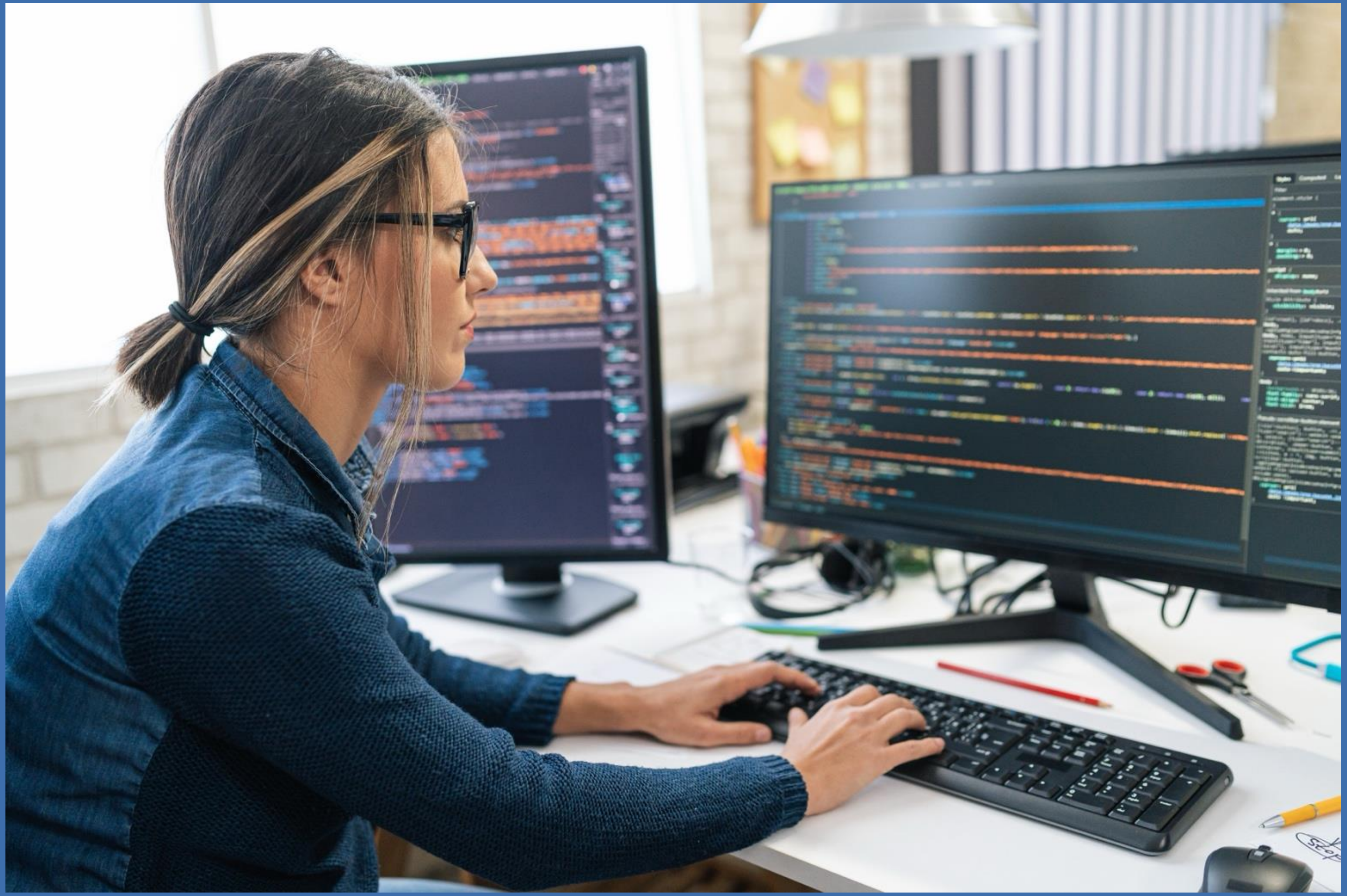
Accessibility Practices for Modern Developers

Amy Hepler
Lead UX/Accessibility Developer, NCDIT
Web Accessibility Specialist (WAS), IAAP

A11y CoP – June 24, 2025

5 Key Concepts

Writing Accessible Code



1. Know there are rules (and legal requirements)

WCAG 2.1 Total Success Criteria = 78

- Level A = 30
- Level AA = 20
- Level AAA = 28

WCAG 2.1 AA is the legal **minimum** standard. (50 criteria)

2. Know how screen readers read (desktop & mobile)

Screen reader users skim headings, landmarks/regions, and links and expect to hear names, roles, and values of components.

