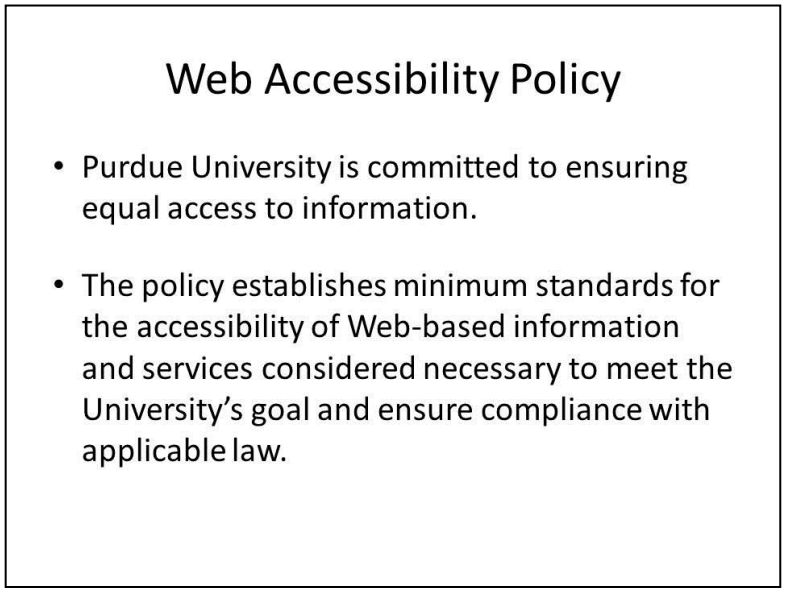
Accessible Web Design

Created by the Web Accessibility Committee
with select content from Web Accessibility in Mind

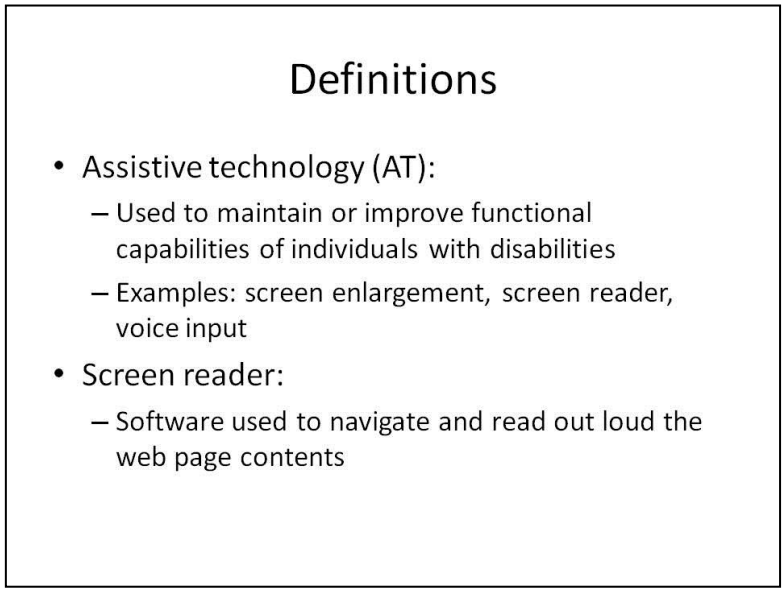
Table of Contents



Web Accessibility Policy

The creation and dissemination of knowledge is a defining characteristic of universities and is fundamental to Purdue University's mission to promote learning, discovery, and engagement. The use of digital and Web-based delivery of information is increasingly central to carrying out the University's mission. Acknowledging this fact, Purdue University is committed to ensuring equal access to information for all its constituencies.

The Web Accessibility Policy establishes minimum standards for the accessibility of Web-based information and services considered necessary to meet the University's goal and ensure compliance with applicable law.



Accessible Web Design

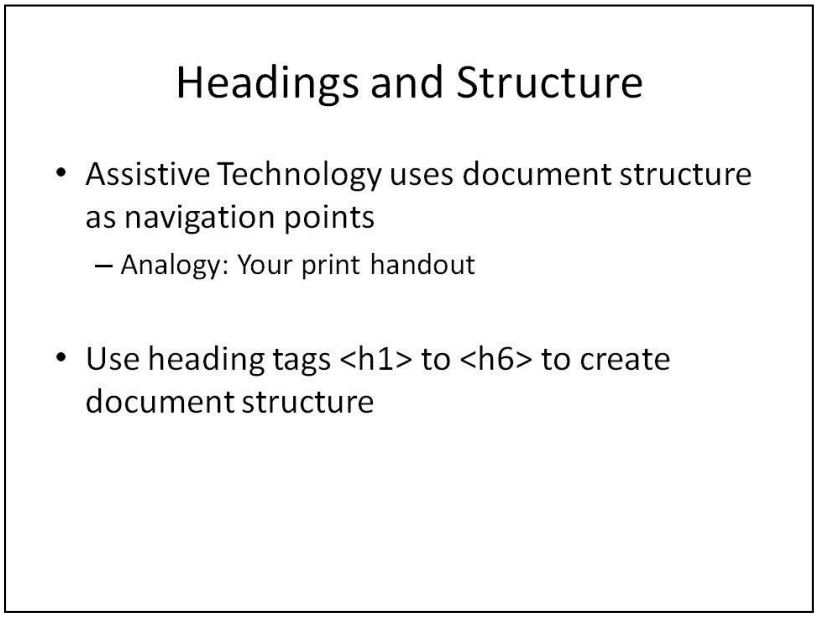
This training addresses common accessibility problems and the design techniques that address them.

We wish to thank the creators of Web Accessibility in Mind, or WebAIM (webaim.org), for allowing us to include information and examples from their site in our training materials.

Headings

Screen readers and other assistive technology use structural information to help make reading web pages more efficient. For example, most screen readers can skip from heading to heading or bring up a list of the headings available on the page. Therefore, it is important to identify headings using the appropriate markup instead of relying solely on formatting. For example, use `<h1>` tags to identify the top-level heading rather than simply making its text large and bold. Do not misuse structural markup for formatting effects. (Illinois Information Technology Accessibility Act Implementation Guidelines)

The content from here to the copyright notice was provided primarily by WebAIM. Their content is also available at: http://www.webaim.org/techniques/semanticstructure/



Using Headings and Lists for Content Structure

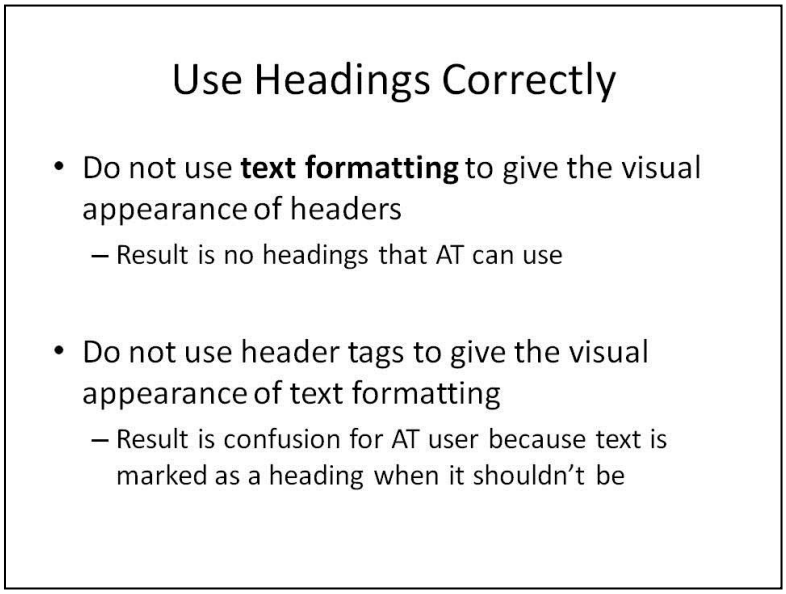
Despite the nature of the Web and the vast change in its role from a structural medium to a visual media, it is still important that Web content be designed with proper structure. With better support for Cascading Style Sheets in recent versions of Web browsers, developers can change the appearance of structural elements to meet their design and visual preferences.

Screen reader and other assistive technology users have the ability to navigate Web pages by structure. This means that the user can read or jump directly to top level elements (`<h1>`), next level elements (`<h2>`), third level elements (`<h3>`), and so on. Viewing or listening to this outline should give them a good idea of the contents and structure of the page.

Pages should be structured in a hierarchical manner, with 1st degree headings (`<h1>`) being the most important (usually page titles or heading), then 2nd degree headings (`<h2>` - usually major section headings), down to 3rd degree headings (sub-sections of the `<h2>`), and so on. Technically, lower degree headings should be contained within headings of the next highest degree.

The following outline shows the hierarchy of what a Web page might contain.

