

Designing Accessibility Into Themes

Before You Begin

- **Do some research.** If you don't understand what accessibility is, or how people with disabilities use computers, take some time to do some research. <http://www.uiaccess.com/understanding.html> is a good place to start, and some videos of people with disabilities using the web can be found at http://www.youtube.com/view_play_list?p=09C317E35FC6D005.
- **Ask an expert.** If you need help, ask an expert. If you don't know any, you can find lots of helpful and knowledgeable people at <http://www.accessifyforum.com>.
- **Accessibility isn't just about sight loss.** Although we usually associate accessibility issues with people who have sight loss, many other people encounter difficulties when trying to use websites. The needs of those with hearing impairments, mobility difficulties, and cognitive impairments or any combination of the above are just as important.
- **Aim for a good experience regardless of ability or technology.** Listening to a site is a very different experience from seeing a site but, with careful implementation, both can be a good experience. Where a drag and drop interface might be great for those who can use a mouse, the equivalent experience for keyboard only users would be something equally easy to understand and use.
- **Include accessibility in the design.** Accessibility is better for users, easier to implement, and less likely to have an adverse impact on the visual design if it is included in the design process rather than being implemented at the end.

Structure and Markup

- **Use appropriate markup.** Lists are lists, headings are headings, and quotations deserve appropriate markup, too. Paying careful attention to web standards automatically increases the accessibility of any page we create.
- **Avoid using structural markup for presentation.** The H1-H6 elements are for providing structure not altering text size and `blockquote` elements are for distinguishing quotations not indenting text.
- **Avoid using unconventional markup.** At best, unconventional markup can be confusing for users, at worst, it can make pages difficult or impossible to use. For example, don't use form elements instead of lists for navigation, and don't use links instead of buttons for form submission.
- **Use the `<title>` element effectively.** Putting the information in the form you'd use for a reverse breadcrumb trail (e.g., [page] - [section] - [site name]) places unique information first and means less repetitive reading for users.
- **Specify the natural language of the document.** This ensures that browsers display characters properly and screen readers use the correct pronunciation rules. For English in HTML add `lang="en"` to your `<html>` element. For XHTML1.0, change your `<html>` element to `<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">`. Language codes can be found at http://www.loc.gov/standards/iso639-2/php/English_list.php.

- **Specify any changes in the natural language.** If you change language midway through a page and don't markup the change correctly, the screen reader will read the second language as if it was the first. The `lang` attribute can be added to any element. The `span` element can be used within a containing element or a `div` can be used if the change includes multiple elements. For example `<p lang="fr">Bonjour</p>`, `<p>Hasta la vista, baby</p>` or `<div lang="la"><p>Lorem ipsum...</p><p>Etiam in risus ipsum...</p></div>`.
- **Use lists for navigation.** This allows screen reader users to build a mental model of the size of the site/navigation without having to listen to and count each item in the menu.
- **Use the most appropriate source code order for the type of site.** Designing a page structure based on user needs allows us to provide a better experience for our users. Putting navigation before content works well for sites where users browse before reading, but putting the content first may be more appropriate for blogs and other content-rich sites where readers are more likely to read first and then browse for other content.
- **Use skip links to provide keyboard shortcuts to content or navigation.** Ideally, skip links should be visible by default, because they are especially useful to those who have mobility difficulties or do not use specialized accessibility software. However, having them appear when the links receive focus is acceptable, if they are placed sensibly. To be most useful, they should be kept to a minimum (no more than three) and be the first links in the source code order. The link text should ideally describe the destination, for example "Skip to main content" or "Skip to main navigation," rather than "Skip navigation."
- **Create a logical page structure.** Check the page without CSS. It should still have a logical structure and be easy to navigate and understand.
- **Use headings appropriately, including section headings if necessary.** `<h1>` is used for main heading of each page (which is not always the site name). Any subheadings (including section headings) of `<h1>` are `<h2>`, and so on. Using this type of heading structure allows screen reader users to create a mental model of the page, quickly determine if it contains the information they seek, and then navigate directly to the section they want.
- **Write good link text.** Link text should contain all important information, be concise, be unique, and make sense when read out of context. "Terms and Conditions (PDF 30kb)" is good. "Click here" is not.
- **Notify users of popups and new windows.** In general, use of pop-ups and new windows should be avoided in favor of allowing users to choose the desired behavior. However, if the use of a popup or a new window is unavoidable, the user should be notified within the link text. This can be done by including the words "(new window)" or "(pop-up)" in the link text or inserting an appropriate icon and giving it appropriate `alt` text.
- **Avoid CAPTCHAs if at all possible.** If their use is unavoidable, provide an audio alternative (e.g., <http://recaptcha.net/>) and a mechanism for users to get timely support if they need it, for example, an email address or telephone number.

