# Web Site and Application Accessibility and Usability Requirements Checklist

2/2017 Version 2.3

## Step 1: Plan the site

### Use Headings to logically structure content

* Use headings to logically divide and structure the site. Use h1 through h6 tags to ensure assistive technology (AT) can programmatically identify headings.
* Use headings 5 and 6 sparingly to reduce site complexity encourage understanding.
* Headings should be properly nested in a hierarchical structure (H1 followed by H2, H2 followed by another H2 or an H3, etc.).

### Page titles

Plan a unique title for each page. Titles must be concise, generally limited to 20-30 characters, and make the contents of the page clear in the context in which it is presented.

### Color and contrast

* Do not use color alone to convey information. If color is used to emphasize the importance of selected text, there must be an alternate method to emphasize.
* Check to ensure that all text is readable and distinguishable from background colors, watermarks, and background images, and that all text is readable in High Contrast mode.
* Avoid flashing/flickering text and/or animated text.
* Contrast ratio of text and images of text must be at least 4.5:1 and 7:1 is even better; contrast ratio of large-scale text and images of large-scale text must be at least 3:1 and 4.5:1 is even better. Application of color is handled by the site (Cascading Style Sheet) CSS and a user-defined CSS (if it exists).

### Formatting

All formatting should be handled using the CSS (Cascading Style Sheet).

Recommended fonts are part of the sans-serif font class and include Times New Roman, Verdana, Arial, Tahoma, Helvetica, or Calibri.

#### References

508: 1194.21(a); 1194.21(d); 1194.21(i); 1194.22(c); 1194.22(o); 1194.31(a); 1194.31(b); 1194.31(f).

WCAG: 1.3.1; 1.4.1; 1.4.3; 2.1.1; 2.4.6;

## Step 2: Perform manual tests

### Verify headings.

* Are headings 5 and 6 sparingly to reduce site complexity encourage understanding?
* Are headings properly nested in a hierarchical structure (H1 followed by H2, H2 followed by another H2 or an H3, etc.)?
* Headings do not provide formatting; that is done in the CSS.

### Verify images and non-text content have appropriate alt text

Verify all non-text content (pictures, images, clip art, shapes, SmartArt, charts) have alternative text descriptions that convey the same information to the user that the image or picture conveys.

#### References

508: 1194.21(a); 1194.21(d); 1194.21(f); 1194.22(a); 1194.22(i); 1194.22(l); 1194.22(n); 1194.31(a); 1194.31(b); 1194.31(f)

WCAG: 1.1.1-H37; 2.4.3

### Verify list formatting

* Ordered lists are used where order is important
* Unordered lists are used where list order is not important

#### References

508: 1194.21(a); 1194.21(d); 1194.31(a); 1194.31(b)

WCAG: 4.1.2; 3.2.4

### Verify links

* Make sure that links (and/or their associated title attributes) are labeled correctly and that the label displayed (may be in addition to the URL).
* Links should obviously be links
* Do all links provide enough information about where they lead?
* If a link opens a new window, is this announced in advance?

### Verify keyboard only functionality

Use the keyboard exclusively to navigate through webpages and Applications. All functionality must be available via keyboard only (i.e., without using a mouse). This does not forbid or discourage providing mouse input or other input methods in addition to keyboard operation.

* Start by using the tab key and see if the focus moves from one link to another.
* You should be able to see the focus displayed as a dotted line from link to link, or see the web URL in the lower left hand corner change as you press tab.
* Move to the beginning of a paragraph and use the arrow keys to move the cursor up and down. Verify that you can use the arrow keys to move to all text lines.
* Ensure that repeated actions (i.e. Next and Submit buttons) have consistent hotkeys throughout the application.
* Most common keystrokes used to navigate a page are tab, arrow keys and hot keys such as Ctrl, Alt, Shift and Enter. To activate a link once you tab to it, press enter.
* All functionality must be available via keyboard only (i.e., without using a mouse).
* Use either device independent event handlers (that work with both the mouse and the keyboard) or use both mouse dependent and keyboard dependent event handlers.
  + A button or element may be coded to activate on mouse-over (i.e., hovering over an object), if any important content or functionality is also available without using the mouse. For example, when using onMouseOver and onMouseOut, use onFocus and onBlur as well.
  + If the mouse interaction is purely cosmetic (such as the addition of a glow or drop shadow), there are likely no accessibility issues, so long as the style change does not indicate some function (such as to indicate that an element is clickable).

### Forms

Forms are used to enter data, search for content, make selections, fill out surveys and questionnaires, register for courses, and more.

* Check the following three things to make forms more accessible:
  + Use the <label> element to explicitly associate form controls and their descriptions. Make note of any fields that are not labeled.
  + Use the <fieldset> element to group related form elements. This usually includes, but is not limited to, checkboxes and radio buttons.
  + Avoid JavaScript jump menus.
* Tab through the form to ensure that the focus goes directly to each form field in the correct reading order of top to bottom, left to right.
* Verify error handling is accessible:
  + User must be able to identify any form fields that are not filled out correctly.
  + The error listing must receive focus directly after the user attempts to submit information to advance to the next screen. The error notification should include the error and take focus to the specific area in which the changes is needed.
  + Each individual error listing must begin with the text “Error” to convey information to assistive technologies.
  + Verify the user is can navigate directly to fields to make corrections.

### Verify text resize/zoom

* Verify that text can be resized (at minimum up to 200%) without breaking layout.
* Verify that full justification is not used anywhere.

### Test key pages with assistive technology

* For important applications, this is a critical step, but requires time and knowledgeable staff resources.
* Concentrate on testing key pages, especially those containing forms.
* Use a screen reader (JAWS, NVDA, or Window-Eyes), screen magnification (MAGIC), speech to text (Dragon Naturally Speaking), a Braille reader, or other AT to perform detailed testing of key pages.

#### References

508: 1194.21(a); 1194.22(l);

WCAG: 2.1.1; 2.1.2

## Step 3: Perform automated tests

### Plain language and readability

Verify page or site text meets [DHS plain language](https://dhs.intranet.mn.gov/training/create-accessible-content/index.jsp) and readability guidelines.

### **Test web pages individually and often during development**

* Using WAVE toolbar for Firefox or WAT toolbar for IE or Compliance Sheriff, test individual pages or modules often and remediate problems as found.
* For new development, at minimum, all pages should be tested individually before being deployed to formal user test or quality assurance processes.
* For modifications, at minimum, each changed page should be tested individually before being deployed to formal user test or quality assurance processes.

### **Test entire website or application early in user test or quality assurance process.**

* Use Compliance Sheriff to test the all website or application pages.
* All Priority 1 errors need to be remediated or addressed; other errors and warnings, at minimum, should be evaluated for potential effects and possible remediation.

#### References

508: [508 compliance checklist from WebAIM](http://workplace/projects/ASP/WebAccess/Shared%20Documents/Checklist%20508.pdf)

WCAG: [WCAG 2.0 checklist from WebAIM](http://workplace/projects/ASP/WebAccess/Shared%20Documents/Checklist%20WCAG2.pdf)