- Heading 1
    - Heading 2
        - Heading 3
        - Heading 3
    - Heading 2

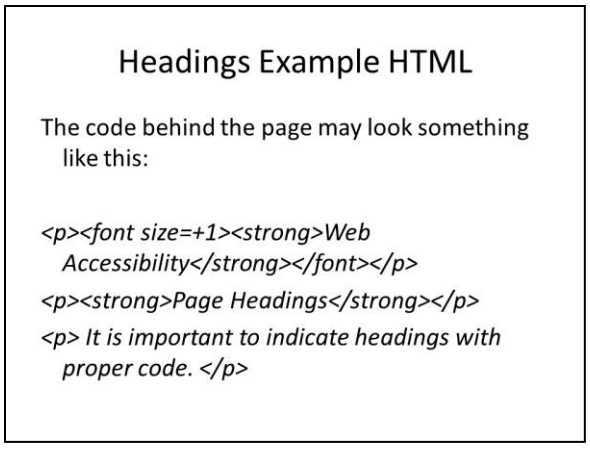


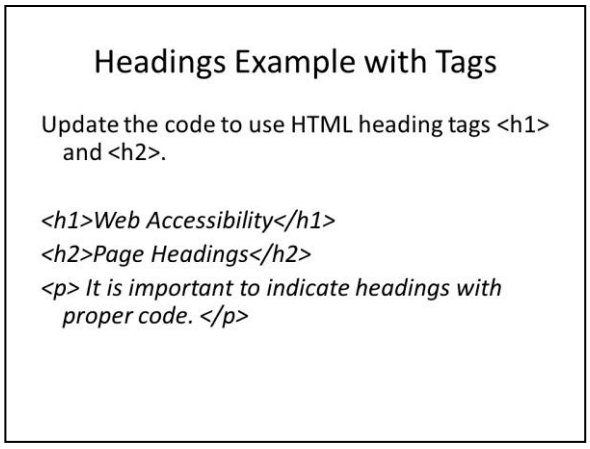
Using Headings Correctly

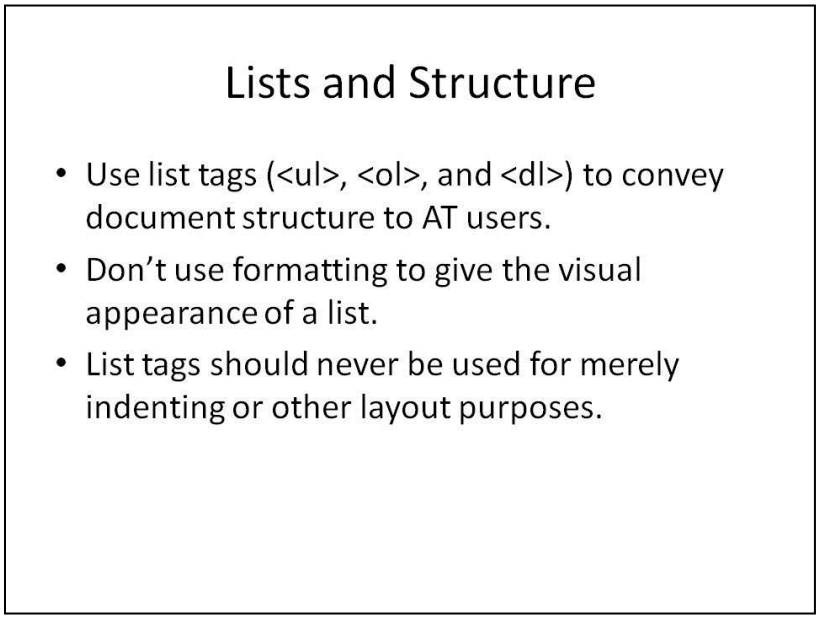
Think of heading tags as structural elements, not design elements. If you want header tags (h1, h2, etc.) to display differently, use CSS to change the appearance. This will not change how assistive technology treats the tag, but will change how other users see the header on the page.

Because headers are a structural element, **do not use text formatting such as font size or bold to give the visual appearance of headers**. Instead, use heading tags in their proper order to give your page structure. Assistive technologies and other browsers rely upon the literal markup of the page to determine structure. Items that are bolded or display in a bigger font are not interpreted to be structural elements.

Likewise, **do not use headers to achieve visual results only**. For instance, if you want to highlight or emphasize an element within your content that is not a heading (such as with the previous sentence), do not use heading tags to achieve the visual appearance you want. Instead, use font size, bold, or italics.







Using Lists Correctly

HTML lists - `<ul>`, `<ol>`, and `<dl>` - also convey a hierarchical content structure. Each of these has rules regarding their use as well.

Unordered lists should be used when there is no order of sequence or importance.

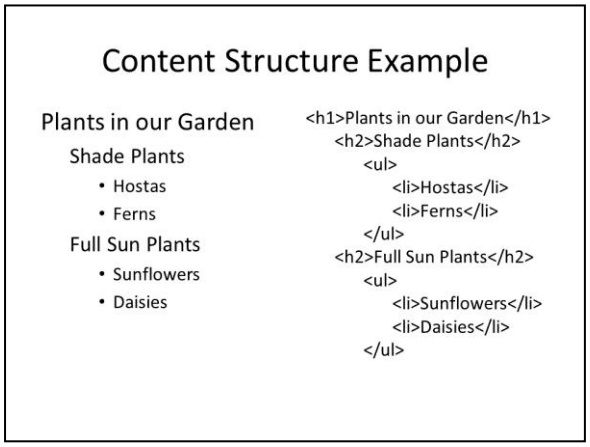
Ordered lists suggest a progression or sequence.

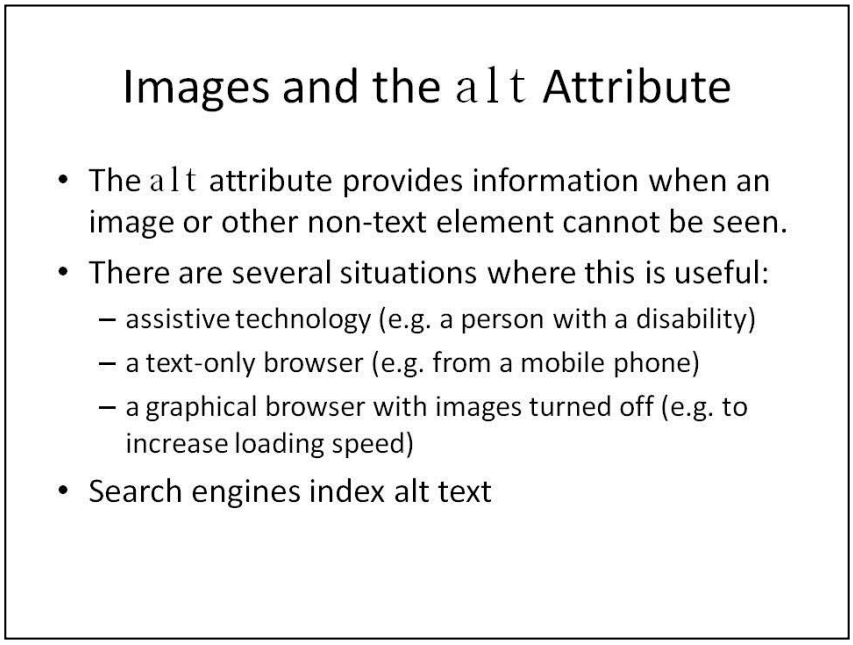
Definition lists should be used explicitly for presenting a structure for definitions.

As with heading, lists should be used correctly and for the right purposes. Unordered and ordered lists should always contain list items. Definition lists must always have definition descriptions.

Empty lists are incorrect HTML. Lists should never be used for merely indenting or other layout purposes. Nested lists should be coded properly.

Copyright © 1999-2010 WebAIM (Web Accessibility in Mind)





Images

Adding alternative text (alt text) to your web pages is one of the biggest improvements you can make for users. When an image or other non-text element cannot be seen, alt text provides information in place of the element. This is important when someone is using:

- assistive technology (e.g. a person with a disability)
- a text-only browser (e.g. from a mobile phone)
- a graphical browser with images turned off (e.g. to increase loading speed)

Search engines also index the alt text—helping the page to be found by potential visitors.

Note that non-text elements are used in several ways on Web pages, and including appropriate alt text—or no alt text—depends on the way the image is used. Let's look at 3 main ways images are used, and what alt text is appropriate for each.

3 Ways Images Are Used

For Information

If the purpose of the image is to communicate information, it needs appropriate alt text. Appropriate alt text expresses the *purpose* or *meaning* of the image, not its *appearance.*

If you're uncertain if the image is meant to convey information, try the following. Imagine you're reading the page out loud to someone over the phone. Is the information in the image important enough to mention? If so, the image is conveying information and needs alt text.

For Navigation

Images used to link to other pages (or other places within the page) require alt text. The alt text will almost always be a word-for-word repeat of the destination of the link (e.g. Home, Services, Contact Us).

Image maps need alt text for each of the hot spots and the main image. The main image might require a null alt attribute. The author of the page must make this decision based on the circumstances of the page.

For Decoration

These images are used for visual appeal, and do not provide any content. Such images require no alt text. Instead, they need to be given what is called an *empty* or *null* alt attribute, written as alt="". There is no space between the quotes.

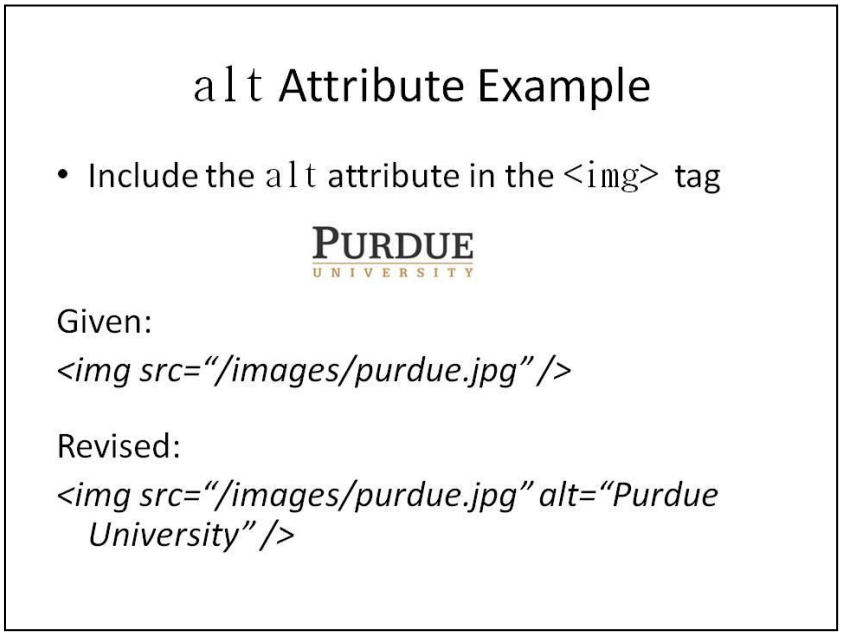
Why provide a null alt attribute? Why not leave out the alt attribute entirely? Because a missing alt attribute leads to more confusion, not less. When there is no alt attribute available, some assistive technology will read the file name of the image instead. Since the filename is rarely descriptive of the image, the individual is left to wonder whether the image is important for conveying information. With a null alt attribute, assistive technology skips over the image without reading anything at all. This is exactly what is desired for decorative images.

Remember that **every image/non-text element needs either alt text or a null alt attribute**.

