Accessibility in e-Learning

Accessible Content Authoring Practices

JUNE 2014

Contents

Introduction

There are a number of strategies that content authors can use to help ensure that the web content they create is accessible, and these strategies are described in detail in this paper. To better understand these strategies, it would be helpful to have a basic knowledge of HTML.

In the HTML lexicon, you will often hear the words “**element**” and “**attribute**.” An element is often referred to as an HTML tag, and an attribute is a property of a tag. The example that follows illustrates an image tag, represented by the <img> element; it has two properties: the “src” attribute, which contains the path to an image file or “source” file; and the “alt” attribute, which contains the text alternative that describes the image.

```
// HTML elements are often referred to as tags
// The <img> element below has a “src” attribute containing the path to the image file
// and an “alt” attribute containing the text alternative describing the image

<img src="images/big_cat.jpg" alt="A big black cat" />
```

Visual Content

Content authors must ensure that their e-learning content is presented in multiple formats and, therefore, accessible through multiple senses. For people who are blind, visual materials require an alternative, typically in the form of text. While an audio version of visual content (such as an MP3 audio file that describes a complex image) could be used as an alternative, chances are the audio will interfere with the screen reader output. Text alternatives, on the other hand, are easily adapted or even simplified, read aloud by a screen reader, turned into Braille to be output through a Braille display, or translated into other languages.

Alt: The most common form of text alternative is the Alt attribute, which is included with the HTML for an image. The Alt attribute should describe the meaningful information that might be understood by a person viewing the image, or if the image is being used as a link or button, it should describe its destination or function. Alt should also be added to the Area element when creating image maps, which are images with clickable areas that lead to other web content. The four examples that follow illustrate how Alt is used in HTML.

Images and Alt

```
// Using alt with meaningful images
// Here alt is used to provide a meaningful
// description of the content in a photograph

<img src="images/black_cat.jpg" alt="A big black cat" height="244" width="325" />
```

Image Maps and Alt

```
// Using alt with map areas
// Here each area in the image map has a corresponding alt attribute

<map id ="planetmap" name="planetmap">
 <area shape ="rect" coords ="0,0,82,126" href ="sun.htm" alt="Sun" />
 <area shape ="circle" coords ="90,58,3" href ="mercur.htm" alt="Mercury" />
 <area shape ="circle" coords ="124,58,8" href ="venus.htm" alt="Venus" />
</map>
```

Meaningless Images and Alt

```
// Using alt with meaningless images
// Here a decorative image includes an empty alt attribute
// to force assistive technologies to ignore it

<img src="images/corner_decoration.jpg" alt="" />
```

Images Used as Buttons

```
// Using alt with images used as buttons
// Here type="image" is used to create a login button
// so the image requires an alt attribute

<form name="myform" action="http://www.mydomain.com/myformhandler.cgi"
method="post">
<input type="text" size="25" value="Enter your name here!" /></br />
<input type="image" src="login.png" name="image" width="60" height="60" alt="Login Now!"
/><br/>
</form>
```

Content Structure

If you incorporate structural HTML into your e-learning content, your content becomes more meaningful to learners who use assistive technologies (ATs).

Headings Nested in Proper Sequence

Structural HTML describes relationships between topics and sub-topics – using **HTML headings** (H1, H2, H3, etc.) – that a sighted reader would likely understand through a number of visual presentations (such as heading size or indentation). It also provides AT users with a means to navigate each web page. Most assistive technologies can provide users with a listing of a page's headings for a summary of its content, but they also enable users to jump directly to a heading, skipping over the other content on the page.

