapted from: ["Visual Design and Navigation Guidelines" By Ann Tohill, Association for Progressive Communications \(APC\) May 2001.](#)

[return to contents](#)

## Using Pop-up Windows

 The logo for the M.W.A.S. (Masters of Web Accessibility Standards) is located in the top left corner of the box. It consists of a red square containing the letters 'M', 'W', 'A', and 'S' in white, stacked vertically. To the right of the square, the text 'OHIO STATE UNIVERSITY' is written in red.	<p><b>17) Do not change the current window without informing the user. (<a href="#">M.W.A.S. 17</a>)</b></p>
---	--

Links that open in new windows can be extremely disorienting to users of assistive technology. These links break the "BACK" and "HISTORY" list. Screen readers may or may not indicate a new window has opened, causing links to appear "dead" -- e. g., nothing appears to happen when the link is activated because new window opens without notice.

Carefully consider when to use links that open in new windows. Generally, there is little advantage to forcing users to keep your site in an open window unless you are certain they will need to return to your site after viewing other pages (e.g. when leaving a form to get more information on required inputs, users may then be expected to return to the form page without the form being reloaded and entered data being cleared). Identify any links that open in a new window using a referenced symbol or plain text.

Example:

Find out more [information about this topic\\*](#).

Find out more [information about this topic \(opens in new window\)](#).

Note: Links with an asterisk open in a new window.

[return to contents](#)

## Skip Navigation Link


 508

A method shall be provided that permits users to skip repetitive navigation links. ([508.o](#)) ([M.W.A.S. 15](#))



Web developers routinely place a host of routine navigational links at a standard location – often across the top, bottom, or side of a page. If a nondisabled user returns to a web page and knows that he or she wants to view the contents of that particular page instead of selecting a navigation link to go to another page, he or she may simply look past the links and begin reading wherever the desired text is located. For those who use screen readers or other types of assistive technologies, however, it can be a tedious and time-consuming chore to wait for the assistive technology to work through and announce each of the standard navigational links before getting to the intended location.

Because the "skip navigation" link is designed to be used primarily by screen-readers, it does not need to be located in a prominent or even visible part of the page. It should, however, appear as close to the start of the page as possible, right after the BODY tag.

Here is how the link appears in HTML code:

Skip navigation link:

```
< a href="#content">skip navigation</a>
```

Anchor is placed before the start of the main body of text:

```
< a name="content"></a>Welcome to the ADA Coordinator's Office . . .
```

For more ideas on how to setup your skip navigation link, see the [WAC 508 Tutorial: Skip Navigation](#).

[return to contents](#)

## Using Image Navigation.

Several year's ago, a trend began to make image-based navigation with navigation "buttons." Here is an example of image navigation:



Each box in the above navigation bar is an image with an associated link. With the addition of [ALT tags](#), navigation based on images can still be accessible. However, something else to consider when using text in a graphic: some users with low vision use programs that enlarge the elements on their screen so that they can more easily see them. Unfortunately, when the text in an image is enlarged it often becomes pixilated and difficult to read.

For example, take a look at the [Department of Molecular Genetics home page](#) (opens in new window). If you increase the text size of the view (use View, Text Size in Internet Explorer or View, Text Zoom in Netscape Navigator), the text in the menu does not increase. This is because this is an image navigation -- each entry in the menu is a small graphic.



As a general design rule, avoid using images to represent text. Instead, format menu text using a [style sheet](#).



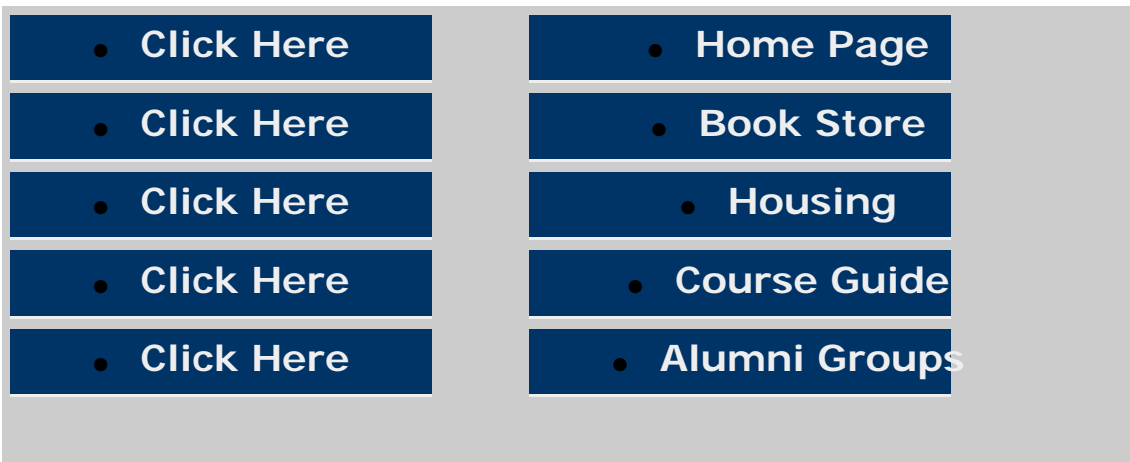
Clearly identify the target of each link. ([WCAG 13.1](#)) ([M.W.A.S. 18](#))



[return to contents](#)

## Repeating Link Text

Link text should be different for each target. Use link text that makes sense in a list.



[return to contents](#)

## Identify Plug-in Links

Identify "plug-in" links: PDS, DOCs, Movies, Audio.

[view my \[.pdf\] then listen to my \[mp3\]](#)

[return to contents](#)

## Mystery Meat Navigation

While there are ways to make image navigation accessible (e.g., using ALT tags), web designers have moved to an even more disturbing trend -- Mystery Meat Navigation. Notably, Mystery Meat Navigation represents a greater problem for sighted users than blind users. Here is an example of this problematic navigation technique:



There are eight links in this navigation bar, yet, it is unclear where any of them go. Only the home page link, represented by an image of a house, is a somewhat universally recognized link symbol. But where does the arrow-up lead to? How about little man running? Or the open book? Not only will users have to guess where the information they want is, returning visitor's will have to play a version of

the [memory game](#) to remember which images they clicked on before.

Vincent Flanders' site [Web Pages That Suck](#) offers an excellent [overview to the problems of Mystery Meat Navigation](#), along with some examples from current sites.

You can avoid the problems of Mystery Meat Navigation by following these simple guidelines:

1. Do not use images for menu items, or, if you must use images for menu items:
2. Provide visible captions to menu images,
3. Provide detailed ALT tags to menu images, and
4. Indicate visited and non-visited links in menus.

[return to contents](#)

## Link Order.



When a user uses the TAB button to scroll through the links on a page, the page is usually followed left to right, top to bottom. While this read-order may work well for most navigation schemes, some become confusing. Here is an example of navigation table:

Links for Student	Links for Faculty	Links for Staff
<a href="#">Registration</a>	<a href="#">Meetings</a>	<a href="#">Schedule</a>
<a href="#">Course Descriptions</a>	<a href="#">Reimbursement Requests</a>	<a href="#">Special Events</a>
<a href="#">Future Course Offerings</a>	<a href="#">Internal Procedures</a>	<a href="#">Team Leaders</a>

If tab order is not specified, the links will be highlighted left to right -- "Registration," "Meetings", "Schedule" -- rather than up and down.

To change this, you must set the `tabindex` attribute in the link element. Example:

1. `<a href="#sample" tabindex="1">Registration</a>`
2. `<a href="#sample" tabindex="2">Course Descriptions</a>`
3. `<a href="#sample" tabindex="3">Future Course Offerings</a>`

Note: If you use the `tabindex` attribute, it is important to set the `tabindex` for all links and elements on the page.