Keep alt text as succinct as possible. While it needs to communicate the purpose of the image it's replacing, users should not be burdened with excessively long alt text. In addition, many browsers render alt text without word wrapping, so long alt text can result in a page that exceeds the width of displays for sighted users.

If an image requires a longer description, it is best to use one of several alternatives to alt text. It is outside the scope of this training workshop to describe those alternatives, but a web search of "providing long descriptions" should reveal available resources.

The content from here to the copyright notice was provided primarily by WebAIM. Their content is also available at: http://www.webaim.org/techniques/images/alt_text.php

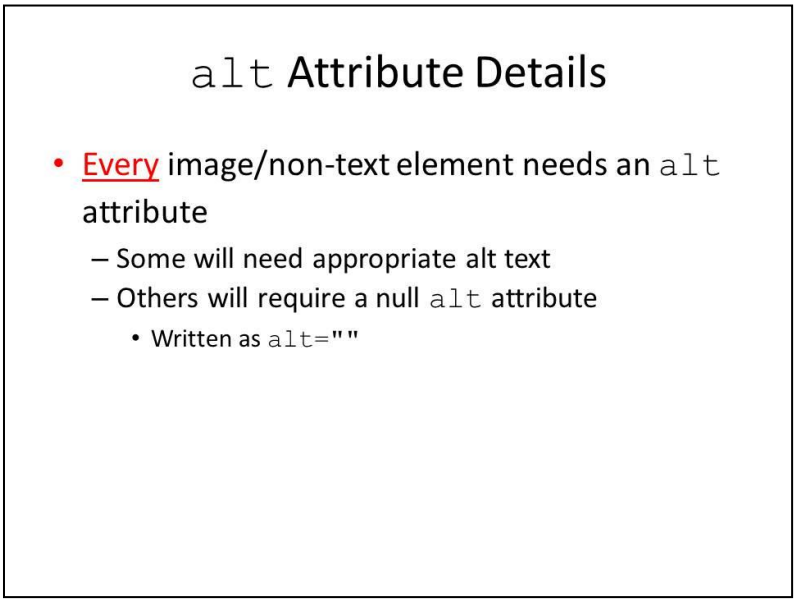


Overview

Most people are at least somewhat familiar with `alt` text. By the way, there is no such thing as an `alt` ***tag***, though people often refer to `alt` text by this name. To be technically correct, it is the `alt` attribute of the `<img>` tag. Its name is not as important as its function, though, so let's take a look at what it means to have effective `alt` text.

Guidelines for `alt` text:

1. Ensure that the text alternatives communicate the purpose of the graphic accurately and succinctly.
2. Provide empty or null `alt` text for graphics that do not convey content.
3. Provide `alt` text for both the main image and the hot spots of image maps.
4. Do not repeat the `alt` text of an image in the adjacent text.
5. Do not put important images in the background.



The Importance of Alternative Text

One of the biggest accessibility problems on the Web today is the widespread disuse of alternative text for graphics and images. Individuals who are blind often use screen readers or refreshable Braille devices that read the text on the page to them. When these assistive technologies come across images without `alt` text, they are unable to communicate their meaning.

When a screen reader comes across an image with no `alt` attribute, there are a couple of things that could happen:

1. It could simply skip the image as if it were not even on the page.
2. It could find some text that is associated with the image such as the file name and read that instead.

The exact behavior of the screen reader varies between brands of screen readers and the circumstances of the Web page itself, but either way, the end result is undesirable. The user either misses the image content completely or gets some text that is probably meaningless.

Perhaps you are interested in hearing what information a screen reader provides when an image has alt text and when it doesn't. If so, WebAIM has audio recordings available at: http://www.webaim.org/techniques/images/alt_text.php#importance

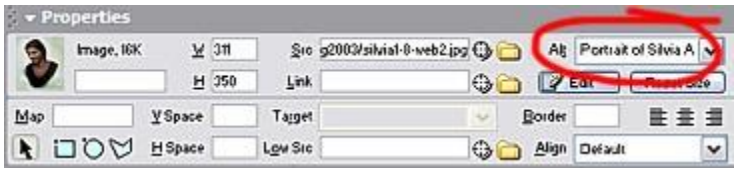
Adding `alt` Text

Let's look at another example image:

The HTML code for this image is as follows:

```
<img src="silvia1-8-web2.jpg" alt="Portrait of Silvia Alvarez" width="311"
height="350" />
```

You can type the code exactly as you see it above into a text editor, or you can use the interfaces of software tools such as Dreamweaver, FrontPage, or GoLive to accomplish the same thing. In Dreamweaver, `alt` text is added through the Properties window as pictured below:



Other editors have similar functions for adding alternative text. Consult your editor's documentation for instructions on how to add an `alt` attribute.

Now that we have a better idea of what an `alt` attribute is and how simple it is to add an `alt` attribute to an image, let's talk about what the `alt` attribute should contain.

Which Alt Text is Appropriate?

- Including appropriate alt text or no alt text depends on the way the image is used

- 3 Ways Images Are Used
  - Information
  - Navigation
  - Decoration

Tips on Communicating the Purpose of the Graphic

Images that contain important content

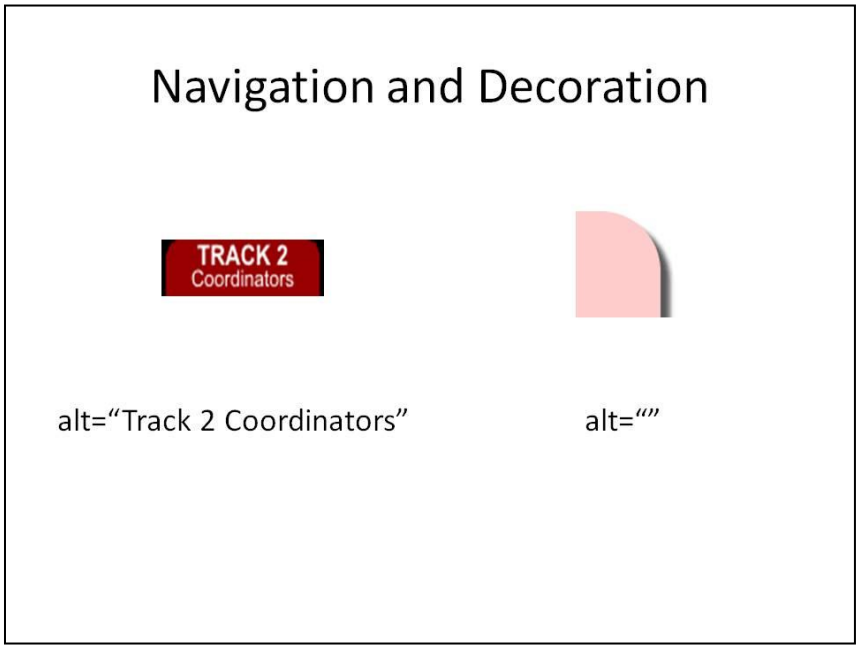
Example 1—Information

Take another look at the portrait of Silvia below:

This particular graphic could be used in many different ways, with many different purposes. Here are a few scenarios:

- An elementary school teacher creates a Web site to explain the difference between paintings, drawings, and sculpture. She includes several different examples of each type of art. In the text of the page itself, she describes the differences between these three media. She uses the portrait of Silvia as one of 4 examples of paintings. One possible `alt` text in this case would be **"A painting of a young lady."** This is probably sufficient, as long as the teacher has adequately described what a painting is within the document itself.
- A family member is compiling a list of people in her family, along with portraits of these individuals. Since all of the images are portraits, an appropriate `alt` text would be **"Silvia Alvarez."**
- An art instructor in a high school creates a Web site showing different types of paintings. He uses this painting as an example of a portrait, and explains within the text of the page what a portrait is. An appropriate `alt` text could be **"Portrait."**
- An art historian is creating a catalogue of different portrait artists. His purpose is to show portraits by various artists. The `alt` text could include information relevant to art historians, such as the title of the work of art, the name of the artist, the medium, and the date. The `alt` text could say **"Silvia Alvarez, oil on canvas, by Paul Bohman, 2002."**

We could go on with different scenarios, but I think you get the point. There is no one right `alt` text for any particular image. It all depends upon the context and the purpose of the image. This is a judgment call that the page's author must make.



Example 2—Navigation

The WebAIM Training CD site uses images for its main navigation, such as the one pictured below.

These images look like tabs on file folders. Some of these tabs are maroon, and others are blue. When the tab is selected, it turns white. Part of the text is in upper case; part is in lower case. All of these details are important to the look and feel of the Web site, but to someone who cannot see how the site looks, its look and feel is mostly irrelevant.

The important aspect of these graphics is that they link to other areas of the site. With that in mind, we would want to provide alternative text that conveys the fact that the user can click on this image to go to another area of the site. In this case, the link destination would be Track 2 of the training event, which is the track for coordinators. The most appropriate `alt` text for this image is as follows:

"Track 2: Coordinators."

In this case, the `alt` text exactly matches the text in the graphic. In most cases where there is text within images, this is the best solution. Don't worry about describing the image. Tell the user about the ***purpose*** of the image, not its appearance.

Accuracy and brevity

Alternative text for images should be as accurate and as succinct as possible. Make sure that your `alt` text conveys all of the important information relevant to its purpose, but don't burden users with excessively long `alt` text. Screen readers or refreshable Braille devices always read the `alt`

text, which can make image-heavy pages rather long. If you need a longer description of the image, you should add a `longdesc` tag to the image.

Null alt Attribute

- Why provide a null alt attribute? Why not leave out the alt attribute entirely?
- A null alt attribute causes AT to skip over the decorative image without reading anything.

Null `alt` text

Decorative images

The Web has become a graphical environment in which developers often add images to their pages simply to enhance the visual appeal of the site. For example, the image below could be used to form part of a rounded border on a page.

Images in this category do not provide any content to the user; they are simply used for decorative purposes. These images have no value to someone who cannot see the page. The proper HTML markup for this type of image is what is often referred to as an empty or null `alt` attribute, written as `alt=""`. That is `alt` equals quote quote, with no space in the middle. The source for the image in this example would look like this:

```
<img src="corner.gif" width="84" height="90" alt="" />
```

Screen readers will ignore graphics with empty `alt` text, which is exactly what we want in this case. You may be wondering why it is necessary to specify a null `alt` text. Wouldn't it make more sense to simply leave the `alt` attribute off entirely? This is a good question, but the answer is that missing `alt` text is worse than null `alt` text because some screen readers read the file name of the image, which can be confusing to listen to. When you add null `alt` text, screen readers skip over the image without reading anything at all.