Data Tables

- **Specify table headers.** Row and column headings should be marked up using the `<th>` element and `scope="row"` or `scope="col"` attributes as appropriate.
- **Associate headers with cells.** If more than one level of heading is required, give each header cell a unique `id` (e.g., `<th id="id1">`) and use the `headers` attribute to explicitly associate the cell and its headers. Each heading `id` should be referenced, in order, within the `headers` attribute, for example `<td headers="id1 id2">`. This ensures that the headers are read out to screen reader users in the correct order.
- **Describe the purpose of the table.** Use the `caption` element to provide a visual description which concisely describes what information is being provided. For example, using "June 2009" for a calendar.
- **Describe the structure of the data.** Use the `summary` element to provide a concise description of the data structure of the table. Using the calendar example, the summary could be "Columns contain days of the week from left to right, starting on Sunday."

Forms

- **Place important information/instructions at the top of the form.** Providing instructions first reduces user error and provides a better user experience.
- **Always indicate required fields.** In addition, inform users at the beginning of the form how required fields will be indicated. The convention is to use a red asterisk, but this can also be provided by a small icon accompanied by `alt` text that says "required".
- **Ensure all form fields have a `label` and explicitly associate it using the `for` attribute.** Explicit association allows screen readers to announce the correct label, and moves the focus to the associated element when the `label` text is clicked. If there is absolutely no room for a visible label, the `label` may be moved offscreen using `{position: absolute; top: -999em}` in CSS, or the `title` attribute used instead.
- **Place labels close to their associated element.** Too much white space between label and element can mean that screen magnification users cannot see the label along with the field, which may lead to increased errors when completing the form.
- **Include the required field indicator within the `label` element.** If this is placed after or below the form field, it may be missed by those who use screen reader or screen magnification software.
- **Place help text between the `label` and the form field.** If guidance is needed to enable users to complete the field correctly, it should be provided before the relevant field, not after.
- **Allow users to hide/show help text.** This allows users to control how much information they are presented with and cuts down on visual or auditory "clutter" for frequent users.
- **Group form controls and label them appropriately.** Use a `fieldset` with an appropriate `legend` to provide context for groups of radio buttons or checkboxes.

- **Write legend text with care.** Be aware when writing legend text that screen readers read the legend and label text for each item enclosed within the fieldset. Try to ensure that each label makes sense when read together with the legend, as well as on its own. A great example of this can be found at <http://uk.tv.yahoo.com/> where a fieldset legend of “Search” combines well with radio button labels of “the web”, “for images”, “for video”, and “for audio”.
- **Segment lengthy forms.** When working with lengthy forms, consider splitting them into multiple pages or using headings in addition to fieldsets with legends. In this case, headings can be worded slightly differently to the legend to minimise repetition and hidden offscreen using CSS.
- **Summarize errors at the top of the form.** This provides an overview of all action required to complete the form correctly. It is also beneficial to provide anchor links to fields which require attention.
- **Highlight fields with errors clearly.** This could be accomplished by using an icon at the beginning of the label with alt text of “error: [concise description of error]”.
- **Allow users to confirm or undo important actions.** This helps increase user confidence and reduce errors.

Interactivity, JavaScript and Rich Internet Applications

- **Build a solid base.** Design the base experience to be a great one before jumping into dynamic interactions. This ensures that the users who cannot access or use the rich experience do not feel deprived.
- **Be aware that screen readers parse JavaScript.** Many modern screen readers have some level of support for JavaScript and care should be taken to ensure that its use does not cause pages to become less accessible.
- **Be careful when implementing functionality that changes areas of the page without refreshing the page, such as AJAX.** Screen reader software works by taking a virtual copy of the page, which it then interrogates. If the page changes without a refresh (or some indication that the user should refresh their virtual copy) users may be unaware that content has changed.
- **Research best practice techniques before implementing rich interactions and functionality.** As this is a relatively new area, techniques are continually being developed and improved and support for rich interactions is increasing. The Paciello Group is doing great work in this area and documenting it at <http://www.paciellogroup.com/blog/>.
- **Understand the impact of the desired functionality.** It is better to exercise caution when considering the use of new techniques or functionality. If, following research, it is not possible to implement the desired functionality in an accessible manner, consider using an alternative until an accessible solution is found.

Color

- **Specify background with text color.** When specifying any text color, be sure to also specify a default body background color.

- **Check color combinations for sufficient contrast.** Check all background/text color combinations to ensure that they meet WCAG2.0 guidelines <http://www.w3.org/TR/WCAG20/#visual-audio-contrast>. A great tool for this is the The Paciello Group Color Contrast Analyzer (available from <http://www.paciellogroup.com/resources/contrast-analyser.html>).
- **Check that readability is maintained when using patterned backgrounds.** Ensure that text maintains a suitable contrast ratio against any textured or patterned background. This can be done by using a strip of solid colour between the text and the background pattern to separate the text from the background and maintain readability.
- **Don't rely on color alone to provide meaning or understanding.** For example, don't refer to colored text or change the text to red to indicate an error. Provide some other method such as underlining, adding a border or outline, or adding an icon with appropriate `alt` text.