[return to contents](#)

## Keyboard Shortcuts.



**Provide keyboard shortcuts to important links. ([WCAG 9.5](#))**

Keyboard access to active elements of a page is important for many users who cannot use a pointing device. Some assistive technology software may include features that allow users to bind keyboard strokes to certain actions. HTML 4.01 allows content developers to specify keyboard shortcuts in documents via the "accesskey" attribute. In general, it is a good idea to add "accesskey" options to frequently used elements, such as navigation links, as well as any element other than links that might otherwise require the use of a mouse click (such as the SEND button on a form). The WAC uses access keys for our navigation menus. Here is a sample of our HTML code:

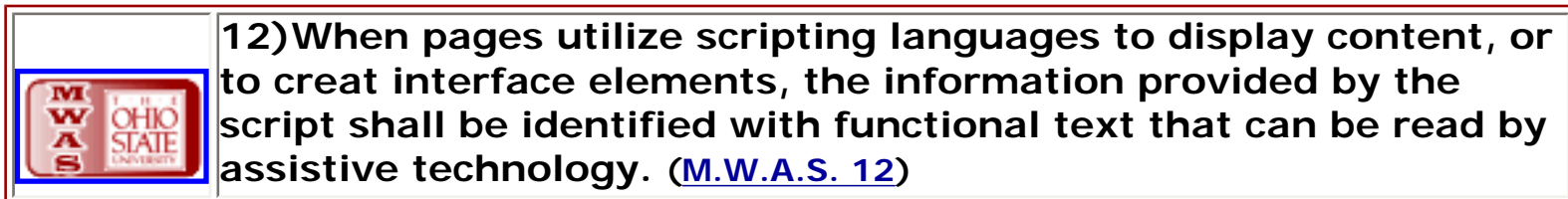
```
<ul class="ulnobullet">
<li class="smallertxt"><a accesskey="W" href=" ../why_howi.htm"><acronym title="Web
Accessibility Center">WAC</acronym> How and Why?</a></li>
< li class="smallertxt">< a accesskey="L" href=" ../links.htm"> Links and Resources.</a></
li>
< li class="smallertxt">< a accesskey="T" href=" ../tutorials.htm">Tutorials.</a></li>
<li class="smallertxt"><a accesskey="S" href=" ../stafflist.htm">Staff.</a></li>
<li class="smallertxt"><a accesskey="A" href=" ../mission.htm">About the <acronym
title="Web Accessibility Center">WAC</acronym>.</a></li>
</ul>
```

[return to contents](#)

# Scripts and Plug-ins

[return to contents](#)

## Using Scripts:



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.

Javascript is one of the most popular scripting languages for web development. In addition to libraries of Javascript code available online, two of the most popular web development tools, FrontPage and Dreamweaver, include easy-to-access code snippets that add little "extras" to web pages. Developers should use caution when deciding to include Javascript or other scripting elements on their pages. Here are a few of the most popular uses for Javascript and some of the accessibility issues to consider:

- **onClick** – The onClick event handler is triggered when the user clicks once on a particular item. It is commonly used on links and button elements and, used in connection with these elements, it works well with screen readers. If clicking on the element associated with the onClick event handler triggers a function or performs some other action, developers should ensure that the context makes that fact clear to all users. Do not use the onClick event handlers for form elements that include several options (e.g. select lists, radio buttons, checkboxes) unless absolutely necessary.
- **onMouseOver and onMouseOut** – These two event handlers are very popular on many web sites. For instance, so-called rollover gif's, which swap images on a web page when the mouse passes over an image, typically use both of these event handlers. These event handlers neither can be accessed by the mouse nor interfere with accessibility – a screen reader simply bypasses them entirely. Accordingly, web designers who use these event handlers should be careful to duplicate the information (if any) provided by these event handlers through other means.
- **onChange** – This event handler is very commonly used for triggering

JavaScript functions based on a selection from within a <select> tag. Surprisingly, it presents tremendous accessibility problems for many commonly used screen readers and should be avoided. Instead, web developers should use the onClick event handler (associated with a link or button that is adjacent to a <select> tag) to accomplish the same functions.

For more information on accessibility issues when using Javascript, see:

- University of Wisconsin-Madison, Trace Center's [Javascript Accessibility Issues](#).
- Web Accessibility In Mind (WEBAIM), "[Creating Accessible Javascript](#)."
- World Wide Web Consortium (W3C), "[Script Techniques for Web Content Accessibility Guidelines 2.0](#)"

[return to contents](#)



**12) 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 1-9 of the MWAS. Modification of 29 U.S.C. 794d 1194.22(m). (M.W.A.S. 13)**

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).

While most web browsers can easily read HTML and display it to the user, several private companies have developed proprietary file formats for transmitting and displaying special content, such as multimedia or very precisely defined documents. Because these file formats are proprietary, web browsers cannot ordinarily display them. To make it possible for these files to be viewed by web browsers, add-on programs or "plug-ins" can be downloaded and installed on the user's computer that will make it possible for their web browsers to display or play the content of the files. This provision requires that web pages that provide content such as Real Audio or PDF (Adobe Acrobat's Portable Document Format) files also provide a link to a plug-in that will meet the software provisions. It is very common for a web page to provide links to needed plug-ins. For example, web pages containing Real Audio almost always have a link to a source for the necessary player. This provision places a responsibility on the web page author to know that a compliant application exists, before requiring a plug-in.

Here is an example from the WAC's links page:

[Training Video on PDF Accessibility from AccessIT](#) (requires [Windows Media Player](#)): "PDF Accessibility" a presentation by Terry Thompson, Technology Specialist with [AccessIT](#), offers detailed help with evaluating PDF files for accessibility and creating accessible PDF using MS Word and Adobe Acrobat plug-ins.

WAC provides a link to the streaming video file, and a link to the Windows Media Player download page, to allow users who don't have the WMP plug-in installed to access video. WAC also provides a link to the home page of AccessIT, in case the video file moves or is offered in other formats on the host site.

[return to contents](#)

## Using Time Out Scripts .

 **16) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required. (M.W. A.S. 16)**


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

Web pages can be designed with scripts so that the web page disappears or "expires" if a response is not received within a specified amount of time. Sometimes, this technique is used for security reasons or to reduce the demands on the computer serving the web pages. Someone's disability can have a direct impact on the speed with which he or she can read, move around, or fill in a web form. For instance, someone with extremely low vision may be a slower-than-average reader. A page may "time out" before he is able to finish reading it. Many forms, when they "time out" automatically, also delete whatever data has been entered. The result is that someone with a disability who is slow to enter data cannot complete the form. For this reason, when a timed response is required, the user shall be alerted via a prompt and given sufficient time to indicate whether additional time is needed

[return to contents](#)

## Color

## Color used for layout.

	<p><b>Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup. (508.c) (<a href="#">WCAG 2.1</a>) (<a href="#">M.W.A.S. 3</a>)</b></p>
	
	

How does colored text affect the blind or the visually impaired? What happens to color-related information when presented on a monochrome monitor, printed from a black and white printer, read by someone who is color blind or read by a screen reader? The color-coded information is lost.

### EXAMPLE:

"Here is a list of questions from the final. The items in RED are mandatory for an A in the class; the items in BLACK are provided for extra credit.

What is the name of the course?  
 What is your name?  
 What is my name?  
 How many times did you come to class?"

Ensure that text and graphics convey the correct meaning when viewed *without* color.

The above information, when restated, will convey importance to those unable to utilize color while still using color as visual clues for the visual user.

"Here is a list of questions from the final. The first 2 items are mandatory for an A in the class; items 3 and 4 are provided for extra credit.

1. What is the name of the course?
2. How many times did you come to class?
3. What is your name?

#### 4. What is my name?"

Color is used, however the information is conveyed in a way that relays the importance and layout with dependence on color.

[return to contents](#)

### Contrast Colors.



Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen.

([WCAG 2.2](#)) ([M.W.A.S. 4](#))



Background color and text color should have the highest contrast. Persons with low vision, monochrome monitors or learning disabilities (and those who don't want to strain their eyes) will be unable to view your text if it is not easily read. Lighthouse International has [excellent information on the use of high-contrast colors](#) and [useful guidelines on legible text](#).

Red on black is a favorite of some people

AND SOME CAN'T RESIST  
THESE TYPES OF FONTS.

even if the contrast is good

and who can resist the scrolling text

Test your pages by printing them or by setting your browser to black, white and a few shades of gray. If the information is readable, contrast is good and the information conveyed without color then the text and color layout of the page is good.

[return to contents](#)

# Cascading Style Sheets

[Using Style Sheets for Consistency](#) || [Linked v Embedded Styles](#) || [Guidelines for Style Sheets](#) || [Using Relative Sizes](#) || [Color and Links](#) || [Spacing and Positioning](#) || [Creating User-Friendly Style Sheets](#) || [CSS Guides and Tutorials](#)

[return to contents](#)

## Using Style Sheets for Consistency.



**Create a style of presentation that is consistent across pages. (WCAG 14.3)**

If used properly, Cascading Style Sheets (CSS) can help you create a uniform "look" for your site by defining font type, colors, and backgrounds site-wide.

The design benefits of using Cascading Style Sheets include:

- more precise control than ever before over layout, fonts, colors, backgrounds, and other typographical effects;
- a way to update the appearance and formatting of an unlimited number of pages by changing just one document;
- compatibility across browsers and platforms; and
- less code, smaller pages, and faster downloads.

[return to contents](#)

## Linked v Embedded Styles.



**Use linked style sheets rather than embedded styles. (WCAG 14.3)**

Style sheets can be employed using either embedded styles, where the style definitions are inserted between the HEAD element on each page, or linked styles, where a CSS file is created and linked to the pages using those styles.

