Introduction to Web Accessibility for Front-end Developers

Global Accessibility Awareness Day 2025

What you need to know

- Digital accessibility & disabilities
- Accessibility guidelines and standards
- Accessible development
  - Semantic HTML
  - CSS accessibility
  - JavaScript and ARIA accessibility
- Testing for accessibility

Digital accessibility and disabilities

Web accessibility definition

“Web accessibility means that websites, tools, and technologies are designed and developed so that people with disabilities can use them.”

-- W3C Web Accessibility Initiative

The disability community

Worldwide

- 1.3 billion people (one in seven) have a disability

In the United States

- Approximately 64 million people live with disabilities

People with disabilities

- Work, go to school
- Use technology
- Have buying power

Digital accessibility issues are far too common

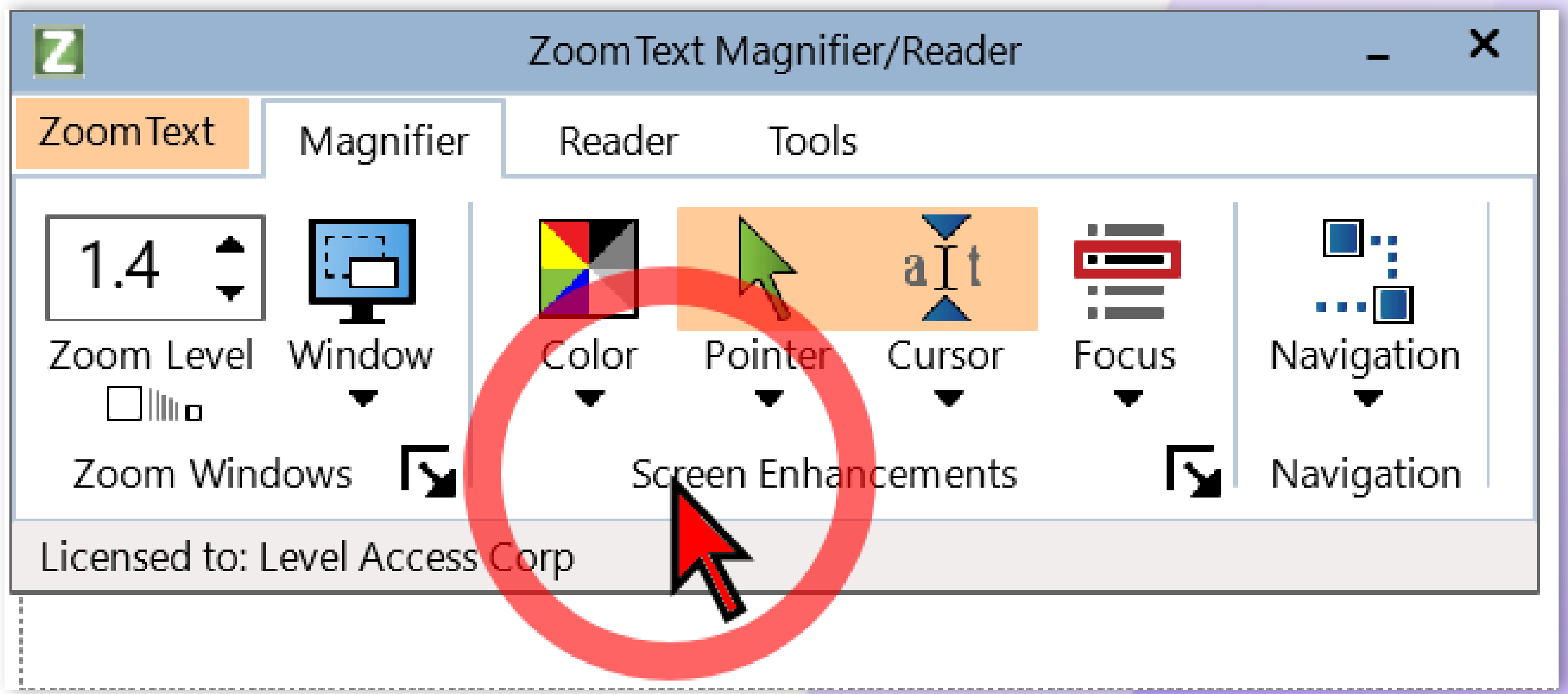
Approx. 95% of the top 1 million home pages were detected to have WCAG 2 (Web Content Accessibility Guidelines) failures

- Common categories were low contrast text, missing alt text, empty links/button, missing form field labels, and missing document language
- Missing, questionable, or duplicative alt text for images was reported as 33.3%
- Low contrast text was the most common issue at 83.6%

Source: The WebAIM Million

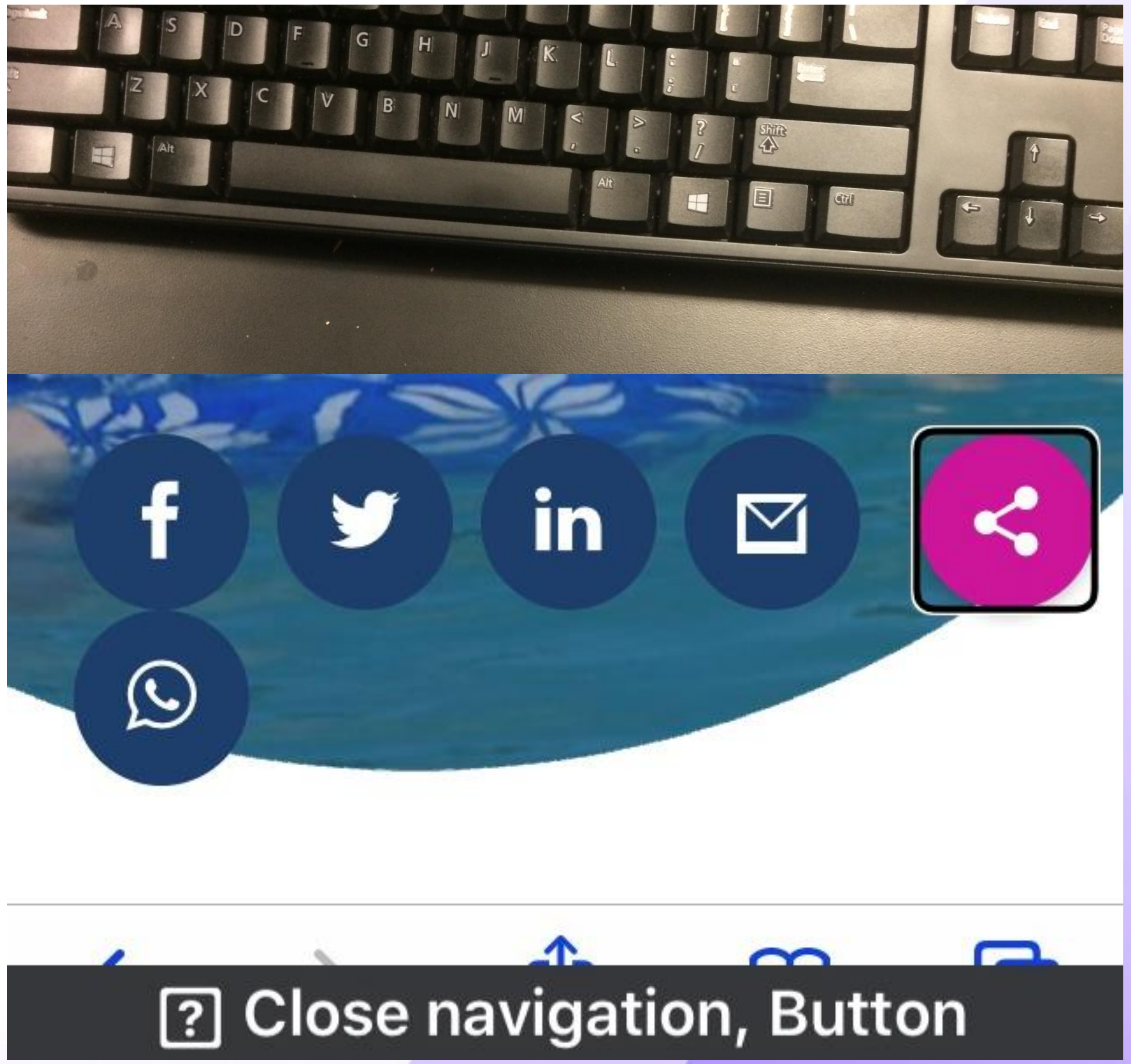
Technology used by people with disabilities

- People with disabilities may use assistive technology or access features
- Can be prebuilt into platforms like Windows, macOS, iOS, Android, Fire TV, etc.
- You likely use accessibility features every day without knowing (e.g., zoom on smartphones, audiobooks, captions, predictive text, or magnification).