3. Semantics and Structure: Don’t reinvent the wheel

Whenever possible, use semantic elements that assistive devices already understand.

4. Test as you work

Testing with an automated testing tool, a keyboard, and a screen reader will help you uncover and address most common issues.

The earlier you find them, the easier to fix.

5. Every code change can be the difference between an accessible and inaccessible product.

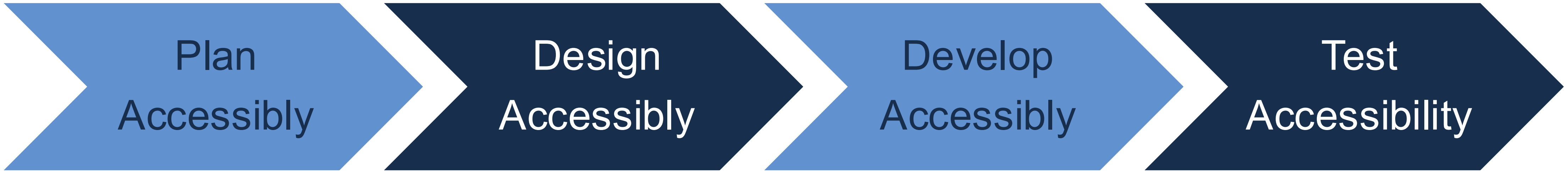
Today’s Agenda

- Shift left: Accessibility in the dev lifecycle
- Semantics & structure
- Keyboard & focus
- Color & contrast
- Responsive layouts
- ARIA
- Dynamic content
- Form validation
- JavaScript framework tips
- AJAX & single-page applications
- Beyond WCAG 2.1 AA
- Resources

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Shift Left: A11y in the Development Lifecycle

- Accessibility starts at planning, not QA
- Pair with designers early
- Document accessible requirements



NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Why Developers Should Shift Left on A11y

- Fixing issues early is cheaper and faster
- Prevents technical debt
- It’s easier to bake in than bolt on
- Reduces rework across teams
- Enables inclusive design and iteration
- Mitigates legal risk
- Improves code quality
- Promotes team ownership

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Start with Semantic Structure

- Gives meaning to structure
- Improves accessibility for assistive technologies
- Enables keyboard and assistive technology interactions – **built-in behaviors**
- Reduces the need for ARIA
- Better SEO and performance
- Easier to maintain and debug

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Semantic Structure in the A11y Tree

- Elements with semantic markup are included in the accessibility tree.
- Things that do **not** end up in a11y tree include:
  - `<div>`
  - `<span>`
  - `<p>`
  - CSS styles and background images
  - colors

Best:

```
<h1>Semantic Structure</h1>
```

Bad:

```
<div style="font-size: 36px; font-
weight: bold;">Semantic Structure</div>
```

