Web Accessibility

Make Accessible Rich Internet Applications

Web Accessibility

Goal

Make advanced Web applications accessible to people with disabilities.

People with disabilities:

- Visual impairment: 285 million ~4%
- Hearing loss: 5.3% of the total world population

UK:

- 2 million with sight loss

Web Accessibility benefits

Accessibility benefits people with or without disabilities. For example, those impacted include:

- People who are not fluent in English.
- People who do not have or are unable to use a keyboard or mouse.
- People with temporary disabilities due to accident or illness.
- Older people.
- New users.

Someone in the world goes blind every 5 seconds.

50% of sight loss cases cannot be avoided.

Reasons to do Accessibility Conformance

- ❏ Economic (more people will use your website)
- ❏ Ethical
- ❏ Legal: **UK Law**

If your business has a website, it **should be accessible** to disabled users.

Equality Act (2010): make "**reasonable adjustments**"

Best way to comply: **test by disabled users**

WCAG **conformance levels:** A, AA and AAA.

What do disabled people use ?

Screen readers:

- JAWS for Windows ~$1000
- Window-Eyes ~£630
- Narrator on Windows 8
- NVDA on Windows 7
- VoiceOver on Mac
- Google ChromeVox FREE

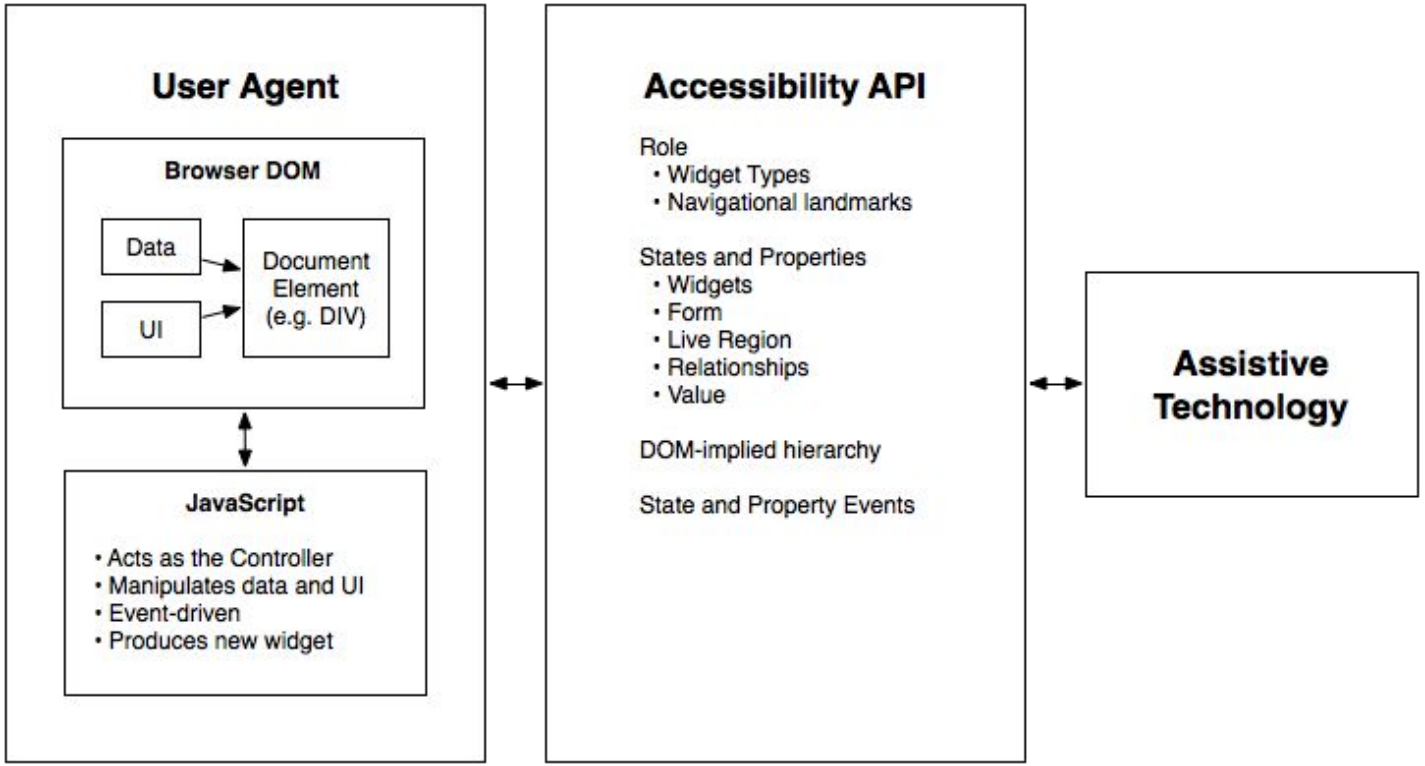
Other technologies:

- Impulse transmitter, Eyegaze Edge

What standards exist for Web content accessibility?

