XXX company is committed to providing our enterprise applications in a manner that is accessible to all individuals. To help meet our goal of universal design, we recommend that our development teams learn about and apply internationally recognized best practices documented in [Section 508 of the Rehabilitation Act](https://www.section508.gov/content/learn/laws-and-policies) and the [Web Content Accessibility Guidelines (WCAG)](https://www.w3.org/WAI/intro/wcag.php) Level AA to the extent possible on current and future projects.  We firmly believe in the spirit of these standards and their goals to provide accessibility to all individuals, **however** we do not invest in a process to certify or validate our products against those standards today, because it is difficult to validate compliance in the variety of technologies our products touch.

Why is Accessibility Important?

1. **It’s the right thing to do**Nearly 1 in 5 people in the United States have a disability and the World Health Organization estimates that 15% of the world's population has some kind of a disability. Access for all users is a moral imperative and we at XXX company can set the standard in our industry. If you think your product can’t possibly be used by an individual with a disability, think again. Accessibility inquiries and modification have come from some of our largest customers.­
2. **Increase the customer base of our products**All federal agencies must comply with Section 508. This means they can NOT purchase a product if it does not comply with the guidelines.
3. **Lower litigation and employee discrimination risk**Public facing and employee facing web content are both covered under the Americans with Disabilities Act (ADA title III and ADA title I). We lower the risk of costly settlements in the future if we develop correctly now.

Types of Disabilities

To create an inclusive experience for our customers and employees, it is helpful to understand the ways in which different disability types need to access web content and what we can do to make our products and content more accessible to them. There are four key areas of disability to consider: visual, hearing, motor, and cognition.

**Visual**

Users with vision loss rely on screen enlargement, keyboard-only navigation, and/or the use of screen reader technology.  Access to information via these means is dependent on: sizable fonts, good color contrast, and well-structured websites that label all graphics, icons, buttons, and multimedia, using web standards for coding tables, forms, and frames.

**Hearing**

Users with auditory disabilities need captions for multimedia content (any video content that also has audio) and transcripts for audio-only content. Without captions or transcripts, only the visual content is accessible.

**Motor**

Users with motor impairments are more likely to use only a mouse, only a keyboard, and voice or other inputs to control and navigate the web. Websites developed with flexibility of input options are more accessible to these individuals.

**Cognition**

User with cognitive impairment need a consistent navigation structure. Overly complex presentation can be confusing. Flickering or strobing designs can be disorienting and in some cases these designs can cause seizures.

Reference W3C: <https://www.w3.org/WAI/intro/people-use-web/>

Learn More About Accessibility

[WebAIM’s Introduction to Accessibility](http://webaim.org/intro/) is a great place to start learning. WebAIM is a part of the Center for Persons with Disabilities At Utah State University and is one of the largest accessibility resources on the web.

To learn more about ADA compliance, see [Is Your Website ADA Compliant? How to Check and How It Affects Your Business](http://www.business.com/web-design/is-your-website-ada-compliant-how-to-check-and-how-it-affects-your-business/) from [business.com](http://business.com/).

Accessibility Best Practices

All roles at XXX company should be familiar with design practices that address accessibility. The links below provide access to best practices, related tools, and training. Review the general content design best practices, and then locate and read the best practices specific to your role.

* General Content Design
* Designers
  + Tools and Training for Designers
* Developers
* Product Owners & Project Managers
* QA

General Content Design

Anyone involved with creating content such as editors, copy writers, marketing specialists, and UX designers should be aware of and apply these general content design best practices.