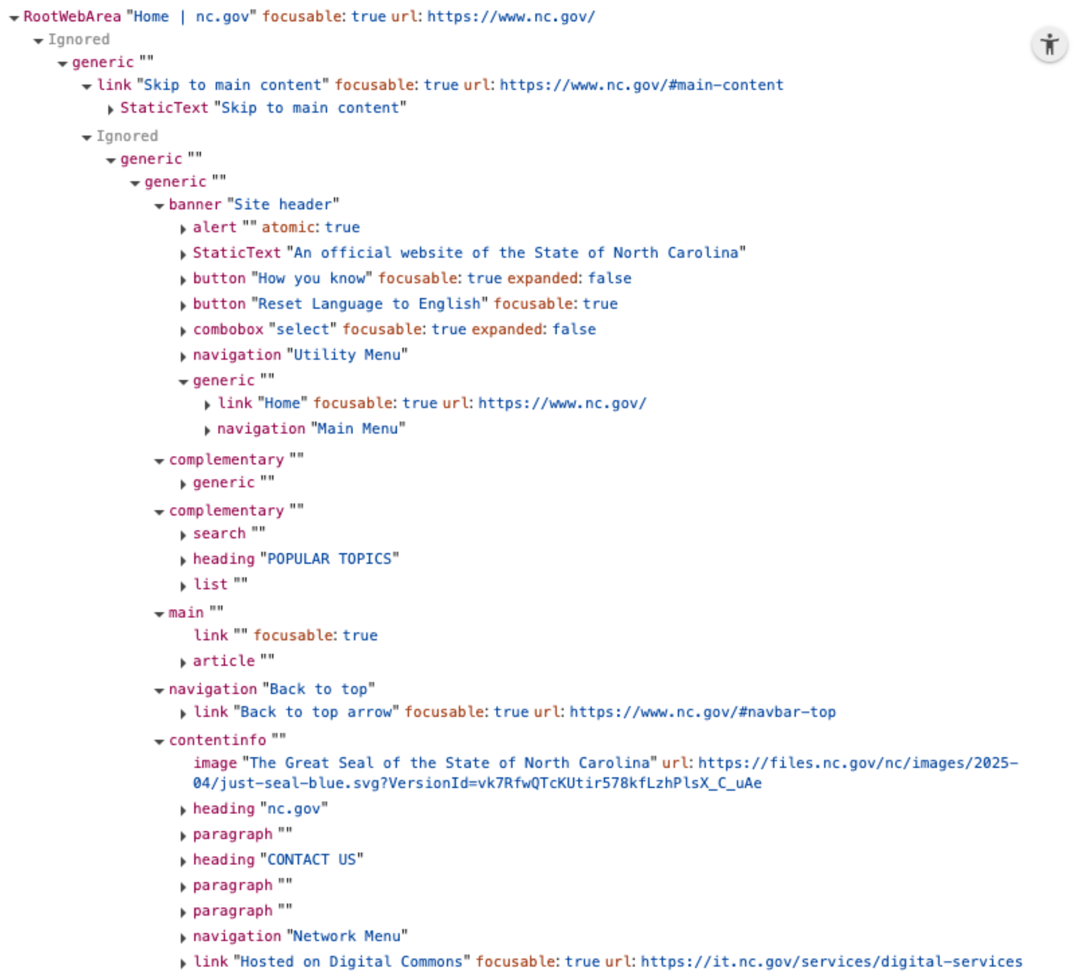
Acceptable:

```
<div style="font-size: 36px; font-
weight: bold;" role="heading" aria-
level="1">Semantic Structure</div>
```

