Accessible Development For All!

Hello! My name is Joni.

- Sr. Javascript front-end dev @ Georgetown
- Pseudo-avid Tweeter: @jonihalabi

Accessibility
Is
User Experience

Users have different needs.

- Device needs
- Visual needs
- Motor needs
- Cognitive needs

This is where standards come in:

- **WCAG**: Web Content Accessibility Guidelines
- **Section 508**

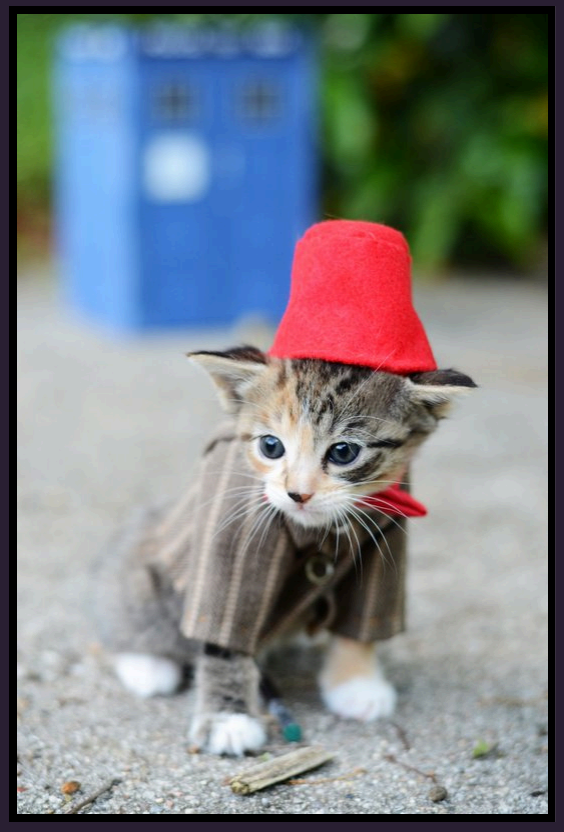
Accessibility is not a separate project that you tackle every once in a while.

- Clean markup
- Know your HTML5 tags
- Be **SEMANTIC**
- Use ARIA and role attributes when necessary

A Quintessential Example

```
<img src="kitten.jpg">
```

```
<img
  src="kitten.jpg"
  alt="A kitten wearing a fez because fezzes are cool">
```



Making Your Sites More Accessible

A 5-Part Primer

Part 1: Landmarks

Landmarks are your (and your users') friends.

Landmarks allow users to navigate through the major sections of a site.

HTML 5 Tag	Role
`<header>`	banner
`<nav>`	navigation
`<main>`	main
`<aside>`	complementary
`<section>`	region
`<article>`	article
`<footer>`	contentinfo
`<form>`	form

```
<header>
  <!-- Logo, etc. go here -->
  <nav> <!-- Site navigation --> </nav>
</header>

<main>
  <section> <!-- Important stuff goes here --> </section>
  <section> <!-- More stuff goes here --> </section>
  <aside> <!-- Incidental information goes here --> </asid
</main>

<footer>
  <!-- Footer info goes here -->
</footer>
```

Landmarks are great... but they're kind of useless without names.

- The `aria-label` attribute gives your landmark a name!
- You can also use `aria-labelledby` to specify an ID of another element that can be used as a label.

```
<header aria-labelledby="site-name">
  <h1 id="site-name">Georgetown University</h1>
  <nav aria-label="Full site navigation"> ... </nav>
</header>

<main>
  <section aria-label="Important content"> ... </section>
  <section aria-label="More important content"> ... </sect
  <aside> ... </aside>
</main>

<footer> ... </footer>
```

One more thing about landmarks

All content on the page must be inside some sort of landmark container.

Part 2: Headings

Headings help users figure out the major themes of a page and the sub-topics for each theme.

Sounds like a term paper, right?

- H1 = Doctor Who
  - H2 = Doctors
  - H2 = Companions
    - H3 = Human companions
    - H3 = Not-so-human companions
      - H4 = K-9!
  - H2 = Enemies

Headings dos and don’ts

- Make sure your headings are in a logical order.
- It is OK to have more than 1 `<h1>` tag on the page!
- **DO NOT** use a `<h*>` tag just because it looks pretty.

Part 3: Skip links

Allows keyboard users skip all of your navigation so they can go straight to the content.

Skip links HTML

```
<body>
  <nav aria-label="Skip links">
    <a href="#main-content" class="skip-link">
      Skip to main content
    </a>
  </nav>

  ...

  <main id="main-content">
    All the important stuff!
  </main>
</body>
```

Skip links Sass

```
.skip-link {
  /* Make it pretty */
  background: #333;
  color: #fff;
  padding: 0.5rem;

  /* Make it hidden */
  position: absolute !important;
  clip: rect(1px,1px,1px,1px);
  overflow: hidden;
  height: 1px;

  /* Show it on focus */
  &:focus {
    clip: auto;
    overflow: visible;
    height: auto;
```

```
}
```

While we are on the subject of keyboard users and links...

DO NOT turn off the outline around your links.

```
a {
  outline: 0; /* This is really bad */
}
```

Part 4: Hidden content (and transitions)

Actually hide things.

- It really is OK to hide things *(like your off-canvas navigation)*.
- However, you need to provide a clear action to display that information.

Hiding content by moving it off the screen or giving it no height or width only hides that content from sighted users.

Screen readers still read that "hidden" (not really hidden) content.

I mean, we've all done this at some point in our lives:

```
.my-hidden-thing {
  height: 0;
  width: 0;
  overflow: hidden;
  position: absolute;
  right: -500000000;
}
```

This actually hides things:

```
.my-hidden-thing {
  display: none;
}
```

If you still want transitions:

```
.my-hidden-thing {
  display: none;
  width: 0;
  transition: all 2s ease;

  &.show {
    display: inline-block;
    width: 500px;
  }
}
```

Part 5: Testing your stuff!

Step into your users' shoes!

You already test different browsers and screen sizes. Testing with your keyboard or screen reader also needs to be part of the QA process.

Testing for keyboard ability?

Ignore your mouse/trackpad!

Keyboard navigation

- Keyboard users expect to be able to tab through the page from left to right, top to bottom.
- This order is determined by your DOM.
- Make sure that any item that has focus is *visibly focused*.

Testing for screen reader usability?

Close your eyes!

Screen reader testing

- Actually test with a screen reader
  - Mac users: VoiceOver
  - PC users: JAWS or NVDA
- Make sure everything is read correctly.

To sum up:

- Know the semantic meaning behind the code you write.
- Ask yourself: Does what I'm creating really make sense?
- Remember about all of the different kinds of users who can visit your site.

Thank you!

http://talks.thatdevgirl.com/accessibility-edui/

Accessibility guideline documentation

- W3C
- Section 508
- WCAG
- WAI-ARIA
- Roles Specification
- Siteimprove

References

- Accessibility statistics
- Accessibility and phone number formatting
- I thought title text improved accessibility
- Semantic differences between <b>, , <i>, and
- Testing with a screen reader
- Accessibility: Bold and italic formatting in HTML