Designing for Accessibility

This presentation was done as part of The Designer's Studio series on October 2, 2014, in Oakland, CA, hosted and sponsored by the lovely folks at futuredraft.

What we'll cover

- Understanding your audience
- Overview of assistive technologies
- Accessible visual and user experience design
- Tools and resources

Understanding your audience

rethinking what disability means

Who is disabled?

- ~12-20% (depending on the population, who's counting, what's counted as a disability, etc.) of people have some kind of visible or invisible disability
- There's a multiplier effect of disabled people's friends and families, who may not use services or products their friend/family member can't
- People not normally considered disabled can also have temporary disabilities due to injury or illness
- You will probably have a disability as you age, if you don't already!

What does accessibility mean for the web?

- The goals of making the web accessible are inclusion, equal access, and supporting people in their ability to be independent
- The photo in the related slide shows a prototype of the Kenguru single-occupant car, which is designed to be used by people in wheelchairs, without assistance, and without expensive lifts and large vehicles

Major use cases

- Visual impairment
- Mobility and fine motor impairment
- Hearing impairment
- Learning and cognitive disabilities
- Seizure and vestibular disorders

Visual impairment

- Blindness
- “Low vision”, tunnel vision, macular degeneration
- Color blindness

Fine motor impairment

- Forms of paralysis
- Muscle and nerve disorders
- Arthritis

Hearing impairment

- Deafness
- Partial hearing of all kinds
- People with different kinds of hearing aids can hear different volumes, tones, pitches, etc.

Learning and cognitive disabilities

- Reading and math disabilities, like dyslexia
- Attention deficit disorders, and conditions that affect memory and cognitive load
- Autism spectrum

Seizure and vestibular disorders

- Seizure disorders triggered by fast blinking and flashing
- Disorders that create motion sickness, etc., in reaction to movement and parallax effects

Assistive technology overview

software, hardware, settings

Screen readers

- Used with custom keystrokes and gestures
- Create a more linear experience than experiencing software visually
- For web content, generally assume that web standards for HTML, CSS, and Javascript are used according to their specs
- Major screen readers are JAWS (for IE/Windows/closed/$$$$), NVDA (for Firefox/Windows/open source/free!), VoiceOver on iOS (native to iOS and OSX)
- Used primarily by people with vision issues, but also by people with cognitive and learning disabilities that make it difficult for them to read text from a screen

Refreshable Braille

- Used primarily by people who are blind and deaf, but also by those who simply want another means of output while reading different types of content

Input devices

- Keyboards are the primary user interface after the mouse, especially for people who have issues with fine motor control
- People use a wide range of commercial and custom devices to enter content, usually in ways that mimic or map to mouse and keyboard interactions
- Stephen Hawking’s custom input device uses a movement in his jaw to cycle through various options on screen, including a virtual keyboard

Other supports and tools

- Some example accessibility supports in iOS (find them under Settings > General > Accessibility; screenshots are some of the options from iOS 7):
  - Zooming and magnification
  - Text size and weight
  - High contrast
  - Color inversion
  - Custom cursors and control styles
  - Hearing options for various types of hearing aids
  - Caption defaults for video
  - Guided Access to hide certain functionality to maintain focus and attention
  - Others related to interactions, notifications, and shortcuts for major accessibility functionality
- Other operating systems have similar supports

Accessible design principles

color, structure, content, interaction

POUR ≠ POOR

- Main principles of the Web Content Accessibility Guidelines (WCAG) 2.0, developed by the W3C
- **Perceivable**: Can I find it?
- **Operable**: Can I use it?
- **Understandable**: Can I grasp what it means? Do I know what's happening in a dynamic context?
- **Robust**: Is it device/browser/assistive tech agnostic?

