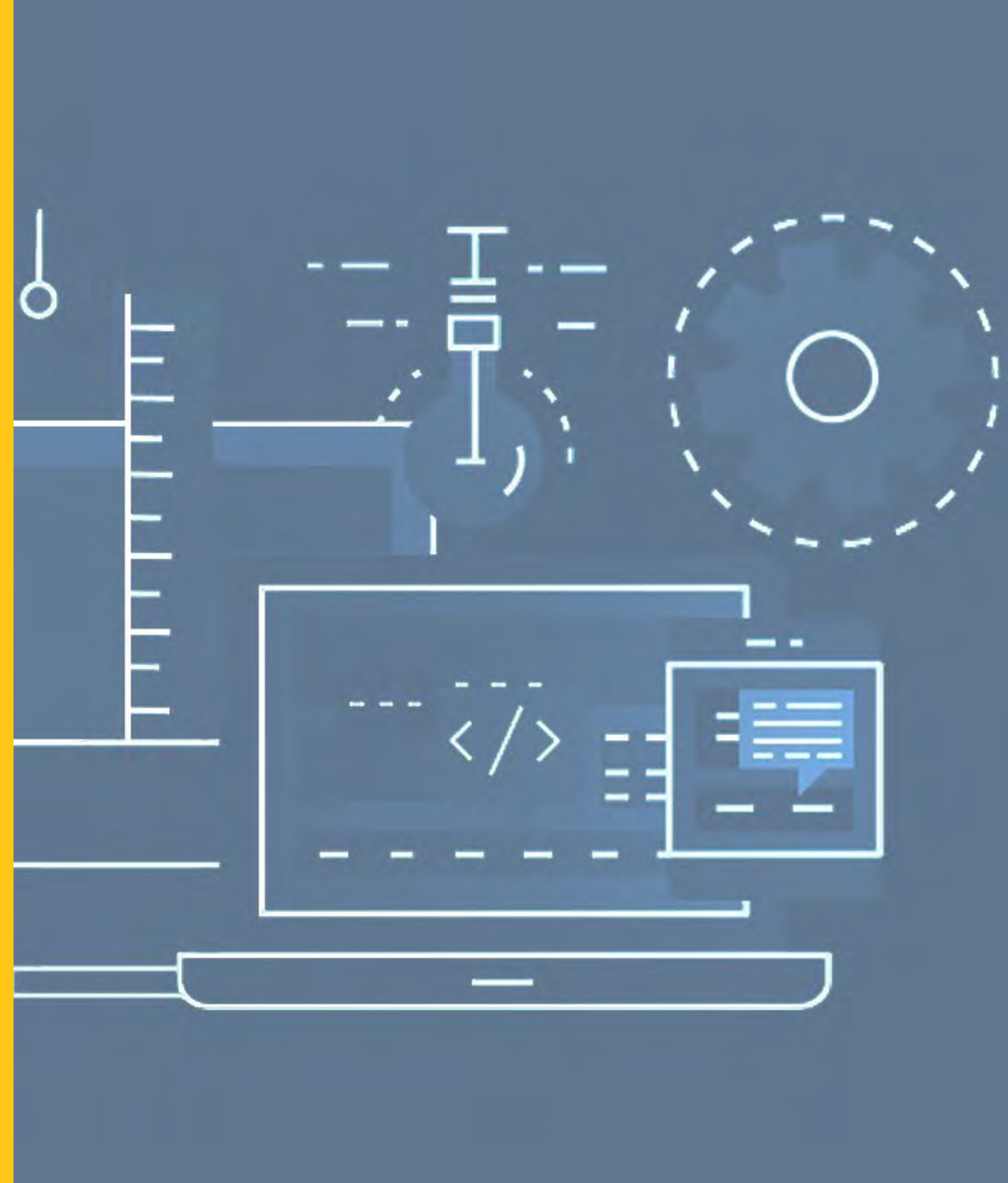


SHELLY GRAHAM, 07/18/2022

# WEB DEV 4

## SUMMER 2022

Week 2: Accessibility



# WEEK 2: ACCESSIBILITY



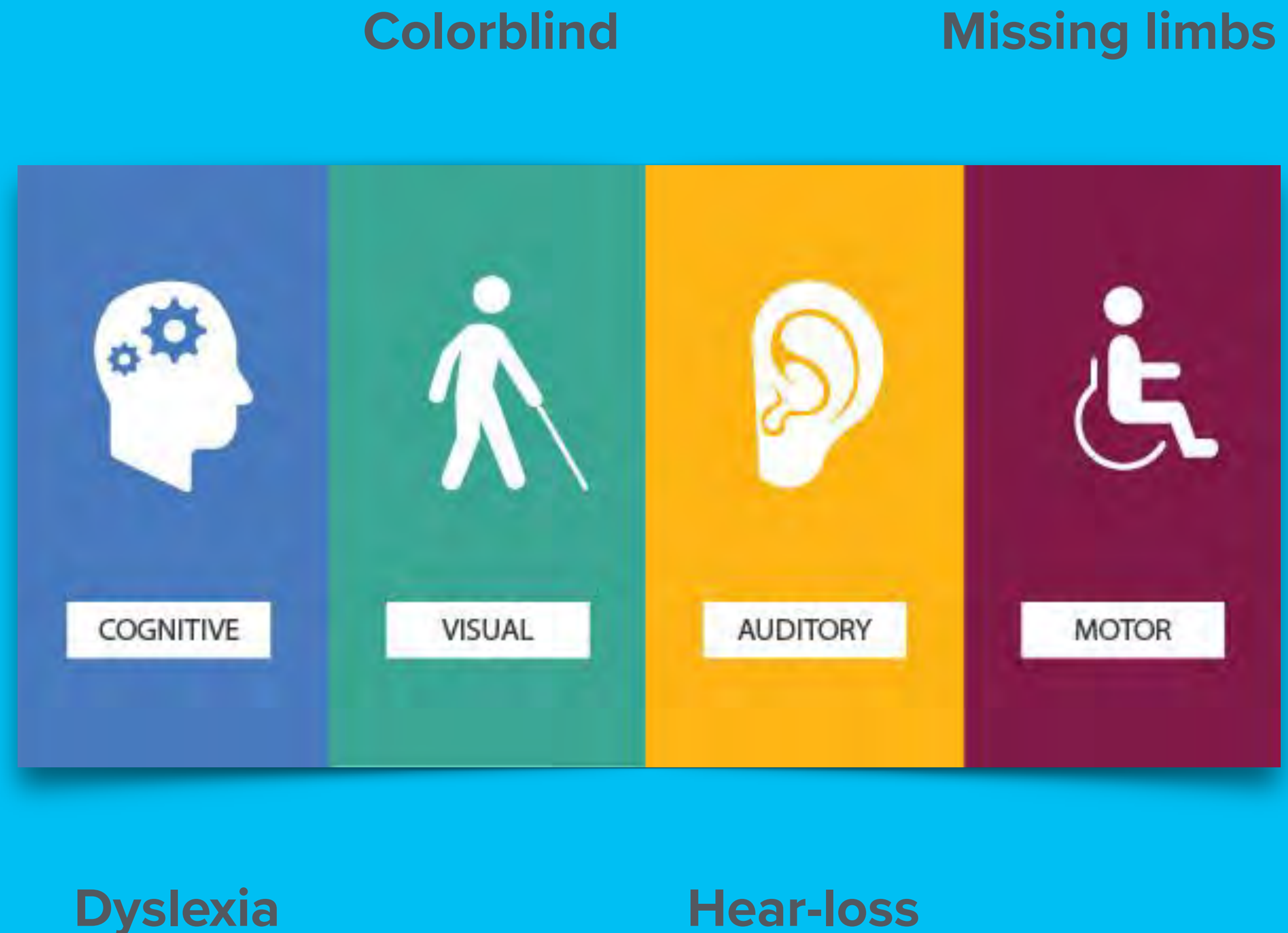


# APPLE COMMERCIAL



# WHAT IS ACCESSIBILITY?

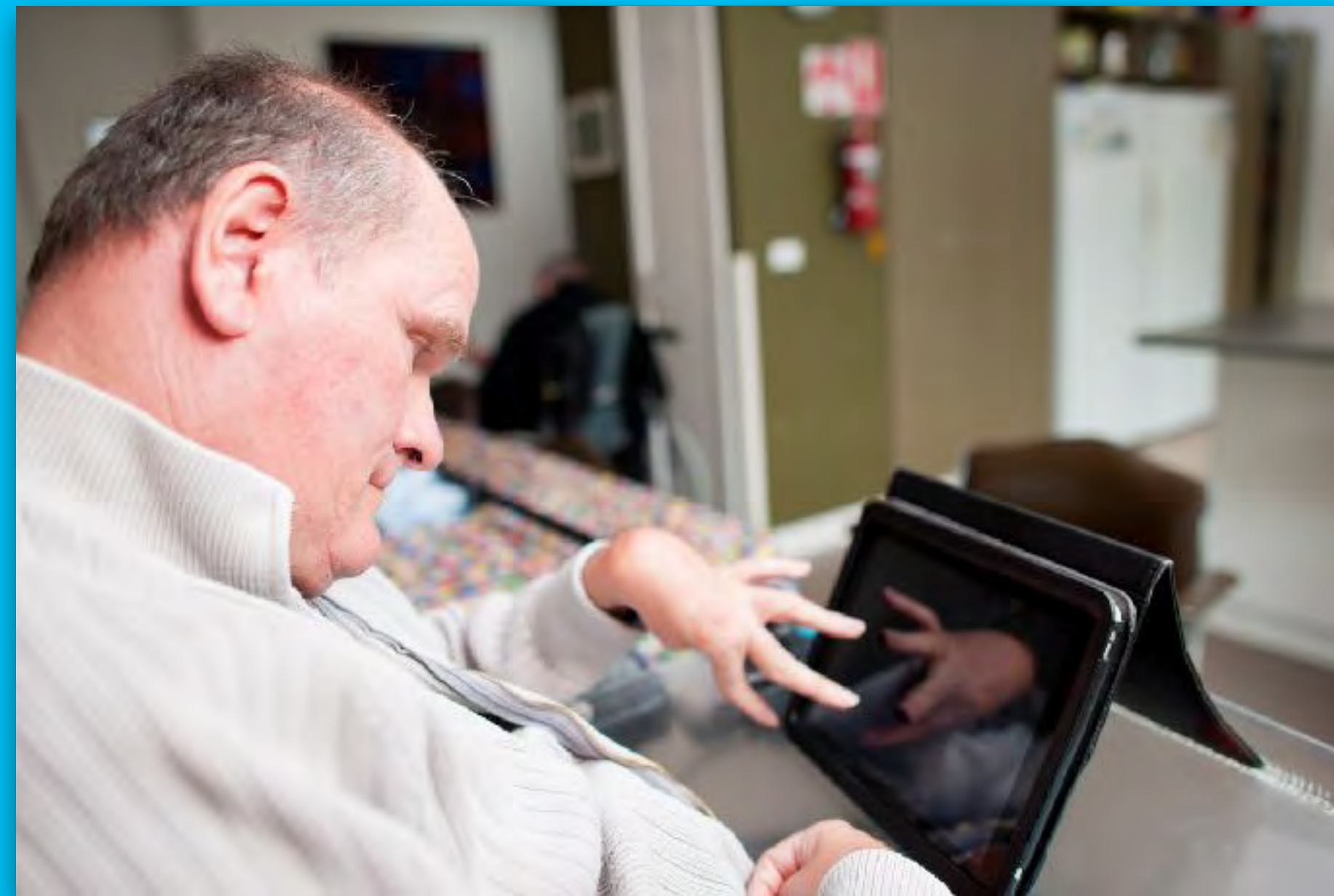
- The ability for users with disabilities to access and contribute to technology content with their accommodations
- Generally 4 categories of disabilities:
  - Cognitive
  - Visual
  - Auditory
  - Motor