**Best Practices**

1. **Organize your content into groups with headings**Pages should be structured in a hierarchical manner, generally with one 1st degree headings (<h1>) being the most important (usually page titles or main content heading), then 2nd degree headings (<h2> - usually major section headings), down to 3rd degree headings (sub-sections of the <h2>), and so on. When using a screen reader, these headings aren’t just for show. Users jump between them and use them to scan the content. That makes it important to try to use them correctly. Plan your work by ensuring your content fits into a logical structure. Keep headings (H1, H2, H3, etc…), lists (ordered and unordered) and table headers for your data in mind when deciding how to present your content.
2. **Write clear and simple content**Assume that your audience is intelligent but don't assume they are subject matter experts. Be as consistent and clear as possible and avoid jargon. Consistency is key in making your content usable and accessible for all users. For example, if using “Modify” in one area of your content, don’t call it “Edit” in another section. Follow established content and design patterns. According to the U.S. Department of Education and the National Institute of Literacy, 32 million adults in the U.S. can’t read and 14% of the population has below average literacy skills. In Canada the figure is 42%. When working on your content you must take into account your primary audience, including those who may have difficulty with some of the content.
3. **Do not convey important information through images, color, or sensory characteristics alone**You should not rely solely on images, shape, size, visual location, orientation, or sound to indicate important instructions for operating or understanding content (ex. “See the image above”). Instead, use a combination of positioning, color, and labeling to identify content.
4. **Create smart, relevant page titles**If you are creating a web document be sure to create good page titles. The page title (<title>) is announced as soon as a new web document is loaded, so it is your opportunity to provide a succinct summary of the page. The title is displayed in search results and is an important way for users to assess the relevance of a page.
5. **Provide text alternatives for images**You can only get meaning from an image if you can see it. Any time you’re adding an image consider what it means to the user. Is it purely ornamental? If it’s a hamburger icon that opens a menu or an arrow that sends you to the next page, then you need text to back it up. Non-sighted users often make use of a talking browser to “read” the web. These specialized browsers convert text to speech so that a user can hear the words on a site. When a talking browser lands on an image, it looks for alt text that it can read aloud; if it finds none, it will often just say “image,” leaving the user in the dark as to what the image is and how it matters to the story. Describe all of the elements that explain what’s happening in the image, rather than just setting the alt text to be something like, “photograph”. If you have to use an image of text, be sure to describe the design if relevant, as well as all of the words in the image. (Ex: Whiteboard drawing of the quote “This is XXX company”)
6. **Use links to navigate**Links point to resources. You can link to a location in the same document or on another document; but if you’re sending somebody somewhere else, you should be using a link rather than a button. Every link should make sense if the link text is read by itself. Screen reader users may navigate through their page reading only the links so certain phrases like "click here" and "more" must be avoided. Make sure your links are recognizable by differentiating them with underlines or something other than color alone.
7. **Use buttons for actions**A button triggers an action. For example, a button could save, close, open a panel, send a message, install, finish, print, confirm, etc... If you’re activating a behavior, use a button. Elements such as next, proceed, continue can be buttons when they are in a part of process in collecting information entered by the user. Eg: Filling up a form in 5 steps, making an online transaction.
8. **Use Animation, video, and audio carefully**If you use multimedia provide a pause button. Avoid flashing or strobing content. Videos and live audio should have captions and a transcript. With archived audio, a transcription may be sufficient.
9. **Provide good color contrast**The Web Content Accessibility Guidelines [recommends a color contrast of 4.5:1](http://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html) to meet color contrast guidelines. You can use various color contrast checkers online to verify your color contrast ratio. [Colour Contrast Analyser](http://www.paciellogroup.com/resources/contrastanalyser/) and [Colour Contrast Check Tool](http://snook.ca/technical/colour_contrast/colour.html#fg=33FF33,bg=333333) are examples of tools available online.

**Tools and training for all**

* [W3C’s Introduction to Web Accessibility](https://www.w3.org/WAI/intro/accessibility.php)
* [How to Create More Accessible Content](https://www.viget.com/articles/how-to-create-more-accessible-content-part-1)
* [Accessibility Quick Reference](http://webaim.org/resources/quickref/)
* [Writing Clearly and Simply](http://webaim.org/techniques/writing/)
* [The Basics of Good Alt Text](http://webaim.org/techniques/alttext/#basics)
* [Creating Accessible Google Docs](http://webaccess.msu.edu/Tutorials/google-drive.html)
* [Creating Accessible PDFs](http://webaim.org/techniques/acrobat/)
* [Powerpoint Accessibility](http://webaim.org/techniques/powerpoint/)
* [Accessible Word Documents](http://webaim.org/techniques/word/)
* [Creating Accessible Videos](http://a11yproject.com/posts/using-caption-services-with-html5-video/)

Designers

Accessibility starts with UX. Accessibility requirements should be considered and clarified before designs are handed over for coding.

