

These web accessibility tips can be used by web designers, developers, or content authors to guide them in creating or deploying web-based resources that are fully accessible to all users. This list is not intended to replace or map to formal standards such as the World Wide Web Consortium's (W3C's) Web Content Accessibility Guidelines (w3.org/WAI/standards-guidelines/wcag/). Have suggestions for how this list can be improved? Please send your ideas to accesscomp@uw.edu.

1. Add proper alt text to images.

Use alt text to provide access to the content of images for individuals who cannot see them, including people using screen readers or Braille output devices. Alt text is supported by most document formats, including HTML, Microsoft Word, and Adobe PDF. For more information, visit uw.edu/accessibility/checklist/images.

2. Use headings properly.

Use headings and subheadings to form an outline of the page. Do not skip heading levels. They help non-visual users (including search engines) to understand how the page is organized, and make it easy for screen reader users to navigate. For more information, visit uw.edu/accessibility/checklist/structure.

3. Create accessible PDFs.

Use "tagged PDF", the only type of PDF that supports headings and alt text for images. Use the PDF Accessibility Checker that's provided by Adobe Acrobat. For more information, visit uw.edu/accessibility/documents.

4. Know when to use PDF.

PDF preserves a document's appearance across operating systems and devices. If this characteristic is not essential, consider using another format such as HTML, which is much more accessible.

5. Use ARIA landmarks.

The W3C Accessible Rich Internet Applications (ARIA) specification makes it possible to produce accessible interactive web applications. One easy entry into ARIA is landmark roles. Simply add an HTML attribute for one of the eight landmark roles (e.g., `role="navigation"`, `role="main"`) and users will be able to jump directly to that section of the page with a single keystroke. Alternatively, use HTML semantic elements that map to ARIA roles. For more information, visit uw.edu/accessibility/checklist/structure.

6. Add labels to form fields.

Use the HTML label element so screen reader users will know which labels or prompts are associated with which form fields. For more information, visit uw.edu/accessibility/checklist/forms.





7. Group related form fields together.

In HTML, wrap groups of checkboxes or radio buttons in a fieldset element, and wrap the question or prompt that applies to them all in a legend element. For more information, visit uw.edu/edu/accessibility/checklist/forms.

8. Markup tables appropriately.

Use HTML markup properly to communicate relationships between column and row headers and the data cells within their scope. For more information, visit webaim.org/techniques/tables/data.

9. Identify language of text.

Since some screen readers are multi-lingual, use markup to identify the default language of a document and of any text that deviates from the default. For more information, visit uw.edu/accessibility/checklist/language.



10. Provide sufficient color contrast.

Be sure foreground and background have adequate contrast. There are many free tools that can help with this. For more information, visit uw.edu/accessibility/checklist/contrast.

11. Avoid using tiny fonts.

Since users may be unaware they can increase font size using browser hot keys, use a reasonably large font size by default; then, users can make it smaller if desired. Note that a font size of 1em uses the default browser font size, therefore is an ideal choice for most text, thereby honoring users' preferences and expectations.

12. Respect white space.

Provide plenty of space between lines and blocks of text. This helps many users to more easily track text horizontally, and generally makes text easier to read.

13. Provide visible indication of focus.

In CSS, use `:hover` to make a page come alive, responding to user's mouse movements. For those who aren't using a mouse, use `:focus` for the same functionality. For more information, visit uw.edu/accessibility/checklist/focus.

14. Use text, not pictures of text.

Use text instead of pictures of text, and control its appearance using CSS. Pictures of text become blurry when enlarged, take longer to download, and are inefficient for the website author to edit.

15. Think twice about the words you choose.

Keep your content simple to read and understand. Often web authors use larger words and longer sentences than is really necessary.

16. Caption video.

Captions provide benefits to all users. There are many free, easy-to-use tools available that support the process of transcribing and captioning videos. For more information, visit uw.edu/accessibility/videos.

17. Describe video.

For people are unable to see video, create a script that includes brief descriptions of important visual content, then provide that as a separate description track, either as timed text or recorded narration. This solution is known as audio description. For more information, visit uw.edu/accessibility/videos/#description.



18. Provide a transcript.

Provide a transcript for video and audio so individuals who are deaf-blind and those with low Internet bandwidth can access the content. Transcripts benefit all users by allowing them to access content quickly.

19. Choose media players that support accessibility.

When choosing a media player, ask questions like: Does this player support closed captions? Does it support description? Can it be operated without a mouse? Are buttons and controls accessible to screen reader users? Able Player ([ableplayer.github.io/ableplayer](https://github.com/ableplayer/ableplayer)) is a free player created with accessibility in mind by the University of Washington, with help from the open source community.