# WHY BOTHER?

- Because everybody uses the internet - but not everybody uses it the same way!
- Code for as many people as possible **INCLUDING** users with permanent and/or temporary disabilities
- Accessibly helps to elevate the overall user experience
- Depending on the industry it can be mandated by the government





# WHY BOTHER?

- But besides that, Accessibility helps with:
  - SEO
  - Maintainability

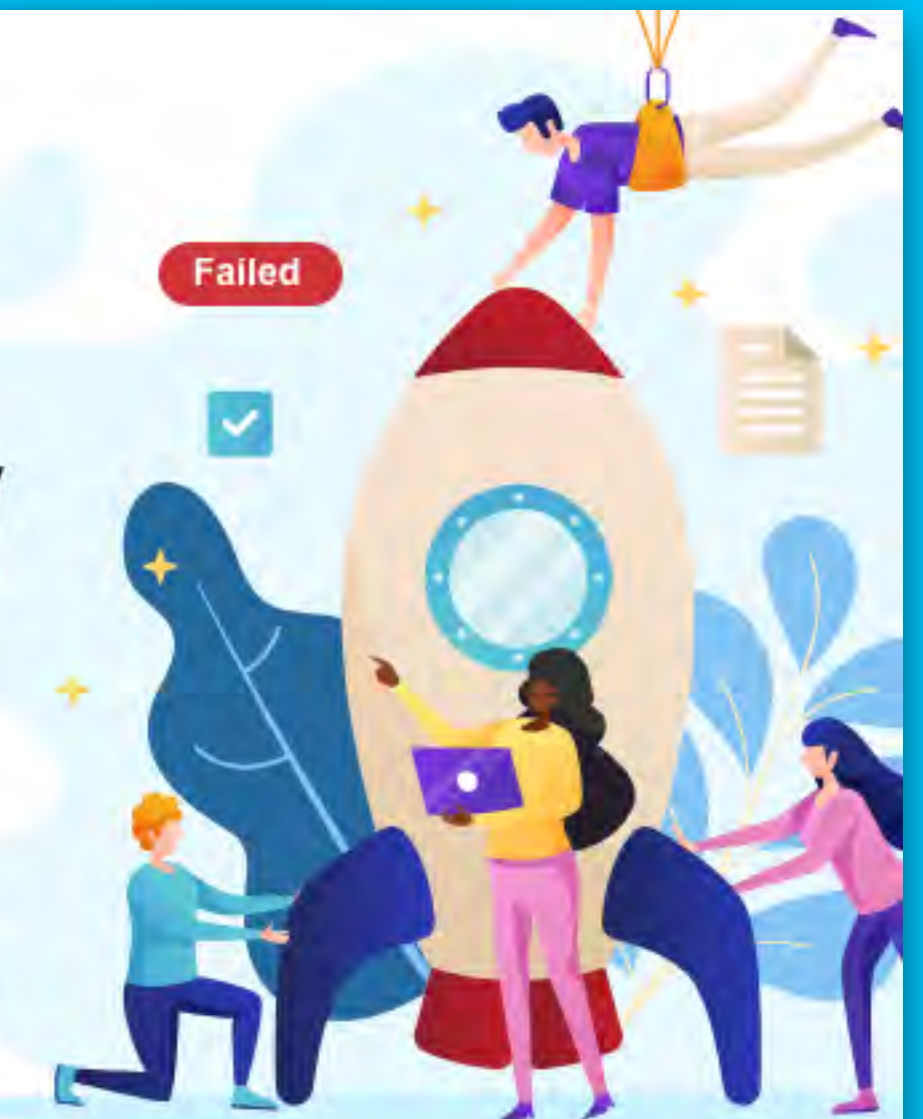


**Maintainability  
= Productivity**



Passed

Failed





# EXAMPLES

- Disabled users use screen readers and voice over technology to navigate websites and apps
- People with different native languages may use closed captions during shows or movies to better understand the context
- A strained or broken wrist can hinder you from using a mouse for a short period of time





# STANDARDS

- WCAG = Web Content Accessibility Guidelines
- WebAim Checklist = Web Accessibility in Mind
- The A11Y Project = The Accessibility Project
- Government guidelines like Section508

W3C

WebAIM  
web accessibility in mind



The A11Y Project





# WHAT IS THE ISSUE HERE?



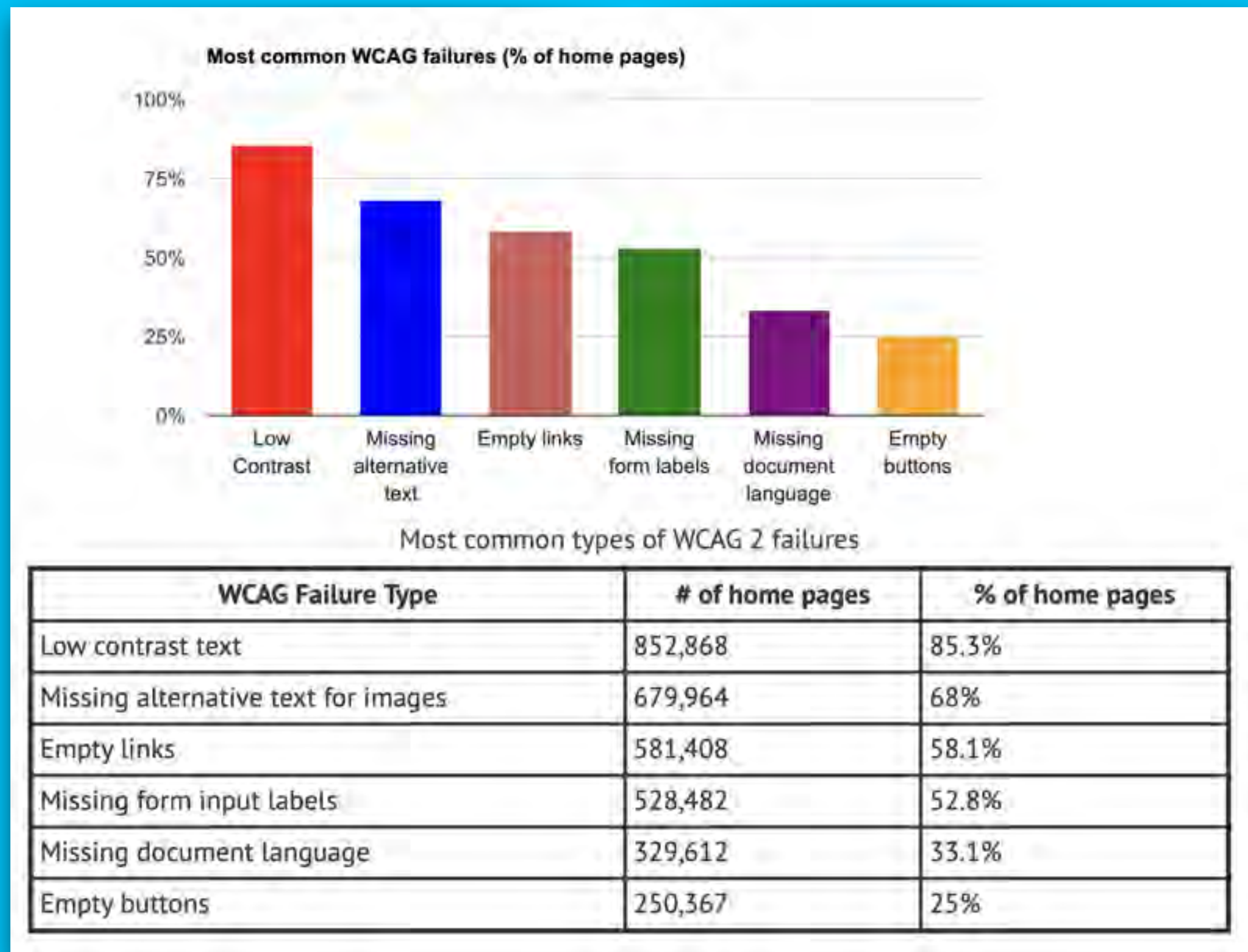
HOW TO FIX IT: LINK



# WE ARE FAR FROM STANDARDS...

- Analysis of the Top 1 Million websites
- **98% of these sites had fixable and preventable errors**

■ Source





# ACCESSIBILITY IS THE LAW

- Inaccessible technology excludes people with disabilities
- This was made illegal in 1990 with the Americans with Disabilities Act
- For physical AND digital spaces!
- There is a global version of this as well - so you really have no excuse!





# EVOLUTION OF THE WEB

- Started from the bottom now we (still) here...
- We left a LOT of pages that don't work (for today's standards) behind or simply never updated them

## Web 1.0 / 2.0 / 3.0 Summary

Crawl	Walk	Run
Web 1.0	Web 2.0	Web 3.0
Mostly Read-Only	Wildly Read-Write	Portable & Personal
Company Focus	Community Focus	Individual Focus
Home Pages	Blogs / Wikis	Lifestreams / Waves
Owning Content	Sharing Content	Consolidating Content
Web Forms	Web Applications	Smart Applications
Directories	Tagging	User Behavior
Page Views	Cost Per Click	User Engagement
Banner Advertising	Interactive Advertising	Behavioral Advertising
Britannica Online	Wikipedia	The Semantic Web
HTML / Portals	XML / RSS	RDF / RDFS / OWL

MySpace, Friendster  
Facebook, Tumblr  
Vine, Google+

Crypto  
Decentralization



# ASSISTIVE TECHNOLOGY

# ASSISTENT TECHNOLOGY = AT

- **Settings, Soft- or Hardware that maximizes mobility of users to operate a device**
- **Three main categories:**
  - **Operating System Settings**
  - **Software**
  - **Hardware**