## Embedded Styles.

Here is an example of how to create embedded styles:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<HTML lang="en">
<HEAD>
<TITLE>WAC Tutorial: Before You Build -- Style Sheets</TITLE>
<STYLE type="text/css">
p { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: small; color: #000000; }
a:link { font-family: Verdana, Arial, Helvetica, sans-serif; color: #000099; text-decoration:
underline; font-weight: bold; }
a:visited { font-family: Verdana, Arial, Helvetica, sans-serif; color: #666666; text-decoration:
underline; font-weight:bold; }
a:hover { color:#FFFFFF; background-color: #000066; }
</STYLE>
</HEAD>
```

Embedded styles look and work just like a linked style. However, it is more difficult to keep the site consistent, because changes to one style class must be changed on each page of the site. Also, embedded styles are more likely to override user preferences in the browser.

[return to contents](#)

## Linked Styles.

Using linked styles involves two steps. First, you must create your styles in a separate CSS file (designated by using the .css extension in your file name -- e.g., wacstyles.css, mystyles.css, prettytext.css). CSS files can be created using a wizard in your web editor (e.g. Dreamweaver or FrontPage) or you can create it in a plain text editor, like NotePad. The style definitions look the same as the embedded styles.

```
p {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: small;
color: #000000;
}
```

```

a:link {
font-family: Verdana, Arial, Helvetica, sans-serif;
color: #000099;
text-decoration: underline;
font-weight: bold;
}
a:visited {
font-family: Verdana, Arial, Helvetica, sans-serif;
color: #666666;
text-decoration: underline;
font-weight:bold;
}
a:hover {
color:#FFFFFF;
background-color: #000066;
}

```

Next, you must link the CSS file to each of your pages using this style. Let's say this file is called "mystyles.css." In order to link this style sheet to a particular page in your site, your HTML would look like this:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<HTML lang="en">
<HEAD>
<TITLE>WAC Tutorial: Before You Build -- Style Sheets</TITLE>
<LINK href="mystyles.css" rel="stylesheet" type="text/css">
</HEAD>

```

By using the link method, changes made to styles in the mystyles.css file will update all the linked pages in the site, without changing any of the code in those pages. This is particularly helpful when changing the color scheme of a site and helps insure a consistent look across pages.

[return to contents](#)

## Guidelines for Using Style Sheets.



Use style sheets to control layout and presentation. ([WCAG 3.3](#))

Specifically covered within this guideline:

1. Use the CSS 'font' property instead of the HTML FONT element to control font styles. -- [WCAG CSS Techniques #8 Text Formatting and Position](#).

- **Incorrect method:** Here is some `<font color="#000066">blue text</font>`.
- **CSS method:** `<style type="text/css">.bluetext { color: #000066; }</style>`  
Here is some `<span class="bluetxt"> blue text </span>`.
- **CSS Example:** **This text is colored using CSS.**

Rationale: Unlike CSS formatting the FONT element in HTML is more likely to override user-selected formatting in the browser. For instance, if the FONT element is used to set the size of text on a page, the user cannot use the magnification settings in the VIEW menus of either Internet Explorer or Netscape Navigator to increase the text size.

2. Do not use incorrect HTML elements for formatting (e.g., using BLOCKQUOTE to indent, rather than creating an indent style) -- [WCAG CSS Techniques #8 Text Formatting and Position](#).

- **Incorrect method:** `<p><blockquote> This text is indented. </blockquote></p>`
- **CSS method:** `<style type="text/css"> .indent {margin-left: 5%; margin-right: 5%;}</style>`  
`<p class="indent">This text is indented.</p>`
  - **CSS Example:** This text is indented using CSS.

Rationale: Special HTML elements were created to identify special text and help assistive technology fully render the meaning of the text. For instance, whenever text is surrounded by the BLOCKQUOTE tag, JAWS screen-reader identifies the text as a "quote." Consider then, how JAWS would render this text:

```
<p> I want you to focus on the text below, so I'm going to indent it:
<blockquote>
  <blockquote>
    <blockquote>
      This is the text I want to emphasize. </blockquote></
blockquote></blockquote></p>
```

JAWS would read this text as:

"I want you to focus on the text below, so I'm going to indent it. Quote. Quote.

Quote. This is the text I want to emphasize. End quote. End quote. End quote."

If much of the text on the page is indented in this fashion, you can imagine how annoying this technique can become for users of assistive technology.

3. Use style sheets to style text rather than representing text in images.-- [WCAG CSS Techniques #7 Text Instead of Images](#).

- **Incorrect method:** **Red Large Text**
- **CSS method:** `<style type="text/css">.redargetext { font-family: Garamond, Arial, Helvetica, sans-serif; font-size: x-large; font-weight: bold; color: #CC3300;}</style><p class="redargetext">Red Large Text</p>`
- **CSS Example:** **This text appears larger and a different color using CSS.**

Rationale: Text represented in images cannot be enlarged using built-in magnification tools in most browsers. And, if image-text is magnified, it typically becomes pixilated and unreadable. CSS formatted text will transform gracefully, regardless of the users screen or window resolution.

[return to contents](#)

## Using Relative Sizes.



**Use relative rather than absolute units in markup language attribute values and style sheet property values. ([WCAG 3.4](#))**

The goal of CSS is to make pages that transform gracefully regardless of users screen-size, resolution, or magnification. When designers set font and element sizes to absolute values, text and elements on the page cannot not "resize" to fit the needs of the user. Relative sizes create a relational system of size. For instance, if Header 2 text is set to be bigger than paragraph text and Header 1 text is set to be bigger than Header 2 text, the user can set the paragraph text to whatever size (i.e., 12 point, 14 point, or 20 point) and both the Heading 1 and Heading 2 text will increase in relation to that size. In the same way, defining the menu section of a page to take up 30% of the page and content 70%, versus 150 pixel width menu and 650 pixel width content, insures that users with smaller screens and lower resolutions will still see a page that is primarily content.

## 1. Use the "em" unit to set font sizes.

Named after the letter "M", the em unit has a long-standing tradition in typography where it has been used to measure horizontal widths. For example, the long dash often found in American texts (--) is known as "em-dash" since it historically has had the same width as the letter "M". Its narrower cousin (-), often found in European texts is similarly referred to as "en-dash".

In CSS, the em unit is a general unit for measuring lengths, for example page margins and padding around elements. You can use it both horizontally and vertically. 1 em unit equates to a 10 point font.

Here is an example of a style defined using em:

```
H1 { font-size: 2em }
```

For more information on using the em unit of measure, see W3C's [The amazing em unit and other best practices](#).

## 2. Use relative length units and percentages.

CSS allows you to use relative units even in absolute positioning. Thus, you may position an image to be offset by "3em" from the top of its containing element. This is a fixed distance, but is relative to the current font size, so it scales nicely. Only use absolute length units when the physical characteristics of the output medium are known, such as bitmap images.



For more good ideas for formatting your text, see W3C's Quality Assurance article: ["Care with Font Size."](#)

[return to contents](#)

## Color and Links.

Most web browsers use different colors to identify visited links (links to pages the user has seen before) and unvisited links (links to pages the user has not seen before). The generally accepted standard typically displays unvisited links in blue and visited links in either purple or red. In most cases, it is best to use this standardized color scheme to identify links on your pages. When non-standard link colors are used, users lose the ability to quickly identify which parts of the site they have already seen and which parts they have yet to visit.

If you must change use an alternate color scheme for your links, consider using a close variation on the blue/purple standard. Otherwise, help make it clear to visitors which is which by using a dark/strong color for unvisited links and lighter/faded color for visited links. In this way, you de-emphasize visited links – places where visitors already know about and possibly did not find the information they were looking for.

Use these CSS properties to specify colors:

- 'color', for foreground text color.
- 'background-color', for background colors.
- 'border-color', 'outline-color' for border colors.
- For link colors, refer to the :link, :visited, and :active pseudo-classes.

[return to contents](#)

## Spacing and Positioning.

Spacing and positioning continues to be a difficult problem for those conscious of accessibility issues. Many users have developed strategies for adding spaces and indents, such as manually entering spaces (&nbsp;) or incorrectly using the BLOCKQUOTE tag, that creates an acceptable visual result, but also makes for less-accessible content. Creating and applying appropriate CSS definitions can help avoid this problem.

Use CSS properties to create space between elements, words, and letters:

- 'text-indent', 'text-align', 'word-spacing', 'font-stretch'. Each of these properties allows users to control spacing without adding additional spaces. Use 'text-align: center' instead of the deprecated CENTER element.
- 'margin', 'margin-top', 'margin-right', 'margin-bottom', 'margin-left'. With these properties, authors can create space on four sides of an element's content instead of adding non-breaking spaces (&nbsp;) or BLOCKQUOTE.