Heading markup should never be used to create visual formatting, such as large, bold text. Headings should also be nested in proper sequence, never skipping heading levels when arranging sub-topics. For example, you can arrange headings in the order H1, H2, H3, H2, H3, H4 H2, H3, but not H1, H3, H4, H2, H4 where subtopic heading levels have been skipped. In the example that follows, the heading levels are indented to make it easier to understand the structure of topics.

```
// Order headings to reflect relationships
// Don't skip heading levels when presenting subtopics
// Don't use heading markup for visual presentation

<h1>Our Solar System</h1>
 <h2>The Sun/h2>
 <h2>Inner Planets</h2>
   <h3>Mercury</h3>
   <h3>Venus</h3>
   <h3>Earth</h3>
   <h3>Mars</h3>
 <h2>Outer Planets</h2>
   <h3>Jupiter</h3>
   <h3>Saturn</h3>
   <h3>Uranus</h3>
   <h3>Neptune</h3>
```

Table Headers Added to Data Tables

Structural HTML can also be added to data tables. When using a screen reader to navigate through a table, the learner would typically use the arrow keys to move from cell to cell. By the time s/he would get to, say, the fourth row, fifth column, it would be challenging for the individual to keep in mind the text descriptions at the top of each column, her/his position within the column, let alone understand what the data in that particular cell represents.

When tables are created with the **<TH> table header** element around the text descriptions in the first row, and sometimes in the first column, these descriptions are read along with the content of the data cells (<TD>). For the learner who uses assistive technologies, this removes the need to keep track of her/his location within the table.

Consider the data table that follows. The content of the first row and the first column have been enclosed in the TH table header element. Using a screen reader for navigation, when an AT user enters the table's data area, s/he would hear the content of the two headers intersecting at each data cell as "John, January, 20," followed by "John, February, 23," and so on. If there are no header elements, the screen reader would simply announce "20," followed by "23," then "18," and so on for each cell, making it difficult for the AT user to understand what the numbers represent.

Percentage of Monthly Savings by Leaving the Car at Home

Name	**January**	**February**	**March**	**April**	**May**
John	20	23	18	5	25
Mary	21	14	24	24	17
Diane	22	23	24	25	26
Mark	29	25	27	22	28

What follows is the HTML markup that illustrates how the header rows and columns of the preceding data table were created. Also notice the Caption and Summary elements, which describe the table and summarize its data, respectively.

```
// Use TH for table header rows and columns
// so row and column descriptions are read
// along with the content of each data cell.

<table summary="Percentage of travel costs saved by employees taking public transit"
border="1">
<caption>Percentage of Monthly Savings by Leaving the Car at Home</caption>
<tr>
 <th>Name</th>
 <th>January</th>
 <th>February</th>
 <th>March</th>
 <th>April</th>
 <th>May</th>
</tr>
<tr>
 <th>John</th>
 <td>20</td>
 <td>23</td>
 <td>18</td>
 <td>5</td>
 <td>25</td>
</tr>
<tr>
 <th>Mary</th>
 <td>21</td>
 <td>14</td>
 <td>24</td>
 <td>24</td>
 <td>17</td>
</tr>
<tr>
```

```
 <th>Diane</th>
 <td>22</td>
 <td>23</td>
 <td>24</td>
 <td>25</td>
 <td>26</td>
</tr>
<tr>
 <th>Mark</th>
 <td>29</td>
 <td>25</td>
 <td>27</td>
 <td>22</td>
 <td>28</td>
</tr>
</table>
```

Lists

Another useful structural HTML element is a **list**. Lists can be an ordered list (<ol>), with numbers for each list item; an unordered list (<ul>), with bullets for each list item; or a definition or description list (<dl>), where a term (<dt>) is followed by a description (<dd>).

Each list type has semantic meaning for AT users. An ordered list is used where the order in which list items appear is important; an unordered list is used where order is not important; and a description list is used where the first item in a list pair (<dt><dd>) is associated with the second. When AT users encounter a list, they will hear the list identified as a list, the number of its items, and the position of each item in the list as they navigate through it. This added information makes navigation through lists easier and more productive for AT users.