**Best Practices**

1. **Understand the** [**General Content Best Practices**](https://confluence.jda.com/pages/viewpage.action?pageId=233354857#AccessibilityatJDA-General)This section highlights important principles of accessible design.
2. **Design accessible forms**Labels should be visible unless the purpose of the form is totally clear; and must be associated with the input field. Provide instructions with your forms and pay close attention to form validation errors and recovery mechanisms.
3. **Consider reading order**The reading order of things should be the same as the visual order. People who navigate by keyboard (e.g., using the Tab key) expect to move sequentially from left to right and top to bottom through the focusable elements on the page. Annotate your wireframes to provide direction on tab order.
4. **Don’t depend on color alone**[Approximately 4.5% of the world's population](https://nei.nih.gov/health/color_blindness/facts_about) are color blind to some degree. In order to ensure that someone with color blindness can successfully use your content, you cannot rely completely on color alone to communicate information. By extension, using colors or patterns alone to indicate something will not be accessible to a blind users. In instances where you are using color as an indicator (for instance, green for something active and grey for something inactive), include a text equivalent description for the state.
5. **Watch the use of CAPS**All uppercase letters can be difficult to read and can be read incorrectly by screen readers.
6. **Ensure link text makes sense out of context**Avoid ambiguous links such as “more”, “click here” and “continue”.
7. **Design link focus indicators**A link focus indicator is similar to a mouseover effect. Unlike mouse users who point directly to UI elements, users with mobility/dexterity disabilities need a clear on-screen indication of a focused link so they know where they are on a web page as they navigate with the keyboard from element to element. Otherwise, they have limited ability to perform tasks such as navigate form elements, follow links, and select buttons.
8. **Annotating wireframes and UX designs**How you convey accessibility considerations to product owners, developers, and others in your team will depend on how you work.  A best practice technique for this communication is  to annotate wireframes or UX designs and documents. It should always be clear what is annotation and what is design.  You could combine a non-pallette color with lines, arrows, numbering, overlap, or other techniques.
9. **User testing**Even knowledgeable, well-advised designers may not get the best solution straight away. Don’t make assumptions or hope things are OK. Always test your ideas and concepts with a broad range of people. This should include those with at least moderate cognitive, motor, hearing or vision impairments. Design researchers will be able to advise and help with testing.
10. **As part of your Definition of Done, ensure the following as you complete your design review:**
    1. You are able to navigate through the page or process using only your keyboard & the tab order is logical
    2. Each page has a succinct title
    3. There is a hierarchical heading structure
    4. Font sizes can be increase
    5. Videos have captions and audio has a transcript
    6. Focusable elements should have a clearly identifiable visual style when they have focus
    7. Changes to the page take place only when initiated by the user
    8. Features that scroll or update automatically have  controls which enable users to pause or advance
    9. Images have alt attributes
    10. Form fields have labels
    11. Tables have a caption and header rows

**Tools and Training for Designers**

* [Designing for disability](https://www.ted.com/playlists/372/designing_for_disability) Playlist (10 talks) from [ted.com](http://ted.com/)
* [Accessibility by Design](http://laurakalbag.com/accessibility-by-design/)
* [7 Things Every Designer Should Know About Accessibility](https://medium.com/salesforce-ux/7-things-every-designer-needs-to-know-about-accessibility-64f105f0881b#.vg1ptgupb)
* [Usability Testing By People With Disabilities: Guerilla Tactics](http://uxpamagazine.org/guerilla_tactics/)
* Firefox Extensions:
* [Web Developer Toolbar](http://chrispederick.com/work/web-developer)
* [Accessibility Extension](http://firefox.cita.uiuc.edu/index.php)
* [FANGS](http://www.standards-schmandards.com/projects/fangs/) this extension creates a textual representation of a web page similar to how the page would be read by a modern screen reader.
* [Color Contrast Analyzer](http://juicystudio.com/article/colour-contrast-analyser-firefox-extension.php)
* [OAA Accessibility Extension for Firefox](https://addons.mozilla.org/En-us/firefox/addon/openajax-accessibility-exte/): browser add-on that can test Web pages for compliance for compliance with the WCAG 2.0 guidelines.
* [Accessibility Checklist](http://webaim.org/standards/wcag/WCAG2Checklist.pdf)
* [Teach Access Tutorial](https://teachaccess.github.io/tutorial/#/0)
* You can examine the way that assistive technologies see web content by using [Accessibility Inspector](https://developer.apple.com/library/mac/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXTestingApps.html) (Mac), or [Windows Automation API Testing Tools](http://msdn.microsoft.com/en-us/library/windows/desktop/dd373661%28v=vs.85%29.aspx) and [AccProbe](http://accessibility.linuxfoundation.org/a11yweb/util/accprobe/) (Windows). Additionally you can see the full accessibility tree that Chrome creates by navigating to chrome://accessibility.

Developers

These guidelines are based on global industry web standards created by the [W3C Web Accessibility Initiative](http://www.w3c.org/WAI) and have been adapted from the BBC Accessibility Standards and Guidelines under an [Open Government License for Public Sector Information](http://www.nationalarchives.gov.uk/doc/open-government-licence/version/2/). Be sure to read the General Content Design section to understand basic accessibility concepts.