Examples of accessibility features

- Screen reader and text to speech
- Browser zoom and text enlargement
- Captions, audio description, and transcripts
- Keyboard access, on-screen keyboard
- Eye tracking and switch control
- Voice control and input
- Word prediction, reduce/pause animation, plain language



Accessibility guidelines and standards

Digital accessibility standards

WCAG

- Adopted into global regulations (Section 508, EN 301 549)
- Current version is WCAG 2.2
- WCAG 3.0 is expected in 3+ years
- Relevant to those, designing, building, and buying technology
- Minimum level of accessibility; do not address the needs of all users

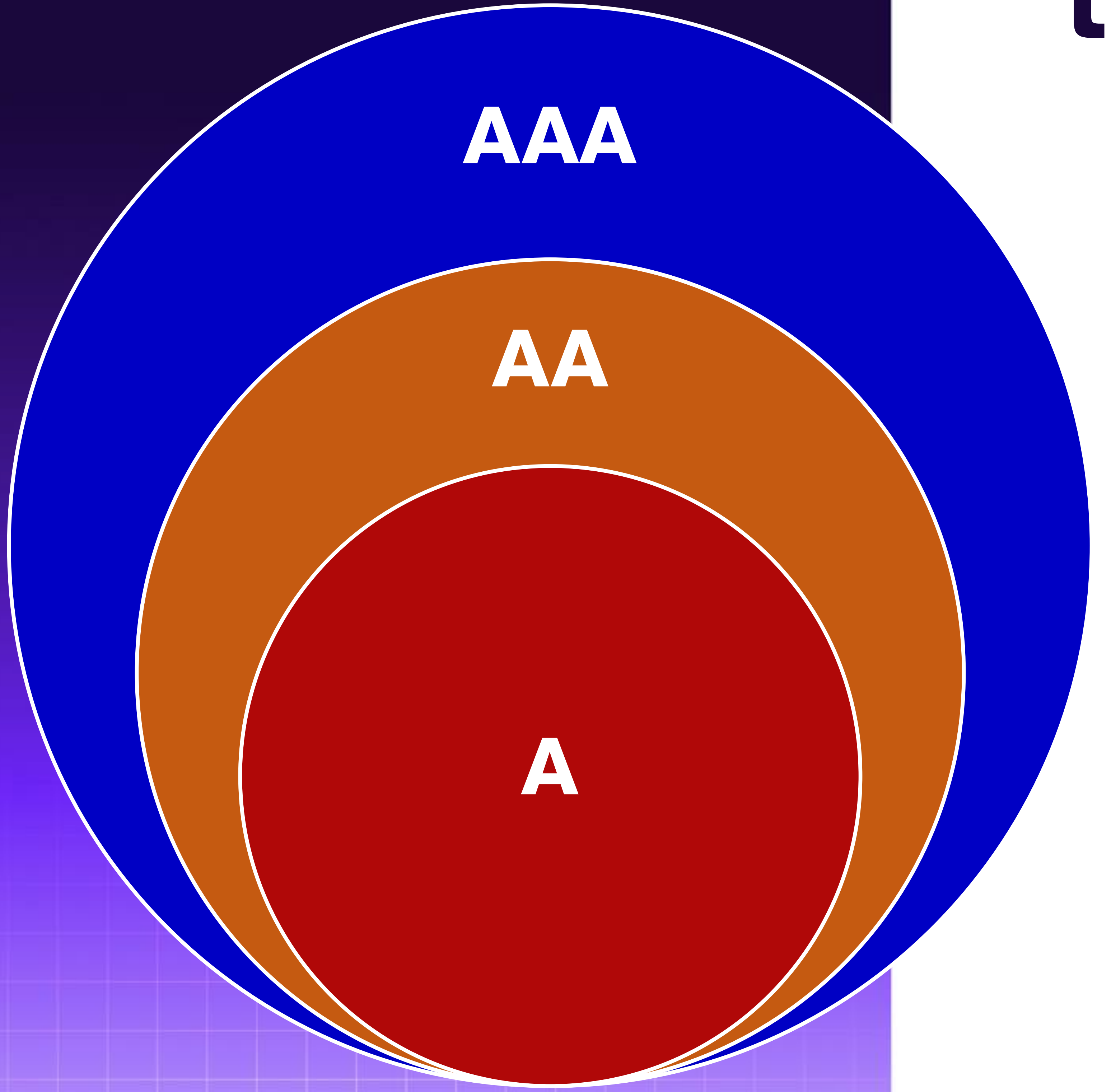
Section 508 and EN 301 549 layer in addition standards for:

- Software including mobile apps
- Documents
- Video
- Two-way voice communication
- Support
- Hardware

Web Content Accessibility Guidelines (WCAG): **P.O.U.R.**

Perceivable	Operable	Understandable	Robust
Text alternatives for non-text content	Keyboard operable	Text is readable and understandable	Content is compatible with current and future user tools
Captions and other alternatives for multimedia	Time affordances	Content appears and operates in predictable ways	
Content can be presented in different ways	Easy to navigate, find content, and know where you are	Users are helped to avoid and correct mistakes	
Content is easier to see and hear	Allows for different input modalities beyond keyboard		