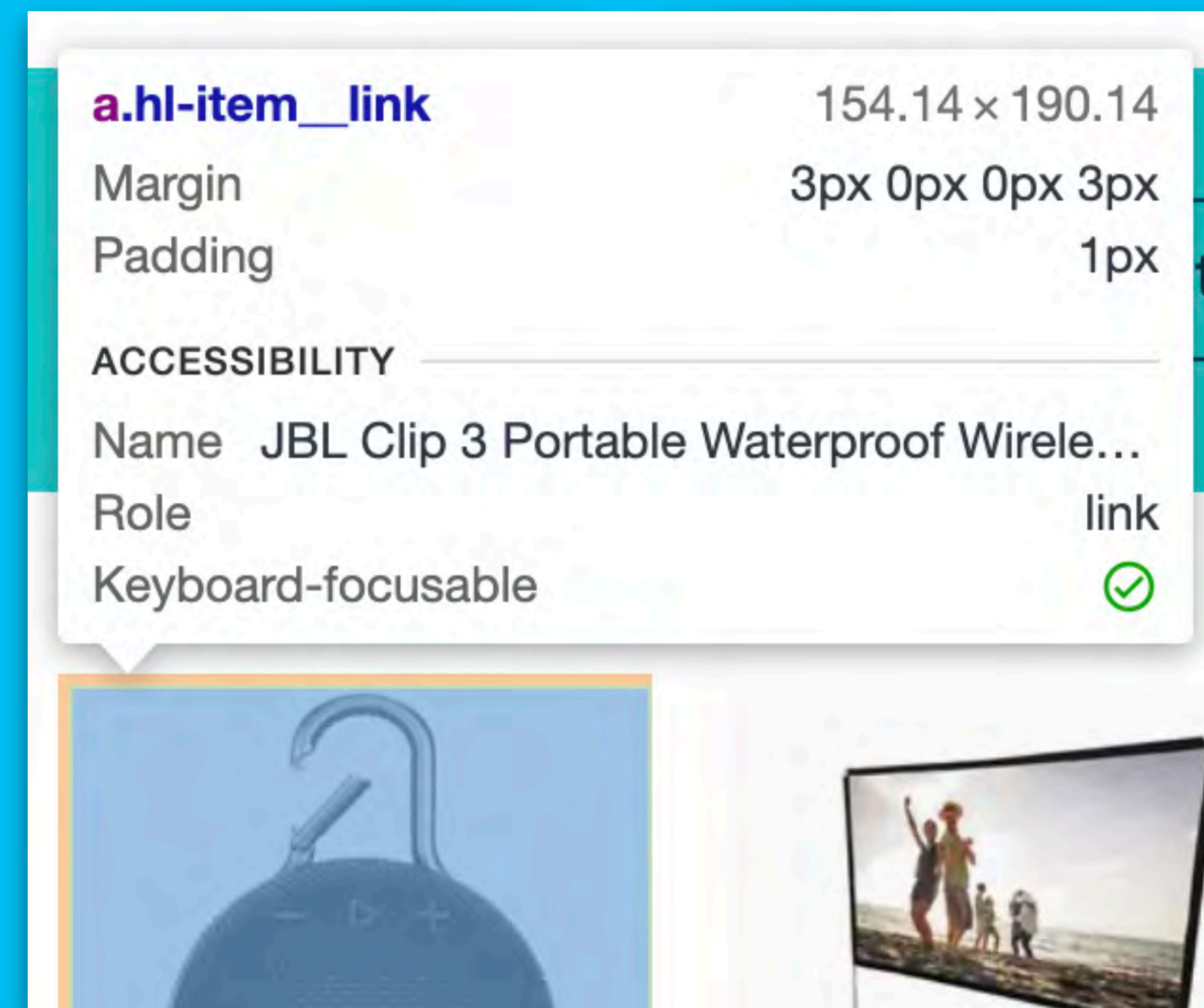
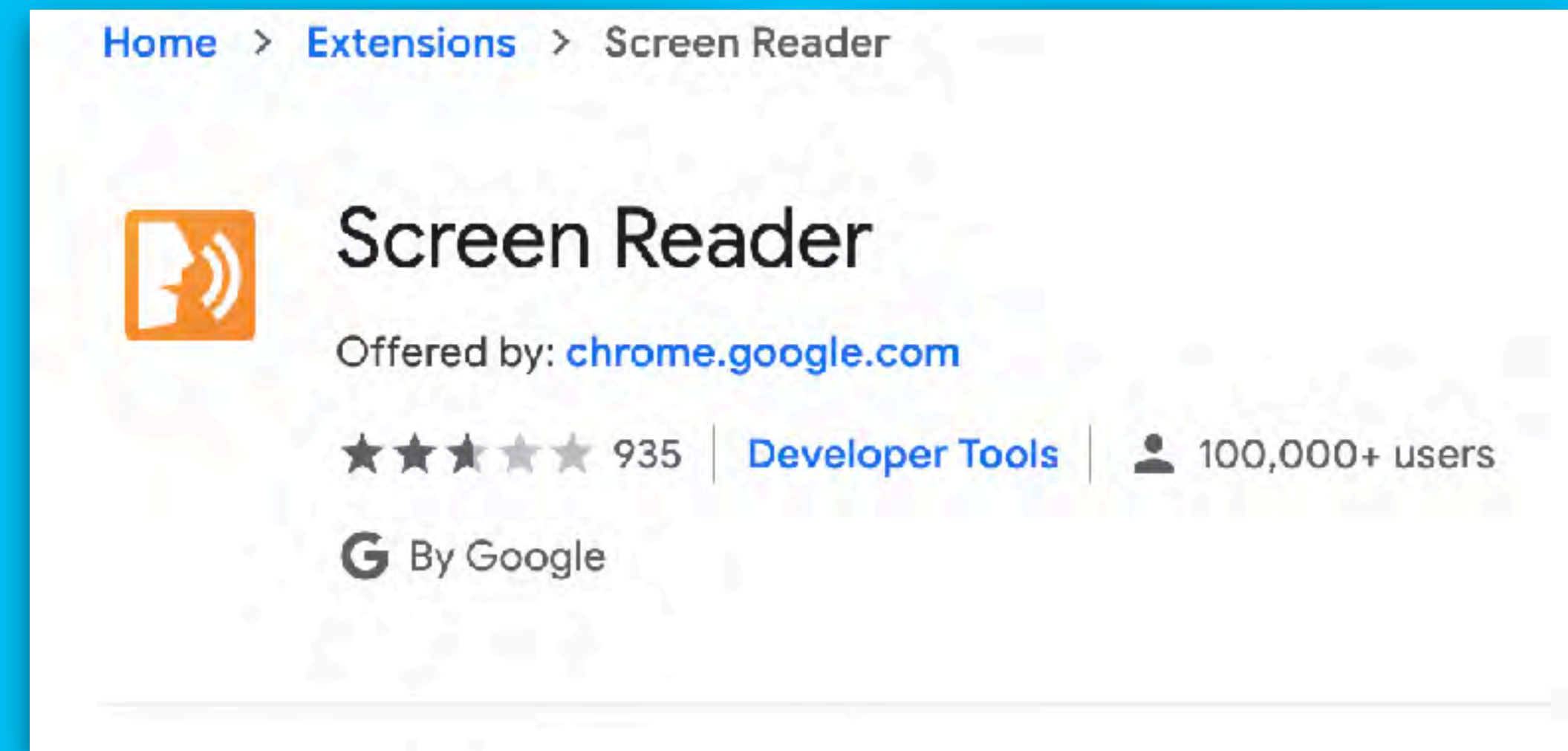
# 1. OPERATING SYSTEM SETTINGS

- Color Filters
- Text Size
- iOS Accessibility Options
- Zoom
- Classic Invert
- Voice Over



## 2. SOFTWARE

- ChromeVox Chrome Extension
- Chrome Accessibility Inspector
- TextHelp from “read-write”
- JAWS Screen Reader
- ZoomText (iOS Zoom Alternative)
- Optical Character Recognition (OCR) via Adobe





## 3. HARDWARE

- **Mouth Stick**
- **Switch —> button(s) that mimics keyboards**
- **Eye tracking software**



# SEMANTIC HTML



# SEMANTIC HTML

- Using HTML tags that are relating to the meaning of the context of code
- We learn to write custom code which is great and all - but over time we have deviated away from “the basics”
- Custom code can look great but can still be shitty code. Worst of all, if it works for the “main” user, it can spread like wildfire through StackOverflow, templates - all without intervention!
- HTML is JUST as important as CSS and JS - if not more!

# SPACE JAM

- Cult website but there is SO much wrong with it!
- No HTML structure
- They used a table for the navigation!!!
- Back in the old days of Web 1.0 there was no layouting...so people “McGyvere” things like this...





# SO MANY DIVS

- Web 2.0 was born and suddenly everything was a div. Until today!
- Ebay is a div website!



# SEMANTIC ELEMENTS

<article>

<aside>

<address>

<details>

<figcaption>

<figure>

<footer>

<header>

<main>

<mark>

<nav>

<section>

<summary>

<time>

<map>

<video>

<audio>

<picture>

<canvas>

<fieldset>

<form>

So many more...



# EXAMPLE STRUCTURE

```
<header>
  <nav>
    <ul>
      <li><a href=""></a></li>
    </ul>
  </nav>
</header>
<main>
  <h1></h1>
  <article>
    <h2></h2>
    <section>
      <h3></h3>
      <video src=""></video>
    </section>
  </article>
  <aside></aside>
  <article>
    <h2></h2>
    <figure>
      <img src="" alt="" />
      <figcaption></figcaption>
    </figure>
    <h3></h3>
    <form action="">
      <label for="">
        <input type="text" />
      </label>
    </form>
  </article>
</main>
<footer>
  <nav></nav>
  <address></address>
  <map name=""></map>
</footer>
```

# TOOLS

- 1. Code Validators for HTML, CSS or JS**
- 2. Screen Readers - External or Build-In Software**
- 3. VoiceOver / ChromeVox**
- 4. Browser Extension like Wave**



# CODE VALIDATORS

- Code validators are often the first step to check if your website is as accessible as it can be
- Good HTML validators:
  - AChecker
  - Nu Validator
- Good CSS validator:
  - CodeBeautify
- Good JS validator:
  - jsHint

The logo for AChecker, featuring the word "ACHECKER" in a bold, sans-serif font with a registered trademark symbol.The logo for Code Beautify, featuring a small icon of a head with a brain inside, followed by the text "Code Beautify" in a bold, sans-serif font.The logo for JSHint, featuring the text "JSHint" in a bold, sans-serif font, with "JS" in white on a dark background and "Hint" in dark gray on a lighter background.

# SCREEN READERS

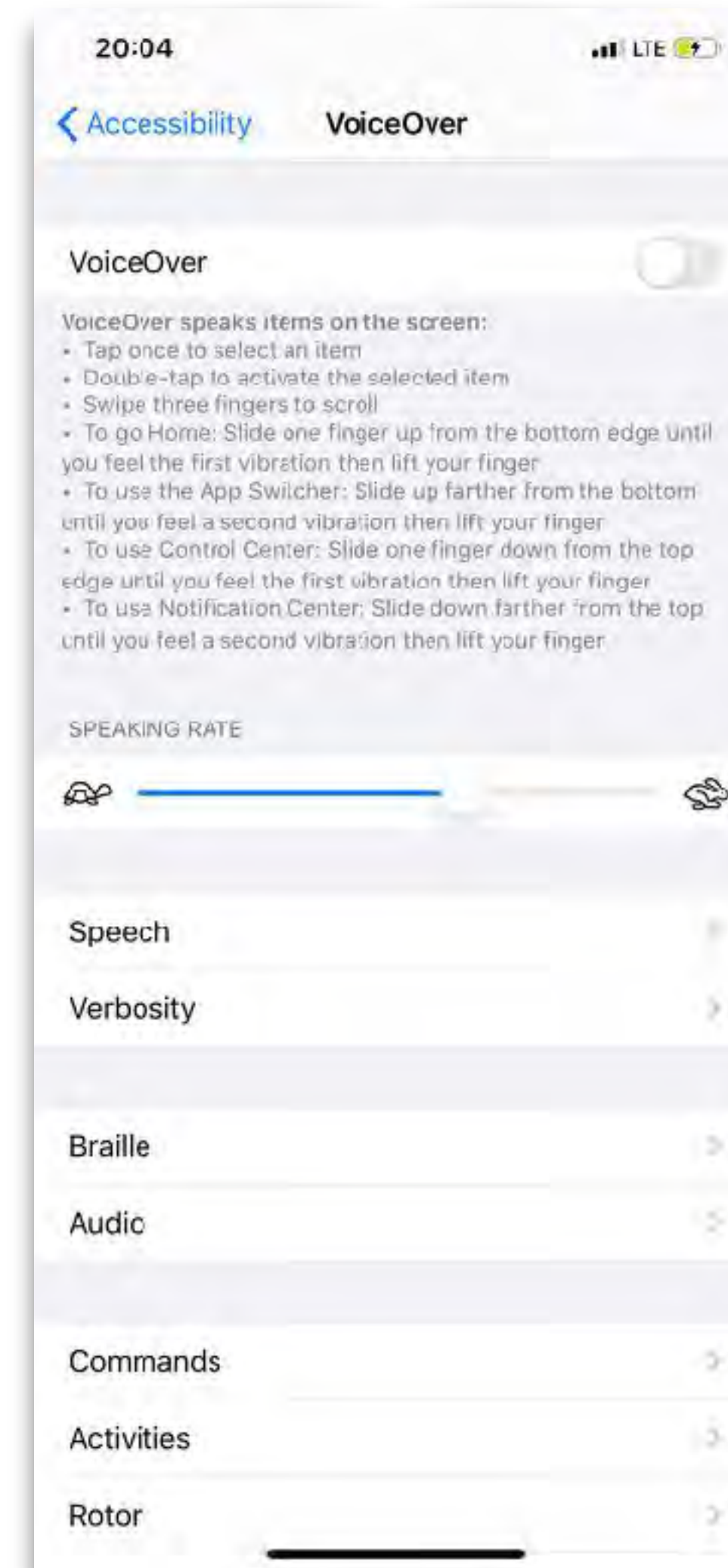
- A screen reader is a form of assistive technology that renders text and image content as speech or braille output.
- Screen Readers can be external machines, browser extensions or built-in machines like in mobile phones
- You can test every website simply by pressing the “tab” key.
- Popular screen reader: JAWS