**Best practices**

1. **Use patterns consistently**   
   Patterns help everyone understand information.
2. **Apply consistent semantic markup**   
   Write what content IS vs what it looks like, and be consistent. If you can use a native HTML element or attribute with the semantics and behaviour you require already built in, instead of repurposing an element and adding an ARIA role, state or property to make it accessible, then do so. For example, a button is a <button> not a <div> with a button style.
3. **Separate content, presentation and behavior.**   
   Use each web technology for its intended purpose. Use XHTML or HTML for marking up what content is and use CSS for controlling the visual presentation and JavaScript for controlling the behavior of the content. Keeping each of these separated is what enables progressive enhancement.

**HTML Recommendations**

1. **Validation**Designate the [W3C recommended Doctype](https://www.w3.org/QA/2002/04/valid-dtd-list.html)
   1. **Why**While assistive technologies such as screen readers generally do a good job of interpreting invalid HTML, there will be less risk of problems arising if the document follows a recognized standard doctype. Omitting the doctype declaration can result in unexpected and broken behavior for users. Without a doctype declared, the browser doesn’t know how to interpret the content and can regress into a non-compliant and incompatible mode, often called quirks mode.
   2. **How**<!DOCTYPE html>
2. **Language**Specify the main [language](https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes) of your pages.
   1. **Why**This helps screen readers with appropriate pronunciation of the language. For instance, lang="zh-Hans” invokes the rendering of a simplified Chinese font.
   2. **How**<html lang="en-GB">
3. **Page titles**Documents must have a page <title> that identifies its main content.
   1. **Why**Document titles help users orientate themselves within web sites and apps. The document <title> element content is often the first thing a speech output user will hear and acts as a confirmation of what page they have arrived at. Document titles commonly have the same content as the main <h1> element. Use meaningful text that is informative but not too long. Place the name of the specific page you are working with first in the title and then put the site or company name (when applicable). This will reduce repetition as the user navigates. They can just listen to the start of the title being read, then skip over the rest and get straight to the content of the page. This also makes it easier for sighted users too, as the amount of space in each browser tab or block on the taskbar is very limited.
   2. **How**<title>BBC Weather</title>
4. **Headings**A logical heading structure is invaluable for people using assistive technologies to help navigate content. Each page should contain an H1 heading. The H1 should identify the main content for a document. Web pages should follow a hierarchical heading structure, without skipping levels.
   1. **Why**When headings appear out of order, it can be disorienting for the person who cannot see your page.
   2. **How**<H1>Warehouse Configuration</H1>  
       …  
       <H2>Additional Configurations</H2>
5. **Test font sizes**Make sure users can enlarge their text in the browser. [WCAG guidelines recommend allowing zooming up to 200% without loss of content.](http://www.w3.org/TR/2008/REC-WCAG20-20081211/#visual-audio-contrast-scale)
   1. **Why**For people with low vision, it is important to support the ability to increase text size without text overlapping or getting truncated.
   2. **How**In the CSS, use em or rem for font sizes rather than px. Define the font size for the body tag in %.  
      (The px values assume that the root font-size is 16px.)   
      .content { font-size: 1.25em; /\* 16px × 1.25 = 20px \*/ }   
      .content h1 { font-size: 1.25em; /\* 16px × 1.25 × 1.25 = 25px \*/ }
6. **Tabindex**The tab order and read order must be logical and intuitive.
   1. **Why**Users who are navigating by keyboard (e.g., using the tab key) expect to move sequentially from left to right and top to bottom through the focusable elements on the page.
   2. **How**When creating web pages be sure the order of items in the source code matches the visual order by both designer and style guide.
7. **Title attribute**Title attributes must not be used for critical information and must not repeat content that is already visible and associated with the same control or content.
   1. **Why**Title attributes are inaccessible to keyboard users without additional Assistive Technology. They are dependent on user settings in Screen Readers and similar Assistive Technology.
   2. **How**
      1. Do not use the title attribute unless on a form input as title text is not well supported on links on mobile
      2. Do not use title attributes and explicit labels together on form fields
8. **Focusable controls**Create features that scroll or update automatically (e.g., slideshows, carousels) that have prominent accessible controls which enable users to pause or advance these features on their own
   1. **Why**Features that scroll or update automatically present a variety of problems for users with disabilities
   2. **How**
      1. All controls should be operable with a keyboard
      2. Use a prominent means for pausing the display. [Read more at w3](https://www.w3.org/TR/WCAG20/#time-limits-pause).
9. **Control styles**Links must be self-evident, identifiable by their visual style, and distinguishable from the surrounding content. Text links should have a mouseover state change.
   1. **Why**To aid discoverability, all links must be made self-evident with their visual style with either an underline or a different colour, meeting contrast standards, to the surrounding text. When hovered over links should have a change in style as confirmation that they are links.
   2. **How**<style>  
         body {  
             background: #fff;  
             color: #000;  
         }  
        