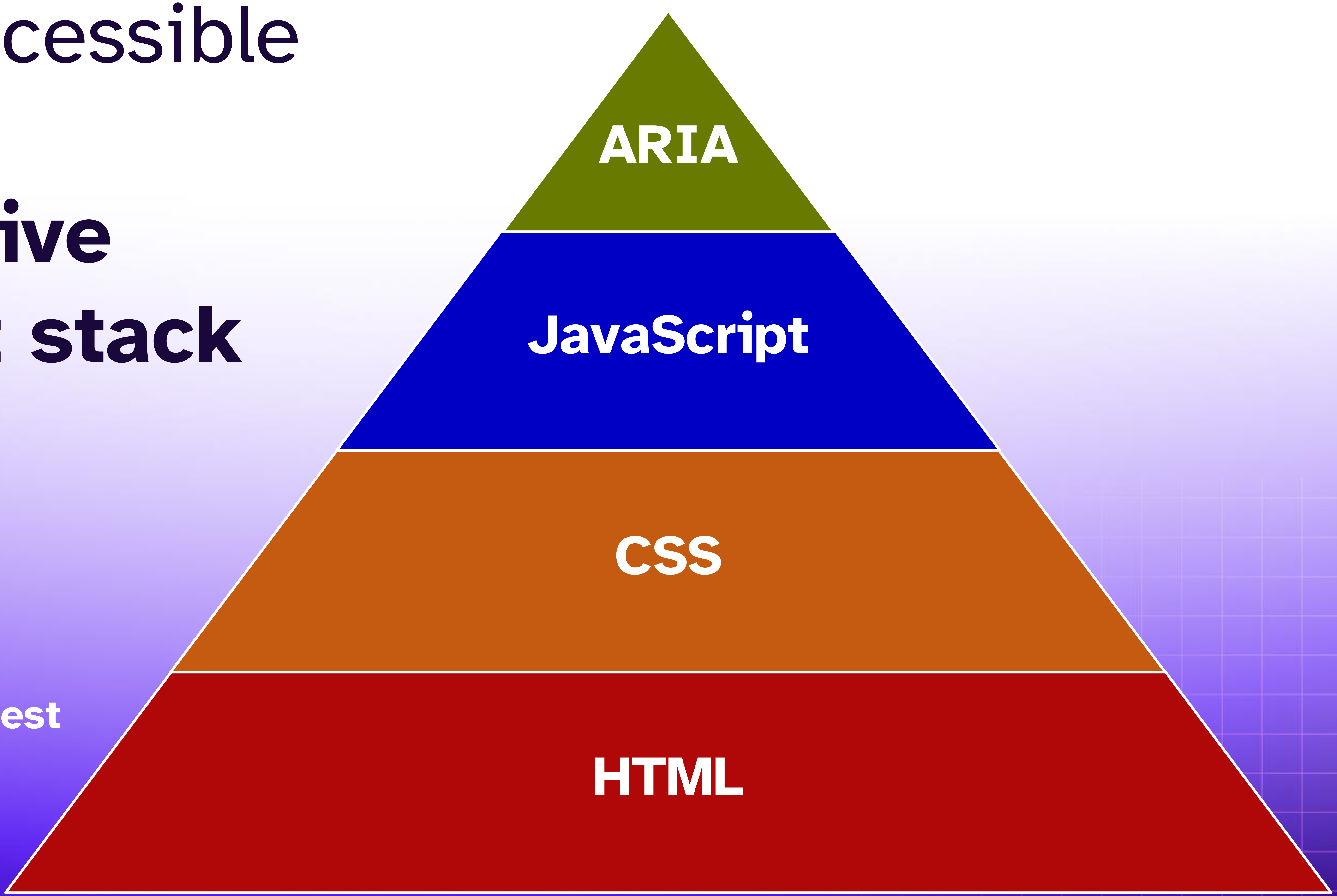
Success Criteria levels



- Level A: Lowest level of conformance
  - Bare minimum requirements
- Level AA: Middle level of conformance
  - The typical legal and / or organizational minimum
- Level AAA: Highest level of conformance
  - More robust support for largest group of users

Accessible development

Building an accessible foundation: **The progressive enhancement stack**



Solve the problem the lowest in the stack you can.

Semantic HTML

Building an accessible foundation: **HTML**

Solve the problem the lowest in the stack you can.

Structure considerations

Use the document's inherent **structure** to reinforce the **meaning** of the visual information.

Use the most **suitable and valid elements** to express the structure of the page.

Create a **hierarchy and order** of structural elements to identify key regions.

Avoid implicit implementations of structures into your content.

level access

Semantics

When you use standard control you get built-in accessibility support

Landmarks / regions

(header, banner, main, article, section)

Headings

(h1…h6)