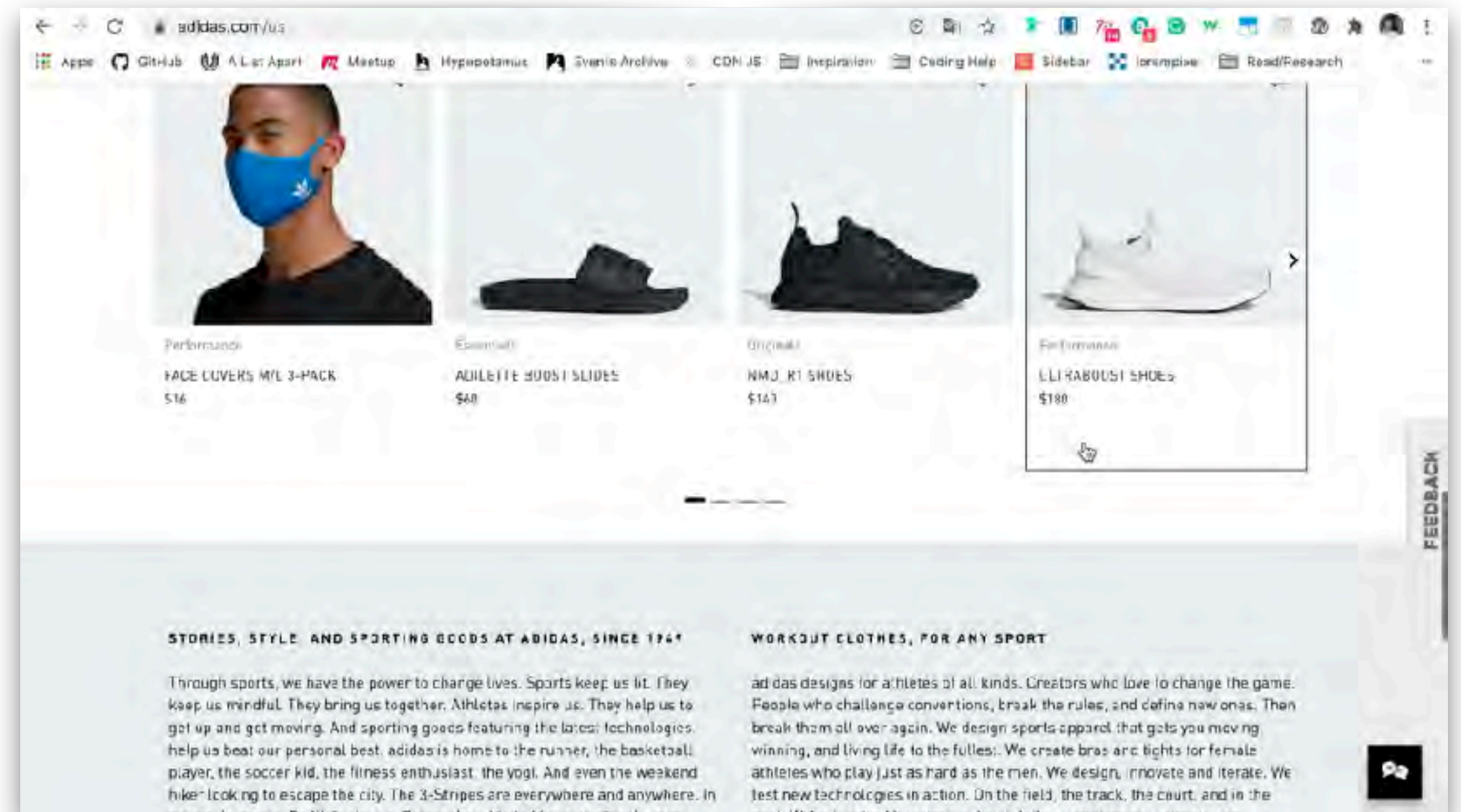
# VOICE OVER

- VoiceOver Software will read all HTML elements with focus out loud
- The software will read the elements out based on the order in the DOM tree
- You can change the speed of the spoken word



# BROWSER EXTENSIONS

- Use browser extensions to help with accessibility
- Good extensions:
- WAVE can be used as website as well as browser extension
- Click play on video to see how to use WAVE



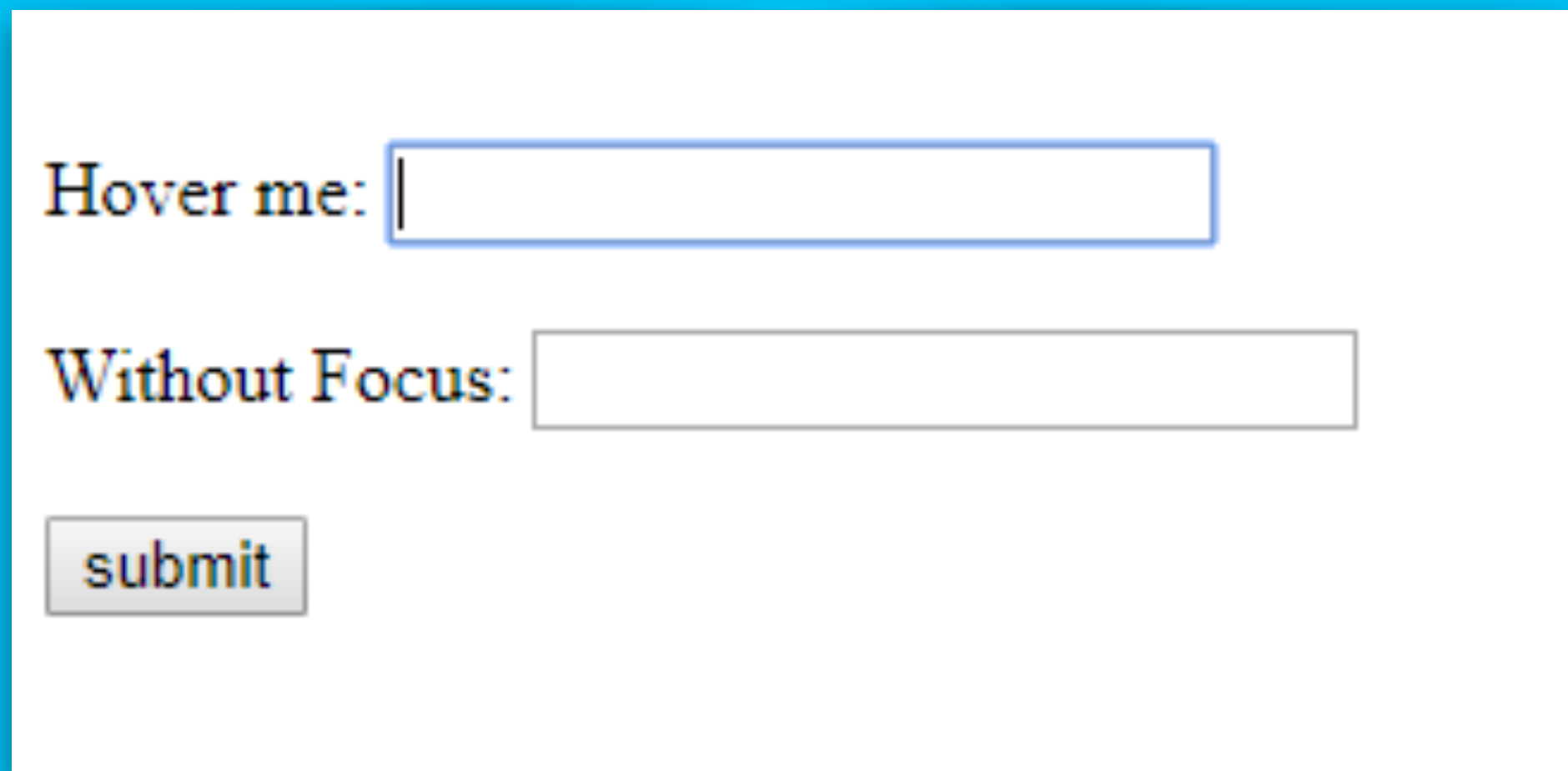


# WAYS TO IMPLEMENT

- 1. Focus**
- 2. Semantic HTML Elements**
- 3. Styles**
- 4. ARIA**

# 1. FOCUS

- Focus is the currently selected element on a page when using a keyboard to navigate
- Most HTML elements are focusable by default
  - Links
  - Form fields with input fields
  - Buttons
- HTML elements that the user does not interact with are not accessible by default
- Order is determined by DOM order - which you can change!



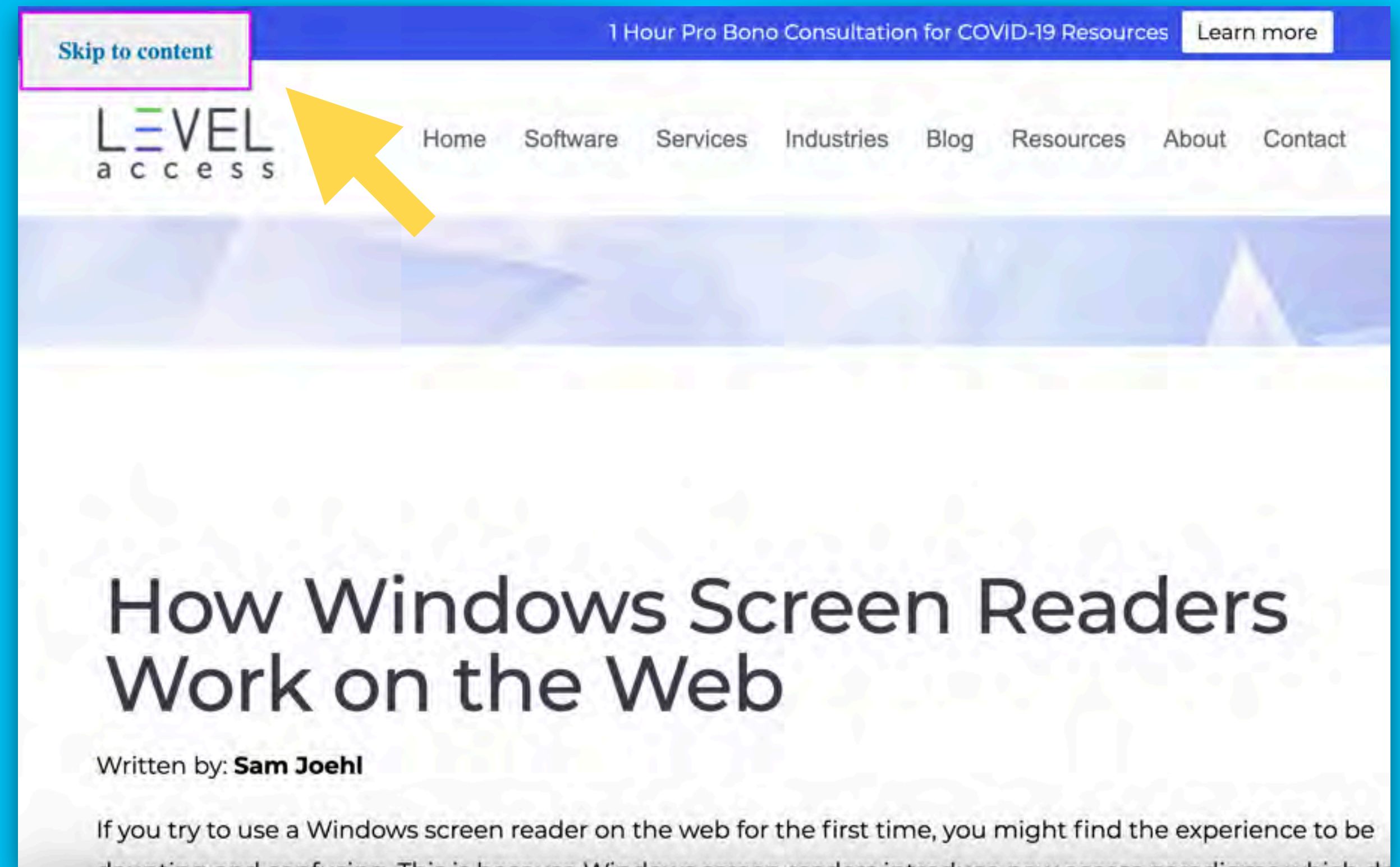
Hover me:

Without Focus:



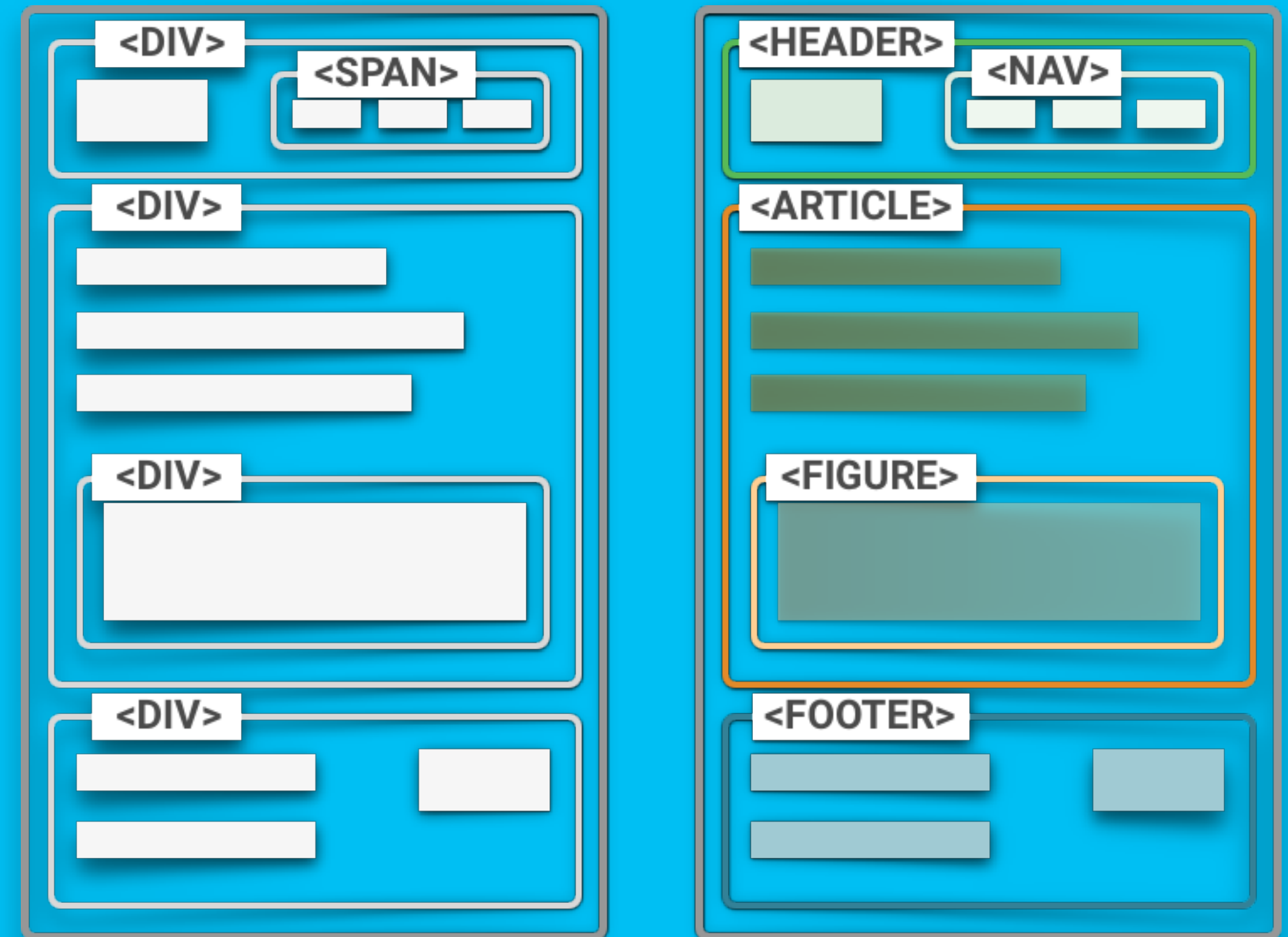
# 1. FOCUS CONTINUED

- Skip-to-content-button
- Remember to include `tabindex="-1"` on the target of the skip link. IE and Chrome both require this attribute to be present to make the skip link work consistently!
- Source
- **Tab Order:** Structure your HTML in the same way you would expect the user to ready the website, top to bottom, left to right
- **Tab-Index-Attribute:** Use to set focus order when you can't change the HTML order, use sparingly



## 2. SEMANTIC HTML

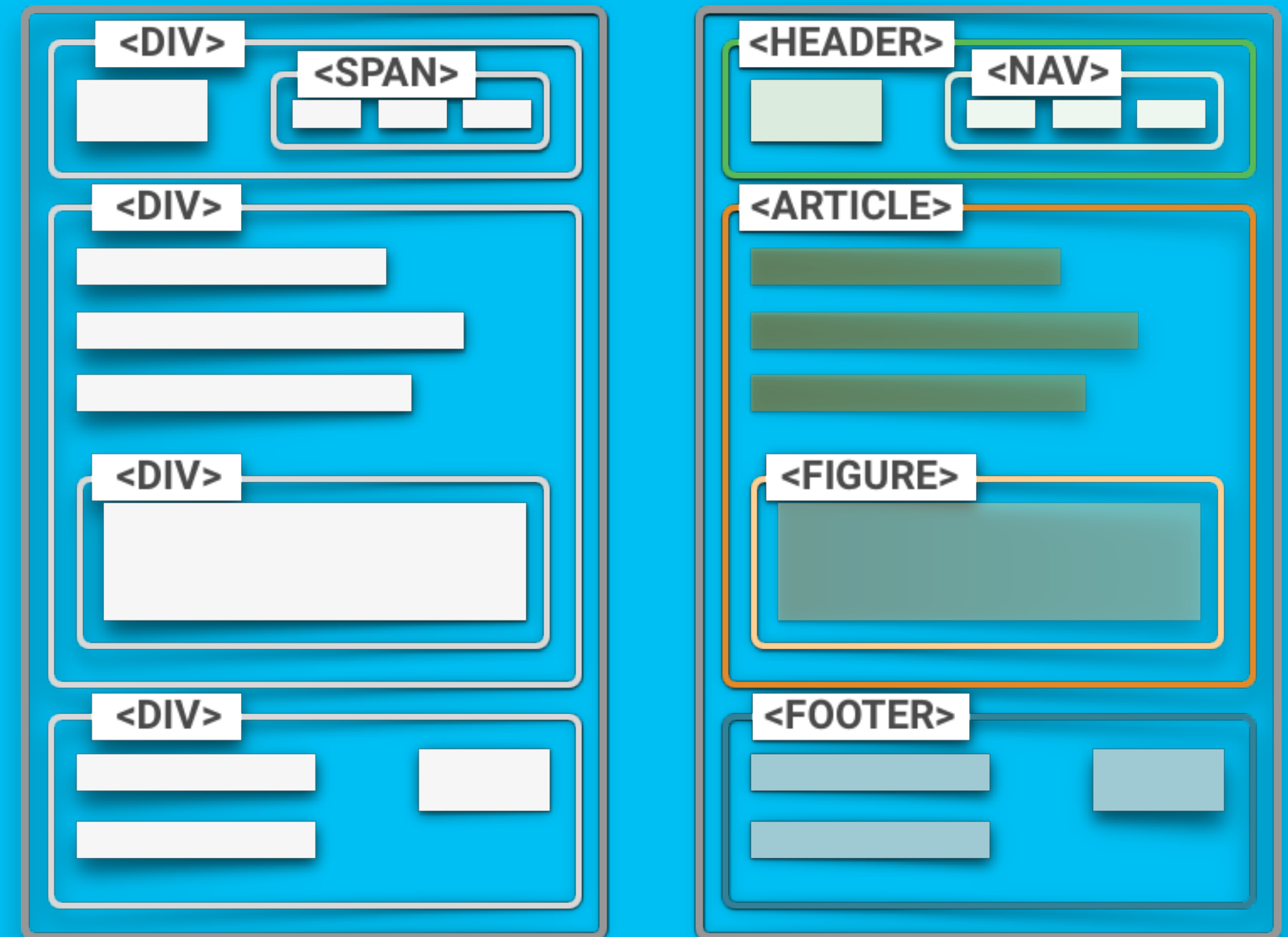
- Not only important for SEO but also for accessibility
- Instead of using just divs, group your website into meaningful section
- General guideline: header with nav, main including articles, sections and possibly asides and lastly the footer





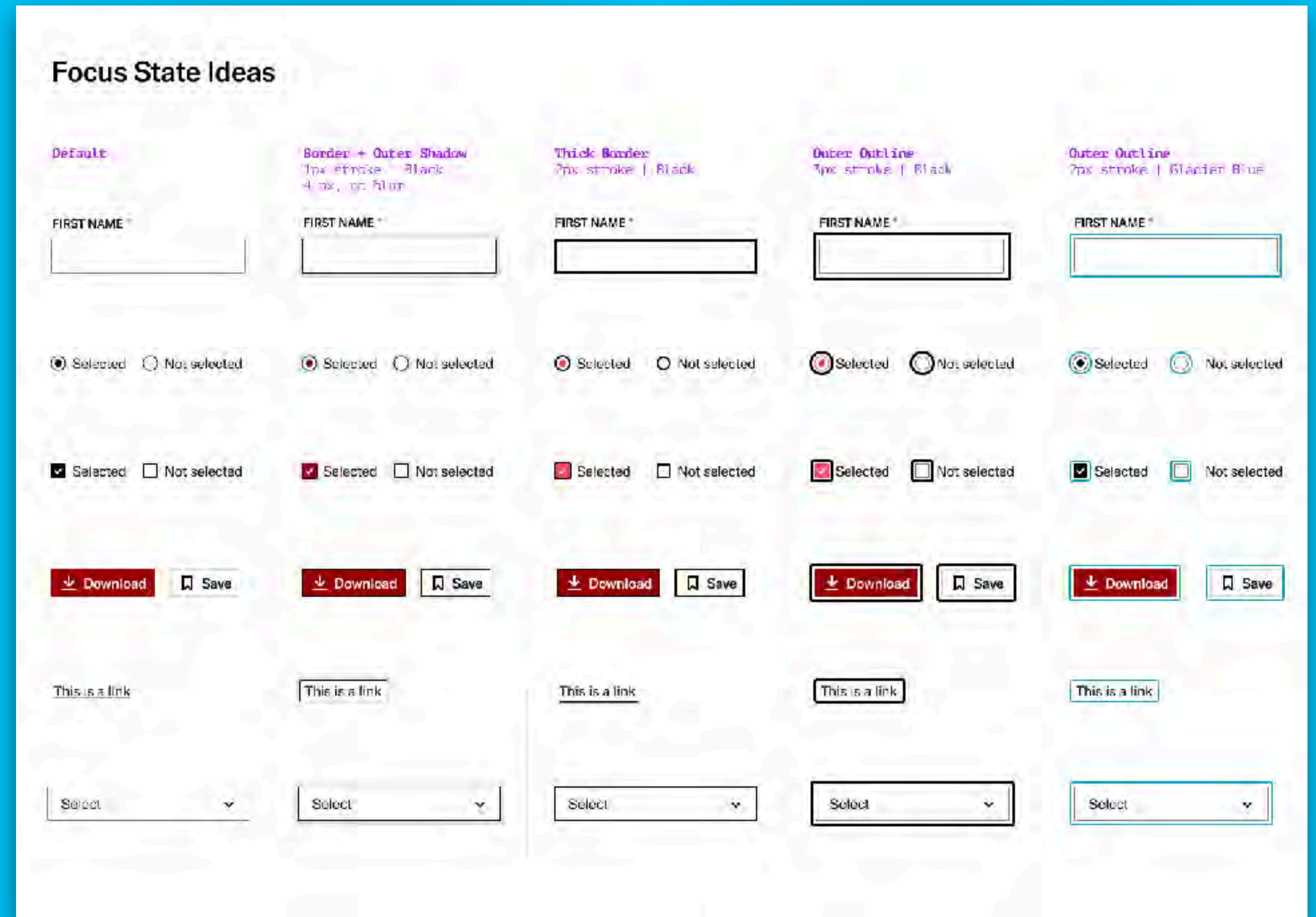
## 2. SEMANTIC HTML CONTINUED

- Use the form-tag for forms along with input fields and labels that describe the input field
- Always define the type of the input field. Don't use type: text for a password field!
- If no alt-text is used, screen readers will read the file name, which are often only symbols and number. Use alt-text on images so screen readers can read them out loud
- Use an empty alt-tag for decorative images alt="".



