Accessibility

What is accessibility?

In the most basic terms, accessibility means making your app or website available and easy to use for as many different people and situations as possible. This includes everything from disabilities and impairments, to situational needs like when a user has a poor connection or can only access one type of device.

Why do we need to care about accessibility?

Above all, the main reason we spend time making considerations for accessibility is to make sure everyone can access and enjoy a product we build. It's simply the right and most empathetic thing to do!

Additionally, there are legal obligations to have at least some basic accessibility in place. This is particularly true if you work in or with the public sector.

Rules to follow

There are lots of considerations to make for accessibility, but fortunately there are guidelines we can follow to help write accessible code. These guidelines are called the Web Content Accessibility Guidelines (WCAG).

You can find the guidelines at this website, but it can be hard to read and follow. Instead, you might like to look at MDN's breakdown of the guidelines. The UK government also provides a pretty readable breakdown of the guidelines.

The four principles

The guidelines each belong under one of four principles: perceivable, operable, understandable and robust. Again, it might be worth looking at something like MDN for a breakdown of these, but here's an overview of the four principles.

- Perceivable: Users are able to see / read / hear content
- Operable: Users are able to navigate and to their own preference
- Understandable: Users are able to understand the content and the content behaves as expected
- Robust: Users can use a variety of devices

Conformance levels

There are three levels of accessibility that we can aim to meet. These are A (essential), AA (ideal) and AAA (specialized). Each of the guidelines and principles will list aims for each of these levels.

A is the bare minimum. This means making sure that your site doesn't have clashing colours and all inputs have labels, for example.

AA is the level most developers should aim for and is the level for the legal obligation.

AAA is usually when you are providing specialist support e.g. your site could be explored with any keyboard device imaginable.

Examples of real world implementation

Semantic tags

Semantic tags have no style or function, but are used to essentially label parts of a page. We should aim to use semantic tags as much as possible and avoid general tags like `div` and `span`, though they do have their place! Both of these general containers are often great for helping with styling and layout.

Here is a list of common semantic tags:

- main
- section
- article
- footer
- header
- aside
- nav

Imagine we have this piece of code:

```
<div id="main">
    <ul>
        <li><a href="/home">Home</a></li>
        <li><a href="/about">About</a></li>
        <li><a href="/Contact">Contact</a></li>
    </ul>
</div>
```

Most developers coming to this code would likely understand that the div is meant to show the main content and the ul is some kind of menu, however, users won't see this and need more context, particularly if using something like a screen reader.

We could change it to this:

```
<main id="main">
    <nav>
        <ul>
            <li><a href="/home">Home</a></li>
            <li><a href="/about">About</a></li>
            <li><a href="/Contact">Contact</a></li>
        </ul>
    </nav>
</main>
```

A small change that makes a huge difference!

Use of colour

Whenever presenting users with information or asking them to do something, it's important we don't do things like ask them to 'click the red link'. Any prompts like these should be presented in at least two ways to account for the very possible situation that the user cannot see colour or see the content for some other reason.

For example, we would avoid something like a link coloured in red with instructions to 'click the red link' and instead perhaps add to the link some information about what it does in addition to the instructions e.g. 'click this link for more information'. This ensures that even if somebody cannot see red or has some kind of visual impairment, they can still get feedback on where the link goes.

Good alt text

All images should have alt text. This text will be read to devices like screen readers and will also render in place of an image if it is broken or cannot load.

Alt text should not contain words like image or photo e.g. an image of... As it's an image tag, the browser already knows it's an image. Aim for as much detail as you can to recreate the scene.

Bad alt text: 'an image of a dog'

Good alt text: 'A big white, fluffy dog looks happily into the distance. In the background is a forest'

Helpful tools

There are a number of tools available to help us with accessibility. These range from packages where we can write tests to simple browser extensions. Here are some tools you can start with:

- Lighthouse: in Chrome's DevTools, you will find a tab labeled Lighthouse. In here, you will find the option to run basic tests on your page. This will give feedback on things like the colour contrast, lack of semantic tags, and header stepping (e.g. moving from h1 to h3 with no h2 in between)
- aXe: available as both a package and browser extension, aXe gives more detail than Lighthouse. This appears as an option in your DevTools options, right at the bottom. One caveat is that aXe is best used on local projects with the extension as many sites will block its usage. As mentioned, aXe can be used as a package but this will require your project to have a `package.json` and usually some set up.

Get aXe here

- WAVE: available as a browser extension and also gives more information than Lighthouse. WAVE will advise you where the issue is and give some suggestions on how to solve the issue.

Get WAVE here

Please remember, none of these tools are a fix-all and a truly accessible app or website will have been thoroughly tested by the developers and have gone through user testing by a multitude of users with different needs.

ARIA

ARIA (Accessible Rich Internet Applications) is probably one of the most difficult aspects of accessibility. It is used to make content accessible by using things like extra labels or linking content together to better show users with assistive technology what is going on on a page.

We would advise you to focus on all the other elements of accessibility before tackling ARIA and would also advise you only use ARIA tags if you know what they do. It's often said that using ARIA badly is almost as bad as not using ARIA at all.

Here's an example of the `aria-label` in action:

`<button aria-label="Like">❤</button>`

Many users would understand that the heart represents a 'like' on most platforms. For assistive technology, we can ask the browser to provide additional information through an extra label, which here says 'like', so the user knows what the button does.

Quick tips - easy wins!

- add labels to any inputs you use
- use semantic tags, like section, in place of divs as much as you can
- make sure your colour contrast meets at least the A conformance level
- use buttons for actions and links to go places
- describe images in a way that users can imagine what would be there
- use descriptive text for anchor / link tags e.g. not just 'click here' but 'click here to visit our Instagram', for example