         a {  
             color: #005580;  
             text-decoration: none;  
         }  
        
         a:focus,  
         a:hover {  
             background:   
             text-decoration: underline;  
         }  
      </style>
10. **Focus styles**Focusable elements should have a clearly identifiable visual style when they have focus.
    1. **Why**It is often difficult for keyboard users to tell where they are on the page. If links are not styled so their focus state is indicated, it’s impossible for the user to see which element is indeed in current focus.
    2. **How**Avoid overriding browsers’ default focus indicator with outline:none. In order to provide users with an easy-to-see focus indicator that is consistent across all browsers, use the :focus selector in CSS to define a style change that happens when an element has focus.  
       <style>  
          a {  
              text-decoration: none;  
          }  
          a:focus {  
              text-decoration: underline;  
          }  
       </style>
11. **Images**Every image must have an alt attribute, including font icons. Ensure that the page remains readable and functional when images are disabled.
    1. **Why**Images without an alt attribute are likely inaccessible. Some users set their machines to not load images. Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language. Read more on how to [make your font icons accessible](https://www.filamentgroup.com/lab/bulletproof_icon_fonts.html).
    2. **How**<img src=company-logo.png" alt="company">  
       Or you can provide context surrounding the image by way of a caption. For images that are decorative or are described in the surrounding text content it is appropriate to use a null (alt="") value for the image alt attribute. This will tell screen readers to skip that image.
12. **Media**Video should have captions and audio should have a transcript.
    1. **Why**Some people are unable to hear audio. Some people are unable to see video.
    2. **How**Captions: <track kind="captions" src="myvideo.vtt"/>  
       Transcripts: <track kind="transcript" title="English transcript" href="#theText">
13. **Form labels**Form fields that allow input (select, and textarea elements, and all input element types other than image, submit, reset, button, or hidden) must have an associated label, either in the form of a <label> element or, for simple forms when no visible label is required, a title attribute.
    1. **Why**If we put text on the screen near an input such as a text field, the screen reader cannot reliably pick up on that visual association. It needs to be told which nearby text is the associated label.
    2. **How**<label for="search">Search WM</label>  
       <input type="text" id="search" name="q" />
14. **Form interactions**Changes to the page location must only take place on explicit user action i.e. when a submit button is activated. They must not take place when change (particularly for select elements), focus, or blur events are fired. Forms should contain a button which provides a clear call to action.
    1. **Why**It is particularly important to users with cognitive disabilities, but is also beneficial to low vision users as an indication of the end of the form.
    2. **How**<form action="/search">  
          <label for="q">Search term:</label>  
          <input type="text" name="q" id="q">  
          <input type="submit" value="Search">  
       </form>
15. **Tables**Use semantic table markup to create accessible tables.
    1. **Why**When creating a data table, understand that sighted users can easily scan and understand the table by looking at a given data cell along with its row and column headers, and comparing that with other data cells. People who are unable to see the table do not have access to all the visual cues that make this possible. Therefore, they depend on the table author to have provided markup that explicitly defines the roles and relationships of all the parts of the table.
    2. **How**See the WebAIM article [Creating Accessible Tables](http://webaim.org/techniques/tables/data) for specific HTML techniques and examples.
       1. Use table captions
          1. **Why**The caption for a table is a table identifier and acts like a title or heading for the table.
          2. **How**<table>  
             <caption>Schedule for the week of March 6</caption>  
             ...</table>
       2. Identify all column headers and row headers
          1. **Why**TH elements, table header cells, can tie related pieces of related information in a table together
          2. **How**<table style="width:100%">  
             <tr>  
             <th>Firstname</th>  
             ..</table>
       3. Associate cells with headers via the [scope attribute](https://www.w3.org/TR/WCAG20-TECHS/H63.html)
          1. **Why**The scope attribute tells the browser and screen reader that everything within a column that is associated to the header with scope="col" in that column, and that a cell with scope="row" is a header for all cells in that row.
          2. **How**<table border="1">  
             <caption>Contact Information</caption>  
             <tr>  
              <td></td>  
                <th scope="col">Name</th>  
                <th scope="col">Phone#</th>  
                <th scope="col">Fax#</th>  
                <th scope="col">City</th>   
              </tr><tr>  
                <td>1.</td>  
                <th scope="row">Joel Garner</th>  
                <td>412-212-5421</td>  
                <td>412-212-5400</td>  
                <td>Pittsburgh</td>  
              </tr><tr>  
                <td>2.</td>  
                <th scope="row">Clive Lloyd</th>  
                <td>410-306-1420</td>  
                <td>410-306-5400</td>  
                <td>Baltimore</td>  
              </tr><tr>  
                <td>3.</td>  
                <th scope="row">Gordon Greenidge</th>  
                <td>281-564-6720</td>  
                <td>281-511-6600</td>  
                <td>Houston</td>  
              </tr>  
             </table>