[return to contents](#)

## Creating User-Friendly Style Sheets.

508

Documents shall be organized so they are readable without requiring an associated style sheet. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document. ([508.d](#)) ([WCAG 6.1](#)) ([M.W.A. S. 5](#))

W3C 1  
WCAG 1.0M  
W  
A  
S  
OHIO STATE  
UNIVERSITY

One of the accessibility advantages to using style sheets is the option for the user to either override your style sheet with his/her own or to turn-off style sheets altogether and render the page in settings established by the user in the browser. Thus, the user can turn off color, change the default color scheme to increase contrast, or even change the font type and size to a more readable setting.

To insure your site can accommodate these user preferences, you need to test your site both with and without style sheets. When style sheets are removed does all the text appear in the correct order on the page? Does the page maintain its hierarchy with headings indicating the structure of the document? Do the default colors interfere with the readability of the text?



**Quick Tip:** the free, online validator, [Bobby](#) automatically turns off style sheets when evaluating your site. Run your page through Bobby to see what it will look like without your style sheet.

[return to contents](#)

Developers should pay special attention when [using CSS to create content](#), such as background images, lists, or content properties and when [using CSS for layout](#). It should also be noted that most of these CSS techniques are not yet fully supported by browser technology. Be sure to test any CSS2 techniques in multiple browsers and versions.

[return to contents](#)

## Lists



## Mark up lists and list items properly. ([WCAG 3.6](#))

Lists offer particular advantages for users of assistive technology when interpreting the organization of information. In particular, lists help group like and related information together and provide information on the extent and relationship of information. For instance, when JAWS encounters a bulleted list, it will read "List of X entries" (where X=the number of bullets in the list) before reading the items in the list.

Lists become problematic when used improperly; in particular, when used for formatting, when the format or organizational structure is indecipherable, when the list includes inaccessible or annoying elements, such as repeating spaces, graphic bullets, and too many "filler" characters used for aesthetic purposes only (e.g., à entry 1 || à entry 2, etc.). To avoid these list pitfalls, be sure to follow these list guidelines:

1. Do not use incorrect HTML elements for formatting. The HTML list elements DL, UL, and OL should only be used to create lists, not for formatting effects such as indentation. Formatting should be done using [style sheets](#).

**2. For numbered lists, use compound numbers whenever possible.** Thus, a list numbered "1, 1.1, 1.2, 1.2.1, 1.3, 2, 2.1," provides more context than the same list without compound numbers, which might be formatted as follows:

1.
  - 1.
  2.
    - 1.
    - 2.
    - 3.
2.
  - 1.

and would be spoken as "1, 1, 2, 1, 2, 3, 2, 1", conveying no information about list depth.

**3. Use style sheets to change list bullets.** Avoid using images as bullets in definition lists created with DL, DT, and DD. However, if this method is used, be sure to provide a text equivalent (ALT tag) for the images.




[return to contents](#)

## Images and Multimedia

[Text Equivalents](#) || [Captioning Multimedia](#) || [Flicker](#) || [Text-only Sites](#)

[return to contents](#)

### Text Equivalents for Images, Media, and Scripts

	<p>A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content). <a href="#">(508.a)</a> <a href="#">(WCAG 1.1)</a> <a href="#">(M.W.A.S. 1)</a></p>
	
	

Non-text elements include: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video

#### Sample Code:

Image without alternate description:



```

```

Image with alternate description:



```
< img src="../../images/wacapp.gif" alt="WAC Approved icon" width="61" height="34">
```

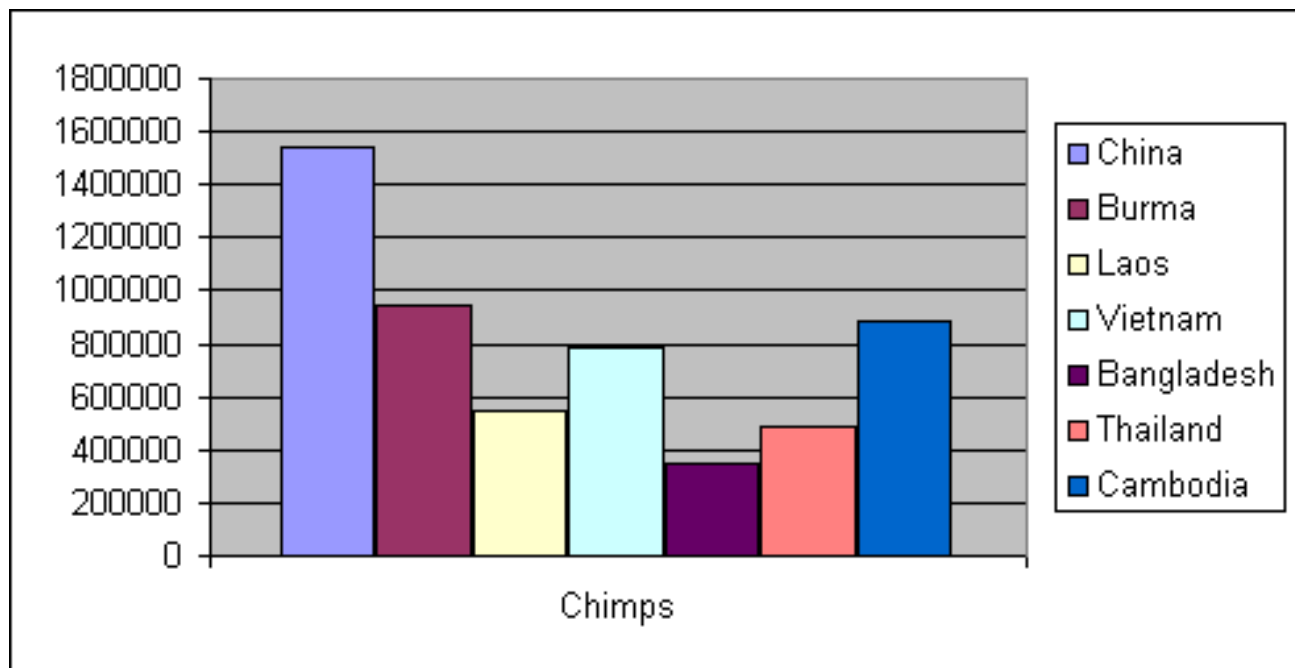
Put your mouse over each image to see the difference. Alternate descriptions are read by screen-readers and appear when an image is unavailable from the server or when the user chooses to "turn-off" or "hide" images in his/her browser.

## Caption or LONGDESC?

Keep in mind, for simple content, a text equivalent may need only describe the function or purpose of content. For complex content (charts, graphs, etc.), the text equivalent may need to be longer and include descriptive information. The LONGDESC attribute is designed for these situations. However, the attribute is not yet fully supported by screen-reader technology.



For best results, use the ALT tag even with longer descriptions or include a caption immediately above or below the image and reference the caption in the ALT tag. For example:



Caption: Graph 2.3 shows the population of Chimps in each location: China = 1,590,000; Burma= 900,000; Laos = 500,000; Vietnam= 800,000; Bangladesh = 380,000; Thailand = 470,000; Cambodia = 855,000.

Code Example for Above:

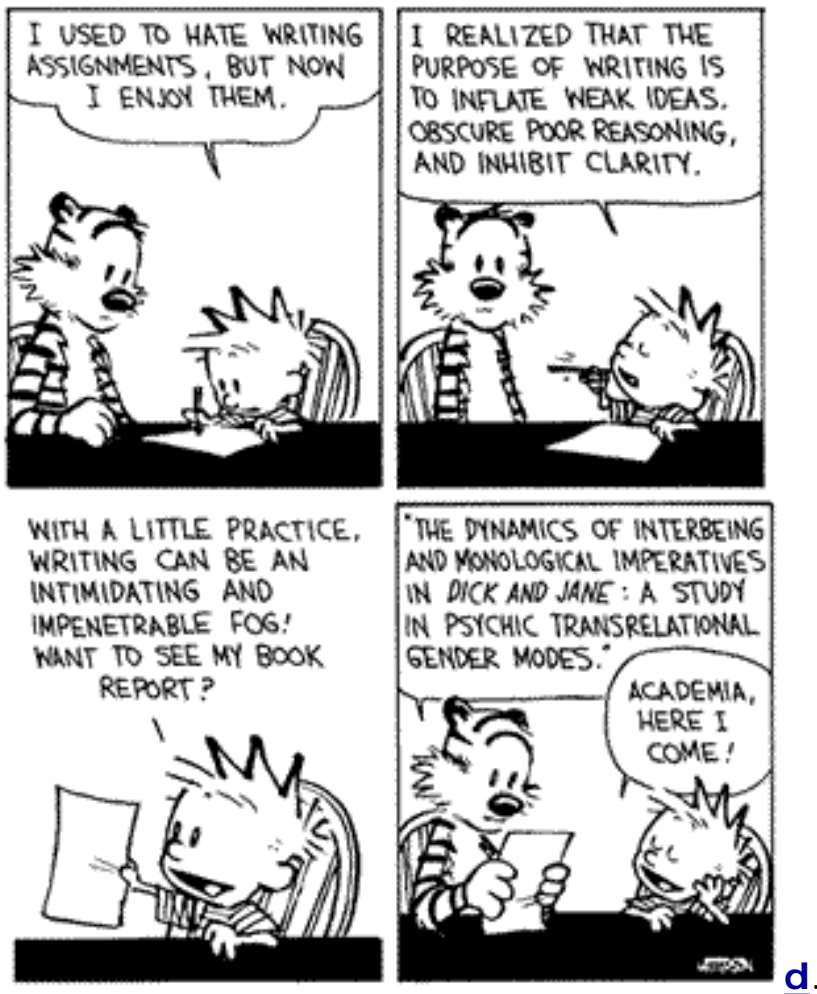
```
<p class="caption">  
<br>  
Caption: Graph 2.3 shows the population of Chimps in each location: China = 1,590,000;  
Burma= 900,000; Laos = 500,000; Vietnam= 800,000; Bangladesh = 380,000; Thailand =  
470,000; Cambodia = 855,000.</p>
```

