PBS.org Best Practices for Web Standards-Based Design

Version 1.1
December 2006

Companion Web site:
http://dipsy.pbs.org/bestpractices/

Table of Contents

Introduction **3**

Web Standards **4**

Semantic Markup **5**

XHTML **6**

CSS **7**

JavaScript **11**

Site Accessibility **12**

- Images **12**
- Navigation **13**
- Content **14**
- Web forms **15**
- Tables **17**
- Accessibility certification and testing **21**

Resources **23**

- Web sites **23**
- Recommended books **25**

Index **27**

Introduction

The ideas expressed throughout this document reflect a forward-thinking, standards-based approach to design. Web standards allow for the separation of presentation from structure by moving code that defines the display of pages to external Cascading Style Sheets (CSS). Removing presentation from Web pages allows developers to concentrate on creating a good semantic structure for XHTML code before addressing how pages will look to users. Web standards provide the foundation for building more flexible and accessible Web sites.

The development of these best practices is a collaborative effort open to all Web developers in the PBS community. Please contact us (http://dipsy.pbs.org/bestpractices/contact/) with your ideas, comments, or questions so we can continue to evolve this document.

Web Standards

The design of all PBS.org shell pages strives to comply with Web standards. Web standards are a collection of World Wide Web Consortium (http://www.w3.org/) specifications for creating products that are accessible to users with different needs and across multiple platforms.

Web standards allow for the separation of presentation from structure by moving code that defines the display of pages to external Cascading Style Sheets (CSS). The separation of structure from presentation has some significant benefits including:

- Cleaner code
- Global control of site presentation
- Easier editing and maintenance
- Smaller pages that eat up less bandwidth

Removing presentation from Web pages allows Web developers to concentrate on creating a good semantic structure for XHTML code before addressing how pages will look to users.

Web pages making use of Web standards render faster and require less bandwidth because display information only needs to be downloaded once. It's then served up locally from the browser's cache for use by multiple pages.

Coding with Web standards also allows the code to be validated by the W3C validation service (http://validator.w3.org/). This improves quality by allowing code to be checked for errors.

Sites designed to Web standards are:

- **HIGHLY ACCESSIBLE:** They are more easily experienced by people with disabilities. Web standards fully support the use of assistive technology products such as screen readers.
- **MORE VISIBLE:** Structuring Web pages correctly makes it easy for search engines to access information, evaluate it, and index it more accurately.
- **FLEXIBLE:** They can be made available on a wide range of platforms, not just traditional Web browsers.
- **SELF-RELIANT:** Web standards do not require plug-ins (such as Flash) or proprietary code that caters to specific browsers.
- **BUILT TO EVOLVE:** Web standards continue to evolve, but they remain mindful of past technology. Pages built to Web standards degrade gracefully in older Web browsers and pages built with older standards continue to work in newer browsers.
- **CONSISTENTLY MAINTAINED:** Web standards offer an easy set of rules that any Web developer can follow and understand.

Resources

What are Web Standards and Why Should I Use Them?
http://www.webstandards.org/learn/faq/

Web Standards for Business
http://www.webstandards.org/learn/reference/web_standards_for_business.html

Semantic Markup

Semantic markup is a key part of designing with Web standards. When display information is moved to Cascading Style Sheets (CSS) two kinds of information remain: structural XHTML code and content. The separation of content from presentation allows developers to write semantic code that describes what the content *means* versus how it *looks*. For example, an H1 tag is used to identify text as a heading, not to dictate the size of that text. (The size, font, style and spacing of the text are defined in a CSS file.)

The use of semantic markup creates pages that are ordered, logical, clean, concise and more accessible. Some examples of semantic markup in XHTML:

Headings: `<h1> <h2> <h3> <h4> <h5> <h6>`
When using semantic markup, the most important heading on the page gets `<h1>` tags, sub headings get `<h2>` tags, sub-sub headings get `<h3>` tags and so on.

Paragraphs: `<p>`
Paragraph tags contain paragraphs of content; they should not be used to create blank space or line returns.

Line breaks: `<br />`
Line break tags force a break in a line of content; they should not to be used to create blank space.

Lists: `<ul> <ol> <li>`
These tags identify text as a list of items. CSS can be used to adjust spacing and indents; to remove bullets or numbers; and to display a list horizontally rather than vertically.

Quotes: `<blockquote>`
Blockquote tags identify a paragraph-long quote; they should not be used to indent content.

Tables: `<table> <th> <tr> <td>`
Table tags describe tabular data; they should not be used for the layout of a Web page. TV schedule grids and calendars are both good examples of data where table markup is appropriate.

Using tags

Semantic code doesn't just apply to XHTML tags with specific uses such as paragraphs or headings. Semantic use of `<div>` container tags can be accomplished by giving the `<div>` containers class or id names. How class or id names are selected is very important. When using class or id names for div containers, make sure the name describes the content of the container rather than describing what the container looks like. For example: `<div id="footer">` vs. `<div id="bluebox">`.

Sample page using semantic markup

```
<div id="header">
      <h1>Page Heading</h1>
</div>
<div id="nav">
      <ul>
            <li><a href="">Site nav link</a></li>
            <li><a href="">Site nav link</a></li>
            <li><a href="">Site nav link</a></li>
      </ul>
</div>
<div id="content">
      <h2>Subheading</h2>
      <h3>Sub-sub heading</h3>
      <p>Content<br />
      more content</p>
      <blockquote>Quote</blockquote>
      <h3>Sub-sub heading</h3>
      <p>Content</p>
      <ul>
            <li>List item</li>
            <li>List item</li>
      </ul>
      <table>
            <tr>
                  <td>Tabular data</td>
                  <td>Tabular data </td>
            </tr>
      </table>
</div>
<div id="footer">
      <p>Copyright</p>
</div>
```

Resources

Integrated Web Design: The Meaning of Semantics
http://www.informit.com/articles/article.asp?p=369225&rl=1

The Meaning of Semantics: Naming Conventions for Class and ID in CSS
http://www.informit.com/articles/article.asp?p=170514

Semantics Simple Quiz
http://www.simplebits.com/bits/simplequiz/

XHTML

XHTML is a markup language similar to HTML, but with stricter syntax. XHTML is less forgiving than HTML; if its coding rules aren't followed, it will not work correctly.