ARIA Recommendations

A web page containing JavaScript will typically be fully accessible if the functionality of the script is device independent and the content is available to assistive technologies.

ARIA enhances accessibility of interactive controls (such as tree menus, drag and drop, sliders, sort controls, etc.), provides content roles for identifying page structure (navigation, search, main content, etc.), areas that can be dynamically updated (called "live regions" in ARIA), and better support for keyboard accessibility and interactivity.

1. **Landmark roles**Use the available [Document Landmark Roles provided by WAI-ARIA](https://www.w3.org/TR/wai-aria-1.1/#landmark_roles)
   1. **Why**One of the easiest [ARIA](https://www.w3.org/TR/wai-aria-practices/) features to implement, and one that provides significant immediate benefits to screen reader users, is landmark roles. A screen reader has no way of knowing what portion of the page contains each element. Accessing commonly used page functionality, such as search requires that the user browse through or listen to the page and find or discover it. Identifying landmark roles allows the user to jump to specific structural elements using shortcut keys.  This will allow users to skip the repetitive navigation read to them upon page load.
   2. **How (example)**<form role="search">  
      And   
      <div role="main" id="main-content">...</div>
2. **Live region**Identify regions that dynamically change as a [live region](https://www.w3.org/TR/wai-aria/states_and_properties#attrs_liveregions).
   1. **Why**A live region allows content updates in a way that a screen reader understands. It also allows you to add additional functionality to alert the user, provide controls for the live region, determine the amount of new content that would be read, and much more. What happens if a screen reader is currently reading an element that is updated? If the updated content is important, should you interrupt the user and set focus immediately to the new content, do you simply inform the user of the update, or do you do nothing? How do you set focus or allow the user to jump to the updated content? With WAI-ARIA, the developer can identify regions that dynamically change as a live region. The value of this property specifies what the screen reader should do when the element is updated.
      1. Off  
         A value of off results in no notification. This would be used for non-important or irrelevant content updates.
      2. Polite  
         A value of polite results in a screen reader notifying a user once current task is complete. This value would be the most common for content updates, especially for things like status notification, weather or stock updates, chat messages, etc.
      3. Assertive  
         An assertive value would result in the screen reader interrupting a current task to notify user. Assertive would be used for important updates, such as error messages
   2. **How**An example of filtering products…  
        
      <div aria-live="assertive" role="alert">  
      Please wait. Loading products.  
      </div>  
        
      When products are loaded…  
        
      <div aria-live="assertive" role="alert">  
      Loading complete. 23 products listed.  
      </div>
3. **Expand keyboard focus**   
   Set the [tabindex of interactive elements](https://www.w3.org/TR/2009/WD-wai-aria-practices-20090224/#focus_tabindex)
   1. **Why**In HTML, only links and form elements can receive keyboard focus. This means that as you 'tab' through a page, the browser stops or sets focus only on these types of elements. If a user receives an error when submitting a form, the form validation error message might be displayed as text (not a link or form element) within a page using scripting. Visual users can immediately see the error message. However, a screen reader user may not know that the new message is present.
   2. **How**By setting a tabindex value of 0 (tabindex="0"), any element will be placed in the tab order of the document. Read more on the [various values for tabindex](https://www.paciellogroup.com/blog/2014/08/using-the-tabindex-attribute/).   
        