## LONGDESC and the D-LINK

Since the LONGDESC attribute is not yet fully supported, the WAC recommends including necessary long descriptions in the form of a caption that appears above or below the image. However, sometimes it will be necessary to provide more detailed information than even a caption allows. If this is the case, you will need to use the LONGDESC method.

If you do decide to use the LONGDESC option, WCAG standards require you also use a corresponding D-link. Here's how it works: First, you create a separate HTML page that includes the long description or explanation of the image or object. Next, you link the image to that description in the LONGDESC attribute of the <IMG> element. Finally, you create a text link immediately following (to the right) of the image or object and the link text should read "d".

Here is an example:



The cartoon above has a simple, short ALT tag and includes a link, both in the LONGDESC attribute and the D-link, to a more detailed description of what is happening in this cartoon. Hint: click on the "d" to read the long description.

Here is the code used to insert this image and d-link:

```
<p>

<a href="/tutorials/bestpractices/calvin-writing.htm">d</a>.
</p>
```

The potential problem with D-links is that using multiple links on a page creates repeated link text that leads to different locations (an accessibility no-no). One way to address this issue is to number multiple D-links (e.g., d1, d2, d3, etc.). As browsers develop more LONGDESC support, it will no longer be necessary to include the D-link as well.

[return to contents](#)

## Spacer Images.

Web page authors often utilize transparent graphics for spacing. Adding a text description to these elements will produce unnecessary clutter for users of screen readers. For such graphics, an empty ALT attribute is useful.

Example of source code: `<IMG src="transparent.gif" alt="">`

Empty ALT tags can also be used for images and graphics that are purely aesthetic, such as graphic dividers, borders, and lines.




[return to contents](#)

## Testing for ALT tags.

To test your ALT tags, using Internet Explorer, go to the TOOLS menu and click on "Internet Options." Click on the "Advanced" tab and scroll down to the "Show Pictures" check box. Uncheck the box, close the dialog and reload your page. Do all of your image placeholders have ALT tags?

[return to contents](#)

## Captioning Multimedia

	<b>Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation. (508.b) (WCAG 1.4) (M.W.A.S. 2)</b>
	<b>Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation. (WCAG 1.3)</b>
	

Two important elements of captioning are included in these guidelines: synchronization and auditory descriptions.

Synchronization insures that hearing-impaired users see the images associated with

the caption. For instance, a video of a professor discussing a particular piece of art work shows the correct frame of the art as the professor draws attention to each element.

Auditory descriptions provide visually-impaired users descriptions of key elements in the visual track. For example, if the professor says "Don't these two look like they are having fun?" and the image being shown is a picture of two crying children, the caption would describe the picture to make the meaning clear. Also captioned would be other key visual elements including actions, settings, body language, and displayed text.

Synchronization also applies to auditory descriptions. Descriptions should not interfere with the audio or dialogue of a movie or presentation.

[return to contents](#)

## Example.

Caption for professor's lecture.

"Let's take a look at this example."

[puts image of decorative vase on overhead projector.]

[Vase is tall and thin, bud vase, in very pale blue with small etched flowers.]

" Notice the intricate etching in this piece. The artist clearly spent a great deal of time with this piece."

[return to contents](#)

## Flicker



**Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz. ([508.J](#)) ([WCAG 7.1](#)) ([M.W.A.S. 11](#))**




While flashing and flickering elements often appeal to web designers to draw attention to information, changes and updates, or just to add aesthetic value, returning users and those who endure repeated loops of movement while trying to access needed information will quickly tire of your pages. In addition, causing a page or elements on a page to flash or flicker can have serious effects on users with epilepsy and light sensitivity. Flashing and moving graphics can also be extremely distracting and just plain annoying to other users.

Here is an example of a page with too many flickering elements: [North Royalton Police Department](#).

If you must include flashing or flickering elements on your page, be sure animation changes at a slow enough rate to avoid injury to your visitors. Also, consider timing-out animated elements, so elements stop moving after a certain number of seconds or a specified number of animation loops.

[return to contents](#)

## Providing Equivalent Versions.

	<b>(k) A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of these standards, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes. (508.k) (WCAG 11.4) (M.W.A.S. 19)</b>
	
	

OSU Minimum Standards for Web Accessibility require that text-only or other accessible alternate versions of your primary site remain current with primary site. You must have a system in place to insure content of alternate versions is updated as frequently (preferably simultaneously) as the primary version.



Thus, when using the text-only alternative, the WAC recommends using auto-update software, like [LIFT Text Transcoder](#), which creates a server-generated text-only version on demand. Other database solutions are also possible with simple [PHP](#) or [ColdFusion](#) coding.

Keep in mind, OSU Minimum Standards for Web Accessibility and Section 508

require that an accessible alternative version of your site be used ONLY IF you are unable to make your primary site meet the standards. Text-only or alternative versions should be seen as last resort options and not the lazy-coder's solution to poorly designed sites.

[return to contents](#)

## Image Maps

[return to contents](#)

### Client-Side



**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. (M.W.A.S. 6)**

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. Because of the potential accessibility problems with server-side image maps, client-side image maps should be used in a majority of cases.

Here is an example of a client-side image map. There are two link "areas" on this image -- can you locate them?:



Here is the code that inserts the image and creates the image map:

```

<map name="Map">
<area shape="rect" coords="4,5,59,45" href="http://www.osu.edu" alt="OSU Home Page">
<area shape="rect" coords="77,3,396,50" href="http://ada.osu.edu/disabilitystudies"
alt="Return to Disability Studies Home Page">
</map>
```

Notice that each <area shape> tag has its own alt="" tag.

[return to contents](#)

## Server-Side



**Redundant text links shall be provided for each active region of a server-side image map . (M.W.A.S. 7 )**

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

An "image map" is a picture (often an actual map) on a web page that provides different "links" to other web pages, depending on where a user clicks on the image. There are two basic types of image maps: "client-side image maps" and "server-side image maps." With client-side image maps, each "active region" in a picture can be assigned its own "link" (called a URL or "Uniform Resource Locator") that specifies what web page to retrieve when a portion of the picture is selected. HTML allows each active region to have its own alternative text, just like a picture can have alternative text.

By contrast, clicking on a location of a server-side image map only specifies the coordinates within the image when the mouse was depressed. The ultimate selection of the link or URL must be deciphered by the computer serving the web page.

Most novice designers and webmasters of non-dynamic sites use only client-side image maps. Creation and use of server-side image maps requires advanced programming experience.

[return to top](#)

## Layout and Data Tables.

[General Guidelines](#)

[Layout Tables](#)

[Data Tables](#)

- [Types of Tables.](#)
- [Table Summaries.](#)
- [Using Layout Tables.](#)
- [Marking-up Layout Tables.](#)
- [Column and Row Headers.](#)
- [Headers in Practice.](#)
- [Abbreviated Headers.](#)

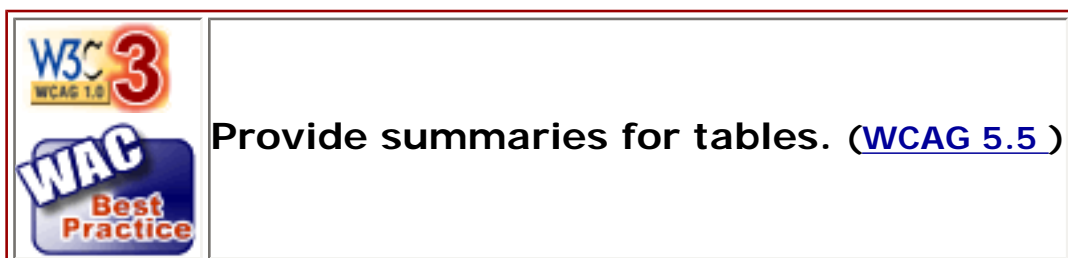
[return to contents](#)

## Types of Tables.

There are two types of tables used in web pages: [layout tables](#) and [data tables](#). In general, data tables require more careful design and coding to insure accessibility. However, layout tables should be used sparingly and designed for quick access to key content on a given page.