While HTML is an application of SGML (Standard Generalized Markup Language), XHTML is an application of XML (Extensible Markup Language), a more restrictive subset of SGML. XHTML can be thought of as the intersection of HTML and XML.

The benefits of XHTML

Cleaner and more logical code

The strictness of XHTML leads to cleaner code. Also, XHTML is written with semantic markup in mind, which makes for more logical pages. A correctly coded XHTML document is organized like an outline: `<h1>` is the main page heading, `<h2>` is a subheading, and `<h3>` is a sub-sub heading, etc.

Interoperability on multiple platforms

Since XHTML Web pages use precise syntax, follow semantic markup rules and adhere to Web standards, they can be viewed on multiple platforms, not just traditional Web browsers. XHTML can be viewed on computers, cell phones, PDAs, and screen readers.

Backwards compatibility

Browsers that can't correctly render Web pages written in XHTML will degrade those pages gracefully. This means that they won't appear exactly as intended, but none of the page's content will be lost, hidden, or distorted.

Validation and error correction

Web pages built using XHTML can be validated through validation services such as the ones offered at the World Wide Web Consortium (W3C) Web site (http://www.w3.org/).

XHTML guidelines

Use the proper DOCTYPE

A DOCTYPE is a tag that tells the browser how to interpret a page's XHTML code. Shell pages on PBS.org use the least strict DOCTYPE, *XHTML 1.0 Transitional*. This DOCTYPE allows the use of tables and some deprecated HTML 4.0 tags such as `<center>`, `<u>`, `<strike>`, and `<applet>`. XHTML 1.0 Transitional works in older browsers better than other DOCTYPES.

This code should appear at the very beginning of every shell page, before any other code:

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

Include an HTML declaration

Place this code after the DOCTYPE:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
```

Follow XHTML coding rules
The code for all PBS.org shell pages must follow these rules:

- **All tags must be lowercase:** XHTML is case sensitive. Uppercase tags affect a page's ability to validate properly.
- **All attribute values must be quoted:** For example: `height="10"` rather than `height=10`.
- **All tags must be closed:** A tag must be closed with its corresponding close tag: .Tags which do not have corresponding close tags, such as <img> tags, are self closing. For example: `<img src="image.gif" />`. Note the single blank space that precedes />. The space is there to avoid confusing browsers that were released prior to the XHTML standard. Some examples:

```
<h1>Title</h1>
<p>paragraph<br />
another line</p>
<img src="banner.jpg" alt="banner" title="return to home
page" />
```

- **All XTHML code must be validated:** The HTML validator at W3C's Web site (http://validator.w3.org/) will identify errors in your code.

Resources

XHTML: Guidelines & Benefits
http://www.nypl.org/styleguide/

Common Ideas Between HTML and XHTML
http://www.webstandards.org/learn/reference/common_ideas.html

W3C XHTML Validator
http://validator.w3.org/

CSS

After developers have coded semantically structured pages with XHTML, they need to be able to stylize the pages' elements. This is where Cascading Style Sheets (CSS) come in. CSS allows Web developers to stylize any XHTML tag and helps them retain a consistent look and feel across pages.

The benefits of CSS

Separation of structure and presentation

Rather than being part of a Web page, CSS code lives in its own file. This not only makes the Web page code cleaner, but allows for smarter development since developers can concentrate on creating a good semantic structure separate from designing how pages will look to users. Pages that use CSS render faster and require less bandwidth because display information only needs to be downloaded once. It's then served up locally from the browser's cache for use by multiple pages.

Global control of presentation

With CSS, Web developers have greater control over the presentation of their Web sites. CSS allows developers to globally control the presentation of XHTML code. For example, if a developer has defined what an <h1> tag will look like in a CSS file and includes an <h1> for every header on every page of the site, they can make a single change and have it affect all of the site's pages at once.

Alternate style sheets

Developers can use CSS to stylize alternate versions of the same content. For example, printable versions of Web pages can be made available without creating an additional set of pages simply by linking to an alternate style sheet designed to maximize printability.

CSS guidelines

Be aware of CSS 2.1 compatibility issues

Developers building Web pages for PBS.org should use all that CSS 1 has to offer such as typographical styles (font-family, font-size, font-weight, line-height, color etc.), spacing styles (such as padding and margin.), and positioning styles (display and float.)

Developers should be cautious with what CSS 2.1 attributes they use because of incompatibilities with some Web browsers (mostly Internet Explorer.) Developers will have to research which parts of CSS 2.1 they can use that will work on all browsers. (NOTE: CSS 2.1 positioning styles such as static, relative, absolute, and fixed are safe to use.)

CSS hacks

Developers may have to use CSS hacks to fix known CSS bugs in some Web browsers (mostly Internet Explorer). Any hacks that are included in a CSS file should be put at the end of the document for easy maintenance. The hack should be labeled and commented appropriately, describing the reason for the hack and what specific browser version the hack fixes. When buggy browsers are updated to render standards correctly, developers can then remove the hacks from the end of their CSS file.

CSS hacks are not encouraged and should only be used when there are no other solutions. Developers should strive to keep their CSS hack free.

Developers can find solutions to common CSS bugs at Position Is Everything (http://www.positioniseverything.net/.)

Validate your CSS
Web developers can validate their CSS files with W3C's CSS validation service (http://jigsaw.w3.org/css-validator/.)

Resources

CSS Crib Sheet
http://www.mezzoblue.com/css/cribsheet/

Cascading Style Sheets, Level 1 Specification
http://www.w3.org/TR/CSS1

Cascading Style Sheets, Level 2 Revision 1
CSS 2.1 Specification
http://www.w3.org/TR/CSS21/

Position Is Everything: CSS Bugs & Fixes
http://www.positioniseverything.net/

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

JavaScript

Javascript is a scripting language that adds functionality to a Web page. Javascript is about the behavior of a Web page, while XHTML is about structure and Cascading Style Sheets (CSS) are about presentation.

JavaScript guidelines

Separate behavior from structure
JavaScript should not be embedded within the XHTML code but called from an external JavaScript (.js) file. Identify XHTML elements with class or id names and give those class or id names behaviors within the external JavaScript file. The link to an external JavaScript file is located within the <head> tag at the top of the Web page.

Example:

```
<head>
      <title>Page Title</title>
      <link rel="stylesheet" type="text/css" href="global.css "
      media="all" />
      <link rel="stylesheet" type="text/css" href="print.css"
      media="print" />
      <script type="text/javascript" src="global.js"></script>
</head>
```

Since JavaScript is called from an external file, it creates cleaner XHTML code and developers can easily modify behaviors across their site globally.

Build pages that do not rely on JavaScript
Developers should build Web pages that degrade gracefully when JavaScript is turned off. This makes the page more accessible to users who do not have devices that understand JavaScript or do not have JavaScript enabled.

Resources

Ten Good Practices for Writing Javascript
http://www.bobbyvandersluis.com/articles/goodpractices.php

Separating Behavior and Structure
http://www.digital-web.com/articles/separating_behavior_and_structure_2/

Separating Behavior and Presentation
http://www.digital-web.com/articles/separating_behavior_and_presentation/

Site Accessibility

Developers who code semantically correct XHTML Web pages have taken the first step towards better accessibility by separating structure from presentation and by giving the page an outlined architecture. Other accessibility issues exist however that can't be solved with Web standards alone.

Making images accessible

Images that are used on Web pages are not always accessible to everyone. Users who use screen readers or text-based browsers can't see images and must get an alternative text equivalent. Here are a few techniques for making images more accessible:

The alt attribute

All tags for meaningful images on PBS.org Web pages must include text alternatives. Guidelines for the correct use of the alt attribute:

1. Include the alt attribute in <img> tags. For example:
   `<img src="apple.gif" width="100" height="100" alt="Red delicious apple" />`
2. Limit alt text to 1,024 characters (1K) or less. Describe each image with just a few words, don't write a book.
3. Describe what the image is or represents or sum up its function, but don't write that it's a picture. For example, don't write "picture of a red delicious apple" or "picture of PBS logo."
4. When writing `alt` text, do not use the image's file name. For example, don't write `alt="apple.gif"`.
5. Images that aren't meaningful, like special borders or horizontal rules, should have a NULL **`alt`**. A NULL **`alt`** is defined as an alt attribute with a blank value. For example, `<img src="spacer.gif" width="10" height="10" alt="" />`.
6. Don't use alt attributes as placeholders for loading graphics. For example, don't write `alt="Loading Image"`.
7. Background images do not need `alt` text.

The title attribute

The title attribute can be used on mostly any XHTML tag (not just <img> tags) and provides more information than an alt attribute. For image accessibility, you can use the title attribute to provide helpful but optional details. The title attribute can:

- **Provide a little more info:**
  `<img src="apple.gif" width="100" height="100" alt="Red delicious apple" title="Red delicious apple, grown in Washington State." />`
- **Describe significance:**
  `<img src="apple.gif" width="100" height="100" alt="Red delicious apple" title="I enjoyed red delicious apples every fall when I was a child." />`

Guidelines for the correct use of the title attribute:

- If you are going to use the title attribute, use it for every image on the Web page.

Making navigation accessible

Navigation as a list

Navigation on a Web page is more accessible when it is coded as a list of links (using `<ul>` and `<li>` tags). The navigation will act differently on different devices:

- Text-based devices don't recognize Cascading Style Sheets (CSS) so the list will just look like a list.
- On a Web browser with CSS enabled, the list will look like site navigation.
- A screen reader will read through the list item by item.
- A user who can't use a mouse will be able to tab through the list using their keyboard's tab key.

Skip navigation link

Providing a skip navigation anchor link at the top of the Web page before the site navigation helps users who use screen readers or text browsers by allowing them to jump ahead to the content of the site. The code looks like this:

```
<a href="#content" title="Skip navigation"
id="skipnavigation">Skip navigation</a>
<ul id="nav">
      <li><a href="portfolio.html"
      title="Portfolio">Portfolio</a></li>
      <li><a href="links.html" title="Links">Links</a></li>
      <li><a href="contact.html" title="Contact">Contact</a></li>
</ul>
```

- `<a href="#content">` is a link to an anchor near the content area elsewhere on the page.
- The title attribute describes the function of the link.
- The `"skipnavigation"` id links to display information in the page's CSS file. CSS hides the skip navigation link on Web devices that render CSS. Since text browsers and most screen readers don't render CSS, they will display the skip navigation link.

Developers can use this method to create other "skip to" parts of the page such as a search box, a user login, or other navigation.

Making content accessible

Color

Web developers must be careful when choosing colors for their Web site. When text colors and background colors are too similar, it can make text hard to read. Developers should also be aware of how their site will look to someone with color blindness. A great tool for testing color blindness is available at Visicheck: Color Blind Simulator (http://www.vischeck.com/vischeck/vischeckURL.php.)

When developers are creating Web pages they should also make sure that text and graphics are still clearly understood when viewed without color.

Text

The size of text and the ability of users to re-size text on their Web devices is a crucial part of Web accessibility. Font size should be set big enough in the CSS file to be read, but should also have the ability to change based on the user's preferences.

Most Web developers set their font size in pixels, but Internet Explorer can't zoom text that is set in pixels. Keywords are an effective alternative to pixels for sizing fonts in CSS. They are not pixel-perfect, but they allow text zooming on any browser. The keywords:

- xx-small
- x-small
- small (about the size of 12px)
- medium
- large
- x-large
- xx-large

Now, if the global font size is set at "small," then percentages of that base size can be used to adjust the other text elements on pages. For example:

```
body
{
      font-size: small;
      /* global base font-size */
}

h1
{
      font-size: 150%;
      /* header 1 font-size is 150% of small */
}

p
{
      font-size: 95%;
      /* paragraph font-size is 95% of small */
}
```

Font-sizes across the site would have a proportional relationship. If a user resized the text in their browser while viewing this site, all text sizes would increase or decrease in proportion to each other.

Making Web forms accessible

Web forms have complicated accessibility issues. To deal with these issues, developers should build forms that are semantically correct and the forms should be built without using <table> tags. Developers should also consider using these techniques:

Field sets
Developers can use the <fieldset> tag to chunk up parts of a form into relevant sets. These field sets make it easier for screen reader and text-based browser users to jump to different parts of the form. For example, first name, middle initial and last name can be chunked into the same field set. (NOTE: Each field set should include <legend> tags to act as a header.) An example:

```
<form action="/cgi-registry/postmaster.pl" method="post"
id="feedback" title="Please provide us with some personal
information.">
      <fieldset>
            <legend>Personal Information</legend>
            <label for="firstname">First Name:</label>
            <input type="text" name="firstname" id="firstname"
            title="Enter your first name." />
            <label for="middleinitial">Middle Initial:</label>
            <input type="text" name="middleinitial"
            id="middleinitial" title="Enter your middle initial."
            />
            <label for="lastname">Last Name:</label>
            <input type="text" name="lastname" id="lastname"
            title="Enter your last name." />
      </fieldset>
      <input type="submit" value="Send" class="button"
      title="Send your feedback." />
</form>
```

Labels
Developers should use <label> tags to associate field labels with their corresponding input field. For example, this “First Name” label corresponds to the input field with id="firstname":

```
<label for="firstname">First Name:</label>
<input type="text" name="firstname" id="firstname" title="Enter
your first name."   />
```

Label tags allow users to click on the label next to an input field and have that input field selected. Users can also tab to a label and have the input field selected. Label tags allow the developer to link a label to an associated input field anywhere on the Web page.

Titles
The title attribute make Web forms more accessible. The title attribute should be used on the `<input>` and `<textarea>` tags as well as anywhere else the user has to enter their own information. Titles should advise the user about what information to put in the input fields:

```
<input type="text" name="firstname" id="firstname" title="Enter
your first name." />

<textarea name="comments" id="comments" title="Enter your
comments." rows="10" cols="10" ></textarea>
```

Form submission
Every Web form should have some sort of "submit" button associated with it. If it's a search form the button could say "Go" or "Search". The form should also be able to be submitted by pressing return or enter keys on the keyboard.

Important: Form actions should not be activated until the user has pressed the submit button. For example, a drop-down menu used for navigation purposes should not bring the user to the requested page until the user has given the go ahead. This method gives the user an escape route if they make a mistake and allows them change their selection before submission.

Other problems with form activation without user submission involve screen readers. Screen readers automatically select the first item in a menu when the user opens it using either a mouse of keyboard. The user then gets automatically directed to the destination of the first selection without getting to see or select the other options.

Tabindex
Developers should use the `tabindex` attribute to allow users to "tab" through a form in a specific order using the tab key on the keyboard. For a majority of pbs.org users the `tabindex` order should be setup to cycle through the form from top left to bottom right. The `tabindex` attribute can be applied to the following form tags: `<input>`, `<select>`, and `<textarea>`. Once the `tabindex` order has been added to the form, test it out using the tab key in a browser. The code looks like this:

```
<form action="/cgi-registry/postmaster.pl" method="post"
id="feedback" title="Please provide us with some personal
information.">
      <fieldset>
            <legend>Personal Information</legend>
            <label for="firstname">First Name:</label>
            <input type="text" name="firstname" id="firstname"
            title="Enter your first name." tabindex="1" />
            <label for="middleinitial">Middle Initial:</label>
            <input type="text" name="middleinitial"
            id="middleinitial" title="Enter your middle initial."
            tabindex="2"   />
            <label for="lastname">Last Name:</label>
            <input type="text" name="lastname" id="lastname"
            title="Enter your last name." tabindex="3" />
      </fieldset>
      <input type="submit" value="Send" class="button"
      title="Send your feedback."  tabindex="4" />
</form>
```

Making tables accessible

To maximize accessibility:

- Tables should only be used for tabular data and not for page layout.
- Tables should never be embedded within tables. This causes big accessibility headaches, especially for screen readers.
- Developers should only use `<thead>`, `<tbody>` and `<tfoot>` tags when coding tabular data. (These tags help Web devices group sections of a table together and are helpful when data tables get very long.)

More table tag guidelines

<thead>
The table header is coded around the table rows and columns which contain the table data headers (<th>):

```
<thead>
      <tr>
            <th scope="col">Year</th>
            <th scope="col">Revenue</th>
      </tr>
</thead>
```

<tfoot>
The table footer is coded around the table rows and columns that contain the table footer:

```
<tfoot>
      <tr>
            <td>Data source: Market Watch</td>
      </tr>
</tfoot>
```

<tbody>
The table body is coded around the table rows and columns that contain the table data:

```
<tbody>
      <tr>
            <td>2002</td>
            <td>$10</td>
      </tr>
      <tr>
            <td>2003</td>
            <td>$15</td>
      </tr>
      <tr>
            <td>2004</td>
            <td>$30</td>
      </tr>
</tbody>
```

<th>

Table data headers are used for column or row headers. Table data headers are like labels for the table's content and helps browsers and other Web devices better understand how the table is organized:

```
<table cellspacing="0" summary="Revenue Per Year: 2002-2004"
id="revenuechart">
      <caption>Revenue Per Year: 2002-2004</caption>
      <thead>
            <tr>
                  <th scope="col">Year</th>
                  <th scope="col">Revenue</th>
            </tr>
      </thead>
      <tfoot>
            <tr>
                  <td>Data source: Market Watch</td>
            </tr>
      </tfoot>
      <tbody>
            <tr>
                  <td>2002</td>
                  <td>$10</td>
            </tr>
            <tr>
                  <td>2003</td>
                  <td>$15</td>
            </tr>
            <tr>
                  <td>2004</td>
                  <td>$30</td>
            </tr>
      </tbody>
</table>
```

The scope attribute

The scope attribute is used in table data header tags (`<th>`). It helps screen readers identify which cells belong to the specified table data header. Developers should add `scope="col"` to column table data headers and `scope="row"` to row table data headers:

```
<table cellspacing="0" summary="Revenue Per Year: 2002-2004"
id="revenuechart">
      <caption>Revenue Per Year: 2002-2004</caption>
      <thead>
            <tr>
                  <th scope="col">Year</th>
                  <th scope="col">Revenue</th>
            </tr>
      </thead>
      <tfoot>
            <tr>
                  <td>Data source: Market Watch</td>
            </tr>
      </tfoot>
      <tbody>
            <tr>
                  <td>2002</td>
                  <td>$10</td>
            </tr>
            <tr>
                  <td>2003</td>
                  <td>$15</td>
            </tr>
            <tr>
                  <td>2004</td>
                  <td>$30</td>
            </tr>
      </tbody>
</table>
```

<caption>

Tabular data tables should be labeled using the <caption> tag. The <caption> tag comes before any table rows or table cells but displays outside of the table when rendered in a browser:

```
<table cellspacing="0" summary="Revenue Per Year: 2002-2004"
id="revenuechart">
      <caption>Revenue Per Year: 2002-2004</caption>
      <thead>
            <tr>
                  <th scope="col">Year</th>
                  <th scope="col">Revenue</th>
            </tr>
      </thead>
      <tfoot>
            <tr>
                  <td>Data source: Market Watch</td>
            </tr>
      </tfoot>
      <tbody>
            <tr>
                  <td>2002</td>
                  <td>$10</td>
            </tr>
            <tr>
                  <td>2003</td>
                  <td>$15</td>
            </tr>
            <tr>
                  <td>2004</td>
                  <td>$30</td>
            </tr>
      </tbody>
</table>
```

The summary attribute

If a developer creates a tabular data table they should use the summary attribute to describe the information contained in the table. The summary attribute is contained within the <table> tag:

```
<table cellspacing="0" summary="Revenue Per Year: 2002-2004"
id="revenuechart">
```

Accessibility certification and testing

WCAG (Web Content Accessibility Guidelines)

Pages developed for PBS.org must pass Level A WCAG accessibility specifications. Passing Level A specifications means the pages have met Priority 1 requirements. Level A compliance is the easiest to attain and does the greatest good in accessibility terms.

Pages must meet Priority 1 requirements. Pages should meet Priority 2 requirements and may meet Priority 3 requirements. Pages that meet Priority 1 and Priority 2 requirements attain Level AA WCAG compliance. If they have meet Priority 1, Priority 2, and Priority 3 requirements attain Level AAA WCAG compliance.

Developers can use these resources to help them get started:

- WCAG guidelines
  http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/

- Techniques for WCAG guidelines
  http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/

- Checklist of checkpoints for WCAG guidelines
  http://www.w3.org/TR/WAI-WEBCONTENT/full-checklist.html

Section 508

Pages developed for PBS.org must also pass Section 508 accessibility specifications. Developers can test their Web pages for Section 508 compliance at Cynthia Says (http://www.contentquality.com/), an automated accessibility checker that can run tests for both Section 508 and WCAG.

More information about Section 508 can be found at the U.S. Access Board (http://www.access-board.gov/508.htm.)

Resources

Building Accessible Web Sites
http://joeclark.org/book/sashay/serialization/

NCAM Rich Media Accessibility
http://ncam.wgbh.org/richmedia/

Accessible Digital Media: Design Guidelines for Electronic Publications, Multimedia and the Web
http://ncam.wgbh.org/publications/adm/

Side by Side WCAG vs. 508
http://www.jimthatcher.com/sidebyside.htm

WCAG guidelines
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/

Techniques for WCAG guidelines
http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/

Checklist of checkpoints for WCAG guidelines
http://www.w3.org/TR/WAI-WEBCONTENT/full-checklist.html

Section 508
http://www.access-board.gov/508.htm

Visicheck: Color Blind Simulator
http://www.vischeck.com/vischeck/vischeckURL.php

Cynthia Says: Accessibility Check
http://www.contentquality.com/

Resources

Recommended Web sites

Web Standards

What are Web Standards and Why Should I Use Them?
http://www.webstandards.org/learn/faq/

Web Standards for Business
http://www.webstandards.org/learn/reference/web_standards_for_business.html

Semantic Markup

Integrated Web Design: The Meaning of Semantics
http://www.informit.com/articles/article.asp?p=369225&rl=1

The Meaning of Semantics: Naming Conventions for Class and ID in CSS
http://www.informit.com/articles/article.asp?p=170514

Semantics Simple Quiz
http://www.simplebits.com/bits/simplequiz/

XHTML

XHTML: Guidelines & Benefits
http://www.nypl.org/styleguide/

Common Ideas Between HTML and XHTML
http://www.webstandards.org/learn/reference/common_ideas.html

W3C XHTML Validator
http://validator.w3.org/

CSS

CSS Crib Sheet
http://www.mezzoblue.com/css/cribsheet/

Cascading Style Sheets, Level 1 Specification
http://www.w3.org/TR/CSS1

Cascading Style Sheets, Level 2 Revision 1
CSS 2.1 Specification
http://www.w3.org/TR/CSS21/

Position Is Everything: CSS Bugs & Fixes
http://www.positioniseverything.net/

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

JavaScript

Ten Good Practices for Writing Javascript
http://www.bobbyvandersluis.com/articles/goodpractices.php

Separating Behavior and Structure
http://www.digital-web.com/articles/separating_behavior_and_structure_2/

Separating Behavior and Presentation
http://www.digital-web.com/articles/separating_behavior_and_presentation/

Accessibility

Building Accessible Web Sites
http://joeclark.org/book/sashay/serialization/

NCAM Rich Media Accessibility
http://ncam.wgbh.org/richmedia/

Accessible Digital Media: Design Guidelines for Electronic Publications, Multimedia and the Web
http://ncam.wgbh.org/publications/adm/

Side by Side WCAG vs. 508
http://www.jimthatcher.com/sidebyside.htm

WCAG guidelines
http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/

Techniques for WCAG guidelines
http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/

Checklist of checkpoints for WCAG guidelines
http://www.w3.org/TR/WAI-WEBCONTENT/full-checklist.html

Section 508
http://www.access-board.gov/508.htm

Visicheck: Color Blind Simulator
http://www.vischeck.com/vischeck/vischeckURL.php

Cynthia Says: Accessibility Check
http://www.contentquality.com/

Recommended Books

Web Standards

Designing With Web Standards
Jeffrey Zeldman

Why Web standards (such as XHTML, CSS, ECMAScript, and DOM) are good for everyone, and why site designers and browser makers should move towards standards compliance.
(http://www.zeldman.com/dwws/)

XHTML

Bulletproof Web Design
Dan Cederholm

This book outlines 10 strategies for creating standards-based designs that provide flexibility, readability, and user control—key components of every successful Web site.
(http://www.simplebits.com/publications/bulletproof/)

CSS

CSS Pocket Reference
Eric A. Meyer

Briefly introduces CSS and then lists all CSS1 properties, plus the CSS1 pseudo-elements and pseudo-classes.
(http://www.meyerweb.com/eric/books/css-pocket/)

The Zen of CSS Design
Molly E. Holzschlag and Dave Shea

Provides an eye-opening look at the range of design methods made possible by CSS (Cascading Style Sheets).
(http://www.peachpit.com/title/0321303474)

JavaScript

JavaScript: The Definitive Guide, Fourth Edition
David Flanagan

Covers the JavaScript language (version 1.0 through version 1.5) and its relatives, JScript and ECMAScript, as well as the W3C DOM standards they're often used to manipulate.
(http://www.oreilly.com/catalog/jscript4/index.html)

Accessibility

Building Accessible Websites
Joe Clark

This book teaches developers of every sophistication and budget level how to improve the accessibility of their Web sites so that people who are blind, deaf, or mobility-impaired can get the most out of them.
(http://www.joeclark.org/book/)

Index

A

a link tag (<a>). *see* anchor link
accessibility , 12-22
  as related to JavaScript, 11
  as related to PBS.org, 3
  as related to semantic markup, 3, 5, 12, 15
  as related to W3C (World Wide Web Consortium), 4
  as related to Web standards, 3, 4
  as related to XHTML (Extensible Hypertext Markup Language), 3, 7, 12