      The following is an example of an error message:  
        
      <div role="group" id="errorSummary" aria-labelledby="errorSummaryHeading" tabindex="-1">  
      <h2 id="errorSummaryHeading">Your information contains three errors</h2>  
      <ul>  
      ...  
      </ul>  
      </div>

**Tools & Training**

* [Good Coding Habits for Accessibility](https://www.joedolson.com/2014/10/good-coding-habits-accessibility/)
* [Teach Access Tutorial](https://teachaccess.github.io/tutorial/#/0)
* [The A11Y Project](http://a11yproject.com/) A community-driven effort to make web accessibility easier.
* [Angular JS Accessibility Documentation](https://docs.angularjs.org/guide/accessibility)
* [Accessibility in AngularJS](https://docs.angularjs.org/api/ngAria)
* [Getting Started with ARIA](http://a11yproject.com/posts/getting-started-aria/)
* [W3C’s WAI-ARIA Authoring Practices](https://www.w3.org/TR/wai-aria-practices/)
* Firefox Extensions:
* [Web Developer Toolbar](http://chrispederick.com/work/web-developer)
* [Accessibility Extension](http://firefox.cita.uiuc.edu/index.php)
* [FANGS](http://www.standards-schmandards.com/projects/fangs/) this extension creates a textual representation of a web page similar to how the page would be read by a modern screen reader.
* [Color Contrast Analyzer](http://juicystudio.com/article/colour-contrast-analyser-firefox-extension.php)
* [OAA Accessibility Extension for Firefox](https://addons.mozilla.org/En-us/firefox/addon/openajax-accessibility-exte/): browser add-on that can test Web pages for compliance for compliance with the WCAG 2.0 guidelines.
* Chrome Extensions:
  + [Accessibility Developer Tools](https://chrome.google.com/webstore/detail/accessibility-developer-t/fpkknkljclfencbdbgkenhalefipecmb?hl=en)
  + [Wave Chrome Extension](http://wave.webaim.org/extension/): The WAVE Chrome extension allows you to evaluate web content for accessibility issues directly within Chrome. Because the extension runs entirely within your web browser, no information is sent to the WAVE server. This ensures 100% private and secure accessibility reporting. The extension can check intranet, password-protected, dynamically generated, or sensitive web pages. Also, because the WAVE Chrome extension evaluates the rendered version of your page, locally displayed styles and dynamically-generated content from scripts or AJAX can be evaluated.
* [HTML5 Accessibility](http://html5accessibility.com/) current accessibility support status of HTML5 features across major browsers.
* [Accessibility Checklist](http://romeo.elsevier.com/accessibility_checklist/) Accessibility Checklist provides an easy way to explore the most relevant guidelines using a simplified language framework and easy to explore user interface. Users can filter guidelines by topic such as images, keyboard, and forms. Users can also filter by standard levels such as A, AA, or AAA.
* [Accessibility Checklist](http://webaim.org/standards/wcag/WCAG2Checklist.pdf) Webaim
* [A Checker](http://achecker.ca/checker/index.php) is an online tool that can test Web pages for compliance with the WCAG 2.0 and Section 508 guidelines.
* [aXe](http://www.deque.com/products/axe/) — automated accessibility testing for your framework or browser of choice
* The [Accessibility DevTools extension](https://chrome.google.com/webstore/detail/accessibility-developer-t/fpkknkljclfencbdbgkenhalefipecmb?utm_source=chrome-ntp-icon) for Chrome provides a helpful audit for discovering accessibility issues, including issues within Shadow DOM. It’s powered by the [Accessibility DevTools module](http://bit.ly/a11y-devtools-module) and a [CLI](http://bit.ly/a11y-ci) is available that also uses this work for continuous integration audits.
* You can examine the way that assistive technologies see web content by using [Accessibility Inspector](https://developer.apple.com/library/mac/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXTestingApps.html) (Mac), or [Windows Automation API Testing Tools](http://msdn.microsoft.com/en-us/library/windows/desktop/dd373661%28v=vs.85%29.aspx) and [AccProbe](http://accessibility.linuxfoundation.org/a11yweb/util/accprobe/) (Windows). Additionally you can see the full accessibility tree that Chrome creates by navigating to chrome://accessibility.
* [tota11y](http://khan.github.io/tota11y/) is a useful visualiser for assistive technology issues built by Khan Academy. It’s a script that adds a button to your document that triggers several plugins for annotating things like insufficient contrast ratio and other a11y violations
* [ally.js](http://allyjs.io/) (by Rodney Rehm) is a library that tries to simplify adding a few accessibility features to your app. It helps query the DOM for all focusable or tabbable elements, traps focus to specific DOM sub-trees, helps determine how focus has changed and comes with several other helpers.
* On Windows, [NVDA](http://www.nvaccess.org/) is a free, open source screen reader which is fully featured and rapidly gaining in popularity. However, note that it has a much steeper learning curve for sighted users than VoiceOver.
* [Accessible Table Builder](http://www.accessify.com/tools-and-wizards/accessibility-tools/table-builder/)
* [Accessible Form Builder](http://www.accessify.com/tools-and-wizards/accessibility-tools/quick-form-builder/)
* [Creating Accessible Videos](http://a11yproject.com/posts/using-caption-services-with-html5-video/)
* [Coding Accessible Patterns](http://a11yproject.com/patterns/)
* [iOS Best Practices](https://www.webaccessibility.com/best_practices.php?technology_platform_id=222)
* [Android Best Practices](https://www.webaccessibility.com/best_practices.php?technology_platform_id=286)
* [Udacity Web Accessibility Course](https://www.udacity.com/course/web-accessibility--ud891)
* [More Tools!](https://www.w3.org/WAI/ER/tools/)