[return to contents](#)

## Summaries for Tables.



Regardless of whether it is a layout or data table, you should provide summaries for all tables via the "summary" attribute.

**For layout tables, the summary can describe the layout.**

Sample Code --

```
<table id="sample" summary="Layout table with two rows and three columns. First row contains a link to the home page and contact information. Second row: the left cell contains
```

the site menu, the middle cell contains the page content, and the right cell contains suggested external links for more help.">

Since many screen-readers are now able to distinguish layout tables from data tables, it is also permissible to simply identify a table as "layout table" in the summary.

## Summaries for data tables should clarify the relationships among cells.

Simple Code --

```
<table id="data" summary="Total required by pollution control standards as of January 1, 1971. Commercial category includes stores, insurance companies and banks. The table is divided into two columns. The left-hand column is 'Total investment required in billions of dollars'. The right-hand column is 'Spending' and is divided into three sub-columns. The first sub-column is titled '1970 actual in millions of dollars', the second is '1971 planned in millions of dollars', and the third is 'Percent change, 1970 versus 1971.' The rows are industries.">
```

[return to contents](#)

## Relative Sizing.



**Use relative rather than absolute units in markup language attribute values and style sheet property values. ([WCAG 3.4](#))**

The tables with absolute sizes do not transform gracefully for users with smaller screens and lower resolutions. A table set to 800 pixel-width will require a user with 640 resolution to scroll left-right to see all the content. This also affects a user who does not have the browser window maximized to take-up the whole screen. Tables with absolute sizes also tend to cut-off important information when the page is printed.

Sample HTML code:

Absolute table: `<table width="780" border="0" cellspacing="0" cellpadding="0" align="center">`

Relative table: `<table width="95%" cellpadding=0 cellspacing=20 align="center">`.




[return to contents](#)

## Guidelines for Layout Tables.

Large tables of data can be difficult to interpret if a person is using a non-visual means of accessing the web. Users of screen readers can easily get "lost" inside a table because it may be impossible to associate a particular cell that a screen reader is reading with the corresponding column headings and row names.

[return to top](#)

## Simple Data Tables .

	
	<b>Row and column headers shall be identified for data tables.</b> <a href="#">(508.g)</a> <a href="#">(WCAG 5.1)</a> <a href="#">(MWAS 8)</a>
	

To get an idea of how data tables are made accessible using the `<TH>` element, consider this table with both row and column headers:

Student Academic Level and Program.		
Student	Academic Level	Current Program
Lori Badia	M.A. completed	Ph.D. English
Susan Bailey	B.S. completed	M.B.A.
Alicia Baker	B.A., in progress	Dept. of Education

Here is the code used to define this table, including headers:

```
<table width="90%" border="2" id="datatable" summary="This table contains information on
```

student programs. Each row lists the student name, academic level completed to date, and current program enrolled.">

```
<tr>
<th scope="col"><p>Student</p></th>
<th scope="col"><p>Academic Level</p></th>
<th scope="col"><p>Current Program</p></th>
</tr>
<tr class="smallertxt">
<th scope="row"><p>Lori Badia</p></th>
<td><p>M.A. completed</p></td>
<td><p>Ph.D. English</p></td>
</tr>
<tr class="smallertxt">
<th scope="row"><p>Susan Bailey</p></th>
<td><p>B.S. completed</p></td>
<td><p>M.B.A.</p></td>
</tr>
<tr class="smallertxt">
<th scope="row"><p>Alicia Baker</p></th>
<td><p>B.A., in progress</p></td>
<td><p>Dept. of Education</p></td>
</tr>
</table>
```

As a screen-reader, like JAWS, reads this table, each data cell can be identified by either row or column header. For instance, when reading across row two, JAWS reads the row header:

"Student Lori Badia, academic level, M.A. completed; current program, Ph.D. English."

When reading down, JAWS reads the row header for each cell. For instance, when reading column three:

"Lori Badia, Ph.D. English; Susan Bailey, M.B.A., Alicia Baker, Dept. of English."

Row and column headers help insure that data tables can be understood regardless of how the user accesses the cells.

[return to contents](#)

## Complex Data Tables

508



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. (508 h) (WCAG 5.2) (MWAS 9)

For instance, consider this data table used in a biology lab:

DATA TABLE 1: Resting heart rate.				
<input type="checkbox"/>	Number of Beats Per Minute			Average Number of Beats Per Minute
Subject	sample 1	sample 2	sample 3	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Assume this table of resting heart rates may include 20 different subjects, each recording 3 measurements of resting beats per minute, with a calculated average across all 3 for each subject. Each row in the table represents a subject and each column represents either a resting heart rate or the calculated average. The table would have at least 100 data cells. Finding the 3rd measurement for the 11th subject may involve navigating as many as 14 rows and 4 columns. Without a method to associate the headings with each cell, it is easy to imagine the difficulty a user of assistive technology may encounter with the table.

Here is another example

Cups of coffee consumed by each senator

Name	Cups	Type of Coffee	Sugar?
T. Sexton	10	Espresso	No
J. Dinnen	5	Decaf	Yes

```

<TABLE border="1" summary="This table charts the number of cups of coffee
consumed by each senator, the type of coffee (decaf or regular), and whether
taken with sugar.">
<CAPTION>Cups of coffee consumed by each senator</CAPTION>
<TR>
<TH id="header1">Name</TH>
<TH id="header2">Cups</TH>
<TH id="header3" abbr="Type">Type of Coffee</TH>
<TH id="header4">Sugar?</TH>
<TR>
<TD headers="header1">T. Sexton</TD>
<TD headers="header2">10</TD>
<TD headers="header3">Espresso</TD>
<TD headers="header4">No</TD>
<TR>
<TD headers="header1">J. Dinnen</TD>
<TD headers="header2">5</TD>
<TD headers="header4">Yes</TD>
</TABLE>

```

This example shows how to associate data cells (created with TD) with their corresponding headers by means of the "headers" attribute. The "headers" attribute specifies a list of header cells (row and column labels) associated with the current data cell. This requires each header cell to have an "id" attribute.

[return to top](#)

## Abbreviated Headers



**Provide abbreviated alternatives for lengthy column and row headers.**

Designers can further support table accessibility by providing abbreviations for long headers. Consider the Student Academic Level and Program table again. When reading across a row, each time a cell is read by JAWS, the data is prefaced with either "student," "academic level," or "current program." We can assign

abbreviated terms for these headers to reduce the repetition. In this code example, we would change "academic level" to "level" and "current program" to "program":

```
<tr>
<th scope="col"><p>Student</p></th>
<th scope="col" abbr="level"><p>Academic Level</p></th>
<th scope="col" abbr="program"><p>Current Program</p></th>
</tr>
```

The abbreviated header still distinguishes the cell information, but results in quicker navigation through the table.



[return to contents](#)

## Forms

[Section 508: General Guidelines](#) || [Form Control Labels](#) || [Label Position](#) || [Logical Read-Order](#) || [Keyboard Shortcuts](#) || [Place-holding Characters](#)

[return to contents](#)

### General Guidelines for Forms.

	<p>(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. (508.n) (MWAS 14)</p>
	

The Section 508 standard for electronic forms addresses the issue from the users perspective rather than the designers. Instead of itemizing specific design elements that must be included in forms, Section 508 simply states that, however the form is design, it must be functional for users of assistive technology. In practice, this means designers must have a better understanding of how form elements are rendered in assistive technology and sites relying on forms for functionality should be tested using a variety of assistive technology and in a variety of platforms/

browsers.

In addition, the WAC recommends form pages clearly display a contact or help link that encourages users who have difficulty using the online form to report their problems. Of course, the contact or help method should not also require the completion of a form (such as the "Tell us what problem you are having" forms found on many commercial sites).

To further help form designers, the WCAG includes more specific guidelines for form development.

[return to contents](#)

## Form Control Labels.



**Associate labels explicitly with their controls. (WCAG 12.4)**

Each form element should have an associated label. In the past, form designers relied on nearby text to identify the purpose of a particular form element. For example, putting "First Name:" in front of a text box. However, if users of assistive technology read the page out-of-order (as frequently occurs) or must move around the form to make changes or re-enter information, the preceding text may be omitted. Without an associated label, JAWS reads a text box as "Blank, edit." -- an unhelpful description to say the least.