Dreamweaver allows users to create null `alt` text within the Properties dialogue box.

Unfortunately, many other HTML editors do not allow you to create empty `alt` attributes within the graphical interface so you must edit your HTML source code directly. To do this, locate the image in the code and add `alt=""` to the `img` tag.

Transparent and spacer images

Developers often use transparent and spacer images to create space between elements on a page. Although users with sight do not see the transparent images, they may be visible to individuals using text browsers or screen readers. You should add an empty `alt` attribute (`alt=""`) to all transparent and spacer images.

Redundant images

Sometimes, Web developers add `alt` text to an image that is exactly the same as the text next to it, or the same as another graphic next to it, as in the example below:

In cases like this, you should add null `alt` text, so that screen reader users do not have to hear the same information twice. The JAWS screen reader would say: "image, international students; image, international students" when reading this section of Web content, which can be confusing, or at least annoying. (In this case, both the photograph of the girl and the adjacent text are images.)

Background Images

> **Important**
>
> It is impossible to add `alt` text to background images, so you should put images in the background only if they do not convey any important content.

If your background image contains important text or other visual cues, you should rewrite the HTML so that the image is in the foreground so that you can apply the proper `alt` text.

Copyright © 1999-2010 WebAIM (Web Accessibility in Mind)

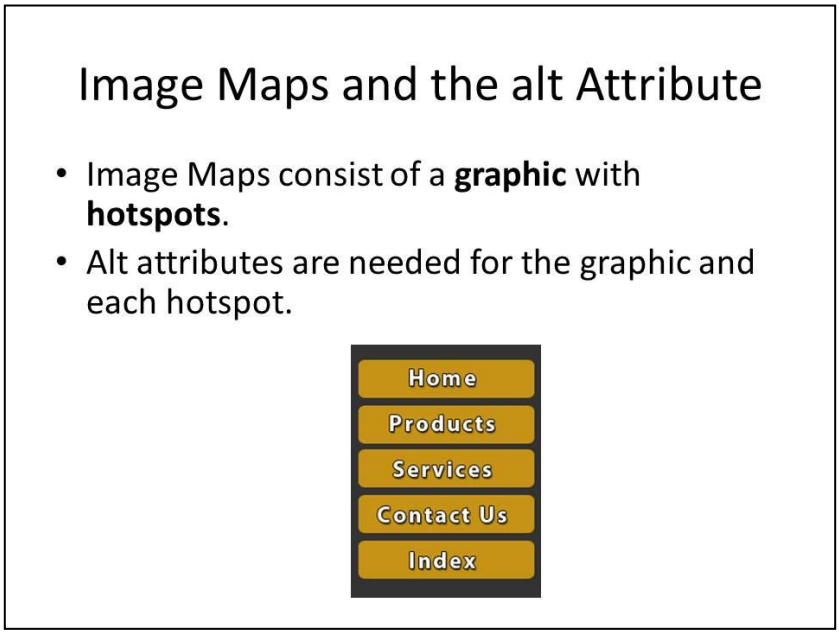


Image Maps

The content from here to the copyright notice was provided primarily by WebAIM. Their content is also available at: http://www.webaim.org/techniques/images/alt_text.php

Client-side image maps

Every common Web development tool creates client-side image maps, rather than server-side image maps. As the names suggest, server-side image maps require special scripting on the server, whereas client-side image maps are processed only in the client's browser. Unless you have purposely chosen to create a server-side image maps, you probably will never create one. Client-side image maps can be accessible, whereas server-side image maps cannot.

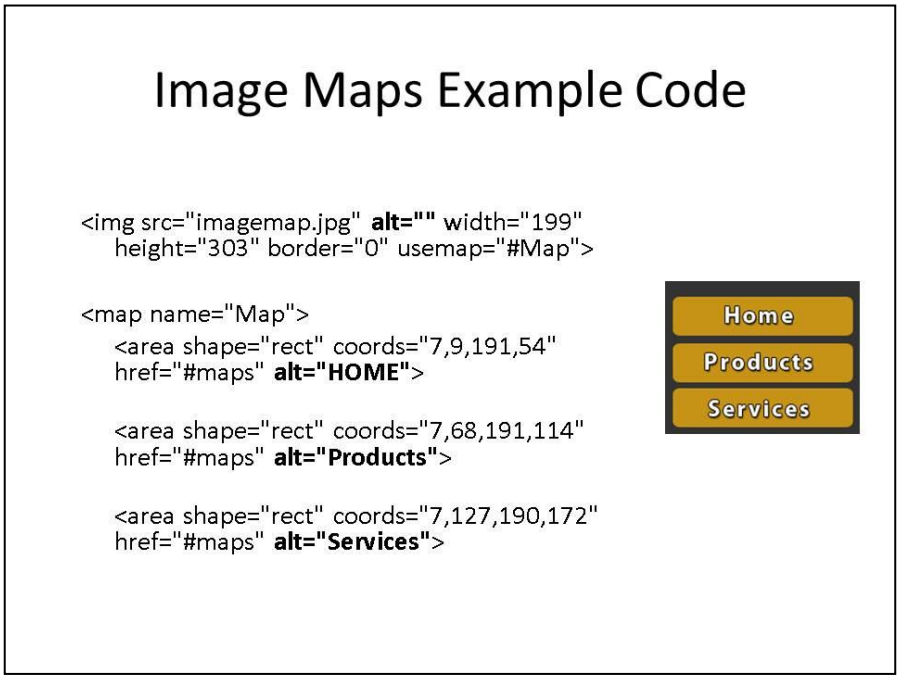
Client-side image maps require `alt` text for both the image and the hot spots. Take a look at this example.

There is only one image above, but there are 5 hot spots. Each of these hot spots leads to a different location in the Web site, so it is necessary to convey the navigational purposes of each of the links. The `alt` text for these hot spots should be exactly the same as the text in the image. The `alt` text for the hot spots is ***HOME***, ***Products***, ***Services***, ***Contact us***, and ***Index***.

We have the `alt` text for the hot spots, but what about the image itself? Aside from the hot spots, this image does not convey any meaningful information. The most appropriate `alt` text for the image is a null `alt` text. Here is the code for the image and its hot spots:

```
<img src="imagemap.jpg" alt="" width="199" height="303" border="0"
usemap="#Map"><map name="Map">
<area shape="rect" coords="7,9,191,54" href="#maps" alt="HOME">
<area shape="rect" coords="7,68,191,114" href="#maps" alt="Products">
<area shape="rect" coords="7,127,190,172" href="#maps" alt="Services">
<area shape="rect" coords="6,186,190,229" href="#maps" alt="Contact us">
<area shape="rect" coords="7,245,189,289" href="#maps" alt="Index">
</map>
```

Not all images used as image maps will have null `alt` text. The content author must determine the most appropriate `alt` text for the situation.



Important

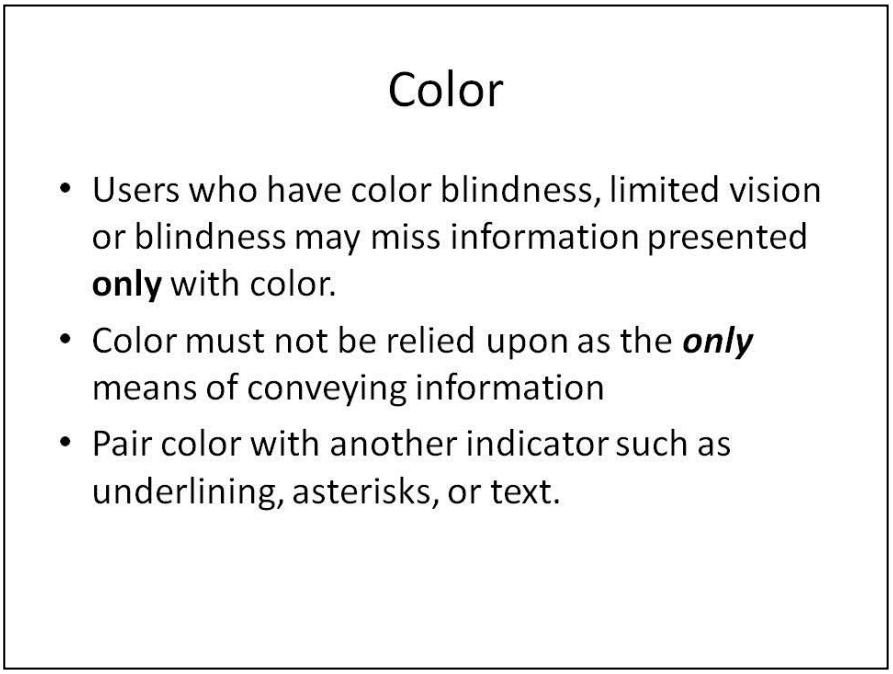
One thing to keep in mind when creating client-side image maps is that screen readers read the literal order of the HTML markup. Tools such as Dreamweaver often do not put the `<area>` tags next to the `<img>` tag. The `<area>` tag, which contains the `alt` text for the hot spots, might end up at some strange location far away from where the hot spot is viewed in the browser. This means that screen readers read the alt text completely out of context. This can be very confusing to screen reader users. If you use Dreamweaver or other similar WYSIWIG tools, be sure to look at the underlying HTML markup. If necessary, cut and paste the `<area>` tags to be right after the `<img>` tag.

Copyright © 1999-2010 WebAIM (Web Accessibility in Mind)

Server-side image maps

Server-side image maps should be avoided because their image can't be made accessible. Assistive technology is unable to access the hotspots. If a server-side map is the only option, then place text links—corresponding to hotspots in the server-side image map—near the image map or as part of the header or footer of the page.

In almost all cases a client-side image map, which can be created accessibly, should be used instead of a server-side image map. Even geometric or unusually shaped hot spots can be created with client-side image maps. See example at WebAIM: http://www.webaim.org/techniques/images/alt_text.php#maps



Color

Users who have color blindness, limited vision or blindness may miss information presented only with color. Because color can't be seen by all individuals, it must not be relied upon as the ***only*** means of conveying information.