Product Owners & Project Managers

The tools and products that you create should make accessibility easier to achieve. Accessibility should be at the forefront of your mind when considering which projects to take on and how to shape the products.

**Best Practices**

* Familiarize yourself with the work associated with making content accessible by reading the recommendations for other roles
* Build in time for accessibility during project planning and sprint planning.
* When sharing good work done by your team, praise efforts to increase accessibility.
* Be an advocate for accessibility. Educate yourself on why accessibility is important in the tools we create.
* Add accessibility QA to your definition of done.

QA

Accessibility in our web applications is an important quality assurance issue.  Our software should be designed so it can be accessed by all. Our users should be able to operate our software with only a mouse, only a keyboard, and voice or other inputs to control and navigate. Familiarize yourself with both the General Content Design recommendations and Developer Recommendations.

**Best Practices**

* Run through your pages with the [Wave Chrome Extension](http://wave.webaim.org/extension/).
* Use only your keyboard to run through your page and ensure that you are able to make each operation work, and you move through the page in logical order.
* Use FANGS to test that users can navigate content using a screen reader.
* Verify that charts and images all have alt-text so that users with screen readers or users on a slow connection will still be able to understand the images.

**Tools**

* [Wave Chrome Extension](http://wave.webaim.org/extension/)
* [aXe](http://www.deque.com/products/axe/) — automated accessibility testing for your framework or browser of choice
* [OAA Accessibility Extension for Firefox](https://addons.mozilla.org/En-us/firefox/addon/openajax-accessibility-exte/): browser add-on that can test Web pages for compliance for compliance with the WCAG 2.0 guidelines.
* [FANGS](http://www.standards-schmandards.com/projects/fangs/) this extension creates a textual representation of a web page similar to how the page would be read by a modern screen reader.

Current Laws

**ADA**

[The Americans with Disabilities Act of 1990 (ADA)](https://www.ada.gov/2010_regs.htm) prohibits discrimination and ensures equal opportunity for persons with disabilities in employment, State and local government services, public accommodations, commercial facilities, and transportation.

**Section 508**

[Section 508](https://www.section508.gov/) is a U.S. accessibility standard that's mandated by law, and is part of the 1998 amendment to the rehabilitation act. Its rules apply to all federal agencies when they develop, procure, maintain or use electronic and information technology. The agencies must give disabled employees and members of the public access to information that is comparable to the access available to others. While Section 508 does not directly apply to our company, it provides technical standards for accessibility AND is applicable to our products and services being marketed to federal agencies.

**WCAG Standards**

The [Web Content Accessibility Guidelines (WCAG)](https://www.w3.org/WAI/intro/wcag.php) are part of a series of web accessibility guidelines published by the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C), the main international standards organization for the Internet.

[WCAG 2.0](https://www.w3.org/TR/WCAG20/) is made up of 12 guidelines. Each of these guidelines contains one or more checkpoints that you can use to evaluate the web accessibility of your site. Then these checkpoints have a priority assigned to them of A, AA or AAA. The conformance A checkpoints are ones that you must satisfy, otherwise one or more groups will find it impossible to access the information on your page. Conformance AA checkpoints are ones that you should satisfy, otherwise one or more groups will find it difficult to access the information. Conformance AAA checkpoints are ones that you may address, otherwise one or more groups will find it somewhat difficult to access the information.

Errors & Suggestions

If you notice an error or have a suggestion for a resource please contact XXX or enter it on our backlog link.