Ordered List

```
// Ordered lists should be used
// when the order of items is important
<ol>
<li>Mercury</li>
<li>Venus</li>
<li>Earth</li>
<li>Mars</li>
</ol>
```

1. Mercury
2. Venus
3. Earth
4. Mars

Unordered List

// Unordered lists should be used
// when the order of items is NOT important
`<ul>`
`<li>Apples</li>`
`<li>Oranges</li>`
`<li>Bananas</li>`
`<li>Pears</li>`
`</ul>`
	● Apples
	● Oranges
	● Bananas
	● Pears

Description or Definition List

```
// Description lists should be used
// with related item pairs
<dl>
<dt>Apples</dt><dd>red, yellow, or green hard fruit</dd>
<dt>Oranges</dt><dd>juicy citrus orange coloured fruit</dd>
<dt>Bananas</dt><dd>elongated, slightly curved, soft yellow colored fruit</dd>
<dt>Pears</dt><dd>soft or hard, various coloured, broad bottom fruit.</dd>
</dl>
```

```
Apples
 red, yellow, or green hard fruit
Oranges
 juicy citrus orange coloured fruit
Bananas
 elongated, slightly curved, soft yellow coloured fruit
Pears
 soft or hard, various coloured, broad bottom fruit.
```

Keyboard Access

An important aspect of web accessibility is the ability to navigate content with both a mouse and a keyboard. This is often referred to as “device independence.” Most learners who are blind will use a keyboard to access web content, not a mouse – a fact that web developers often overlook. Web content authors must use both mouse and keyboard events when creating scripting that adds interactivity to e-leaning content. They must also ensure that changes made through cascading style sheets (CSS) that occur with a mouse also occur with a keyboard. For instance, if a mouse pointer is held over a link to change the link’s colour, the same colour change should occur when the Tab key is used to navigate focus to the link.

Javascript Events

If you use Javascript in your web content, either use device independent event handlers, or both mouse and keyboard event handlers together.

Mouse Events	**Keyboard Events**	**Device Independent Events**
// don’t use these on their	// don’t use these on their	// These can be used on their
// own, use along with a	// own, use along with a	// own, but be careful not to
// keyboard event	// mouse event	// change context
onclick	onkeydown	onchange
ondblclick	onkeypress	onfocus
onmouseover	onkeyup	onblur
onmouseout		

CSS :hover and :focus Styles

Styles for mouse and keyboard access are created by using both the :hover and :focus styles together. The :hover style activates the changes defined for a style when a mouse pointer hovers over an element; the :focus style activates the changes when the element receives keyboard focus. The example that follows shows a link that changes colour when a mouse pointer is placed over it, and when the Tab key is used to bring keyboard focus to the link.

```
// Some simple HTML with a link
// that has been styled to look
// like a button.

<html>
<body>
<a href="" class="button-link">A Button Link</a>
</body>
</html>

// This is the accompanying
// CSS to style the link to
// look like a button. Notice the
// :hover and :focus styles that
// change the background colour to
// blue with a mouse and with a keyboard

<style type="text/css">
.button-link{
 border:1px solid green;
 background-color: red;
 padding:1em;
}
.button-link:hover, .button-link:focus{
 background-color: blue;
 color:white;
}
</style>
```

Forms

When using forms in e-learning, it is important to ensure that the form input elements, such as text fields, radio buttons, and checkboxes, are properly labelled for users of assistive technologies. In many cases, simply locating the text to describe a form element immediately next to the input field will serve this purpose for AT users; however, there is no guarantee that the text will remain immediately adjacent. For example, if the screen resolution changes or the content is viewed on a mobile device, the label may shift position and, as a result, become disconnected from its form field.

To avoid this potential problem, use the HTML <label> element to explicitly associate the text description with a specific form element. Regardless of where the text label may be located on the screen, its association will ensure that the description is always announced, and that forms are described effectively for AT users. The <label> element also makes the text of the label clickable to activate a form element, and provides a larger target area for users who might have difficulty positioning a mouse pointer over a tiny form element such as a radio button or checkbox.