# 3. STYLES

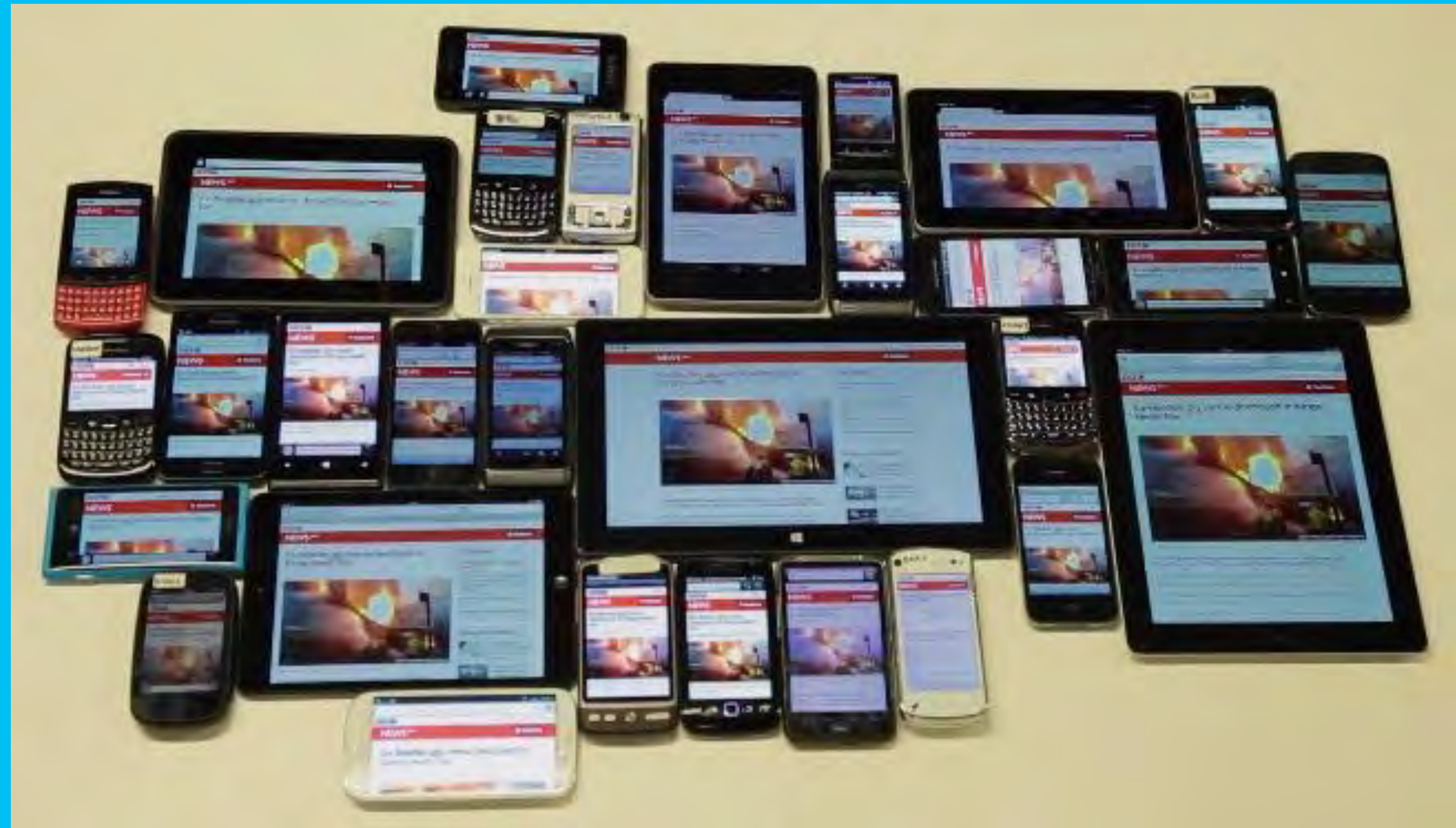
- Generally concerns the responsiveness, size and contrast of the website





# 3. STYLES- RESPONSIVENESS

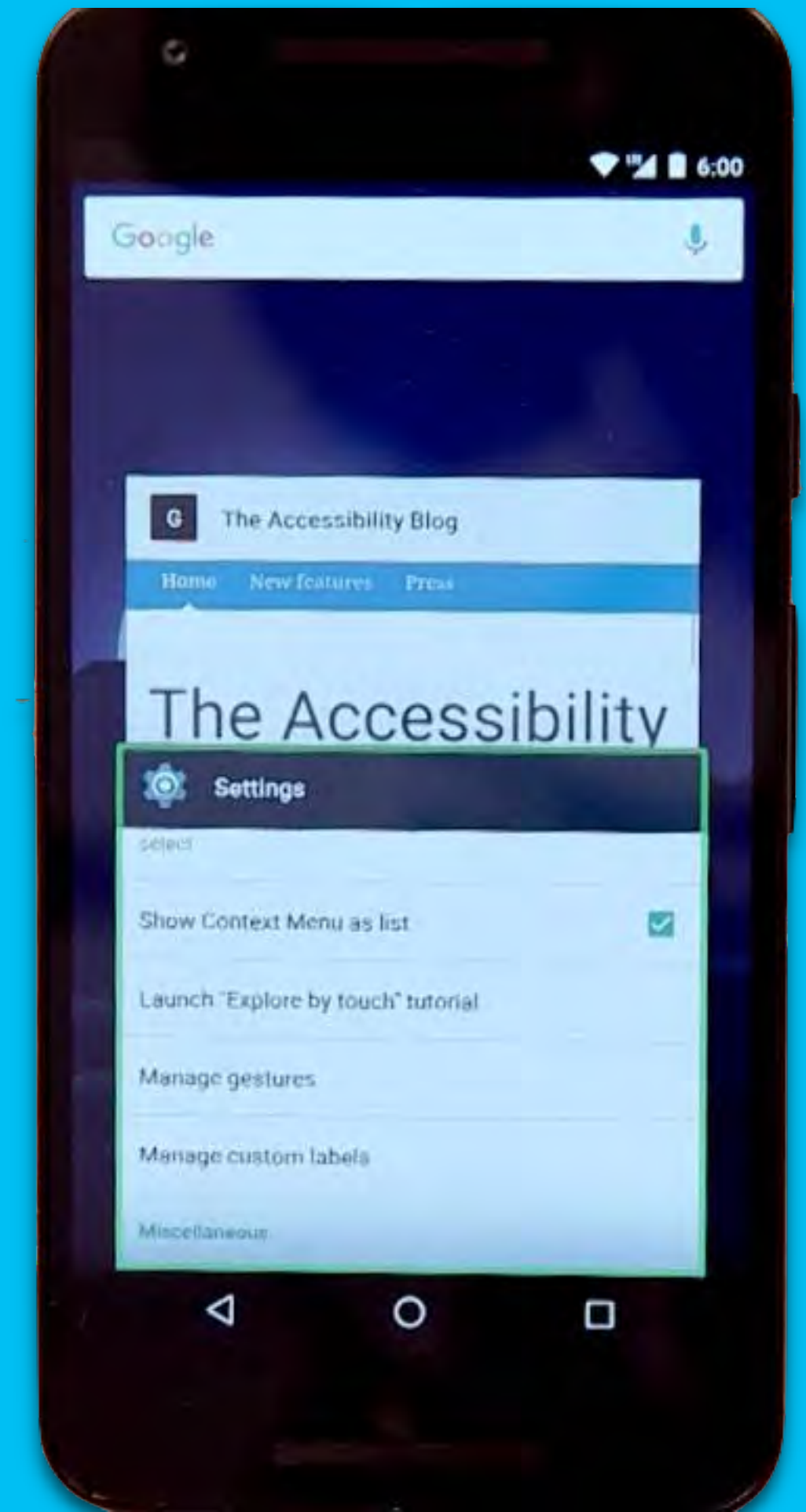
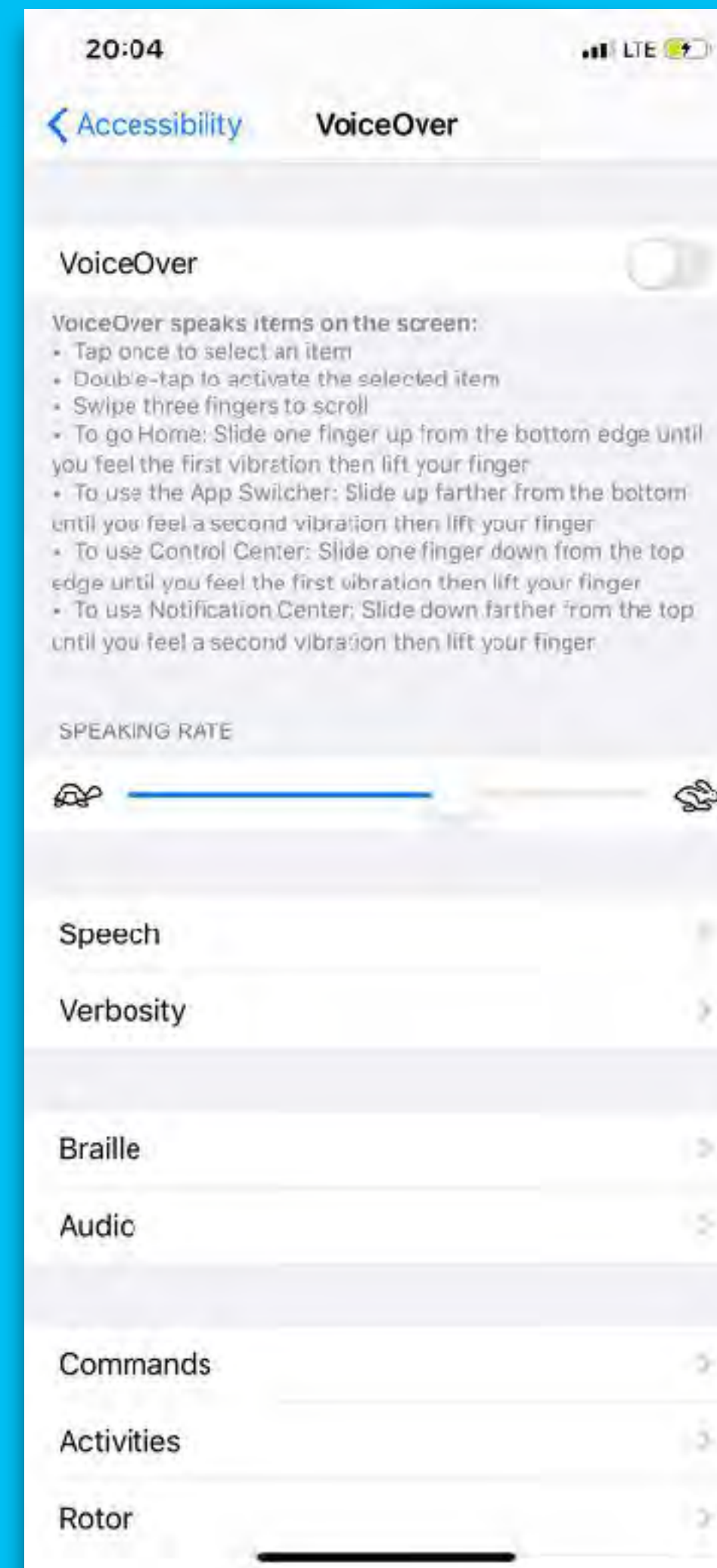
- Responsiveness is best tested on different devices with different operating systems
- Test minimum: iOS and Android
- For mobile devices: different operating systems have different built-in screen readers
- Don't rely on external hardware like bluetooth keyboards