- **WAI**: a working group which develops guidelines and resources
- **WCAG 2.0**: a stable technical standard
- **WAI-ARIA**: guides to develop Accessible Rich Internet Applications

Accessibility API



How should I test my website for WCA ?

1. Test by disabled users - the best way to comply
2. Use software evaluation tools - do not rely on this
   - ➔ No web site evaluation tool has been developed that can completely replace a human being
   - ➔ Pitfalls in using automated tools:
     - ◆ False alarms (use of tables, image alt-s, etc.)
     - ◆ Unnoticed non-conformance

Analysers / Evaluation Tools:

- http://wave.webaim.org
- Google Developer Tools | Audits | Accessibility

Understanding WCAG 2.0

The standard has:

- **4 principles**
  - **12 guidelines**
    - Success criteria for each guideline
      - Techniques for meeting the SC
        - Sufficient techniques
        - Advisory techniques
        - Failures to meet the SC

What is a good website ?

It is a **POUR** website :)

- ☑ **P**erceivable
- ☑ **O**perable
- ☑ **U**nderstandable
- ☑ **R**obust

WCAG Requirements

Principle 1 - Perceivable

Guideline 1.1 - Text Alternatives

Success Criteria 1.1.1 Non-text Content - Level A

All non-text content has a text alternative, except:

- Time-based Media
- CAPTCHA
- Decoration, Formatting, Invisible

Situation A - F, 12 - 16 techniques for each situation.

Principle 1 - Perceivable

Guideline 1.2 - Time-based Media

Audio, video: alternatives for prerecorded audio/video, captions, etc.
1.2.1 - 1.2.9

1.2.1 Audio-only and Video-only (Prerecorded)

Make content available to all users. Example:

- An audio recording of a press conference

  Web page: link to the recording, and also a link to a text transcript of the conference.

Principle 1 - Perceivable

Guideline 1.3 - Adaptable

Content presented in different ways (e.g. simpler layout).

1.3.1 Info and Relationships

Structure and relationships can be programmatically determined or are available in text.

Visual cues:

- headings in a large, bold font
- list items preceded by a bullet
- form fields arranged in groups
- use of different background colour for relationship, etc.

Principle 1 - Perceivable

Guideline 1.4 - Distinguishable

Separating foreground from background.

1.4.3 Contrast (Minimum)

A contrast ratio of at least 4.5 : 1, except for the following (AA):

- Large Text - minimum contrast of 3 : 1
- Decoration, invisible
- Logotypes

Principle 2 - Operable

Guideline 2.1 - Keyboard Accessible

2.1.1 Keyboard

All functionality of the content is operable through a keyboard interface.

2.1.2 No Keyboard Trap

Keyboard users do not become trapped in a subset of the content.

Principle 2 - Operable

Guideline 2.4 - Navigable

2.4.1 Bypass Blocks

2.4.2 Page Titled

2.4.3 Focus Order

2.4.4 Link Purpose

Can be determined from the link text or the link text + link context.

Principle 3 - Understandable

Guideline 3.1 - Readable

3.1.X Language of page, parts, unusual words, abbreviations, pronunciation

3.1.5 Reading Level - AAA

When text requires reading ability more advanced than the lower secondary education level, supplemental content, or a version that does not require more ability, is available.

Principle 4 - Robust

Guideline 4.1 - Compatible

Maximize compatibility with current and future agents, including AT.

4.1.1 Parsing

Use a correct and strict syntax: start/end tags, no duplicate attributes, unique id-s, etc.