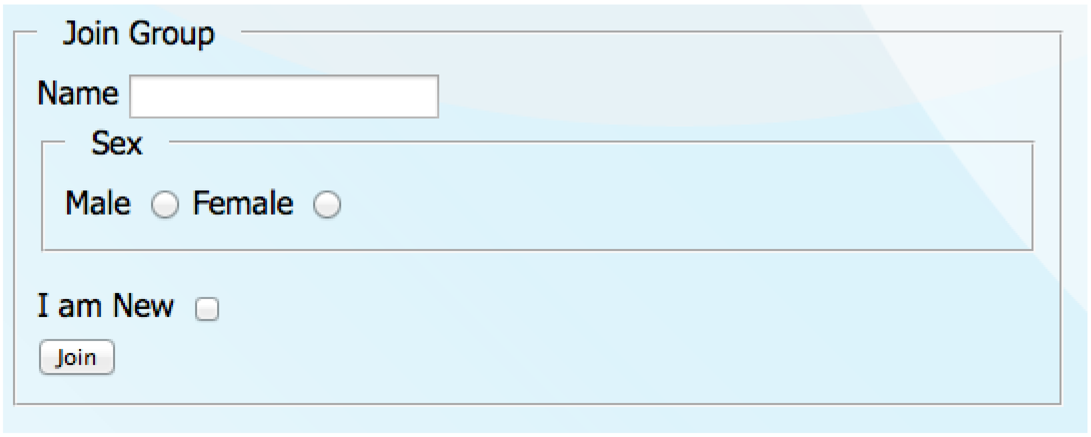
Describing Forms with Label, Fieldset, and Legend

The example that follows shows how the HTML <label> element is used to label forms. Notice that the “for” attribute in the <label> element matches the “id” attribute in its associated form input field. Also notice that the Fieldset element encloses the form as a whole, as well as the radio buttons where a person identifies her/his sex. A Fieldset is used to group and describe related form elements. Each Fieldset has a Legend – which acts like a parent label – containing the text to describe the form elements grouped in the Fieldset.

When an assistive technology encounters the form, it announces the Legend “Join Group,” followed by the labels “Name, edit text” and, then further into the form, “Join Group, Sex, Male radio button unselected.” In each case, the Legend is announced along with the Label of each field providing additional meaning to clarify the purpose of each input element. Without the Fieldset, the assistive technology would announce “Name” for the first element, and then “male, radio button unselected,” which is an unclear description of the input field’s purpose.

```
// Use the <label> element to explicitly
// associate a description with each input
// element, and use the <fieldset> and <legend>
// elements for groupings of related form elements

<fieldset><legend>Join Group</legend>
<form>

 <label for="text1">Name</label>
 <input type="text" id="text1" />
 <fieldset><legend>Sex</legend>

 <label for="radio1">Male</label>
 <input type="radio" id="radio1" name="sex" />

 <label for="radio2">Female</label>
 <input type="radio" id="radio2" name="sex" />

 </fieldset>

 <label for="checkbox1">I am New</label>
 <input type="checkbox" id="checkbox1" />
 <br />
 <input type="submit" value="Join" />
</form>
</fieldset>
```

The preceding markup produces a form that looks something like the following:



Links

A common error in creating web content is using meaningless link text, such as “click here” or “learn more,” which provides no useful information and increases navigation efforts for everyone. It is much easier to scan through a page to discover where links lead to when the link text is meaningful on its own, such as “Register,” “Learn About Us,” or “Login Now.”

While assistive technologies allow users to list all the links on a page, there is no value in getting multiple listings of “click here” or “learn more.” The exceptions are when text surrounding “click here” or “learn more” links can provide context; for example, on news sites, where a headline is presented as a link, followed by a short excerpt from an article, then followed by a “more” link that leads to the full article. In this case, the headline link provides context, adding meaning to the “more” link that follows for the AT user navigating through these links.

Creating context in this way is acceptable at Level A in the WCAG 2.0 guidelines (see Guideline 2.4.4), but it is not acceptable at Level AAA. The reason is that assistive technologies often allow users to alphabetically sort the listing of links, which in the previous example would then inadvertently shift the position of the headline link and therefore lose any meaning that the "more" link once had.

The general rule for links is that they should be meaningful on their own, describing either the function of a link (such as opening up a popup window) or its destination (what the user will find when s/he follows the link).

Consider the following two examples, and the amount of effort you, as a sighted user, must use to discover the locations of the links in the content if you were scanning – as opposed to reading – the paragraph.