# 3. STYLES - DESKTOP VS. MOBILE

- 82% of screen readers use a mobile device!
- Be sure to test on as many devices as possible
- The biggest difference in testing for mobile vs. desktop is the screen reader navigation - you can't use tab on mobile
- Learn essential finger commands

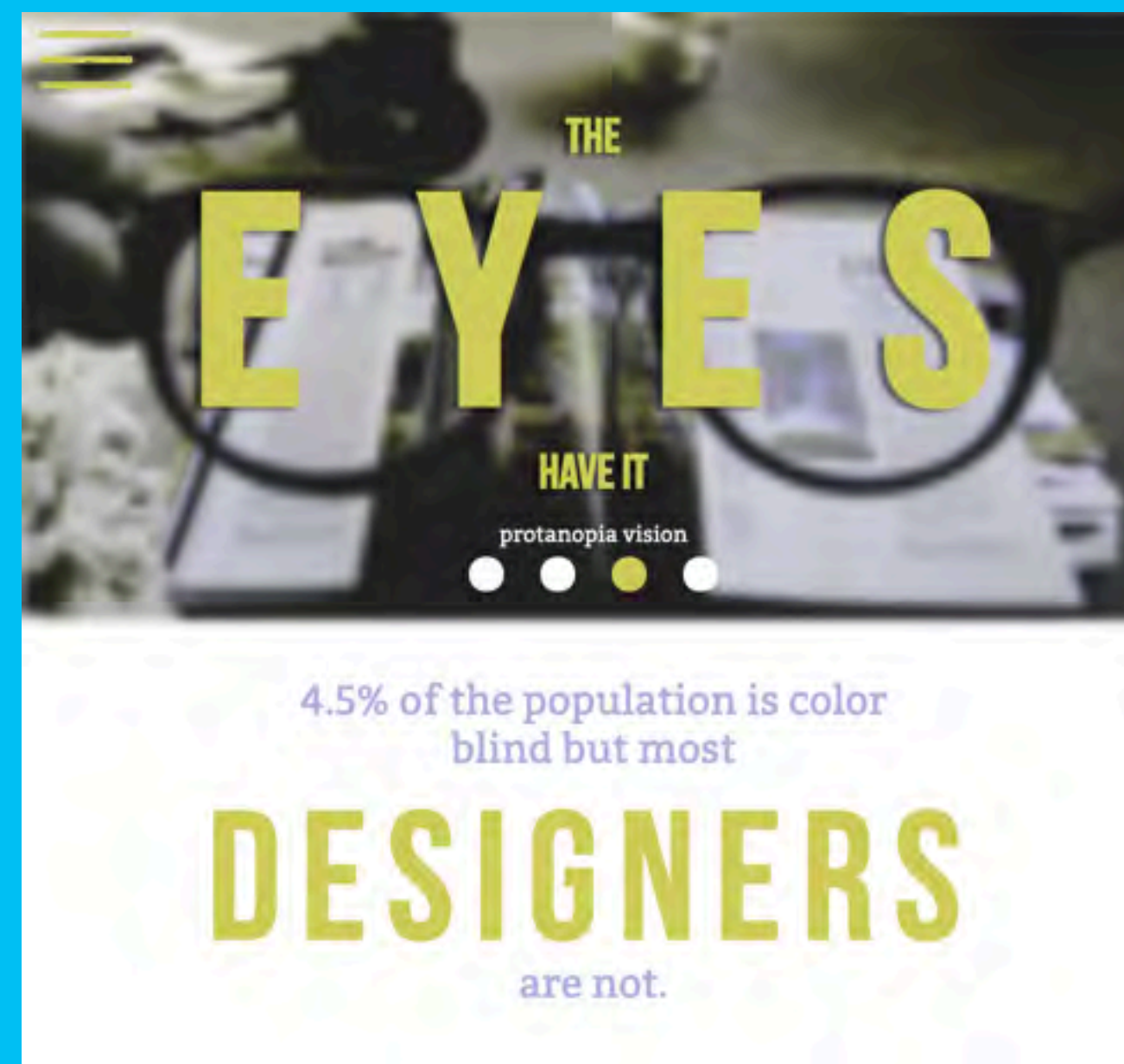




# 3. STYLES- CONTRAST

- Set body size to a minimum of **18px!**
- Use rems and ems instead of pixels so typography can grow with the size of screen
- Use well-readable fonts and provide native alternatives (also, never use more than 2-3 fonts per page)
- Use contrast checker tool: [Contrast Checker](#)

Made with: [Color Oracle](#)





# 4. ARIA

- ARIA allows you to specify attributes on elements which then modify the way they are integrated into the accessibility tree
- Modify semantics
- Adds extra labels and description not available in regular HTML5
- DOES NOT change elements inherent behavior, e.g. make element focusable
- No ARIA is better than bad ARIA!

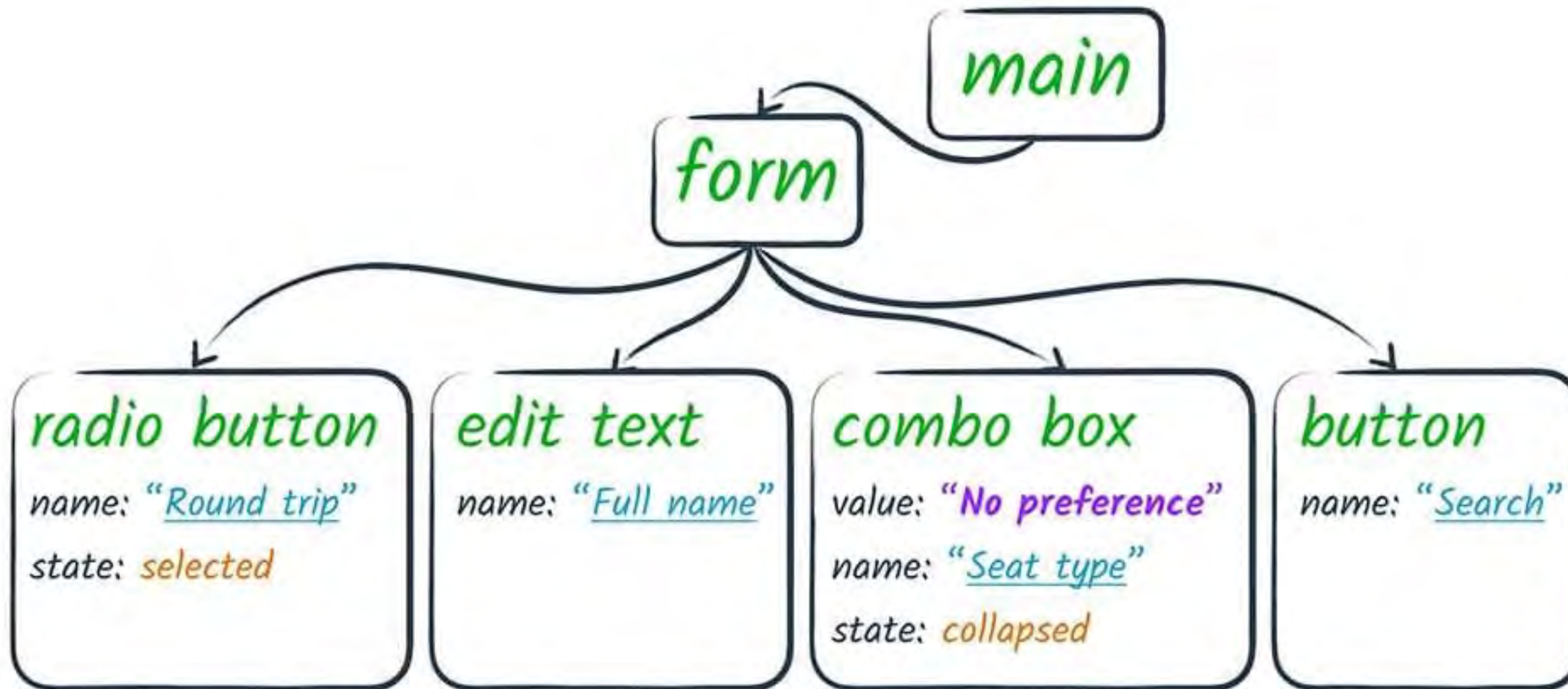
Web  
Accessible  
Initiative

Accessible  
Rich  
Internet  
Applications



WAI - ARIA





Or Accessibility Object Model (AOM) contains accessibility-related information for HTML elements which focus on:

- name
- description
- role
- state

# ACCESSIBILITY TREE



# ARIA TERMINOLOGY

1. **aria-label**: will overwrite native label mechanisms

```
<button class="hamburger" aria-label="menu">Button Image</button>
```

2. **aria-labelledby**: used to show relationship between two tags, e.g. label and input elements

—> can be used with multiple elements

3. **aria-owns**: alters the relationship between elements in DOM tree

e.g. siblings can now be coded as parent-child pair

4. **aria-hidden**: prevents screen readers from seeing elements and reading them out

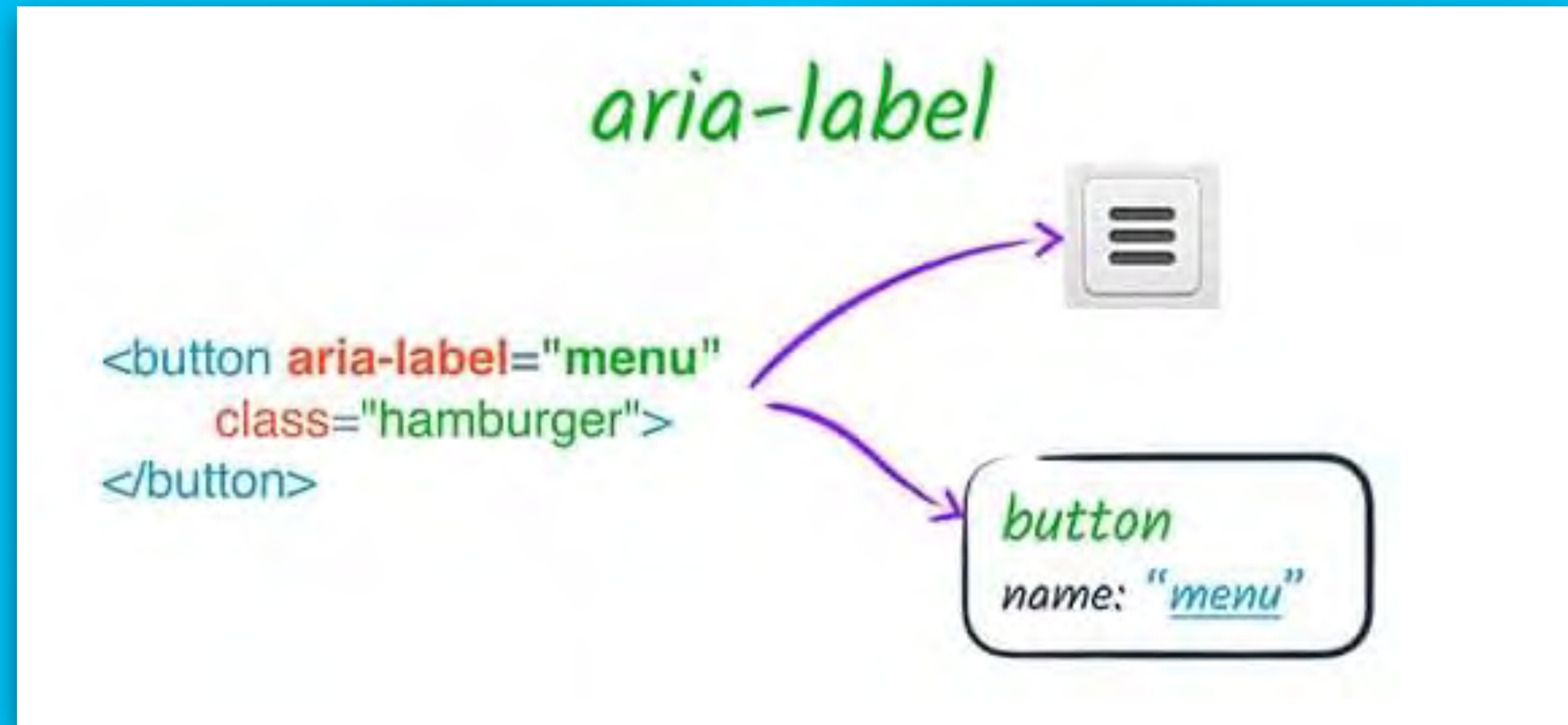
5. **aria-live**: notify user when content is updated during session (e.g. score during sports game, price on a product)

```
<h3 class="connection-status" aria-live="polite">Reconnecting...</h3>
```