Instead, whenever color is used as an indicator, pair it with another indicator such as underlining, asterisks, or text. For example, it is appropriate to use color to highlight links, but use underline and/or bold for links as well. Another example: if the required fields for a form are in red, make that information accessible by also marking required fields with an asterisk or with the word "required".

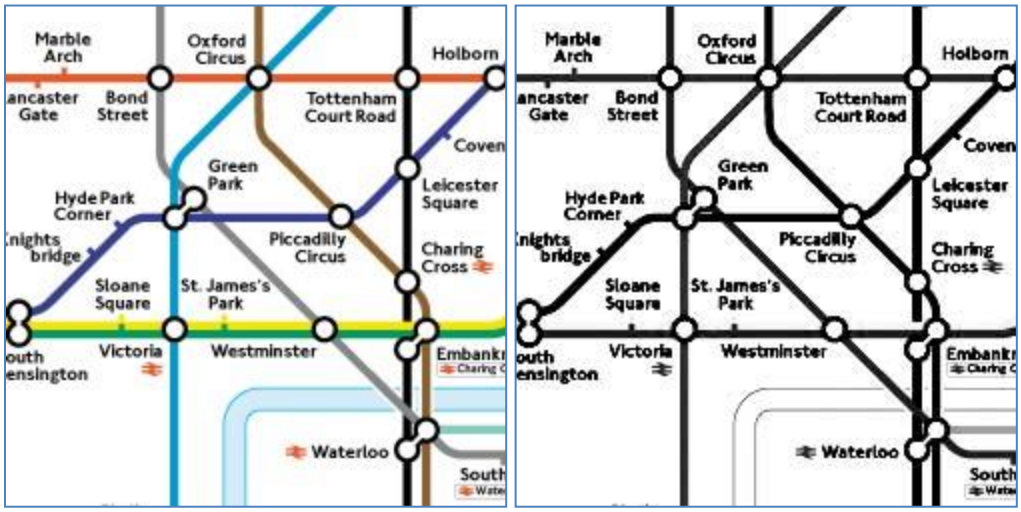
The content from here to the copyright notice was provided primarily by WebAIM. Their content is also available at:http://www.webaim.org/articles/visual/colorblind.php

Designing for Color-blindness

There is no need to convert all of your images to black and white or get rid of your images entirely. You may not have to change any of your images at all. Most of the time when people put images on the web, the fact that they are in color at all is irrelevant. It may be nicer to see the colors, but the viewer can understand the image just fine even with all of the colors removed.

The key to designing with color-blindness in mind is to make sure that colors are not your only method of conveying important information.

For example, if the image shows the routes of the London Underground, where the routes are distinguished only by the color of the lines, as in the graphic below, you would want to somehow annotate either the graphic itself (and supply the appropriate `alt` text) or the text in the web page to supplement the color-dependent method of distinguishing between routes.

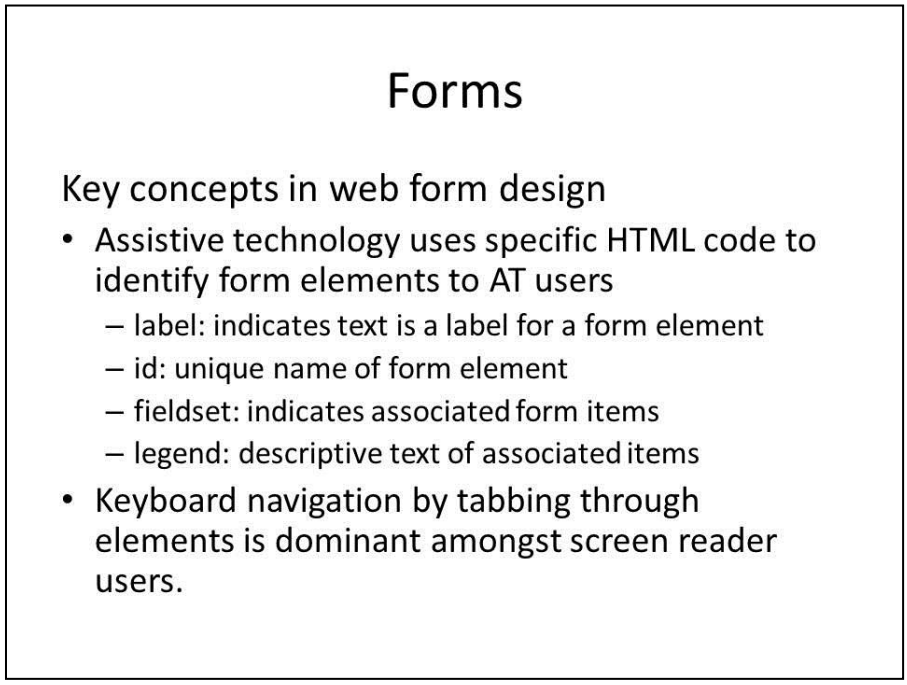


Incidentally, people with color-blindness are not the only ones who will benefit from this technique. Those who are blind are also unable to distinguish between colors, and so are in need of the extra cues given through other methods.

Key Concepts

Key Concepts for Color-blindness	
Challenges	**Solutions**
Reds and greens are often indistinguishable	This is not normally a problem except in cases where the colors convey important information. Under these circumstances you will need to either change the graphic or provide an additional means of obtaining the same information. Oftentimes the most appropriate way to do this is to provide an explanation in the text itself.
Other colors may be indistinguishable	Same as above.

Copyright © 1999-2010 WebAIM (Web Accessibility in Mind)



Forms

Web forms allow users to search for information, register for classes, take part in surveys, and complete a wide variety of other actions. For visual users, it's fairly easy to identify the text label and the form element (e.g. text box, radio button, drop-down menu) that go together.

For a blind person using a screen reader, the visual layout isn't helpful. Screen reading software needs to be told—through the coding of the Web page—that a text label is associated with a specific form element. Without this coding, the screen reader user can identify that there is a form element on the page, but can't tell what information should be entered into the form element.

Screen reader users typically will use only the keyboard to navigate Web pages, including forms. Therefore, design Web forms so they can be completed using only the keyboard.

For additional information on general form accessibility, see http://webaim.org/techniques/forms/.

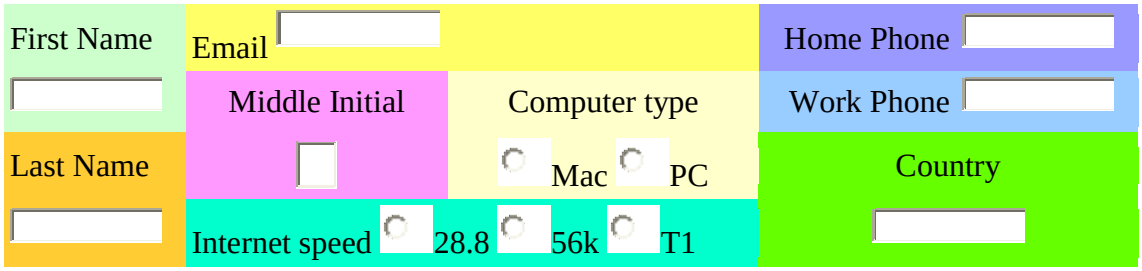
The content from here to the copyright notice was provided primarily by WebAIM. Their content is also available at:http://webaim.org/techniques/forms/screen_reader.php

Provide a Logical Form Layout

One of the most common ways of navigating through a form is using the **Tab** key. A person fills in one field, hits the **Tab** key, fills in the next field, and so on, until the end of the form is reached.

For visual users, this is an easy thing, because the text labels are placed in such a way that they appear to be linked to their corresponding controls. For a blind person using a screen reader, however, visual layout is not going to be of much help.

There has to be some way of linking the label with its control in the markup. The following example demonstrates the possible confusion that can occur with a form that is poorly marked-up and poorly organized.



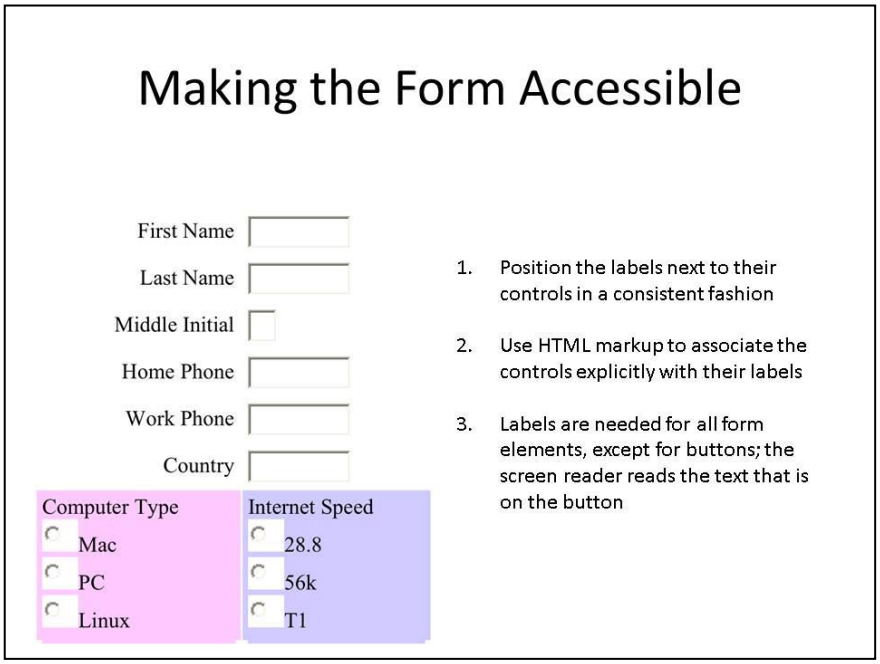
Visually, this table makes at least some sense (even if it is a bit confusing). For the most part, a person with sight can figure out that the *First Name* label goes with the control beneath it, and that the *Email* label goes with the control to the right of it. What happens when we remove the colors and strip it of its formatting, though?