Images

- **Provide `alt` attributes for every image.** Every image placed in HTML **must** have an `alt` attribute which contains a concise, appropriate alternative.
 - For linked images, the `alt` text should describe the destination of the link.
 - For decorative images, the `alt` text should be null (`alt=""`) or empty (`alt=" "`) which lets assistive technology know that the image can safely be ignored. Alternatively, these images can be presented as CSS background images.
 - For images of text, the `alt` text should duplicate the text within the image.
 - For images which contain information, or are important content, the `alt` text should be a concise description of the purpose of the image. A good way to identify this kind of image is to imagine reading the page to someone over the telephone. If the image is not important enough to be described, it can be considered decorative and treated appropriately.
 - If the image is a graph or diagram, the `alt` text should be a concise description of what the graph is (e.g., "Sales figures for June 2009",) accompanied by a longer description or alternative displayed adjacent to the image or provided on a separate but clearly linked page.
- **Avoid using CSS to display images which contain information.** It is better to treat the image as content, display it using the `img` element and provide an appropriate alternative.
- **Ensure text contrast is maintained if a background image is not displayed.** Specify a background colour in addition to the background image to avoid text becoming unreadable if the background image is not displayed for any reason.
- **Use images to enhance understanding.** This can help those whose main language is not the same as the language used on the site, and also those who have difficulty understanding text. For example, a question mark icon could be added next to the word "Help".

- **Combine adjacent image and text links.** Combining adjacent links where an image and text have the same destination reduces repetition for screen reader users. Ensure the image `alt` and link text make sense when read together, leaving the image `alt` null if the alt text would otherwise duplicate the adjacent link text. A link to a downloadable document could be written as `Annual Report 2009 400kb)` or an email link could be written as `email us`.

Typography

- **Use relative units.** Base text sizes from a default of at least 100% or 1em. If absolute units must be used, provide a separate stylesheet for Internet Explorer that specifies sizes in percentages or ems to ensure those users can resize text if they need to.
- **Maintain readability when text size is decreased as well as increased.** In general, avoid specifying text smaller than 75% or 0.75em, but if smaller text is required, ensure it remains readable at two steps smaller than default. This helps users with tunnel vision who may reduce the text size in order to reduce eye strain when reading on screen.
- **Consider making small text bold.** This can improve contrast and readability.
- **Increase line-height to improve readability.** Using a line-height of slightly larger than normal (e.g., 1.2 to 1.6) can prevent ascending and descending characters from “crashing together” making text easier to read for everyone, but those with dyslexia or other reading impairments in particular.
- **Avoid fully justified text.** This can create “rivers” of whitespace and cause significant difficulty to dyslexic users.
- **Use appropriate case.** Screen readers use different pronunciation rules for text in ALL-CAPS. For example, “CONTACT US” would be pronounced “contact U S” rather than “contact us”, which could cause confusion. Avoid this by using the appropriate case in the source and using `text-transform` in CSS to change the visual display if required.
- **Limit use of images of text.** If images of text or image replacement techniques are required, limit their use to only the main heading of a page. This helps users who require specific colour schemes, preventing them from having to make further changes to their browser settings in order to read the page.
- **Make images of text clear and easy to read.** If images of text are unavoidable, ensure that the text size is reasonably large (above 18px) and has good contrast. This helps people who use screen magnification software, as small images of text will pixelate when magnified and may become impossible to read.

Orientation and Way-finding

- **Provide focus states as well as hover states for links.** Ensure that any changes which happen on hover also occur on focus to help keyboard only users identify the focused element.
- **Identify the current location in navigation.** Using a different visual style to identify the current section or page within navigation can help users to quickly orient themselves within the site.

- **Do not remove the default browser focus outline.** The focus outline is vitally important to users who have mobility difficulties and navigate through the site without using a mouse. If the default focus outline seems insufficient, it can be enhanced using CSS.
- **Do not move links out of the viewport using CSS.** If links are moved offscreen, keyboard-only users can become disorientated as the focus indicator will disappear from view until the user has moved to a focusable element within the viewport.
- **Make links easy to identify.** Make links clear and easy to distinguish from emphasized or body text.
- **Make clickable targets bigger.** For example, using `display: block` within a navigation menu to allow users to click the area around the text or making clickable buttons or links larger. In addition to attracting more attention from all users, this particularly helps users who have impaired motor skills.

Testing

- **Validate your HTML and CSS.** Invalid code isn't necessarily inaccessible, but may cause the page/site to behave unpredictably, which may in turn cause problems for assistive technology.
- **Check color combinations.** Use a tool which tests against the Luminosity algorithm and test for color-blindness issues using an appropriate tool or simulator.
- **Ensure fallbacks are in place.** Check that all important functionality works without javascript, CSS and images.
- **Unplug your mouse.** Before trying out a screen reader, unplug your mouse and try using the site/theme with keyboard only. If you can use the site comfortably without using your mouse, you're doing very well.
- **Use an automated testing tool.** Tools like WAVE (<http://wave.webaim.org/>) can help you identify coding errors and suggest areas which may require further manual assessment.
- **Get an expert review.** If you don't feel that you have the experience or knowledge to fully test the site, get an expert to review the site.
- **Do some user testing.** If at all possible, ask some people with disabilities to test your site/theme and observe them while they're using it.