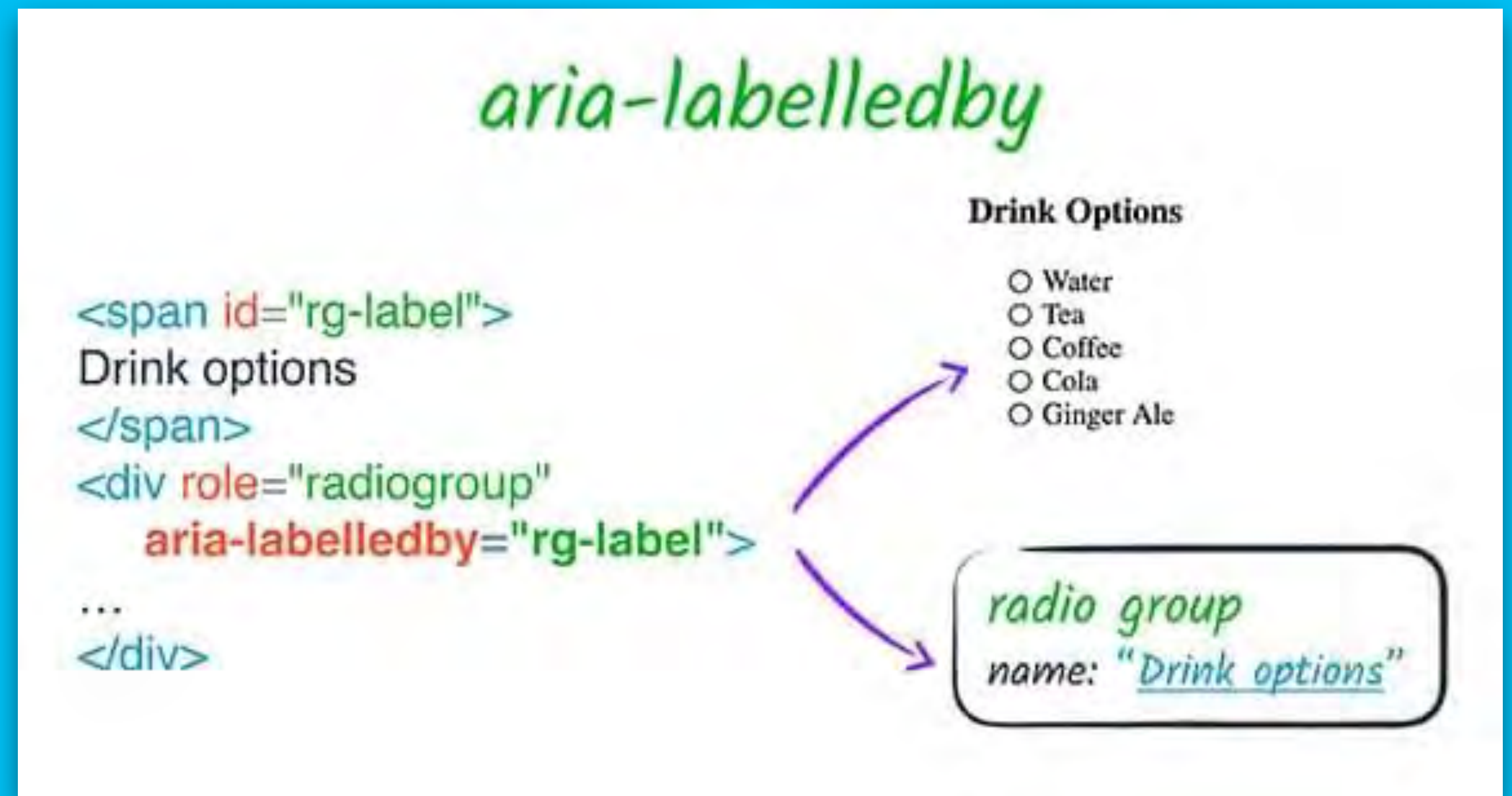
# ARIA-LABEL

- Allows us to specify a string to be used as the accessible label.
- Overrides any other native labeling mechanism, such as a label element — for example, if a button has both text content and an aria-label, only the aria-label value will be used



# ARIA-LABELLEDBY

- Is a relationship attribute
- Allows us to specify the ID of another element in the DOM as an element's label
- Ideally you would just use a label and an input field to achieve the same thing
- May be used on any element, not just labelable elements
- Lets you refer to elements that are hidden and would otherwise not be in the accessibility tree, e.g. add a hidden span next to an element you want to label, and refer to that with aria-labelledby.



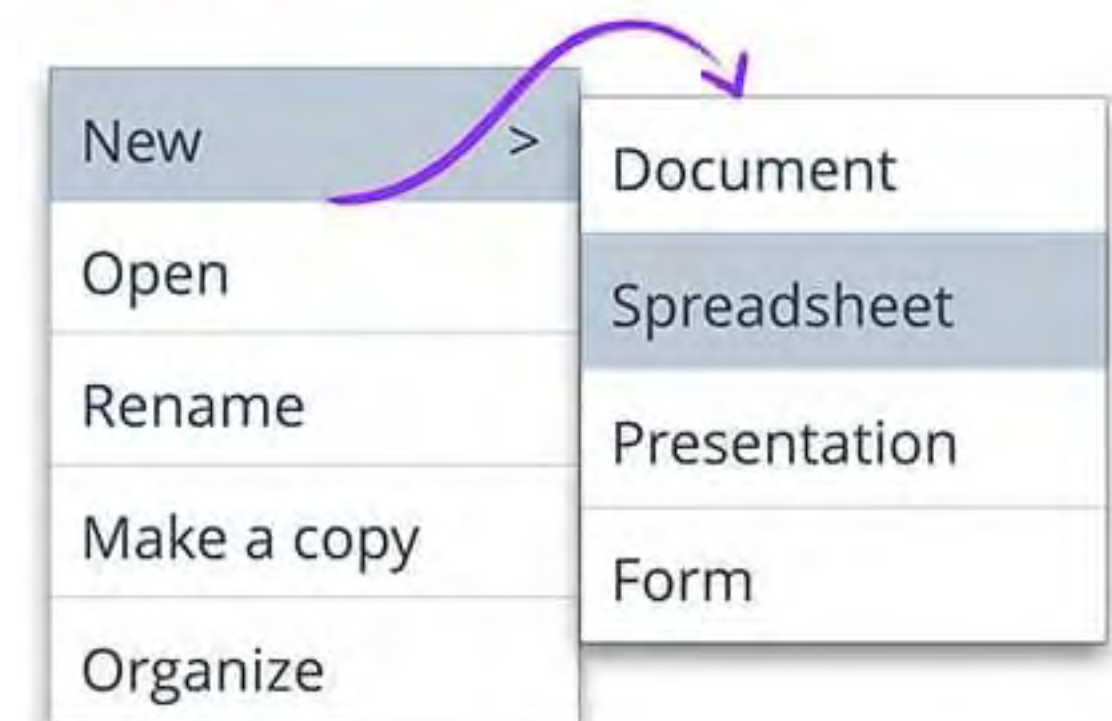


# ARIA-OWNS

- Allows us to tell AT that an element that is separate in the DOM should be treated as a child of the current element, or to rearrange existing child elements into a different order
- For example, if a pop-up sub-menu is visually positioned near its parent menu, but cannot be a DOM child of its parent because it would affect the visual presentation, use aria-owns to present the sub-menu as a child of the parent menu

```
<div role="menu">  
  <div role="menuitem"  
    aria-haspopup="true">  
    New  
  </div>  
  <div aria-owns="submenu">  
  </div>  
  <!-- more items... -->  
</div> <!-- menu -->  
<div role="menu" id="submenu">  
  <div role="menuitem">  
    Document  
  </div>  
  <!-- more items... -->  
</div> <!-- submenu -->
```

*aria-owns*



# 5. FORMS

- Should be one column
- Top align labels
- Group labels with their input
- Avoid call caps
- Show all selection options if under 6
- Resist using placeholder text as labels
- Make CallToActions descriptive
- Group related information

The screenshot shows the Big Cartel 'Sign up' form, which is a single-column layout. At the top, the Big Cartel logo is on the left, and navigation links 'Tour', 'Examples', 'Buzz', and 'Pricing + Sign Up' are on the right. The 'Sign up' heading is in a large, white font on a green background, with a subtext: 'Payment is month-to-month. Upgrade, downgrade, or cancel at any time.' Below this, a green circle with the number '1' indicates the first step: 'Tell us about your store'. The form fields are arranged vertically, each with a label to its left: 'Store name' (text input), 'Store address' (text input with '.bigcartel.com' as a placeholder), 'Password' (text input), 'Password again' (text input), 'Store type' (dropdown menu with 'Accessories' selected), 'Country' (dropdown menu with 'United States' selected), 'Time zone' (dropdown menu with '(GMT-05:00) Eastern Time (US & Canac)' selected), and 'Currency' (dropdown menu with 'U.S. Dollar (USD \$)' selected). The form is set against a light green background with a subtle pattern.



# RESOURCES

**Book a Flight**

**Form Controls**

**WebAim Forms Best Practice**

**Semantic HTML with Forms**

**Reduced Motion Preference in CSS**

**CSS Loading Spinners**

**ARIA Authoring Practices**

**QUESTIONS?**



# HOMEWORK

- Expand your website from last weeks homework and add accessibility features. Also, add in feedback you've received during class.
- If not already responsive, make your site responsive and use semantic HTML elements wherever it makes sense.
- In addition to your landing page, add one subpage. The topic of the subpage is up to you. I'm suggesting a product page to be able to work with different links or a contact page, that includes an input field.
- At a minimum, your site should include:
  - A Name field
  - An Email field
  - At least one set of radio buttons
  - At least one checkbox
  - All inputs should have an associated label.
  - A custom component like an accordion, carousel or combobox where you have to demonstrate ARIA usage to make the component accessible
  - The entire site should be navigable with keyboard only, and each field should be clearly described when read by a screen reader.
  - Your site must have no warnings or errors using accessibility validation chrome extension that we looked at in class.
- You will be graded on the correct usage of tools to help you achieve good usability.
- Bonus Points if you are able to implement a "Skip to Main Content" button.
- Submit a link to the git repository for this assignment.

**FIN**