Data tables

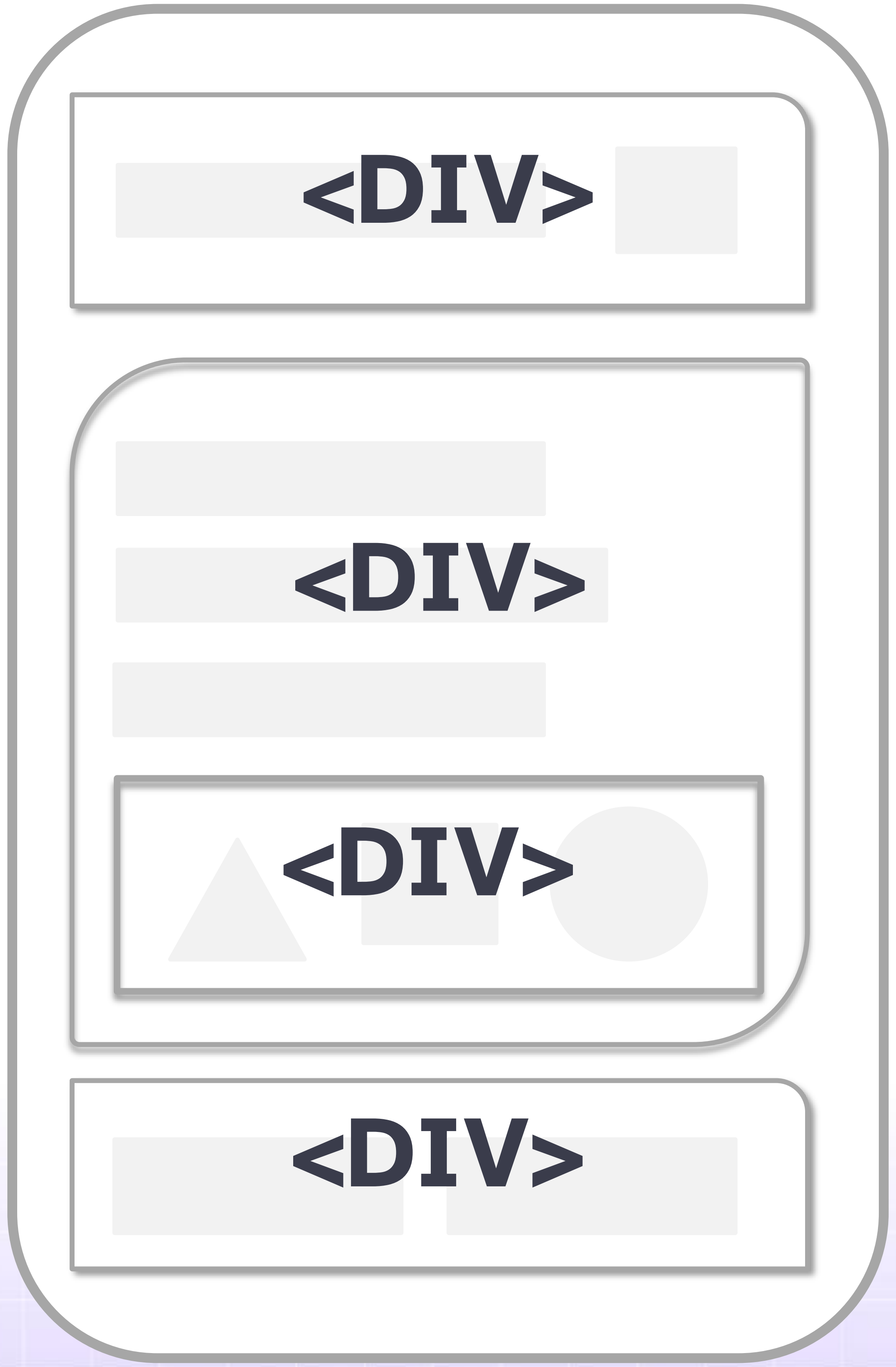
Labels and inputs

Fieldset / legend

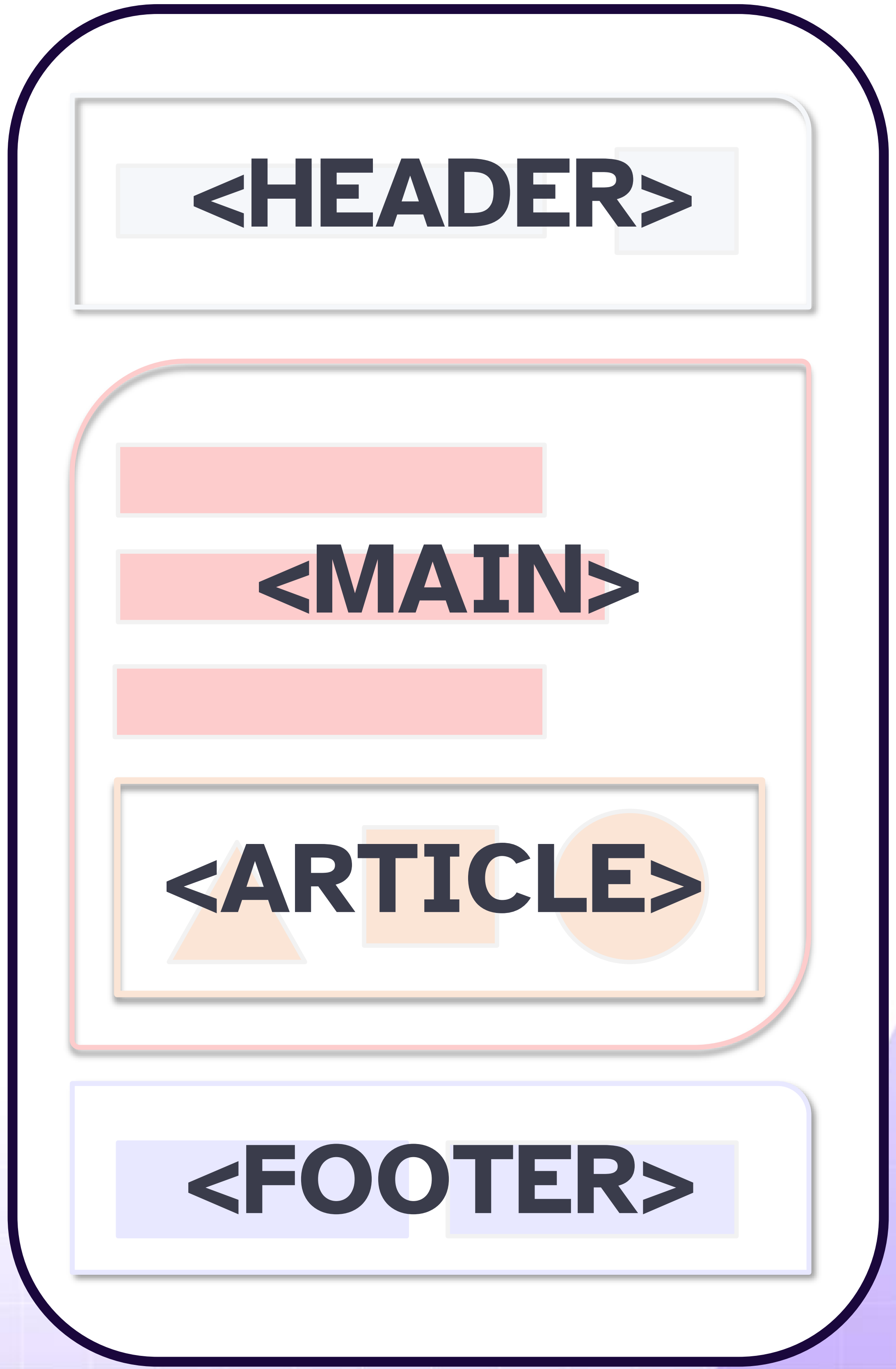
Lists

(dl, ul, ol)

Ambiguous structure
(AKA "<DIV> soup")



Identifiable sections
(AKA "Semantic markup")



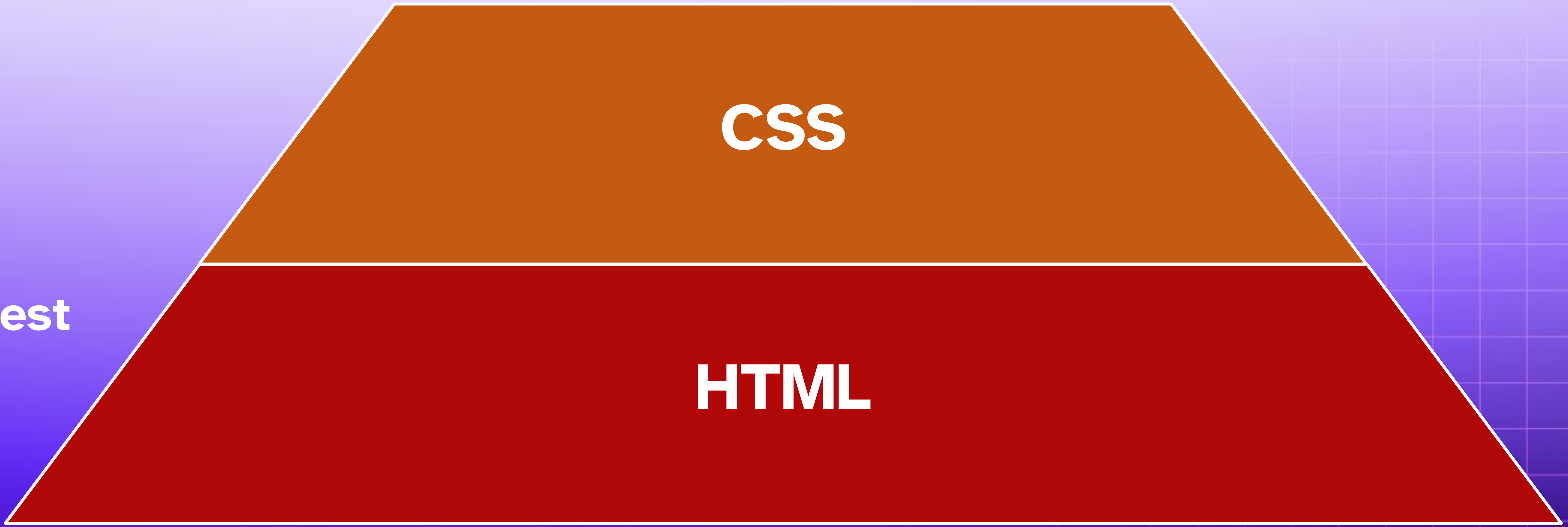
Forms

- Proper use of <label> and for attribute
- Error messages and validation
- Fieldset and legend for groups
- Placeholder vs. label