A screen reader will read the above table in this order (color used for comparison purposes only):

1	2		3
4	5	6	7
8	9	10	11
12	13		14

If you compare the reading order to the visual order in the table above it, you can see that the labels and the form elements do not match up very well. The person using the screen reader is likely to become very confused, and may fill in the form incorrectly, typing information in the wrong places.

You may listen to what the above form sounds like when using a screen reader. WebAIM has audio recordings available at: http://webaim.org/techniques/forms/screen_reader.php#logical



Using Form Labels Appropriately

There are two things that we can do to the previous table to fix it:

1. Put the labels adjacent to their controls
2. Use HTML markup to associate the controls explicitly with their labels

One way to fix the accessibility problems of the above table would be to reduce the table to two columns, placing the labels directly to the left of the form elements, and then adding in the appropriate HTML elements. See the resulting HTML markup at http://webaim.org/techniques/forms/screen_reader#sample_code.

First Name
Last Name
Middle Initial
Home Phone
Work Phone
Country

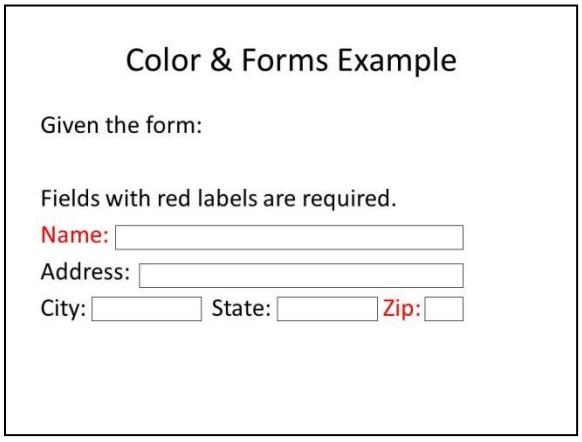
Computer Type	Internet Speed
Mac	28.8
PC	56k
Linux	T1

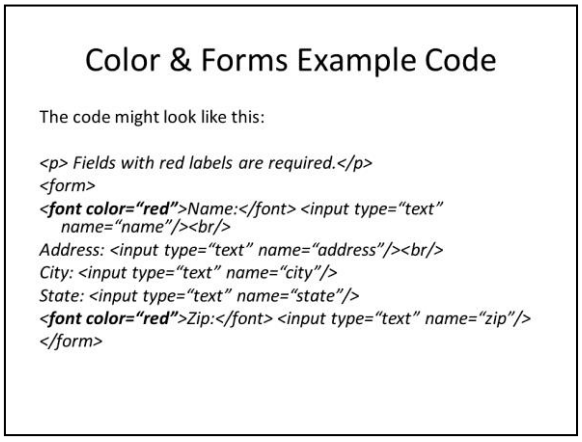
Boxes or Buttons?

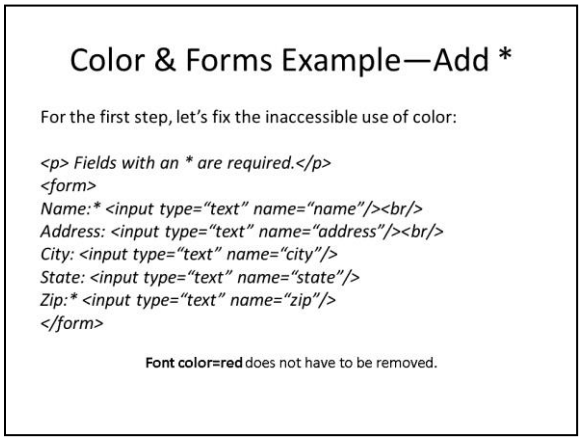
- Check boxes and radio buttons convey to the user how many choices can be selected.
  - Check boxes: many choices within the set
  - Radio buttons: one choice within the set

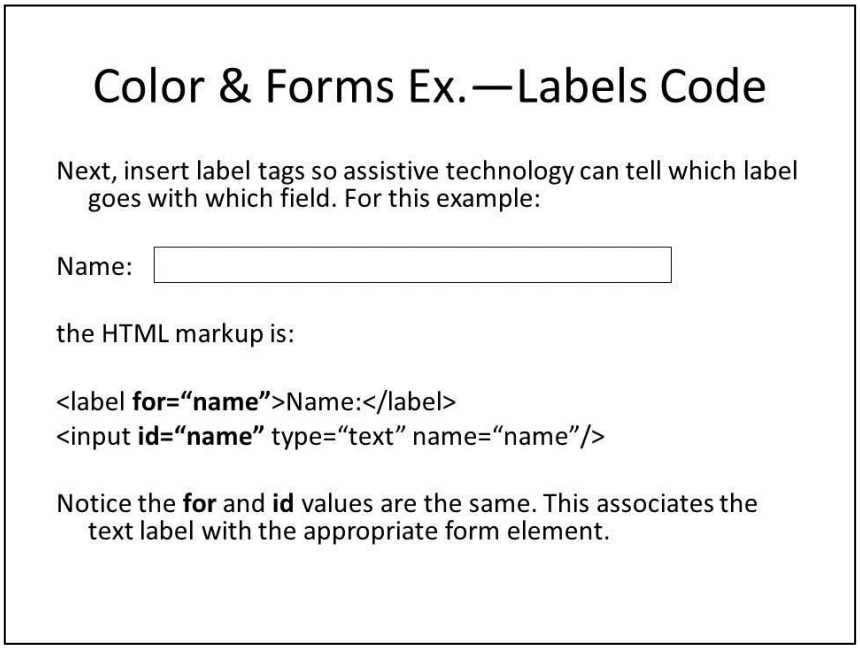
Check Boxes and Radio Buttons

Check boxes are similar to radio buttons, but they are not the same. Any number of check boxes in a fieldset—from none to all—can be selected. A group of radio buttons in a fieldset, by contrast, can have only ONE button selected.









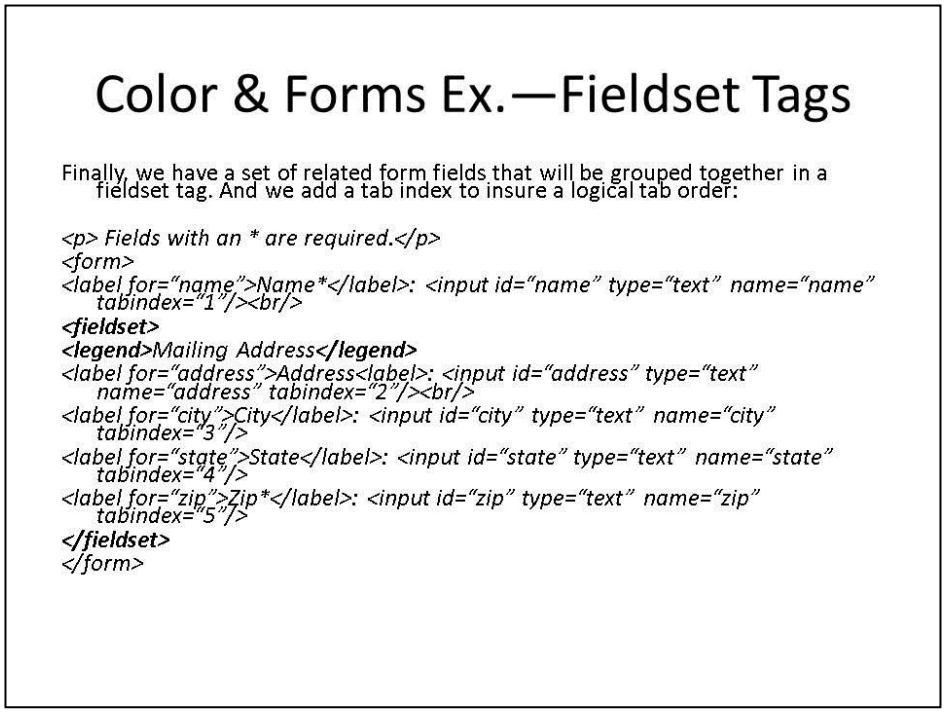
Create labels for form elements using the <label> element

When a screen reader user is navigating through form elements on a web page, the screen reader software will identify the type of form element that is currently selected and will provide them means to complete, select, de-select, or submit that form element.

However, when navigating through the form, there is often no indication as to what information is desired with the particular form item. For instance, if the user navigates to a text box using the Tab key, there may be no indication as to whether the text box is where they should submit their name, address, phone number, a message, or any number of other things.

This can be solved by associating form labels to form items on the page. The label should almost always be located adjacent to the form item itself. When a screen reader accesses a form item that has a `<label>` element associated with it, it will read the text within the `<label>` element and indicate the type of form item it is (e.g., "First Name. Text box." or "Age Range. Radio button.").

Labels are needed for all form elements, except for buttons; the screen reader reads the text that is on the button (e.g., "Submit button").



Group related form elements using the `<fieldset>` element

When you have several associated form elements, they can be grouped together by something called a `fieldset`. Each `fieldset` should have a legend. The `legend` is the text that describes the associated group of form items. In the above example, fieldsets are defined for *Computer Type* and *Internet Speed*.

The fieldset typically generates a visual border in the web browser that surrounds the contents of the fieldset. The fieldset, in this example, is defined by the border that surrounds the three checkboxes and their adjacent labels.

In the above example, the legends of these fieldsets are the words *Computer Type* and *Internet Speed*. The label for the first checkbox in the Internet Speed fieldset is *28.8*. The fieldset is important, because without it, if the screen reader were to access the checkbox, it would read something like, "Twenty eight point eight. Not checked". While this is valuable, it gives no description of what the data is for. Is 28.8 referring to Internet speed, age, height, IQ, or something else? With the fieldset in place, the screen reader would read something like, "Internet speed: Twenty eight point eight. Not checked." Fieldsets should be used when there are groups of check boxes or radio buttons. They can also be used for other form items that are associated.

Even if the screen reader is incapable of reading the `label`, `fieldset`, and `legend` elements that were added above (as is the case with some older screen readers), the improved table above is organized in such a way as to make it possible to figure out which labels go with which controls.