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

The Accessibility Tree

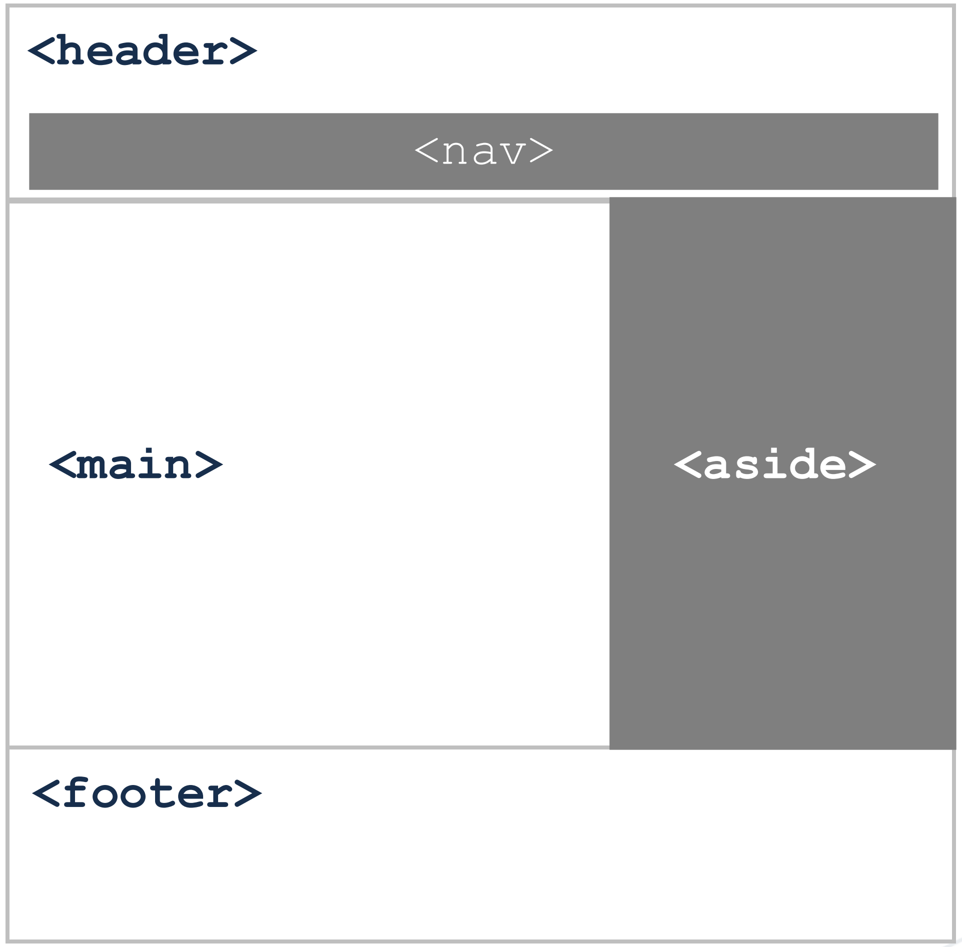
- Browsers have an a11y tree.
- The bridge between code and assistive technology
- Determines what users hear, feel, or navigate
- Mistakes in code or ARIA can break it
- Essential for debugging and testing
- If it’s not in the a11y tree, it’s not in the experience.



```
▾ RootWebArea "Home | nc.gov" focusable: true url: https://www.nc.gov/
  ▾ Ignored
    ▾ generic ""
      ▾ link "Skip to main content" focusable: true url: https://www.nc.gov/#main-content
        ▸ StaticText "Skip to main content"
      ▾ Ignored
        ▾ generic ""
          ▾ generic ""
            ▾ banner "Site header"
              ▸ alert "" atomic: true
              ▸ StaticText "An official website of the State of North Carolina"
              ▸ button "How you know" focusable: true expanded: false
              ▸ button "Reset Language to English" focusable: true
              ▸ combobox "select" focusable: true expanded: false
              ▸ navigation "Utility Menu"
              ▾ generic ""
                ▸ link "Home" focusable: true url: https://www.nc.gov/
                ▸ navigation "Main Menu"
            ▾ complementary ""
              ▸ generic ""
            ▾ complementary ""
              ▸ search ""
              ▸ heading "POPULAR TOPICS"
              ▸ list ""
            ▾ main ""
                link "" focusable: true
              ▸ article ""
            ▾ navigation "Back to top"
              ▸ link "Back to top arrow" focusable: true url: https://www.nc.gov/#navbar-top
            ▾ contentinfo ""
                image "The Great Seal of the State of North Carolina" url: https://files.nc.gov/nc/images/2025-
                04/just-seal-blue.svg?VersionId=vk7RfwQTcKUtir578kfLzhPlsX_C_uAe
              ▸ heading "nc.gov"
              ▸ paragraph ""
              ▸ heading "CONTACT US"
              ▸ paragraph ""
              ▸ paragraph ""
              ▸ navigation "Network Menu"
              ▸ link "Hosted on Digital Commons" focusable: true url: https://it.nc.gov/services/digital-services
```