CSS accessibility

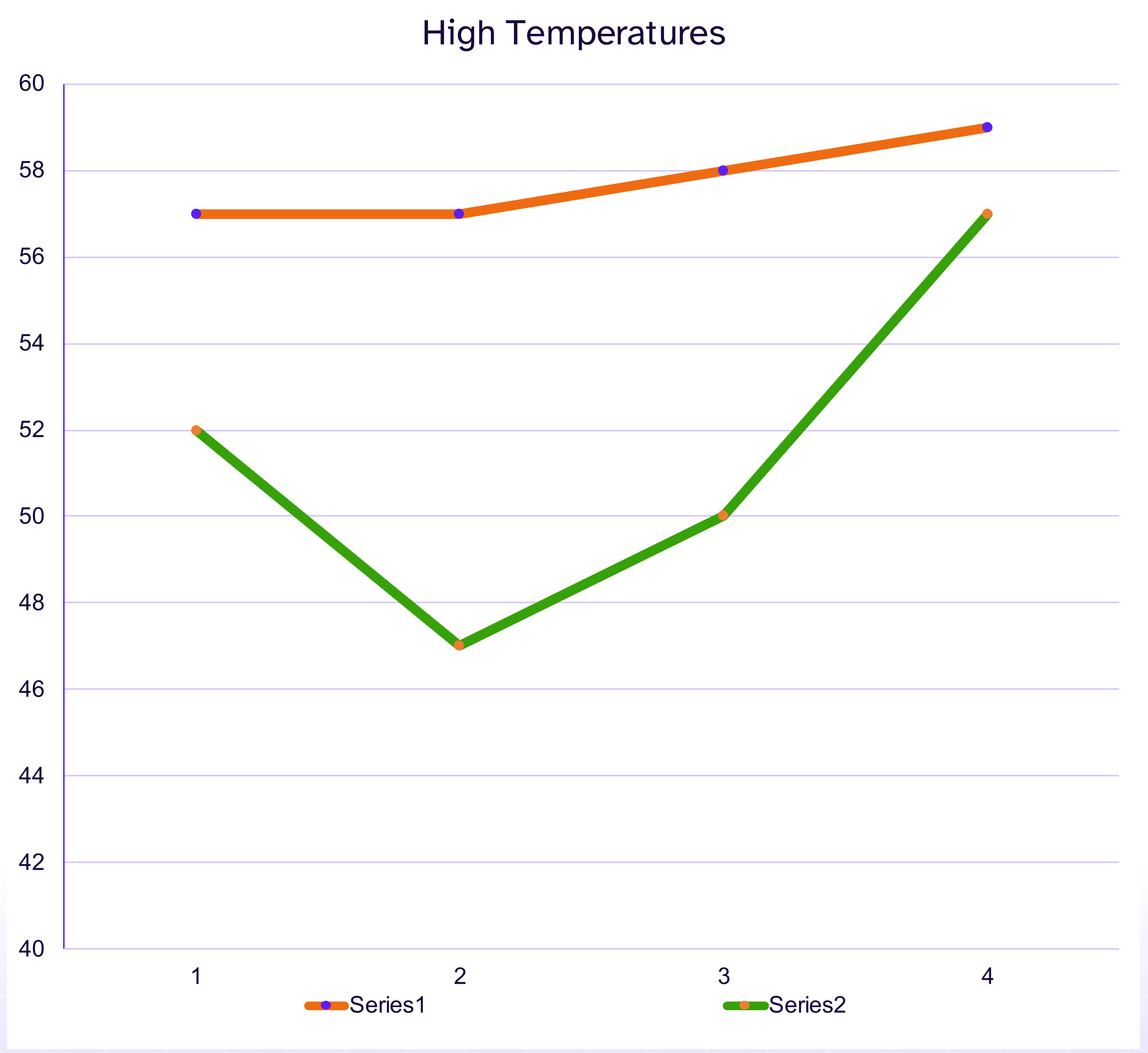
Building an accessible foundation: **CSS enhancement**

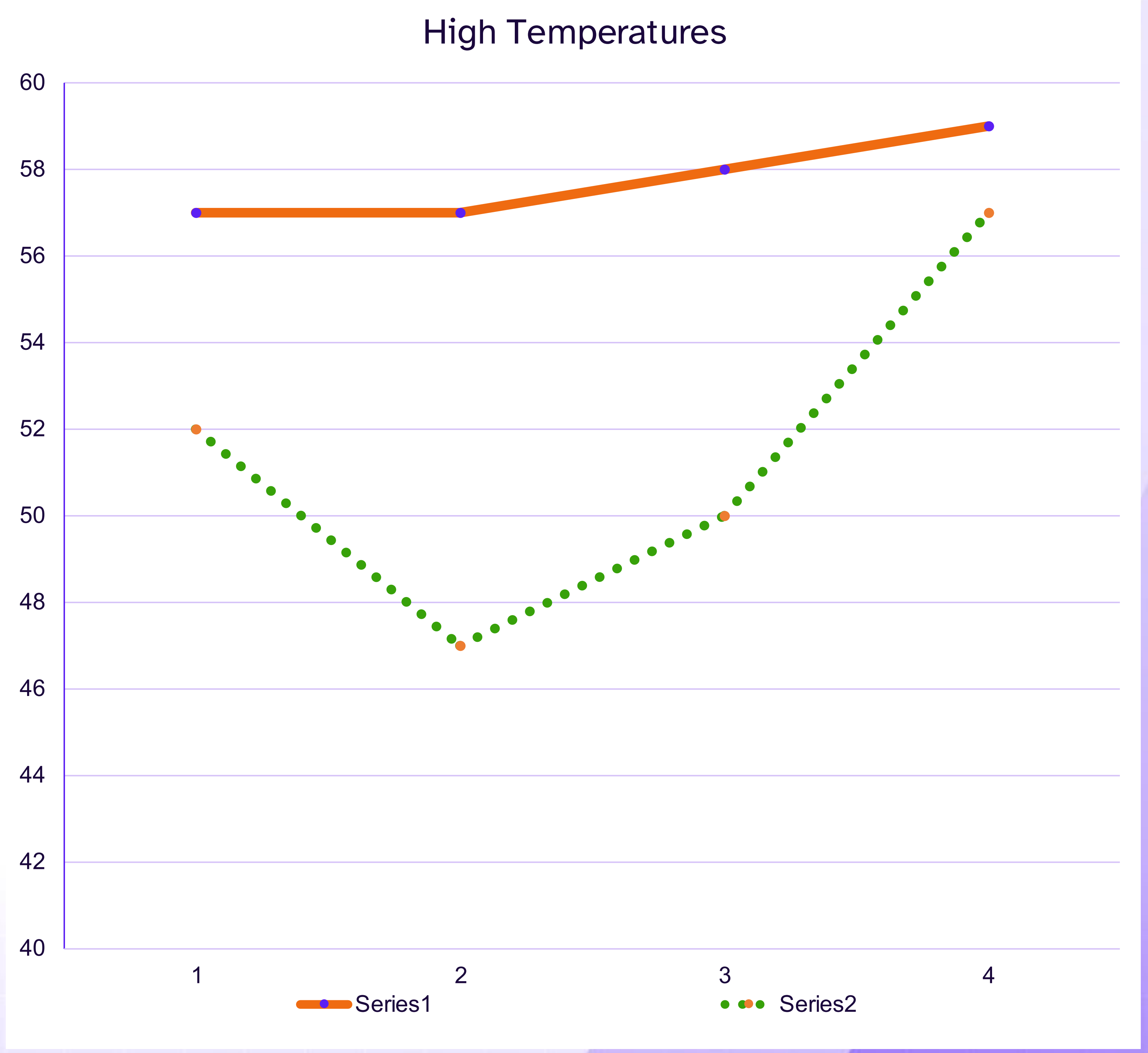
Solve the problem the lowest in the stack you can.



