# Accessibility

**Michael Chang**

**Spring 2020**

# Updates

**Please refer to new pinned Ed post**

**Project is optional, extra credit**

**If remaining assignments, will work with you**

# Plan for today

**Web accessibility challenges**

**Demo: screen reader use and navigation**

**Demo: Accessibility examples**

- Both positive and negative...

**Summary of takeaways from demo**

# Access challenges

**Many are not unique to web**

**Fonts, colors, styling**

Low contrast colors, small text size, hard to read fonts

**Animation, movement**

E.g. text that scrolls by quickly

**Spacing and layout**

E.g. right-aligned text can be easy to miss

# Access challenges

**Some are more specific**

**Navigation challenges**

- Assume use of mouse
- Overridden keyboard shortcuts

**Element labeling and identification**

- E.g. everything is a div
- Some UX (menus, toolbars) don't have HTML elements

# Demo: Screen readers

**See link on website for instructions**

# Takeaways

**Use semantic elements**

- Meaning aids navigation, searching
- Use CSS to suppress default styles

**Label elements clearly**

- Images: use `alt` text
- Form controls: use `<label>`
- When settings colors/fonts/etc. in CSS, ensure there is a text description
  - E.g. don't just mark same text red or green
- For general elements, use `aria-label`
  - But this is a bit of a "hammer;" prefer more specific techniques

# Takeaways

## Ensure DOM element ordering makes sense

- E.g. don't put footer as first child of body
- Beware when using `absolute`/`fixed` `position`, Flexbox `reverse`, etc.

## Manage the user focus

- Use `.focus()` to move focus programmatically
  - Draw attention to new/changed content, dialogs, etc.
  - Set `tabindex="-1"` to make elements focusable via JS
- Restrict focus to subset of elements
  - E.g. when elements aren't interactable (dialogs)
  - See code on resources page

# Takeaways

**Be careful of keyboard and mouse events**

- Provide alternatives to drag/hover
- Be careful about overriding standard keyboart shortcuts and behavior

# ARIA

**ARIA defines HTML attributes used by assistive tech**

- Should still prefer more general techniques (e.g. semantic elements)
- But provides more flexibility

**Examples of ARIA roles**

- Allows you to decouple HTML from asstiive tech behavior
- Often requires a fair amount of JS to work correctly