You may listen to what the form sounds like now when using a screen reader, with the improvements that we made. WebAIM has audio recordings available at: http://webaim.org/techniques/forms/screen_reader.php#group

Copyright © 1999-2010 WebAIM (Web Accessibility in Mind)

Exercise

Original code:

```
<tr>
<td>
<font color="#FF0000">Name</font><br />
<input type="text" name="Name" id="Name" size="25"/>
</td>
</tr>
<tr>
<td>
<font color="#FF0000">E-mail Address</font><br />
<input type="text" name="Email" id="Email" size="25"/>
</td>
</tr>
<tr>
<td>
<font color="#FF0000">Street Address</font><br />
<input type="text" name="Address" id="Address" size="25"/>
</td>
</tr>
<tr>
<td>
<font color="#FF0000">City</font><br />
<input type="text" name="City" id="City" size="25"/>
</td>
</tr>
<tr>
<td>
<font color="#FF0000">State</font><br />
<input type="text" name="State" id="State" size="25"/>
</td>
</tr>
<tr>
<td>
<font color="#FF0000">Zip</font><br />
<input type="text" name="Zip" id="Zip" size="10"/>
</td>
</tr>
```

Accessible code:

1. Add `label` tags around the text that indicates what data should be entered into the form field next to it.

   <label>Name</label>

2. Add the `for` attribute to the label tags and type the `id` of the form field it labels.

   <label for="Name">Name</label>

3. Add the `<fieldset>` element and `legend` to group the related Mailing Address form elements.

```
<fieldset>
    <legend>Mailing Address</legend><br />
    <label for="StreetAddress">Street Address </label><br />
    <input type="text" name="addressline1" id="StreetAddress" size="25"/>
    <label for="City">City </label><br />
    <input type="text" name="addressline2" id="City" size="25"/>
    <label for="State">State </label><br />
    <input type="text" name="addressline3" id="State" size="25"/>
    <label for="Zip">Zip Code </label><br />
    <input type="text" name="addressline4" id="Zip" size="10"/>
</fieldset>
```

Data Tables

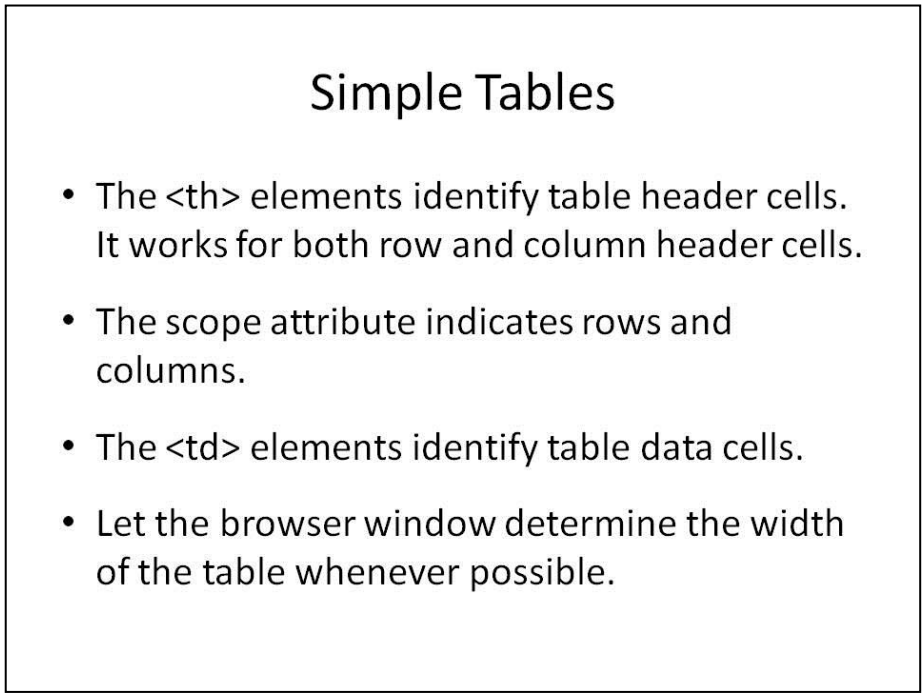
Most sighted people are familiar with data tables, and can easily extract information from a simple table such as the following. To identify the monthly cost of Carole's cable service, a sighted individual locates a specific cell by looking at row and column headings and using them as reference points.[1]

Monthly Cost for Utilities

Name	Phone	Cable	Water	Internet
Alicia	$10	$20	$30	$40
Bob	$20	$30	$40	$10
Carole	$30	$40	$10	$20
Don	$40	$10	$20	$30

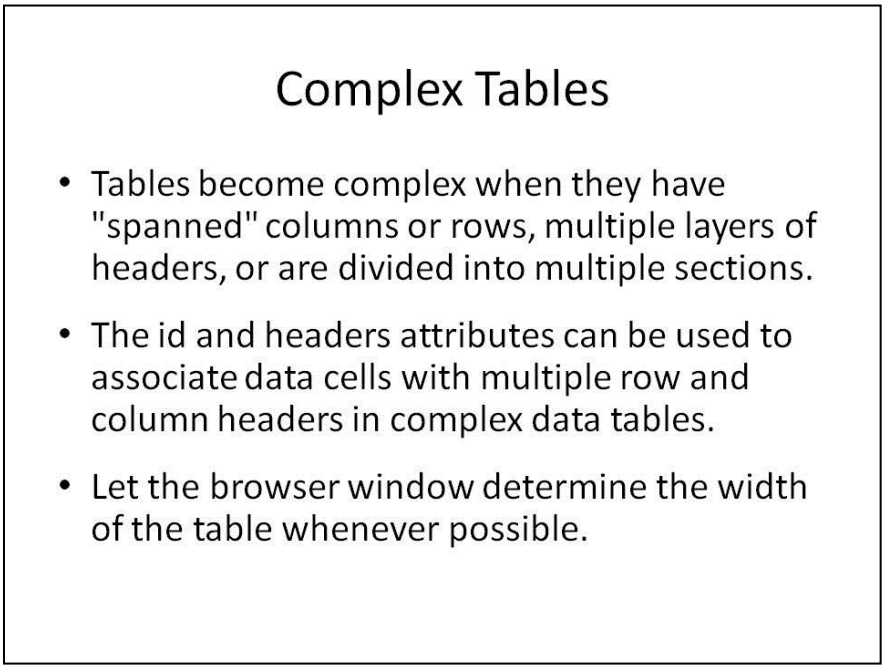
Some computer users are not able to visually access the cell contents and the headers at the same time. This includes users who need to magnify the web page in order to view it, or individuals who need to use screen reader software to access the information in the data table.

These users depend on computer software to assist them in determining the row and column header information. This explains the importance of using specific coding that explicitly defines row and column headers for each cell. The software makes use of that coding to convey row and column information while the user explores data cells within a table. Without such coding, the software may not accurately guess the header information since the placement and style of row and column headers do not follow specific, defined rules.



Identify a header cell for each column and row in simple data tables

A simple table can be defined as having one row of headings or one column of headings, or both.[2] Use the `<th>` element to identify table header cells. It works for both row and column header cells.



Identify relationships in complex data tables using `id` and `headers` attributes

Tables become complex when they have "spanned" columns or rows, multiple layers of headers, or are divided into multiple sections. The `id` and `headers` attributes can be used to associate data cells with multiple row and column headers in complex data tables.[3] Such tables are not common on the Web, so it is unlikely that you will work with tables that are this complex.

The content from here to the copyright notice was provided primarily by WebAIM. Their content is also available at: http://www.webaim.org/techniques/tables/data.php

Marking Up Data Tables

Data tables are different from layout tables. The purpose of data tables is to present information in a grid, or matrix, and to have column or rows that show the meaning of the information in the grid. When screen readers read straight through data tables—especially large ones—it's easy for users to get lost.

In order for a data table to be accessible, it must have the proper markup in the HTML. When the proper HTML markup is in place, users of screen readers can navigate through data tables one cell at a time, and they will hear the column and row headers spoken to them.

Designate Row and Column Headers Using the `<th>` Tag

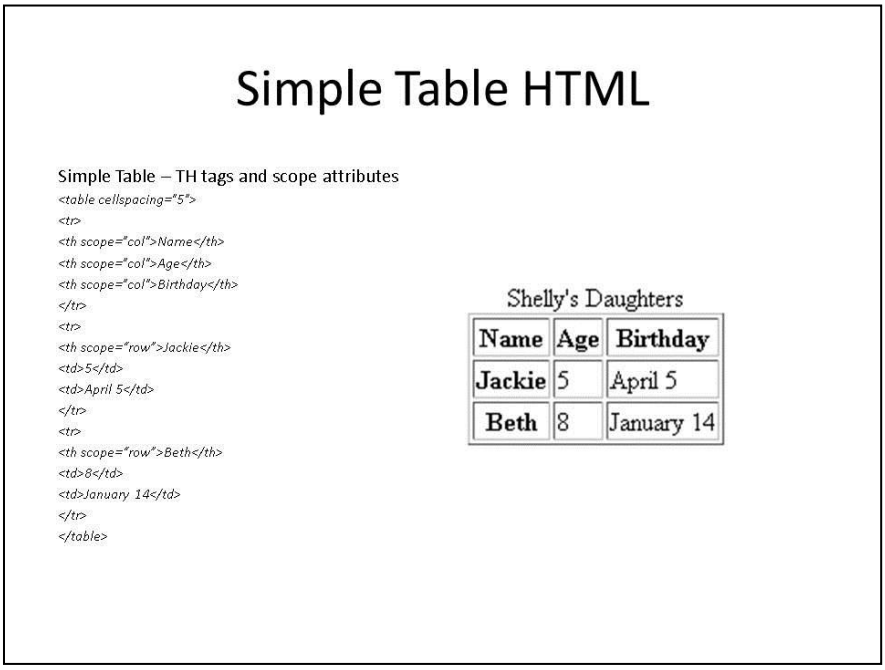
The very first step toward creating an accessible data table is to designate row and/or column headers. Most authoring tools provide a method of changing data cells into header cells.

In the markup, the `<td>` tag is used for table data cells and the `<th>` tag is used for table header cells. For example, the column headers for the table below are *Name*, *Age*, and *Birthday*. The row headers are *Jackie* and *Beth*.