Avoid color alone for meaning

Not just for users with visual disabilities





Use sufficient contrast

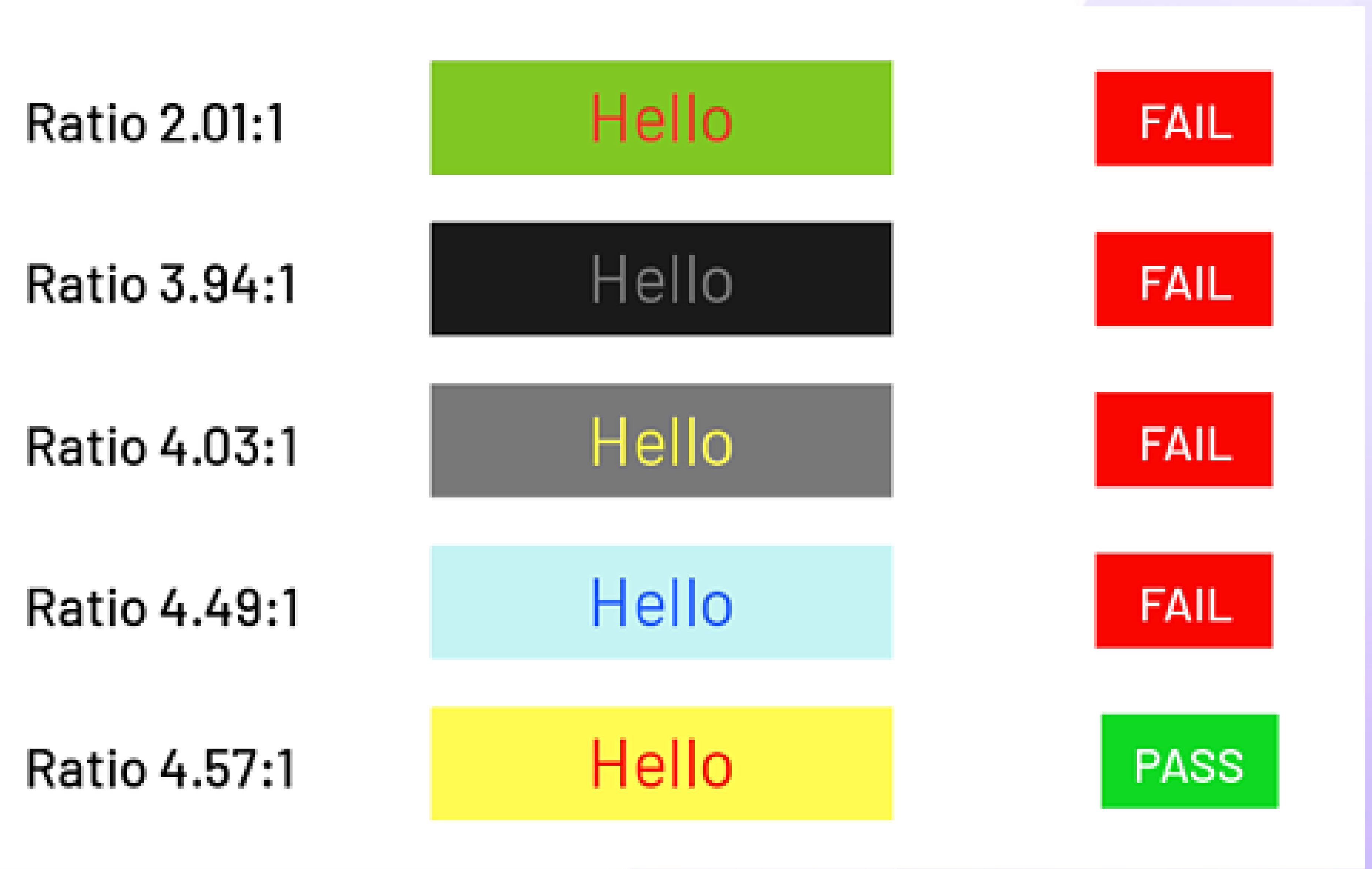
WCAG 2 defines the minimum color contrast ratio as follows to meet Level AA:

3:1: all text that is 14pt and bold (approximately 18px-19px)

3:1: all text that is 18pt and larger (approximately 24px-26px)

3:1: all graphical elements and active interface components

4.5:1: all text smaller than 18pt or 14pt bold



Ratio 2.01:1	Hello	FAIL
Ratio 3.94:1	Hello	FAIL
Ratio 4.03:1	Hello	FAIL
Ratio 4.49:1	Hello	FAIL
Ratio 4.57:1	Hello	PASS

Don't Use Images of Text

*Unless there is no other way.
However, most font effects can be achieved with CSS.

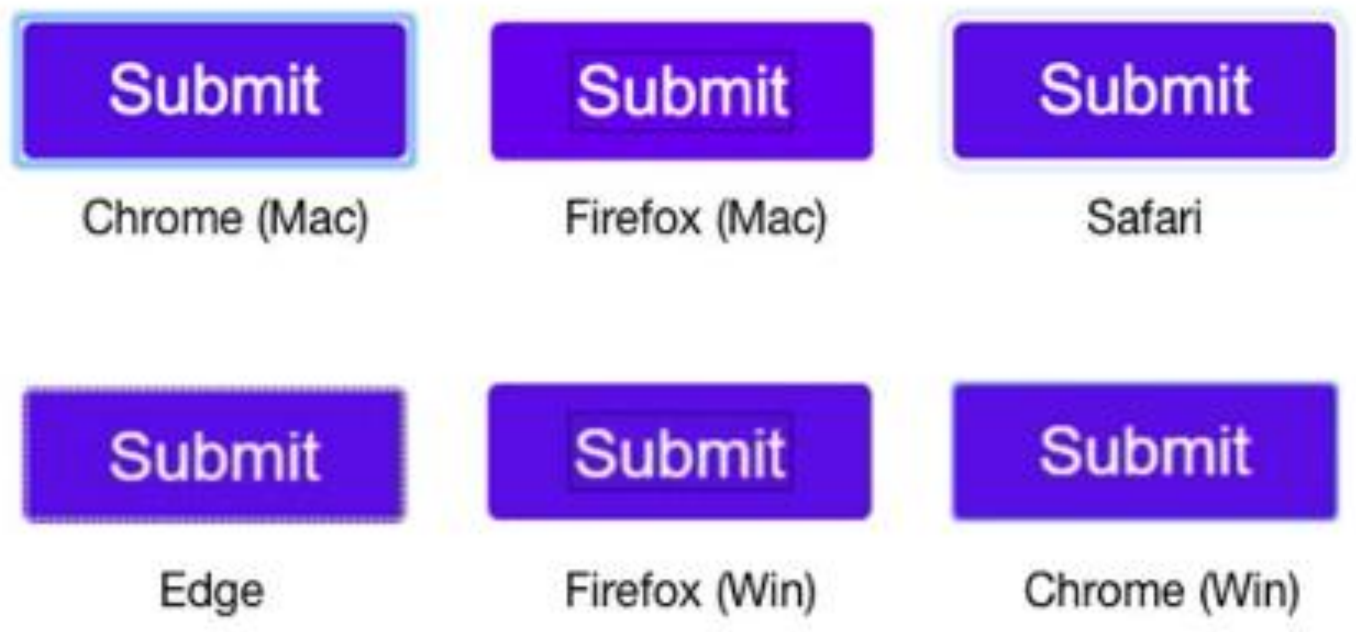
Why do resizing and reflow matter?

"I need the text content to resize on browser zoom to help me read the content better."