Color contrast

- Evaluated based on font size and weight
- Measured by comparing opacity and brightness of text and its background color(s)

Color + (con)text

- Refrain from using color or other Gestalt principles *alone* to convey info
- Conveying information through text in some fashion is the best for the most users

Showing state

- Mouse *and* keyboard visual indications are important for interactive elements; these indications can be made with color and style
- Interactive elements (like links, buttons, and other controls) can’t be nested inside each other to allow for keyboard support for all controls

Content order and outline

1. Welcome
    2. About Us
        3. Our Company
        3. Our Work
        3. Our Philosophy
    2. About You
        3. Your Brand
        3. Your Vision

- Example web page DOM with a global heading (A), main content (B), and a global footer (C)
- Top-to-bottom DOM order supports browsing by screen readers and similar devices, and supports logical reading and keyboard focus order

- Standard HTML headings (even if visually hidden with CSS) provide an outline that screen reader users can skim and navigate with
- Headings also convey which chunks of content have equal weight
- Content added dynamically to the page should appear “below” what the user is currently interacting with, not in an area already visited

1. Your Trip Results
        3. Filter your Results
        3. Sort Your Results
    2. Search Results
        3. Result 1
        3. Result 2
        3. Etc.

- More complex, real life example with an “imperfect” but still useful page hierarchy
- Global header (A), main content (B), sidebar (C), and global footer (D)
- Page title heading and content in B, filtering and sorting options in C

Proximity

- Important to consider for users who have the screen zoomed in, or who have cognitive difficulties
- Things that are related to each other or impact each other should be near each other on the screen

Image alternatives

- `alt` attribute values for images provide descriptions of pictures and graphics for screen reader users
- Alternately, visible captions for images can help all users, and are especially recommended for complex charts, graphs, and the like

Captions and transcripts

- Captions for videos, audio descriptions, and transcripts help people with hearing difficulties understand video content
- Good for SEO since search engines are best at indexing text
- Youtube and Vimeo make captions and transcripts for you automatically, which you can edit
- HTML5 video supports a variety of ways to provide captions to users

Labels and instructions

- Clear, concise instructions (labels) for each form field prevent errors and make the way clear; don't rely on placeholder text in the field itself for this, since they disappear when you interact with the field!
- Links, buttons, and other controls that repeat visually should have unique labels for screen reader users (and ideally all users!)

Notifications and warnings

- A little demo of a notification system that would address proximity concerns, color contrast concerns, and could be wired up to be very usable by a screen reader user
- Changing only the colors of text or borders on fields isn't sufficient
- Notifications and warnings should be placed near the elements they pertain to
- Showing clear instructions about what content is needed, required, and how to format it (if necessary) should be included
- Labels for text inputs should also be stacked on top for easily readability, especially on mobile and for zoomed-in users (unlike in my example)!
- On-screen notifications can help confirm that what the user wanted to do has actually happened, as well

Consistent look and feel

- Interactive controls that look (and sound, for screen reader users) the same should generally behave the same to prevent user error and confusion

- Use semantic, native HTML controls whenever possible
- Use Accessible Rich Internet Application (ARIA) attributes for complex, non-native widgets like tabs, dialogs, sliders, etc., to provide screen reader users with audio output that matches the visual output

> “How do I convince my clients that they want an accessible site?"

Universal design

- Human-friendly
- Robust
- Adjusts to situations, environments, and conditions
- Hardware and software agnostic
- “Future-proofed”

Mobile-first design

If you’re steering toward mobile-first (or at least down-to-mobile) planning, accessibility overlaps nicely!

- Content is ordered by importance
- Complex processes are broken down into digestible pieces
- Flexible layout relies on CSS
- Touch targets are large and obvious/friendly

Tools and resources

testing tools, guidelines, and techniques

Testing color

- Contrast Ratio browser tool for color contrast
- Color Oracle desktop tool for color blindness
- Photoshop proofing for color blindness

Captions and transcripts

- YouTube captions and transcripts
- Vimeo captions and transcripts

OSX and iOS documentation

- Apple accessibility settings and supports
- iOS app accessibility guidelines
- Using VoiceOver on OSX and iOS

Guidelines and specifications

- WCAG 2.0 for accessible UX and development
- a11yTips blog for WCAG in plainer language
- WAI-ARIA for building complex, custom widgets

Photo credits

- Title slide
- What we’ll cover
- “Who is disabled?”
- Kenguru
- Major use cases
- Visual impairment
- Fine motor impairment
- Hearing impairment
- Learning and cognitive disabilities
- Seizure and vestibular disorders
- Screen readers
- Refreshable braille
- Input devices
- Color contrast
- Color and (con)text
- Showing state
- Proximity
- Image alternatives
- Captions and transcripts
- Labels and instructions
- Consistent look and feel
- Universal design
- Thank you slide