More Effort with Meaningless Link Text

The W3C Web Content Accessibility Guidelines (WCAG 2.0) describe what is required to produce web content that will be accessible to everyone. Learn more about WCAG 2.0 here. WCAG 2.0 includes a variety of supporting documents that go into great detail about success criteria and techniques that can be used to meet these criteria. Click here for more about supporting documents. Tools such as the AChecker Web Accessibility Checker, found here, can be used to identify accessibility problems, and to learn about strategies that can be used to resolve problems.

Less Effort with Meaningful Link Text

The W3C Web Content Accessibility Guidelines (WCAG 2.0) describe what is required to produce web content that will be accessible to everyone. WCAG 2.0 includes a variety of supporting documents that go into great detail about success criteria and techniques that can be used to meet these criteria. Tools such as the AChecker Web Accessibility Checker can be used to identify accessibility problems, and to learn about strategies that can be used to resolve problems.

Language and Readability

Language and readability are significant drivers of accessibility in web content. For each web page, **identify the primary language** being presented by simply adding the "lang" attribute to the page's opening HTML element. This ensures that the assistive technology reading the page knows which pronunciation to use. For instance, imagine how French would sound if it were read using an English pronunciation. The example that follows shows the lang attribute in the opening HTML element of a page.

```
// Use the lang attribute in the opening HTML element to
// identify the primary language of a page

<html lang="en">

// When XHTML is being used add both the lang
// and xml:lang attributes

<html lang="en" xml:lang="en">
```

When multiple languages are being presented on a single page, it is important to identify **changes in language** for language that is not the primary language. A common issue found on many Canadian websites pertains to the Français or English link used to switch between the two languages. The language of these links must be identified in the HTML markup containing the words; again, use the "lang" attribute. The same is true for any text on the page that is not the primary language.

```
// For the Français link, identify
// the text as French using lang="fr"

<a href="french_index.html" lang="fr">Français</a>

// For the English link on the French site, identify
// the text as English using lang="en"

<a href="english_index.html" lang="en">English</a>
```

The reading level of web content may also create barriers for those who have difficulty with comprehension. When writing for the web, especially for sites geared to the public, use the simplest language possible, generally at a grade 9 or 10 level. Avoid using sarcasm, idioms, metaphors, and other non-standard forms of writing that run the risk of being misunderstood.

The use of simple language can improve a website's readability – and therefore its understanding – for people whose first language is not the language in which the content is being presented, as well as for individuals who may have a cognitive or reading-related disability. If your website's audience is more defined and advanced, use language appropriate for your audience, but still be mindful of keeping the words simple wherever possible.

More Accessibility Dos and Don’ts

Do’s

- Do use cascading style sheets (CSS) to format appearance rather than using HTML attributes.
- Do also define background colour when text colour has been defined.
- Do use relative measures (em, %) to define sizes, rather than using absolute measures (px, pt, cm) wherever possible.
- Do use text and background colours that provide good contrast.
- Do title all pages uniquely.
- Do attempt to use link text, and a page title and main heading on the link’s destination page that use identical wording for each, or, *at a minimum*, use words that are equivalent in their meaning (for example, when space is limited).
- Do use the same words as link text when multiple links lead to the same location.
- Do use unique link text when links lead to different locations.
- Do provide definitions for unusual or new words.
- Do expand all abbreviation, preferably in each case using the HTML Abbr, or Acronym elements.

Don’ts

- Don’t use the HTML blockquote element to create indentation, unless the text being indented is actually a quote.
- Don’t use images to present text when actual text could be used.
- Don’t use deprecated HTML elements (e.g., <font>,<b>,<s>, <u>, and <center>).
- Don’t use deprecated HTML attributes (e.g., align, bgcolor, border, nowrap, width, and vspace).
- Don’t use colour alone to represent meaning (e.g., don’t write “click the red button”).
- Don’t remove the standard underline from links when links are surrounded by other text.
- Don’t use invisible images to create space. Use CSS for this purpose.
- Don’t open new windows without warning the user ahead of time.