To avoid this problem, designers need only to enclose the existing text label in the <LABEL> tag and add an "id" name to text box. Here is an example:

```
<LABEL for="firstname">First Name:
<INPUT type="text" id="firstname" tabindex="1">
</LABEL>
```

Note that the "for" attribute in the LABEL tag must be identical to the "id" attribute of the form element the label describes. This is what is meant by an "explicit association." The result looks essentially the same:

First Name:

However, when the JAWS user navigates to the text box, JAWS no longer reads

“Blank, edit.” Instead, the text box is identified as “First name, edit,” a much more helpful description of what information should go into the edit box.

Explicit association of label and form element is particularly important for list boxes, combo boxes, radio buttons, and check buttons, in which several entries/options are associated with one form question. Consider this example of a radio button set:

Administrator	Faculty	Lecturer	GTA	Student	Staff
---------------	---------	----------	-----	---------	-------

Since the code that creates this whole table is kind of lengthy, let's just look at the first three cells:

```
<td>
<p align="center">
<label for="adminbtn">
<input type="radio" name="position" value="Administrator" id="adminbtn" tabindex="6">
Administrator</label>
</p>
</td>
<td>
<p align="center">
<label for="facultybtn">
<input type="radio" name="position" value="faculty" id="facultybtn" tabindex="6">
<br> Faculty
</label></p></td>
<td>
<p align="center">
<label for="lectbtn">
<input type="radio" name="position" value="Lecturer" id="lectbtn" tabindex="8">
Lecturer</label></p></td>
```

Each radio button has an associated label, which allows assistive technology to read the text appearing below the button whenever the button is activated. Since labels for buttons often appear above, below, and to the left or right of the button itself, they can be extremely difficult to navigate without associated labels.

Like many forms, the example button options appear within a layout table. While designed to help sighted users identify the grouping of like elements, tables add another layer of navigation difficulty for assistive technology users that can further disassociate the form element from its purpose. If you decide to use layout tables

in your forms, be sure you have reviewed the [elements of good table design](#) in this tutorial.

[return to contents](#)

## Positioning Form Control Labels.



**Until user agents support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned. (WCAG 10.2)**

As assistive technology advances, it includes more support for accessible HTML tags and attributes. However, not all users are working with the latest version of their technology and not all technology supports the same level of code structure. Technically, explicit association of labels with form elements means that the label does not need to be near/next to the element for assistive technology to understand and translate the relationship. Thus, with the use of explicitly associated labels, assistive technology could correctly interpret even a form that looked like this:

Administrator, Faculty, Lecturer, GTA, Student, Staff					

However, WCAG continues to require designers to follow the generally accepted practice of putting the label as close to the form element as possible (in this case, within the same cell as the associated radio button). Although practitioners continue to argue the best arrangement, generally speaking, labels appear immediately before or below for text boxes, and to the right of check boxes and radio buttons.

[return to contents](#)

## Creating a Logical Read Order.



**Create a logical tab order through form controls. (WCAG 9.4)**

If you don't specify a particular read-order for your form, screen-readers will try to interpret the page using a top-down, left-right default structure. But, forms in

particular can cause a number of problems when interpreted using this system. Consider this section of the 2003 U.S. Individual Income Tax Return form 1040:

-----

Election Campaign (See page 19.) **Note.** Checking "Yes" will not change your tax or reduce your refund.  
Do you, or your spouse if filing a joint return, want \$3 to go to this fund?  Yes  No  Yes  No

**Filing Status** Check only one box.

1  Single

2  Married filing jointly (even if only one had income)

3  Married filing separately. Enter spouse's SSN above and full name here. ▶

4  Head of household (with qualifying person). (See page 20.) If the qualifying person is a child but not your dependent, enter this child's name here. ▶

5  Qualifying widow(er) with dependent child. (See page 20.)

**Exemptions**

6a  Yourself. If your parent (or someone else) can claim you as a dependent on his or her tax return, do not check box 6a

b  Spouse

c Dependents:

(1) First name	Last name	(2) Dependent's social security number	(3) Dependent's relationship to you	(4) If qualifying child for child tax credit (see page 21)

No. of boxes checked on 6a and 6b \_\_\_\_\_

No. of children on 6c who:  
 • lived with you \_\_\_\_\_  
 • did not live with you due to divorce \_\_\_\_\_

If the "Filing Status" section of the form is read left-to-right and top-down, it would read "Single," then "Head of household," then "Married filing jointly," then the edit box for "Head of household, then "Married filing separately," and finally, "Qualifying widow." Consider the next section, "Exemptions." At what point would each of the entries on the far right column be read?

Creating a logical tab order resolves the problem of which form control gets read when and helps users input like and related information. The easiest way to create a logical tab order is add the "tabindex" attribute to each form control:

```
<FORM action="submit" method="post">
<P>
<LABEL for="field1">Email:</label>
<INPUT tabindex="2" type="text" name="field1" id="field1">
<LABEL for="field2">Name:</label>
<INPUT tabindex="1" type="text" name="field2" id="field2" >
<INPUT tabindex="3" type="submit" name="submit">
</FORM>
```

In this example, the tab order would read Name, Email, Submit, while reading top-down, the form would appear as Email, Name, Submit.

[return to contents](#)

## Keyboard Shortcuts.



**Provide keyboard shortcuts to form controls, and groups of form controls. (WCAG 9.5)**

Keyboard users often face difficulty with forms when trying to engage "submit,"

"reset," or other buttons or form controls usually activated using a mouse click. One strategy to avoid this problem is to be sure to assign a tabindex attribute to form controls, to insure the cursor can be "focused" on these elements. Another helpful tool is to create alternate key combinations or keyboard shortcuts to activate form controls. Keyboard shortcuts are created using the "accesskey" attribute:

```
<FORM action="submit" method="post">
<P>
<LABEL for="firstname" accesskey="F">First name</LABEL>
<INPUT type="text" tabindex="1" id="firstname">
<INPUT tabindex="2" type="submit" name="submit" accesskey="S">
</FORM>
```

This example assigns "F" as the accesskey. Typing "F" gives focus to the label, which in turn gives focus to the input control, so that the user can input text.

On long forms it may be implausible to assign keyboard shortcuts to every element, but you should still assign an keyboard shortcut to important form elements, like the submit button.

[return to contents](#)

## Place-holding Characters.



**Until user agents handle empty controls correctly, include default, place-holding characters in edit boxes and text areas.**  
([WCAG 10.4](#))

Although support for forms is improving, some assistive technologies require initial text in form controls in order to function correctly. A blank space can be used in most cases. However, if you expect a larger portion of your users to be working with older technology, use alphabetic characters instead. Here are two examples of text boxes:

Name:

Email:

Here is the code used to create these form elements:

```
<p align=left>
<label for="textfield">Name:</label>
<input type="text" name="textfield" accesskey="N" tabindex="1" id="textfield" value=" ">
</p>
<p align=left>
<label for="textfield2">Email:</label>
<input name="textfield2" type="text" id="textfield2" accesskey="E" tabindex="2"
value="enter email address">
</p>
```

In the first text box, the initial value is set to " ", or a blank space. In the second text box, the initial value is set to "enter email address." In both cases, users must delete or type-over the initial value.

[return to contents](#)

## Frames

[return to contents](#)

### General Guidelines for Frames.

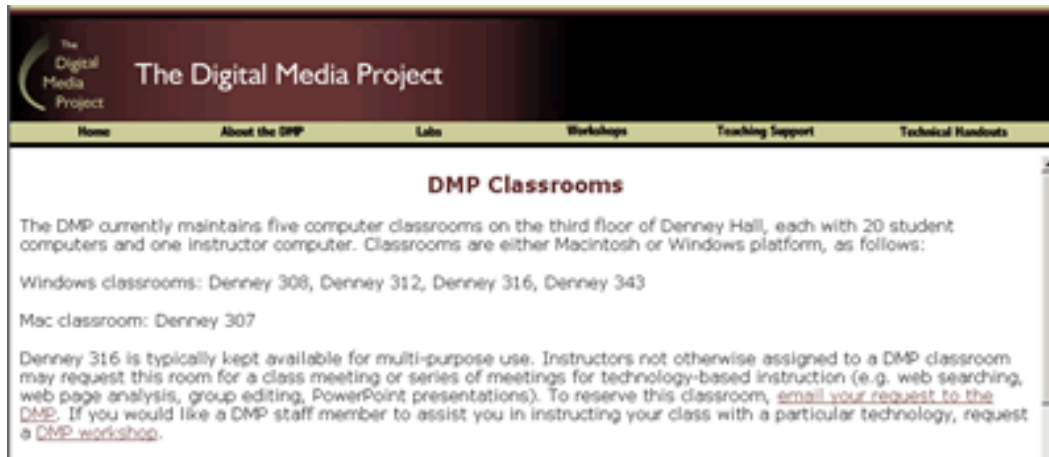


**10) Frames shall be titled with text that facilitates frame identification and navigation. ([M.W.A.S. 10](#))**

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

Frames provide a means of visually dividing the computer screen into distinct areas that can be separately rewritten. Unfortunately, frames can also present difficulties for users with disabilities when those frames are not easily identifiable to assistive technology. For instance, a popular use of frames is to create "navigational bars" in a fixed position on the screen and have the content of the web site retrievable by activating one of those navigational buttons. The new content is displayed another area of the screen. Because the navigational bar doesn't change, it provides a

stable "frame-of-reference" for users and makes navigation much easier. However, users with disabilities may become lost if the differences between the two frames are not clearly established.