  books, 26
  content, 14
    color blindness issues, 14
    color choices, 14
    text resizing with keywords, 14
    text zoom, 14
    Visicheck: Color Blind Simulator, 14
  certification, 21
  Cynthia Says: Accessibility Checker. *see* Cynthia Says
  images, 12
    use of `alt` attribute, 12
    use of NULL `alt`, 12
    use of `title` attribute, 12
  navigation, 13
    skip navigation link, 13
  recommended books, 26
  resources, 21, 22, 24
  Section 508, 21
  screen readers, 13
  tables, 17-20
    as related to DOCTYPE, 7
    as related to Web forms, 15
    as related to Web standards, 4
    example code, 5, 6, 15, 17, 18, 19
    use of `caption` tag (<caption>), 20
    use of `scope` attribute, 19
    use of `summary` attribute, 20
    use of `tbody` tag (<tbody>), 17
    use of `tfoot` tag (<tfoot>), 17
    use of `th` tag or table data header tag (<th>), 18
    use of `thead` tag (<thead>), 17
    use with tabular data, 17
    why not to use in page structure, 17
    why not to use in Web forms, 15
  testing, 21
  text browsers, 12, 13
  Visicheck: Color Blind Simulator, 14
  Web Content Accessibility Guidelines. *see* WCAG
  Web forms, 15-16
    as related to screen readers, 15, 16
    as related to semantic markup, 15
    as related to text browsers, 15
    example code, 15, 16
    form submission, 16
    use of `fieldset` tag (<fieldset>), 15
    use of `input` fields (<input>), 15
    use of `label` tag (<label>), 15
    use of `legend` tag (<legend>), 15
    use of `select` fields (<select>), 16
    use of `tabindex` attribute, 16
    use of `textarea` fields (<textarea>), 16
    use of `title` attribute, 16
`alt` attribute
  as related to screen readers, 12
  example code, 8, 12
  guidelines, 12
  use of NULL `alt`, 12
  use with images, 12
anchor link (<a>)
  example code, 6, 13
  use as skip navigation link, 13
  use with lists, 13
  use with `title` attribute, 13

B

`blockquote` tag (<blockquote>)
  as related to semantic markup, 5
  example code, 5, 6
br tag (
). *see* line break tag
Building Accessible Websites, 26
Bulletproof Web Design, 25

C

`caption` tag (<caption>). *see* tables
Cascading Style Sheets (CSS). *see* CSS
Cederholm, Dan (author), 25
  Bulletproof Web Design, 25
certification. *see* accessibility
Clark, Joe (author), 26
  Building Accessible Websites, 26
contact information, 3
content
  accessibility , 14
  color choices, 14
  text resizing, 14
  text zooming, 14
CSS (Cascading Style Sheets), 9
  alternate style sheets, 9
  as related to accessible content, 14
  as related to accessible navigation, 13
  as related to semantic markup, 3, 5, 9
  as related to skip navigation, 13
  as related to Web browser's cache, 4, 9
  as related to Web standards, 3, 4
  as related to XHTML (Extensible Hypertext Markup Language), 3, 9
  books, 25
  color, 9
  compatibility issues, 9
  display, 9
  example code, 11, 14
  float, 9
  font-family, 9
  font-size, 9, 14
  font-weight, 9
  guidelines, 9
  hacks and fixes, 9
  hack maintenance, 9
  keywords, 14
  line-height, 9
  margin, 9
  padding, 9
  position: absolute, 9
  position: fixed, 9
  position: static, 9
  position: relative, 9
  print style sheets, 9
  recommended books, 25

resources, 10, 23
text resizing, 14
text resizing with keywords, 14
text zooming, 14
validation by W3C (World Wide Web Consortium), 10
CSS 1. *see* CSS
CSS 2.1. *see* CSS
CSS hacks. *see* CSS
CSS Pocket Reference, 25
Cynthia Says, 21
to check Section 508 compliance, 21
to check WCAG compliance, 21

D

Designing with Web Standards, 25
`div` tag (`<div>`)
as related to semantic markup, 5
example code, 5
DOCTYPE
as related to XHTML, 7
definition, 7
example code, 7
examples of deprecated tags, 7
XHTML 1.0 Transitional, 7

E

Extensible Markup Language. *see* XML
Extensible Hypertext Markup Language. *see* XHTML

F

`fieldset` tag (`<fieldset>`), 15
example code, 15, 16
use with Web forms, 15
Flanagan, David (author), 25
JavaScript: The Definitive Guide, Fourth Edition, 25
Flash
as related to Web standards, 4
as related to Web browser plugins, 4
forms. *see* Web forms
form tag (`<form>`)
example code, 15, 16

H

head tag (`<head>`)
as related to JavaScript, 11
example code, 11
`h1`, `h2`, `h3`, `h4`, `h5`, `h6` tags. *see* heading tags
heading tags (`<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`)
as related to CSS (Cascading Style Sheets), 9
as related to semantic markup, 5
example code, 5, 6, 8, 9
HTML (Hypertext Markup Language)
as related to DOCTYPE, 7
as related to SGML, 7
as related to XHTML, 7
declaration after DOCTYPE, 7
differences from XHTML, 7
example code of declaration after DOCTYPE, 7
example of deprecated tags, 7
HTML 4.0. *see* HTML
Holzschlag, Molly E. (author), 25
Zen of CSS Design, The, 25
Hypertext Markup Language. *see* HTML

I

image tag (`<img>`)
as related to XHTML coding rules, 8
as related to `alt` attribute, 12
example code, 8, 12
images
background images, 12
use of `alt` attribute, 12
use of `title` attribute, 12
img tag (`<img>`). *see* image tag
`input` fields (`<input>`), 15
example code, 15, 16
use with `label` tag (`<label>`)
use with `tabindex` attribute, 16
use with `title` attribute, 15, 16
use with Web forms, 15
Internet Explorer (Win/Mac), 9
CSS compatibility issues, 9
CSS hacks, 9-10
zooming of text problem, 14

J

JavaScript, 11
accessibility, 11
as related to XHTML (Extensible Hypertext Markup Language), 11
books, 25
recommended books, 25
resources, 11, 24
JavaScript: The Definitive Guide, Fourth Edition, 25

K

keywords, 14
example code, 14
use with text resizing, 14
use with text zooming, 14
solution to IE (Internet Explorer) text zoom problem, 14

L

`label` tag (`<label>`), 15
use with `input` fields (`<input>`), 15
use with Web forms, 15
example code, 15, 16
`legend` tag (`<legend>`), 15
use with Web forms, 15
example code, 15, 16
`li` tag (`<li>`). *see* list tags
line break tag (`<br />`)
as related to semantic markup, 5
example code, 5, 6, 8
link tag (`<a>`). *see* anchor link
list tags (`<ol>`, `<ul>`, `<li>`)
as related to semantic markup, 5
example code, 5, 6, 13
use with accessible navigation, 13

M

Meyer, Eric A. (author), 25
CSS Pocket Reference, 25

N

navigation
- accessibility , 13
- as a list, 13
- skip navigation link, 13
  - example code, 13
  - guidelines, 13
  - hide with CSS (Cascading Style Sheets), 13
  - use with `title` attribute, 13

O

`ol` tag (`<ol>`). *see* list tags

P

p tag (`<p>`). *see* paragraph tag
paragraph tag (`<p>`)
- as related to semantic markup, 5
