Accessibility or “a11y” is an increasingly important topic in web development. It is important to Go because many Go tools and resources use web technology, like godoc. Ensuring that these tools are accessible ensures that more people can use them and that more people can use Go.

Accessibility is a large topic that ranges from the merely esoteric to the excruciatingly abstruse.

There are, however, a number of simple rules of thumb to follow that greatly reduce the chances of creating an inaccessible page by accident: - Prefer semantic markup. The native html elements have a great deal of accessibility baked in with no need to turn anything on or turn anything off. If you have a list of things use `ul` or `ol` even if you do not want it to be displayed as a list. The extra CSS is worth it. - When using headers (`h1-h6`) never skip a level and make sure they nest logically so that, for example, an `h4` is a logical subsection of the previous `h3`. Screen readers use these as navigational landmarks, similar to a table of contents. - Avoid layouts that work against the source order of the document. It's very easy to rearrange element with flexbox and grid but when you do the focus order remains unchanged. Someone navigating the site by keyboard will jump from place to place seemingly at random, making it very hard to follow what's going on. - Ensure sufficient color contrast. Not everyone has perfect eyes or high quality displays. Either can make it difficult to tell where low-contrast content starts and stops. https://leaverou.github.io/contrast-ratio/ - Ensure the page is usable without color. There are many kinds of color deficiency. Some like red-green color blindness are common. Others can be quite rate. The best way to make sure everyone gets the same information is to encode information in both shape and color. For example, if you're making a dashboard for a test runner do not just have a green circle for passed and a red circle for failed: have a green check mark for passed and a red X for failed. - Have clear hover and focus styles to let the user know that an element is “clickable”. As with the above, this should be clear even without color. - Make sure nothing breaks as you zoom in the page to 200%. Browsers are really good about handling this but are not always perfect. Users with poor vision will appreciate the effort. - Make sure the page is usable entirely by keyboard. Certain medical conditions make it hard to use a mouse or trackpad and a lot of special assistive software and hardware work by creating synthetic key events. Be very careful about elements hidden off page as they will remain in the focus order without special effort. - Ensure all non-textual elements have appropriate alternative text. (If a non-textual item is purely decorative the appropriate alternative text is the empty string.) - Avoid writing in ALL CAPS. Screen readers will always read this as an acronym and spell out each letter (`text-transform: uppercase` is fine, however, and will just add emphasis to the voice). - There are a lot of tools to help find accessibility problems (some listed below) but keep in mind that these are just linters and there are limits to static analysis. They sometimes have false positives and often have false negatives. Also keep in mind that what they are checking is the page as it appears when you run the tool.

Useful Browser Extensions

Firefox

- Fangs is a screen reader emulator. It dumps all the info a screen reader would announce at once as annotated text, allowing a very quick review.

Chrome

Chrome has some quite useful built in a11y tools, but you need to enable them by going to settings > experiments in the dev tools and checking "Accessibility Inspection".

- Google's Accessibility Developer Tools Largely integrated into the Chrome dev tools but still has some features not yet included like access to the full Accessibility Tree.
- aXe Decent static analyzer. It's the basis for the accessibility audit in Chrome Dev tools but this extension includes much more information and runs more tests. (Also available for Firefox, but has much better integration in Chrome).
- ChromeVox A full in-browser screen reader. (You will need to configure keyboard shortcuts to be able to turn it off when not in use, and you may also need to disable the extension completely when not in use as it can do some annoying thing even when not active).
- High Contrast This extension is used by people with special contrast needs. It's a good idea to run a page through each filter (and test any hover/focus states) to make sure that everything is still visible.
- Siteimprove Accessibility Checker This is the best static auditor. The extension will ask you to register for a newsletter on first use, but you do not have to—just skip it.
- Spectrum Simulates various kinds of color blindness to make sure nothing on the page becomes invisible or hard to distinguish when read by a user with that specific medical condition. (For some reason this does not seem to work immediately after installation but it will after a few retries).
- Funkify An easy to use disability simulator.

Further Reading

- http://a11yproject.com/
- https://inclusive-components.design/
- https://accessibility.blog.gov.uk/
- https://www.youtube.com/playlist?list=PLNYkxOF6rcICWx0C9LVWWVqvHlYJyqw7g