Shelly's Daughters		
Name	**Age**	**Birthday**
Jackie	5	April 5
Beth	8	January 14

Associate the Data Cells with the Appropriate Headers

Now that we've created headers, we need to associate the cells with the appropriate headers. There are two ways to associate data cells with their headers: 1) scope attribute and 2) headers and id attributes.



The `scope` attribute

The scope attribute should be used on simple data tables such as the one in this example. Here is the markup for the table, using the `scope` attribute:

```
<table border="1" align="center">
<caption>Shelly's Daughters</caption>

<tr>
<th scope="col">Name</th>
<th scope="col">Age</th>
<th scope="col">Birthday</th>
</tr>

<tr>
<th scope="row">Jackie</th>
<td>5</td>
<td>April 5</td>
</tr>

<tr>
<th scope="row">Beth</th>
<td>8</td>
<td>January 14</td>
</tr>

</table>
```

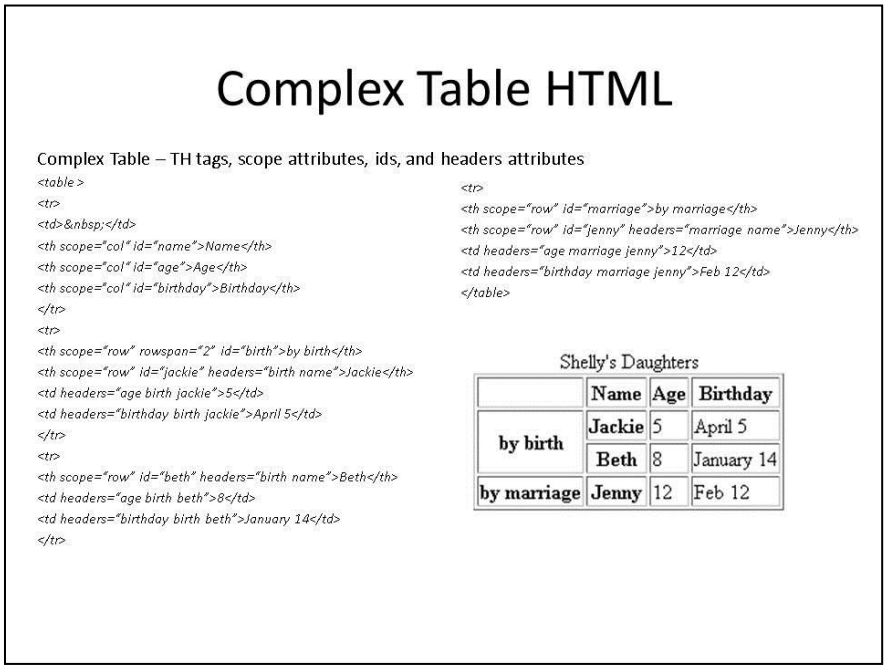
The `scope` attribute tells the browser and screen reader that everything under the column is related to the header at the top, and everything to the right of the row header is related to that header. It's a straightforward concept.

The `headers` and `id` attributes

Another way to accomplish the same purpose is to use the `headers` and `id` attributes. The `headers` and `id` method should only be used when there is more than one logical level in a table, and when it is necessary to link more than two headers with a data cell. **This method is NOT recommended for simple tables such as the first example.**

If we extend our original example, we can create a table that fits this criterion. In the table below, data have three headers each, so it is appropriate to use a more complex technique.

Shelly's Daughters			
	Name	**Age**	**Birthday**
by birth	**Jackie**	5	April 5
	Beth	8	January 14
by marriage	**Jenny**	12	Feb 12



The markup looks like this:

```
<table border="1">
<caption>Shelly's Daughters</caption>

<tr>
<td> </td>
<th id="name">Name</th>
<th id="age">Age</th>
<th id="birthday">Birthday</th>
</tr>

<tr>
<th rowspan="2" id="birth">by birth</th>
<th id="jackie">Jackie</th>
<td headers="birth jackie age">5</td>
<td headers="birth jackie birthday">April 5</td>
</tr>

<tr>
<th id="beth">Beth</th>
<td headers="birth beth age">8</td>
<td headers="birth beth birthday">January 14</td>
</tr>

<tr>
<th id="step">by marriage</th>
<th id="jenny">Jenny</th>
<td headers ="step jenny age">12</td>
<td headers="step jenny birthday">Feb 12</td>
</tr>
</table>
```

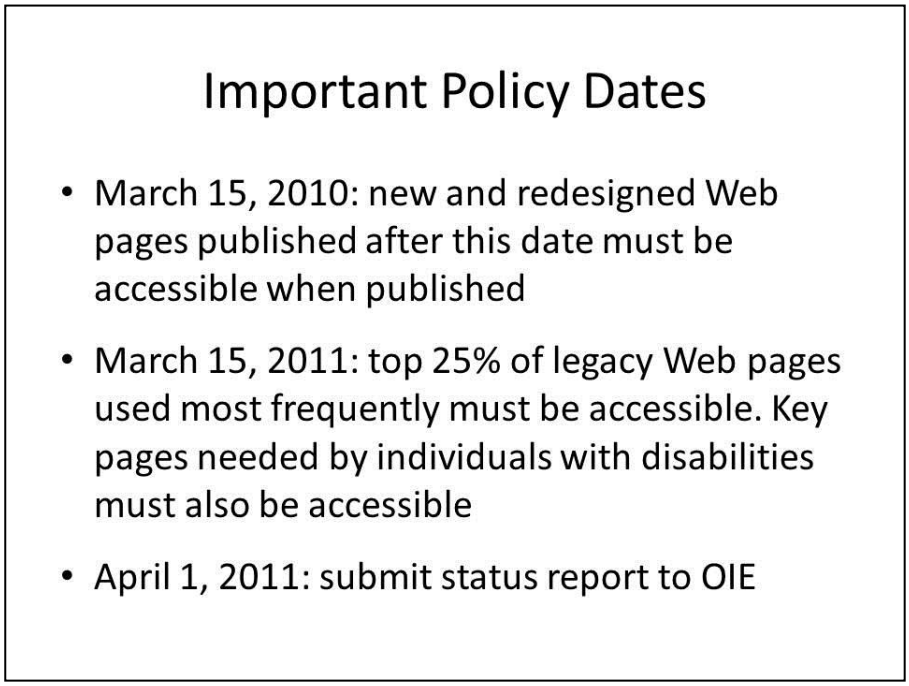
Again, it should be emphasized that this method is more complex. It should be used with tables of a more complex nature, where the `scope` attribute will not work.

Another caveat: spanned rows and columns are not handled well by the JAWS screen reader, which is the most popular brand of screen reader. If at all possible, avoid complex data tables, or represent the data in a way that is less complex, preferably with no more than two headings applying to a single data cell.

Use Proportional Sizing, Rather than Absolute Sizing

The rule that applies to layout tables also applies to data tables. Let the browser window determine the width of the table whenever possible, to reduce the horizontal scrolling required of those with low vision.

Copyright © 1999-2010 WebAIM (Web Accessibility in Mind)



Campus Resources

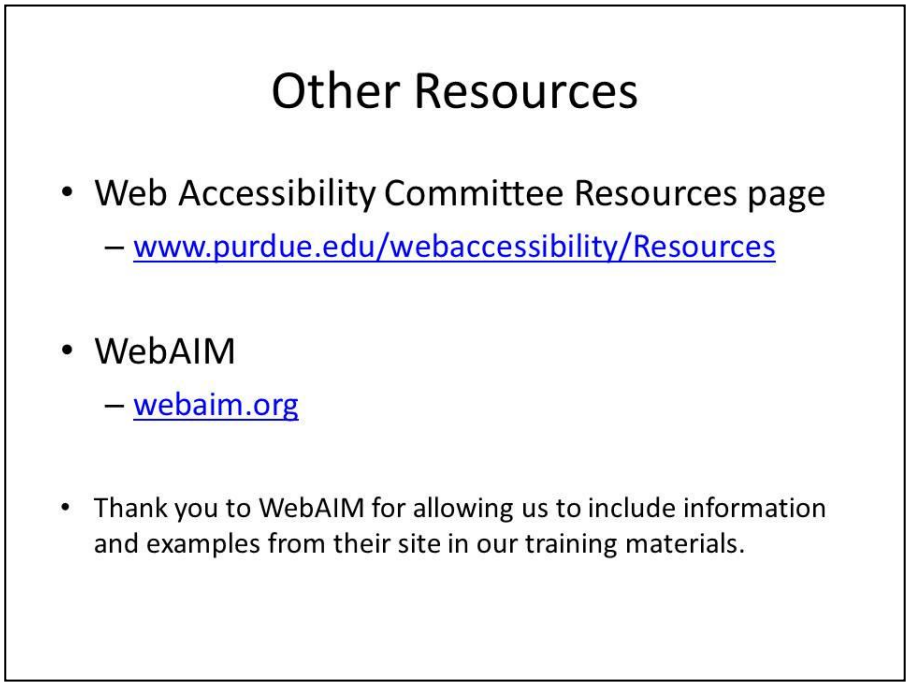
- Questions re: policy, deadlines and reporting

  Office of Institutional Equity
  http://www.purdue.edu/ethics/oie
  494-7253 (voice), 496-1343 (TTY)
  equity@purdue.edu

Campus Resources (cont.)

- Questions re: accessible web design

  Web Accessibility Committee
  http://www.purdue.edu/webaccessibility
  Contacts page: submit questions to committee
  Training page: self guided training available



Appendix

Illinois Information Technology Accessibility Act Implementation Guidelines:
http://www.dhs.state.il.us/IITAA/IITAAWebImplementationGuidelines.html

Further reading regarding Color:

- Horton, Sarah, Designing accessible text—Part 3: Color, Web Design Reference Guide, InformIT
- The Ohio State University, Standard 3 -- Color, Minimum Web Accessibility Standards

Further reading regarding Data Tables:

1. Hudson, Roger, Accessible Data Tables, Accessibility & Usability Services, Webusability
2. University of Wisconsin-Madison, Data and Layout Tables, Web Accessibility 101: Policy, Standards, and Design Techniques
3. Illinois Information Technology Accessibility Act Implementation Guidelines for Web-Based Information and Applications 1.0, Tables