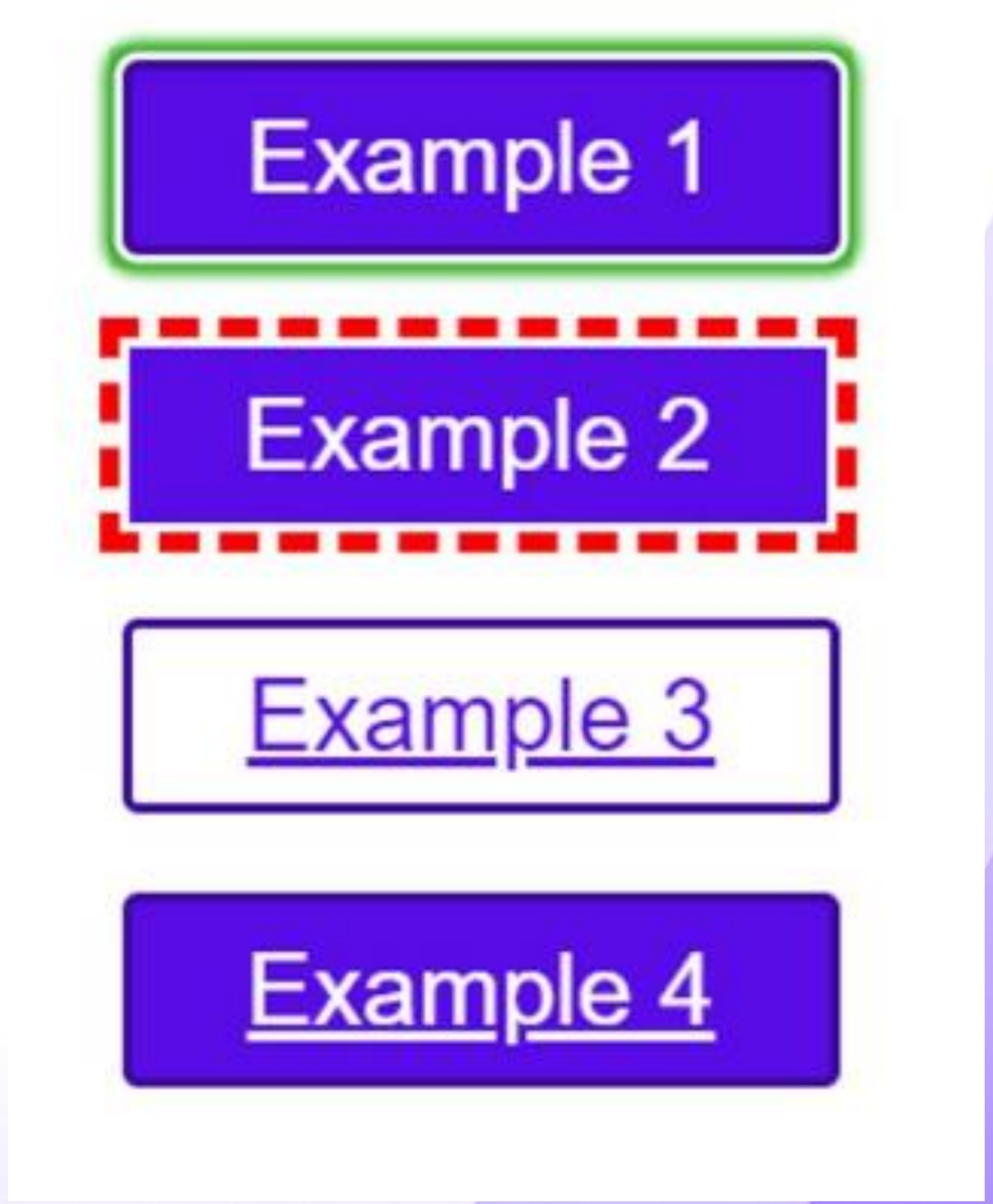
"Scrolling in multiple directions is much more time consuming and difficult for me to read."

Focus visible

Default (varies by browser)



Custom (CSS)



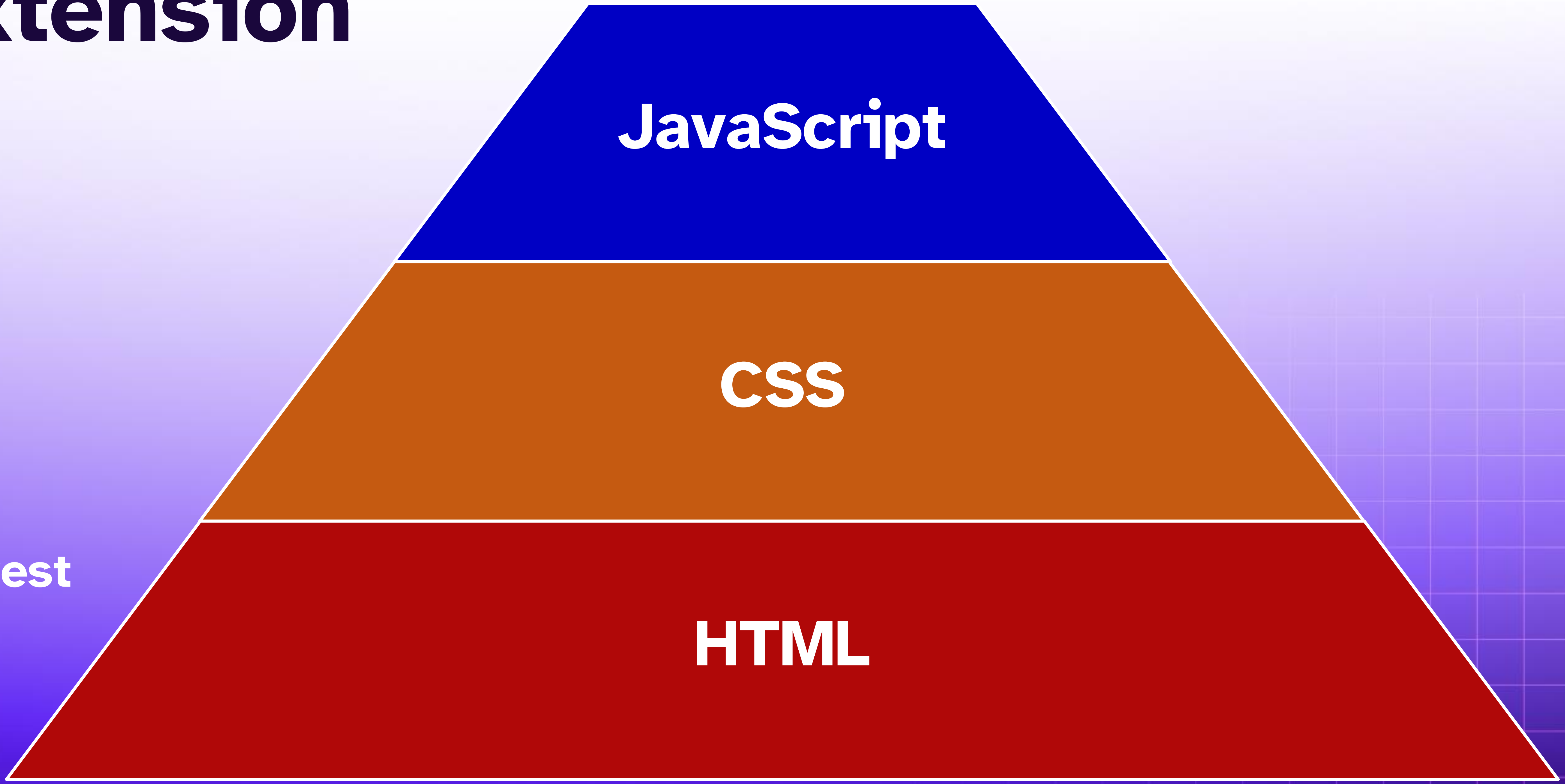
level access

JavaScript and ARIA accessibility

Building an accessible foundation: **JavaScript extension**

- Focus
- Keyboard events
- Custom controls (ARIA)
- Dynamic content

Solve the problem the lowest in the stack you can.