4.1.2 Name, Role, Value

For all user interface components:

- The name and role can be programmatically determined
- State values can be user/or programmatically set
- Notification of changes is available to user agents, AT

Implementation - Techniques

ARIA technique ARIA11 - Using ARIA landmarks to identify regions of a page

This technique relates to:

- Success Criterion 1.3.1 (Info and Relationships)
- Success Criterion 2.4.1 (Bypass Blocks)

ARIA Landmarks

- banner
- complementary - separate from & additional to main content
- contentinfo - copyright and links to privacy statements
- main - usually only one main content
- form , navigation, search, application

Using ARIA landmarks - ARIA11

Some **HTML5 tags** may be treated also as landmarks:

- header
- nav
- main
- section
- article
- footer

Example:

```
<body role=”application”>
<header role=”banner”>
```

Categorization of Roles

Roles are categorized as follows:

- Abstract Roles - must not use, base classes from which all others inherit
- Widget Roles (e.g. alert, button, checkbox, searchbox, tab, tabitem, treeitem)
- Document Structure Roles (e.g. article, document, heading, listitem, toolbar)
- Landmark Roles
- Live Region Roles (e.g. alert, status)

General Technique G1

Adding a link at the top of each page that goes directly to the main content area

Provide a means to skip information **repeated** on each page, which come before the main content in the normal order: banner, logos, menu, a breadcrumb trail, search field, sidebars, ads.

Preferable when there is **one main** content area.

W3C suggests the links to be visible all the time, however this is not a requirement.

Design Patterns and Widgets

How to make rich web applications accessible.

- Modal and Alert Dialog
- Autocomplete
- Combobox
- Button
- Link
- Checkbox
- Listbox
- Menu or Menubar
- and many more...

Modal Dialog

- is a window overlay
- has its own tab sequence
- the parent window is inert

Keyboard Interaction

- TAB: moves focus to the next focusable element inside the dialog
- Shift + TAB: moves focus to the previous focusable element
- Escape: closes the dialog

WAI-ARIA Roles, States, and Properties

- the dialog container has a role=”dialog”
- has an aria-labelledby or an aria-label property
- may have an aria-describedby property

Using `alt` attributes on `img` elements

An easy to understand requirement for SC 1.1.1. Exceptions:

- Situation E: Captcha
- Situation F: **Decoration**, Formatting, Invisible

Using **icons** in the UI

Icons used for pure decoration or visual styling (logo):

```
<i class="fa fa-fighter-jet" aria-hidden="true"></i>
```

Icons with semantic purpose but non-interactive:

```
<i class="fa fa-fighter-jet" aria-hidden="true"></i><span
class="sr-only">Time to destination by bike:</span>
```

Using `alt` attributes on `img` elements

Icons with semantic purpose and focusable

Add an `aria-label` to the interactive element:

```
<a href="path/to/shopping/cart" aria-label="View 3 items
in your shopping cart">
  <i class="fa fa-shopping-cart" aria-hidden="true"></i>
</a>
```

Add a `title` for the sighted mouse users:

```
<i class="fa fa-trash-o" aria-hidden="true" title="Delete
this item?"></i>
```

Using aria-label and other tags

Use **aria-label** if the element is focusable and a text label is not visible on the screen. Sometimes you build a description from the item properties.

E.g.: "Change order ". $amount ." ".$currency_code

aria-labelledby, aria-describedby:

```
<div role="menuitem" id="fileMenu">File</div>
<div role="menu" aria-labelledby="fileMenu">
    <div role="menuitem">Open</div>
```

Usage of tabindex

`tabindex` is used for keyboard navigation of a webpage.

- `tabindex = "-1"` ⇒ element is not tab/keyboard focusable, but can set focus with script/ mouse
- `tabindex = "0"` ⇒ element is keyboard focusable
- `tabindex > 0` is not recommended, so do not use it. There is a proposal at W3C to declare tabIndex values > 0 invalid in the specification.

Usage of tabindex “0” and “-1” in ARIA keyboard navigation

`jquery.aria.key-nav.js`

```
$('#my-menubar').managefocus('a', {role: 'menubar'})
```

ARIA Live Regions

Alert the screen reader to a dynamic change.

aria-live=”polite” (“off”, “assertive”)
aria-controls=”list of ID-s to control”

Simple Use Case: Combobox

```
<select aria-controls="bird-info"><option> .... </select>
<div role="region" id="bird-info" aria-live="polite">
```

As the user selects a new bird, the info is spoken, when the user pauses.

Specialized Live Region Roles

In some well-known predefined cases, use these roles instead:

- **`log`**
  add a redundant `aria-live="polite"`
- **`status`**
  add a redundant `aria-live="polite"`
- **`alert`**
  adding `aria-live="assertive"` ⇒ double speaking issues in iOS
- **`progressbar`**
- **`timer`**

Conclusion

- Do always ARIA programming for production webapps,
  from the project start, if possible
- Do NOT rely on adding it later ⇒ it will require much more effort
- Be motivated by real disabled needs, and not webapp analysers
- ChromeVox and Accessibility Developer Tools is your friend
- Don’t strive to be perfect - you’ll never be

Questions