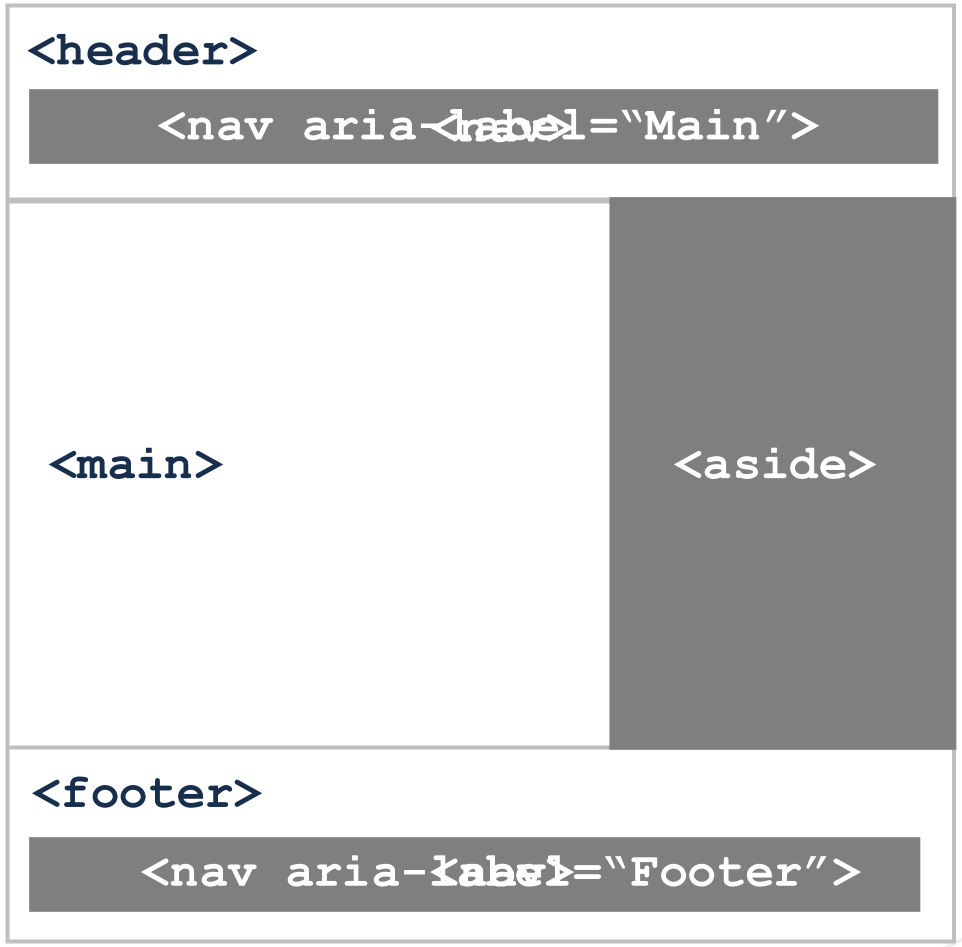
HTML 5 Landmark Regions

- Landmarks specify things like `<header>`, `<nav>`, `<main>`, `<footer>` and others
- Improves screen reader navigation
- Reduces need for ARIA
- Defines structure and layout
- Enhances SEO and machine readability



Landmark Region Tips

- All text should be within a landmark region
- Minimize number of landmarks
- Distinguish multiple instances of landmarks by different programmatic labels (`aria-label` or `aria-labelledby`)



Keyboard & Focus Management

- Primary way many users interact with content
- All UI must be navigable with keyboard alone
- Not everyone uses a mouse
- Focus = orientation

Join the A11y CoP Today!

