Accessibilty (a11y)

- Content is available to as many people as possible
- Disabilities are common
- Disabilities are more than just blindness
    - but blind people are people too

Why a11y?

- Programmer are lazy
    - it's one of the 3 Virtues of a Programmer
- `a cce ssi bil it y`
- a (eleven letters) y
- a11y

Hint: if you put this on your resume, have both forms

Ex: "Exposure to web accessibilty(a11y) options"

How are we accessible?

- it's an entire field of work
    - we are only covering the intro/basics
- Make HTML inform tools like screen readers
- Provide alternatives for visuals
- Allow for physical limitations

Informing tools

- Using Semantic HTML
    - provides a lot of automatic behaviors
    - includes not using semantic tags unsemantically!
- Adding ARIA attributes
    - coming up
    - states (ex: expanded)
    - properties
        - landmarks

Alternatives for visuals

- Image `alt` attributes
  - Have them
  - With **useful** text
  - If it is visually interesting, describe it!
    - Even if it isn't mechanically relevant
    - Example: don't say "logo" or "picture"
  - but use `alt=""` for when basically pointless
  - Don't say "picture of..."
    - just describe contents

Don't rely on visuals alone

- Don't use colors alone to signal info!
    - Have text as well
    - Example: an On/Off slider: say "On" or "Off"
        - in addition to any visual effect
        - Don't assume your visuals make sense!

Allow for physical limitations

- Allow for keyboard OR mouse
- Minimum size for touch controls (54px)
- Don't put info needed under their hand (mobile)
- Think before requiring hold/drag
    - for steadiness
    - and for timing
    - fine motor control isn't even common
        - that's why we call it "fine"
        - don't require it.

Web Content Accessibility Guidelines (WCAG)

- **https://www.w3.org/TR/WCAG20/**
- WCAG by the WAI at W3C (!)
- A set of guidelines for accessible web content
    - used by vendors of tools
    - used by webdevs that care
- 3 levels (A, AA, AAA)
    - A = "must" (absolute minimum, not praise)
    - AA = "should" ("good enough")
    - AAA = "may" ("actually working at it")

How to use WCAG

Rules for 4 areas (POUR):

- Perceivable
- Operable
- Understandable
- Robust

Worth it to read through once

- Notably: vague

Semantic HTML covers most of A and AA

- Not everyone is semantic!

Tooling!

Various tools exist to help!

- Tools to test your site
- Tools to act as the user
- Tools to try to do it for you
    - I've heard only bad things

Why not to rely on validation tools alone!

- Guidelines are vague and subjective
- No tool can test for that
- Tools only recognize clear violations
    - and some might be actually correct
- Human review is needed to find subtle bugs
    - and to verify if reported bugs are real

Why to use validation tools anyway

- Most of us won't know the actual experience
- Good to supplement human review
- Can teach good habits
    - Fix the same issue a few times
    - You start writing it correct the first time

Why to avoid accessibility overlays

- A few companies make these
- Ads/sponsored links in a11y search results
- They offer to make your site accessible
    - Without you changing the site
- These are my personal understanding, not NEU…
    - EVERY a11y expert and disabled user I follow
        - HATES these
    - These tools have lost or settled court cases
- Learn to do it right instead

Example of a validation tool

- aXe, WAVE, etc
- Install WAVE Chrome Extension

Example of a screen reader

- (Demonstrate VoiceOver)
- Using a screen reader is a good confirmation of the experience
    - But involves more work to learn on your part
    - Headsets a must in an office :)

Minimum a11y

- Use Semantic HTML
    - Seriously, not casually
- Provide alt text
- Avoid "Click here" or "Read More"
- Have enough color contrast

Minimum a11y test

(inspired by @geekgalgroks on Twitter)
https://a11y.jenn.dev/posts/bare-bones-cheatsheet/

- Can you tab through all controls?
- Can you operate all controls with enter/spacebar?
- Do you pass a color contrast test?
- Confirm alt tags
    - what you tell someone not looking at it

Accessible Rich Internet Applications (ARIA)

- W3C WAI ARIA, for those keeping score
- "Rich" means JS-driven HTML
- HTML attributes to give more meaning
- Semantic elements automatically assume many of these
- Can be quite complex
- Minimize the need with semantic HTML!

No ARIA is better than Bad ARIA

- ARIA overrides default semantic HTML behavior
  - AND overrides assumptions tools make for apps w/o a11y effort
  - When the ARIA is bad, it's a *trusted* bad
  - ARIA assumes behavior, doesn't provide it
- Avoid by minimizing the need
  - tired of hearing this yet?
  - and minimize the use
  - and understand the use
  - and verify with screen-readers

ARIA Roles

A "role" gives purpose to an element

- a "button" is a role
- a "heading" is a role

Many semantic HTML elements are roles

- but some people use different elements
- there are also roles with no matching element
    - such as "tab panel" and "tab"
- live region roles define areas that change
    - Note: must be on screen with role BEFORE a change to work

ARIA Landmarks

- Define the foundational structure
  - main
  - navigation
  - region
  - search
  - etc
- You want some, but not too many
  - "noisy"
  - You want to make the page easy to navigate
  - Don't become a voice operator system

ARIA States

- States imply changeable states of elements
    - think "checked" or "selected"
    - but also "open", for accordions
- Offer more description than HTML alone
    - and that's when you want a little ARIA

ARIA Properties

Data about an element not expected to change

- such as "label" or "labelled by"

Common use case:

- cards of many articles
- each with intro text and "Read More"
- visually we can see the article title
    - and know "Read more" what?
- ARIA can let us give screen readers more to read

How to ARIA!

- First, do you need to?
- Second, check the Practices document
    - **https://www.w3.org/TR/wai-aria-practices/**
    - Look to see how the ARIA attributes are used

Better Experience through Skiplink

A "skiplink" is a link that moves focus past initial headers/navigation

```
<a href="#main">Skip to content</a>
```

- moves focus to `id=main` element on same page

Nice for everyone if you have big headers and multiple pages

- Even nicer if you're having it read to you on every click

Hiding the skip link

Skiplinks are often visually hidden

- not always, but often
- visually shown if you tab to it
- read/usable by screen reader regardless

Only visually hiding the skiplink

`display: none` would REMOVE the skiplink

- meaning it couldn't gain focus
- and wouldn't be read/usable by screen readers

Instead, move away from visual

- transform it offscreen
- still in rendered document
- move it onscreen when it gets focus

https://css-tricks.com/how-to-create-a-skip-to-content-link/

Summary - A11y

Accessibility is about making content usable

- Semantic HTML does a lot of work
- Small details can make a big impact
- If it is frustrating for you to fix
    - consider what it is like for users!

Summary - A11y Tools

- Validation tools are great
    - but cannot be a pass/fail
- Accessibility Overlays exist
    - have a bad reputation
- Screen readers are hard to learn
    - but are "real" experiences

Summary - ARIA

- ARIA are attributes added to elements
    - Provide additional context
    - used by tools to modify experience
- No ARIA is better than Bad ARIA!
- ARIA doesn't create behavior, it informs tools
- ARIA Practices is great source of learning
    - but take it slow