- as related to XHTML coding rules, 8
- example code, 5, 6, 8

PBS.org Best Practices for Web Standards-Based Design
- Web site, 1

PBS.org shell pages. *see* shell pages

R

recommended books, 25-26
- accessibility, 26
- CSS (Cascading Style Sheets), 25
- JavaScript, 25
- semantic markup, 25
- Web standards, 25
- XHTML (Extensible Hypertext Markup Language), 25

resources, 23-26
- accessibility, 21, 22, 24
- CSS (Cascading Style Sheets), 10, 23
- JavaScript, 11, 24
- semantic markup, 6, 23
- Web standards, 4, 23
- XHTML (Extensible Hypertext Markup Language), 8, 23

S

`scope` attribute. *see* tables
screen readers
- as related to accessibility, 4, 13
- as related to `alt` attribute, 12
- as related to Web forms, 15, 16
- as related to Web standards, 4
- as related to XHTML, 7

search engines
- as related to Web standards, 4

Section 508, 21
- resources, 21, 22, 24
- U.S. Access Board Web site, 21
- testing compliance with Cynthia Says, 21

`select` fields (`<select>`), 16
- example code, 16
- use with `tabindex` attribute, 16
- use with `title` attribute, 16
- use with Web forms, 16

semantic markup, 5
- as related to CSS (Cascading Style Sheets), 3, 5, 9
- as related to Web standards, 3, 4
- as related to Web forms, 15
- as related to XHTML (Extensible Hypertext Markup Language), 3, 5, 7
- definition, 5
- example code, 6
- of `div` containers, 6
- of headings, 6
- of line breaks, 6
- of lists, 6
- of paragraphs, 6
- of quotes, 6
- of tables, 6
- resources, 6, 23

SGML (Standard Generalized Markup Language)
- as related to HTML, 7

Shea, Dave (author), 25
- *Zen of CSS Design, The*, 25

shell pages (for PBS.org)
- as related to Web standards, 4
- example code of DOCTYPE, 7
- example code of HTML declaration after DOCTYPE, 7
- following the CSS (Cascading Style Sheets) guidelines, 9
- following the XHTML coding rules, 8
- following image accessibility rules, 12
- passing Section 508 accessibility specs, 21
- passing WCAG (Web Content Accessibility Guidelines) Level A specs, 21
- using the correct DOCTYPE, 7

skip navigation. *see* navigation
Standard Generalized Markup Language. *see* SGML
`summary` attribute. *see* tables

T

`tabindex` attribute, 16
- example code, 16
- use with `input` fields (`<input>`), 16
- use with `select` fields (`<select>`), 16
- use with `textarea` fields (`<textarea>`), 16
- use with Web forms, 16

`table` tag (`<table>`). *see* tables
tables
- accessibility , 17-20
- as related to DOCTYPE, 7
- as related to Web forms, 15
- as related to Web standards, 4
- example code, 5, 6, 15, 17, 18, 19
- use of `caption` tag (`<caption>`), 20
- use of `scope` attribute, 19
- use of `summary` attribute, 20
- use of `tbody` tag (`<tbody>`), 17
- use of `tfoot` tag (`<tfoot>`), 17
- use of `th` tag or table data header tag (`<th>`), 18
- use of `thead` tag (`<thead>`), 17
- use with tabular data, 17
- why not to use in page structure, 17
- why not to use in Web forms, 15

`tbody` tag (`<tbody>`). *see* tables
`td` tag or table data tag (`<td>`). *see* tables
testing
- of accessibility, 21
- of CSS (Cascading Style Sheets) code
- of XHTML (Extensible Hypertext Markup Language) code, 7, 8

text
accessibility, 14
color issues, 14
resizing of, 14
use with keywords, 14
zooming of, 14
textarea fields (<textarea>), 16
example code, 16
use with tabindex attribute, 16
use with title attribute, 16
use with Web forms, 16
text browsers
as related to accessible navigation, 13
as related to alt attribute, 12
as related to Web forms, 15
tfoot tag (<tfoot>). *see* tables
th tag or table data header tag (<th>). *see* tables
thead tag (<thead>). *see* tables
title attribute
example code, 8, 12, 13, 15, 16
guidelines, 12
use with images, 12
use with input fields (<input>), 16
use with navigation or anchor links, 13
use with select fields (<select>), 16
use with textarea fields (<textarea>), 16
use with Web forms, 15, 16
tr tag or table row tag (<tr>). *see* tables

U

ul tag (<ul>). *see* list tags

V

validation
as related to Web standards, 4
as related to XHTML coding rules, 8
of CSS (Cascading Style Sheets) code
of XHTML (Extensible Hypertext Markup Language) code, 7, 8
Visicheck: Color Blind Simulator, 14
resources, 14, 22, 24

W

W3C (World Wide Web Consortium), 4, 7
WCAG (Web Content Accessibility Guidelines), 21
Level A, AA, and AAA specs, 21
Priority 1,2, and 3 requirements, 21
resources, 21, 22, 24
testing compliance with Cynthia Says, 21
World Wide Web Consortium (W3C). *see* W3C
Web browsers
as related to accessible navigation, 13
as related to accessible tables, 18
as related to CSS (Cascading Style Sheets), 9-10
as related to screen readers, 4, 7, 12, 13
as related to text browsers, 12, 13, 15
as related to Web standards, 4
backwards compatibility with XHTML, 7
compatibility with XHTML 1.0 Transitional DOCTYPE, 7
CSS compatibility problems with IE (Internet Explorer), 9-10
plugins, 4
text zoom problem in IE/Win (Internet Explorer for Windows), 14
Web Content Accessibility Guidelines (WCAG). *see* WCAG
Web forms
accessibility, 15-16
as related to screen readers, 15, 16
as related to semantic markup, 15
as related to text browsers, 15
example code, 15, 16
form submission, 16
use of fieldset tag (<fieldset>), 15
use of input fields (<input>), 15
use of label tag (<label>), 15
use of legend tag (<legend>), 15
use of select fields (<select>), 16
use of tabindex attribute, 16
use of textarea fields (<textarea>), 16
use of title attribute, 16
Web standards, 4
advantages, 4
as related to accessibility, 3, 4
as related to CSS (Cascading Style Sheets), 3, 4
as related to Flash, 4
as related to screen readers, 4
as related to search engines, 4
as related to semantic markup, 3, 4
as related to shell pages, 4
as related to validation, 4
as related to Web browsers, 4
as related to XHTML (Extensible Hypertext Markup Language), 3, 4
benefits, 4
books, 25
definition, 4
introduction to, 3
recommended books, 25
resources, 4, 23

X

XHTML (Extensible Hypertext Markup Language), 7
as related to CSS (Cascading Style Sheets), 3, 9
as related to DOCTYPE, 7
as related to HTML, 7
as related to JavaScript, 11
as related to semantic markup, 3, 5, 7
as related to title attribute, 12
as related to Web standards, 3, 4
as related to XML, 7
benefits, 7
books, 25
coding rules, 8
recommended books, 25
resources, 8, 23
XHTML 1.0 Transitional. *See* DOCTYPE
XML (Extensible Markup Language)
as related to XHTML, 7

Z

Zeldman, Jeffrey (author), 25
Designing with Web Standards, 25
Zen of CSS Design, The, 25