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Keyboard & Focus Management Tips

- Ensure a logical tab order
- Avoid `tabindex` values over 0
- Use native HTML elements when possible
- Never remove/disable focus styles
  - Maintain a visible focus indicator for all interactive elements (`:focus-visible`)
- Avoid `tabindex="-1"` unless for focus management
- Provide skip links
- Use ARIA carefully for focus control
- Avoid keyboard traps

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

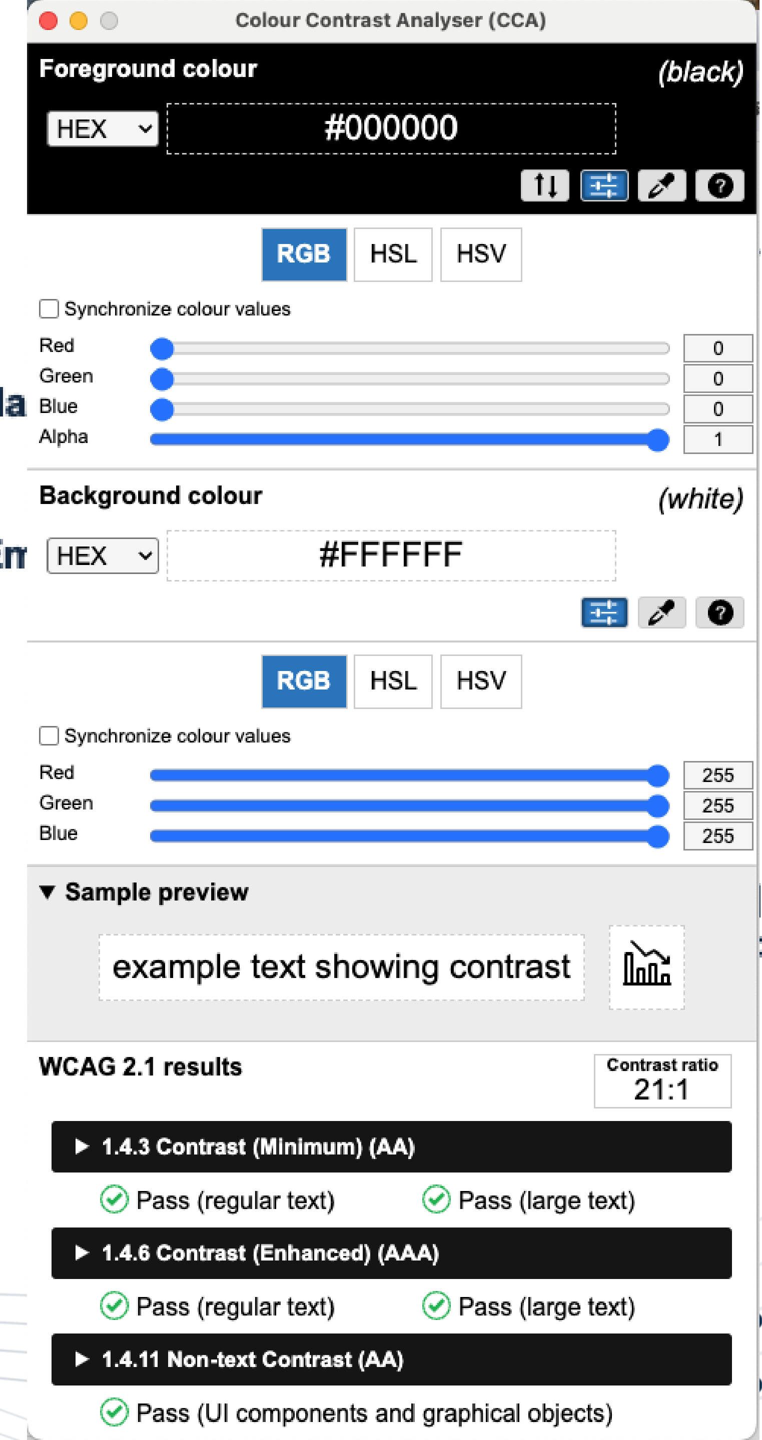
Manage/Restore Focus in Modals & Overlays

- Move focus into the modal on open
- Trap focus within modal
- Return focus when modal closes or escaped
- Use proper roles and attributes

```
<div role="dialog" aria-modal="true"
aria-labelledby="modal-title">
  <h2 id="modal-title">Subscribe to
Updates</h2>
</div>
```