20. Choose a website navigation menu that works for all users.

When creating a navigation menu, Ask questions like: Can this menu be operated by keyboard alone? If it can, is doing so efficient or does it require dozens or hundreds of keystrokes? Consult credible resources such as the WAI-ARIA Authoring Practices ([w3.org/TR/wai-aria-practices](https://www.w3.org/TR/wai-aria-practices)) for accessible design patterns and examples on how to properly code navigation menus.

21. Create JavaScript widgets that support accessibility.

An accessible interactive widget is one that can be operated with the keyboard alone, uses ARIA to communicate with assistive technology (AT), and degrades gracefully for users who don't have the latest AT. Consult the WAI-ARIA Authoring Practices ([w3.org/TR/wai-aria-practices](https://www.w3.org/TR/wai-aria-practices)) for accessible design patterns and examples on how to properly code a wide variety of common widgets.

22. Choose JavaScript widgets that support accessibility.

If choosing to use an existing widget rather than creating your own, perform due diligence. Check the documentation and bug reports for details about accessibility. Test widgets using keyboard alone, or using AT, and ask users with disabilities to test them.

23. Know how your Learning Management Systems (LMS) and Content Management Systems (CMS) support accessibility.

Most LMSs and CMSs provide some level of support for accessibility. Understand the accessibility features of the tools you're using. If there are accessibility shortcomings, understand the workarounds. Know which themes, plug-ins, and modules are accessible, and recommend or require those over inaccessible options.

24. Test web pages using a keyboard.

Take the #nomouse challenge! Try navigating the web page and controlling all its features using the tab key on a keyboard; don't touch the mouse. This simple test is typically a good indicator of accessibility. For more information, see nomouse.org.

25. Test pages using high contrast color schemes.

All major operating systems and some web browsers have high contrast color schemes available. When these are enabled, make sure that all important page content is still visible.

26. Test pages with assistive technologies.

There are free screen readers and other AT available that can be used for testing. You don't have to become an expert user, but simple tests with AT can provide valuable insights into whether certain features on a web page might present accessibility problems. For more information, see the WebAIM article [Testing with Screen Readers \(webaim.org/articles/screenreader_testing\)](https://www.webaim.org/articles/screenreader_testing), as well as their series of quick guides on testing with specific screen readers.



27. Test pages on mobile devices.

Growing numbers of users, including users with disabilities, are accessing the web using phones, tablets, and other mobile devices. Test your website using mobile devices, and when doing so, be sure to check for accessibility.

28. Ask vendors specific questions about the accessibility of their products.

When procuring products from third party vendors, we are dependent on those vendors to provide products that work for all users. Always ask about accessibility of products, even for minor purchases. For major products, consider adopting a more formal process for addressing accessibility. For more information, visit uw.edu/accessibility/procurement.

29. Demand accessibility!

If a product is not accessible, avoid buying it, using it, or supporting it. Work to implement policies on your campus that require IT purchases to be accessible. If it is required that an inaccessible product be used because it is the only alternative, put the vendor on notice that you expect to purchase an accessible alternative when it becomes available.

30. Get involved!

There are many communities of practice that are working together on making the web (and world) more accessible. For example, consider getting involved with the Access Technology Higher Education Network (ATHEN; athenpro.org) or EDUCAUSE IT Accessibility Community Group (educause.edu/community/it-accessibility-community-group). Together we can make a difference!

About AccessComputing

The Paul G. Allen School of Computer Science & Engineering and DO-IT (Disabilities, Opportunities, Internetworking and Technology) at the University of Washington lead *AccessComputing*, a project that aims to increase the participation of people with disabilities in computing careers nationwide.

For further information, to be placed on the mailing list, request materials in an alternate format, or to make comments or suggestions about DO-IT publications or web pages, contact:

University of Washington
Box 354842
Seattle, WA 98195-4842
accesscomp@uw.edu
www.uw.edu/accesscomputing/
206-685-DOIT (3648) (voice / TTY)
888-972-DOIT (3648) (toll free voice / TTY)
509-328-9331 (voice / TTY) Spokane

AccessComputing
Dr. Richard Ladner, PI
Dr. Sheryl Burgstahler, Co-PI
Dr. Amy J. Ko, Co-PI
Dr. Jacob O. Wobbrock, Co-PI
Dr. Brianna Blaser, Project Coordinator

Acknowledgment

AccessComputing is supported by the National Science Foundation under Grant #CNS-0540615, CNS-0837508, CNS-1042260, and CNS-1539179. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Copyright © 2020, University of Washington. Permission is granted to copy these materials for educational, noncommercial purposes provided the source is acknowledged.



University of Washington
Computer Science and Engineering
DO-IT
UW Information Technology