ARIA

Three main areas to support assistive technology

Landmarks: Indicating main structural areas of a page

UI components: Creation of roles and properties of user interface elements

Dynamic content: A method to indicate alerts, page changes, and dynamically updating information. (Live regions)

Custom controls

When you build your own controls,

you are responsible for:

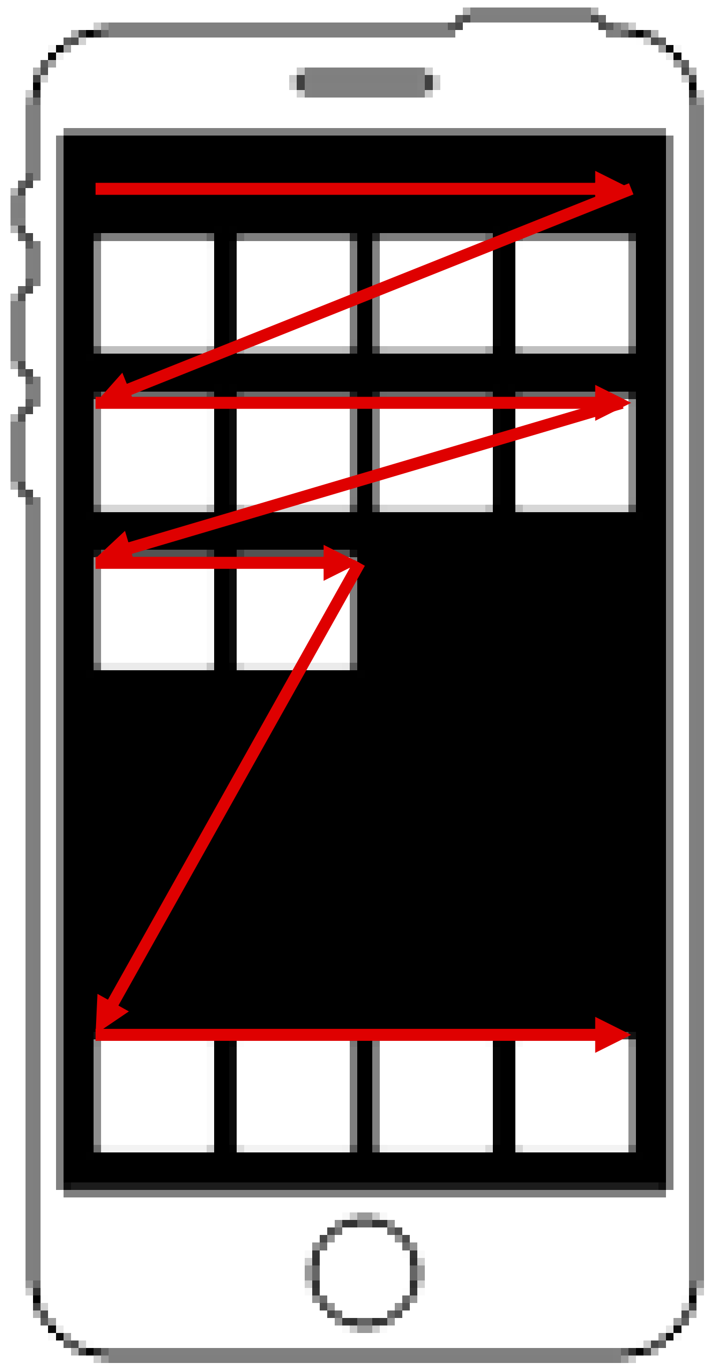
- Semantics (ARIA)
- Focus order
- Keyboard interaction

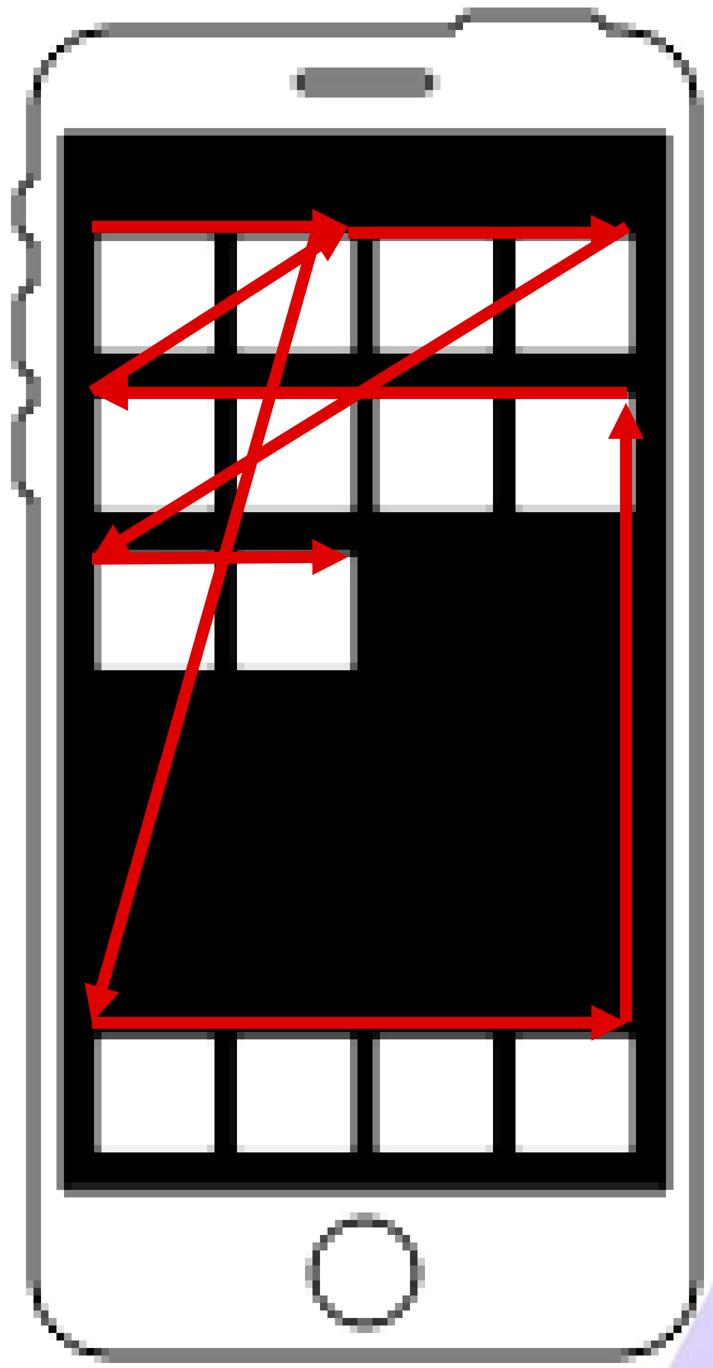
Why does keyboard accessibility matter?

“I need to be able to use the keyboard to perform my tasks on a website.”

“The keyboard helps me navigate a digital product effectively without the use of a mouse.”

Focus order


Top to bottom, left to right (logical exemption and language exemption)


Avoid jumping around wildly, which disorients the user

Good uses of ARIA

Give text equivalent to non-text visual elements	**aria-label** and **aria-labelledby** `<i class="fa fa-exclaim" aria-label="Warning"></i>`
Programmatically associate elements	`<input` **aria-describedby="error1"**`>` `<span id="error1">Error: First name is required</span>`
Inform the user when the page changes	Live regions, alerts, dialogs
Create widgets with roles that don't exist natively in HTML	**role="tablist"**, **role="menu"**

Accessible name and description

Will override text content and native attributes like label and title

aria-label=“Close dialog” (a string)

aria-labelledby=“link_context” (a list of IDs)

aria-describedby=“email_errormessage” (a list of IDs)

Testing for accessibility

Manual testing techniques

Keyboards

Assistive technology

Browser developer tools

Accessibility tree / inspector

Browser zoom

Contrast checkers / gray-scale filters

Favlets / browser extensions apply styles or run checks

Automated test engines

- Access Engine, Axe, WAVE, EqualAccess Lighthouse (subset of aXe)
- Automatic vs. human review
- Understanding false positives
- Not all issues can be addressed automatically

Integrating accessibility testing into development workflows

- Work with designers to make sure designs are annotated for accessibility and patterns are documented.
- Write accessible code from the start.
- Use SDKs in automated testing tools
- Use the Level Access browser extension.

Key accessibility checkpoints

- **CI/CD**
  - Check in
  - Unit testing
  - Build
- **QA testing**
- **Monitoring**

Key takeaways

- People have a diverse set of user needs
- How you write code impacts whether others can use it
- Test for accessibility
- Become familiar with assistive technology and how others use technology
- Learn from people with disabilities
- Share what you learn with others

ARIA Resources

ARIA should be leveraged when you are able to provide all the relevant information about what you are implementing.

ARIA Authoring Practices Guide

- From the creators of ARIA themselves
- Implement custom widgets and other aspects of ARIA