Color & Contrast

- Text/background: 4.5:1 contrast minimum
- Non-text/background: 3:1 contrast minimum
- Consider color blindness and dark mode
- Avoid color as sole indicator
- Helpful tools:
  - [Colour Contrast Analyser]
  - [WebAIM Contrast Checker]


Colour Contrast Analyser (CCA)

Foreground colour (black)

HEX #000000

RGB HSL HSV

Synchronize colour values

Red	0
Green	0
Blue	0
Alpha	1

Background colour (white)

HEX #FFFFFF

RGB HSL HSV

Synchronize colour values

Red	255
Green	255
Blue	255

▼ Sample preview

example text showing contrast

WCAG 2.1 results

Contrast ratio 21:1

▶ 1.4.3 Contrast (Minimum) (AA)

✓ Pass (regular text) ✓ Pass (large text)

▶ 1.4.6 Contrast (Enhanced) (AAA)

✓ Pass (regular text) ✓ Pass (large text)

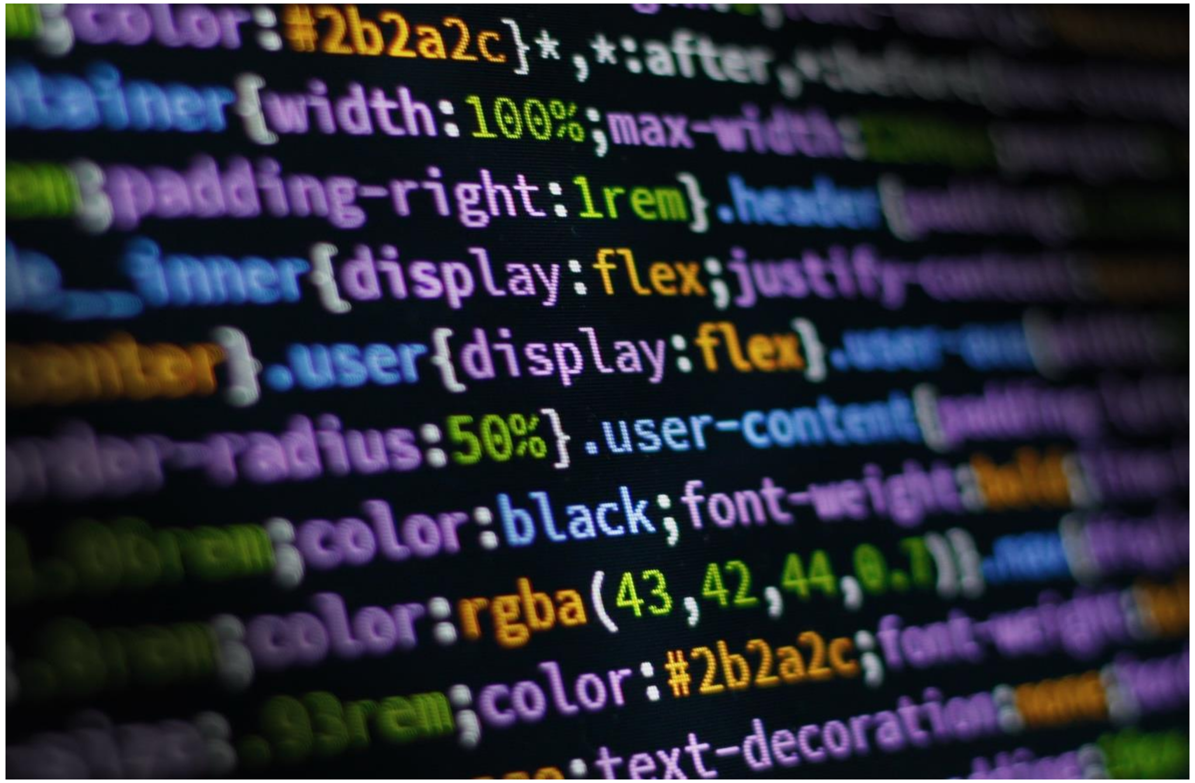
▶ 1.4.11 Non-text Contrast (AA)

✓ Pass (UI components and graphical objects)

NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Responsive & Flexible Layouts