The above image shows a web site that used to be based on frames. The top navigation area, including the program logo and navigation bar remain static while the information changes below. (Note: this site was later updated to remove frames. Visit the new, improved site at: <http://english.ohio-state.edu/programs/dmp>).

While Section 508 guidelines require that frames be titled and labeled to identify the changing areas of information, the WAC recommends that web developers avoid using frames in most cases. Instead, templates can be developed to manage repeating information (such as background color, logos, and navigation menus) across the site.

Before deciding to use frames in your site, we recommend you read more about the problems with frames:

Jacob Nielsen: "[Why Frames Suck \(most of the time\)](#)."

ALT.HTML: "[Why are Frames so Evil?](#)"

Web Design Group: "[Guide to Frame Usage](#)."

If you do decide to use frames, be sure to read the [Access Board's guidelines for creating accessible frames](#).

## Example:

```
<FRAMESET cols="15%, 85%">
```

```

<FRAME src="menu.html" title="Navigation menu" name="menu">
<FRAME src="content1.html" title="Main content" name="content">
<NOFRAMES> <P>This frameset document contains:
<UL> <LI><A href="menu.html">Page navigation</A></LI>
<LI><A href="content1.html">Main content</A></LI> </UL>
</NOFRAMES> </FRAMESET>

```




The WAC recommends including the <NOFRAME> tag as shown above to allow users who do not have frames enabled to still see the content that is displayed within the frame.





[return to contents](#)

## About the Guidelines Referenced in this Tutorial.

This tutorial brings together a collection of accessibility and usability guidelines, standards, and best practices for web developers.

### Key for this Tutorial

Logo	Source for Guideline/Standard	More Information
	<p>Guideline from the "<a href="#">Web-based Intranet and Internet Information and Applications (1194.22)</a>" subsection of Section 508 of the Rehabilitation Act. OSU requires web developers to meet or exceed the <a href="#">OSU Web Accessibility Policies and Standards</a>, which are based these standards.</p>	<p>Specific help for each guideline is referenced throughout the tutorial. Look for the (Sec 508) link at the end of each description. For more help, see the <a href="#">Access Board's Guide to the Standards</a>.</p>

	<p>Priority 1 issue from the <a href="#">Web Content Accessibility Guidelines</a> of the <a href="#">World Wide Web Consortium's Web Accessibility Initiative (WAI)</a>, A Web content developer must satisfy this checkpoint. Otherwise, one or more groups will find it impossible to access information in the document. Satisfying this checkpoint is a basic requirement for some groups to be able to use Web documents.</p>	<p>Specific help for each guideline is referenced throughout the tutorial. Look for the (WCAG) link at the end of each description. See the complete list of <a href="#">Web Content Accessibility Guidelines 1.0</a>.</p>
	<p>Priority 2 issue from <a href="#">Web Content Accessibility Guidelines</a> of the <a href="#">World Wide Web Consortium's Web Accessibility Initiative (WAI)</a>, A Web content developer should satisfy this checkpoint. Otherwise, one or more groups will find it difficult to access information in the document. Satisfying this checkpoint will remove significant barriers to accessing Web documents.</p>	<p>Specific help for each guideline is referenced throughout the tutorial. Look for the (WCAG) link at the end of each description. See the complete list of <a href="#">Web Content Accessibility Guidelines 1.0</a>.</p>
	<p>Priority 3 issue from <a href="#">Web Content Accessibility Guidelines</a> of the <a href="#">World Wide Web Consortium's Web Accessibility Initiative (WAI)</a>, A Web content developer may address this checkpoint. Otherwise, one or more groups will find it somewhat difficult to access information in the document. Satisfying this checkpoint will improve access to Web documents.</p>	<p>Specific help for each guideline is referenced throughout the tutorial. Look for the (WCAG) link at the end of each description. See the complete list of <a href="#">Web Content Accessibility Guidelines 1.0</a>.</p>
	<p>Best Practice recommended by the W.A.C. Although not covered specifically by either Section 508 or the WCAG, WAC recommends web developers follow these guidelines to further usability for all users.</p>	<p>WAC usability guidelines are based on accepted best practices, as identified by, among others, noted experts <a href="#">Vincent Flanders</a> and <a href="#">Jacob Nielsen</a>.</p>

## Section 508

OSU requires web developers to meet or exceed the [OSU Web Accessibility Policies and Standards](#), which are based on Section 508 of the Rehabilitation Act of 1973. Section 508 requires that certain guidelines or priorities are met to insure those with visual, auditory, physical, mental, or learning disabilities are able to access electronic information. Standards covered by Section 508 include:

- [\(a\) Alternate Descriptions for Non-Text Elements](#)
- [\(b\) Synchronization with Multimedia](#)
- [\(c\) Use Color for Design, not Context](#)
- [\(d\) Using Style Sheets](#)
- [\(e\) Server-Side Image Maps](#)
- [\(f\) Client-side Image Maps](#)
- [\(g & h\) Data Table Headers](#)
- [\(i\) Using Frames](#)
- [\(j\) Flashing or Flickering Elements](#)
- [\(k\) Updating Text-Only Versions](#)
- [\(l\) Using Scripts to Create Dynamic Pages](#)
- [\(m\) Using Applets and Plug-Ins](#)
- [\(n\) Using Online Forms](#)
- [\(o\) Skip Navigation Link](#)
- [\(p\) Using Time Out Scripts](#)


For more help with Section 508 guidelines, see the WAC Tutorial: [Understanding and Applying Section 508 Standards](#).


## Using the WAI's Web Content Accessibility Guidelines


Depending on its intended use and audience and the complexity of the site design, web designers may want to adhere to the stricter (more detailed) [Web Content Accessibility Guidelines](#) of the [World Wide Web Consortium's Web Accessibility Initiative \(WAI\)](#), which cover not only basic web pages, but also

use of dynamic content with scripting, plug-ins, multimedia, and other advance design features.

The WCAG are categorized by priority:


- 


• **[Priority 1]**  
A Web content developer **must** satisfy this checkpoint. Otherwise, one or more groups will find it impossible to access information in the document. Satisfying this checkpoint is a basic requirement for some groups to be able to use Web documents.
- 


• **[Priority 2]**  
A Web content developer **should** satisfy this checkpoint. Otherwise, one or more groups will find it difficult to access information in the document. Satisfying this checkpoint will remove significant barriers to accessing Web documents.
- 

• **[Priority 3]**  
A Web content developer **may** address this checkpoint. Otherwise, one or more groups will find it somewhat difficult to access information in the document. Satisfying this checkpoint will improve access to Web documents.

The WAI allows designers to certify their sites under three levels of compliance:

- 

• Conformance Level "A": all Priority 1 checkpoints are satisfied;
- 

• Conformance Level "Double-A": all Priority 1 and 2 checkpoints are satisfied;
- 

• Conformance Level "Triple-A": all Priority 1, 2, and 3 checkpoints are satisfied.

Thus, a designer with a simpler site, may choose to make the site compliant to "Double-A" rating, while more advance sites may want to certify a "Triple-A" rating to insure all plug-in and scripting technologies are compliant as well.

**The WAC recommends that designers in academic institutions who choose to use the WAI's WCAG for validation should certify to at least the "Double-**

## A" rating.

**Note:** In addition to treating, in detail, methods for accessibility in complex web sites, the WCAG also strive to lead the way in establishing accessible design practices as emerging technology and design trends enter the market. Thus, many of the recommendations, such as using Style Sheets for layout as well as formatting, represent design elements that will be, but are not yet, fully supported by existing browser technology. Designers who choose to become fully compliant to the WCAG Priority 1, 2, and 3 issues may need to require use of only the most current version of available browsers and further may only support one or two browser types.