- Use `ems`, `rems`, or % for scalable UIs
- Support mobile reflow, avoid fixed widths
- Test at 400% zoom
  - **Test keyboard on mobile view**
- Be careful of using flexbox to reorder UI



```
color:#2b2a2c}*,*:after,
{width:100%;max-width:
padding-right:1rem}.header{
inner{display:flex;justify-
}.user{display:flex}.
radius:50%}.user-content{
;color:black;font-weight:
color:rgba(43,42,44,0.7)}
.93rem;color:#2b2a2c;font-weight
:text-decoration
```

Accessible Rich Internet Applications (ARIA)

- Bridges a11y gap between HTML limitations and full-blown web applications
- Allows developers to:
  - Add roles to elements
  - Define states and properties
  - Describe relationships
- Doesn't visually change appearance

- Use only when native HTML does not suffice
  - Good: `aria-expanded` to a toggle button
  - Bad: `<div role="button" tabindex="0" ... >` instead of `<button>`

Using ARIA Correctly

> ”No ARIA is better than bad ARIA.”
> – W3C

- Prefer native HTML over ARIA
- Use descriptive labels
  `<button aria-label=“Close settings panel”>X</button>`
- Update states programmatically
  `<button aria-expanded=“false aria-controls=“menu”>Menu</button>`
- ARIA landmarks and regions (`role=“”`)
- Provide keyboard interactions for custom widgets
  `<div role=“button” tabindex=“0”>Join Today!</div>`
- Avoid ARIA misuse

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Dynamic Content

- Tell users about dynamic changes. Three main options:
  1. Load or reload the page, with the page `<title>` reflecting the updated dynamic information.
  2. Move the focus to the updated content or to a confirmation or error message.
  3. Use an ARIA live region to make an announcement
     - Use the right politeness level:
       - `aria-level=“polite”` waits for current speech to finish
       - `aria-level=“assertive”` interrupts current speech immediately (don’t overuse)
     - Live region must be present and empty before making announcement

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Form Validation

- Cognitive clarity (what, where, how)
- Essential for screen reader users
- WCAG 2.1 requirements
  - SC 3.3.1: Error Identification
  - SC 3.3.3: Error Suggestion
  - SC 3.3.2: Labels or Instructions
  - SC 3.3.4: Error Prevention

Form Validation Best Practices

- Provide clear labels and instructions
    - `<label for="">` or `<label>`-wrapped inputs
- Show inline errors near the field
- Announce errors to screen readers
- Set focus to first error on submit
- Avoid color alone for error indication

JavaScript Framework Tips

- Start with semantic HTML
  `<button>` over `<div … onClick …>`
- Manage focus proactively
- Announce dynamic content
- Test with screen readers and keyboards
- Use accessible component libraries

AJAX & Single-Page Applications (SPAs)

- Problem:
  - No full page reload → no trigger for screen reader
  - Focus remains on button/link that triggered update
  - Screen reader says nothing → user confused
- Solutions:
  - Manage focus and announcements during route/view changes
  - Notify screen readers that new content loaded
  - Wait 1-2 seconds after injecting content before sending focus

Beyond WCAG 2.1AA

- Plain language & clarity
- Consistent layouts and components
- Whitespace and visual separation
- Visual hierarchy and content chunking
- Error prevention and forgiveness
- Build with real people in mind

Resources

- Google:
  - Web Accessibility Course
  - Accessible Development Guide
- Official specifications
  - How to meet WCAG
  - Authoring Tools Accessibility Guidelines (ATAG)
- Color contrast:
  - Colour Contrast Analyser
  - WebAIM Contrast Checker
- ARIA Components & Patterns
- MDN A11y Web Docs
- Teach Access – code samples/tips for testing in VoiceOver
- LinkedIn Learning: Accessibility for Web Design

Remember:
Every code change can be the difference between an accessible and inaccessible product.

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Questions?

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

NCDIT NORTH CAROLINA DEPARTMENT OF INFORMATION TECHNOLOGY

Amy Hepler

Accessibility Practices for Modern Developers

amy.hepler@nc.gov