

# Session breakdown

- Presentation (50-60 minutes)
- Short break (~10 minutes)
- Practical lab exercise (rest of session)

# User Interface Design for Vision Impaired Users

*Screen Readers and Other Assistive  
Technologies*

---

James McLean  
Supervisor - Andrew Ireland

# Overview

Purpose - Improve knowledge and understanding of accessibility requirements and how to support assistive technologies.

We will cover:

- How we interact with computers (HCI)
- Accessibility considerations
- How screen readers work
- How we can implement accessibility support

Supported with a practical exercise

HERIOT-WATT UNIVERSITY

MASTER CLASS REPORT

**User Interface Design for  
Vision Impaired Users**  
*Screen Readers and Other Assistive  
Technologies*

James McLean  
*Software Engineering MEng*

*Supervised by  
Andrew Ireland*

Rate your current knowledge

Wooclap code: MCSR

<https://app.wooclap.com/MCSR>



# What is a screen reader?

- Scans elements on the screen.
- Generates audio description of the UI.
- Describes elements as the user navigates through the UI.
- Informs the user of any dynamic element changes.
- Can be controlled with a keyboard, or another keyboard-navigation based device.

How do we interact with computers?

# Computers are inescapable

- Computers are used for many common information gathering tasks which previously would have been done through other means
  - Telephone
  - Letters
  - Face to face
  - etc.
- Computer systems must be usable by all!



# Digital Poverty

“The inability to interact with the online world fully, when, where, and how an individual needs to.”

*Digital Poverty Alliance*



# Human Computer Interaction

“Human-computer interaction (HCI) is a multidisciplinary field of study focusing on the design of computer technology and, in particular, the interaction between humans (the users) and computers.”

*Interaction Design Foundation*

# User Interface Design and User Experience

- UI Considerations - User-centred design
  - Layout
  - Navigation
  - Visual Design
  - Interaction Patterns
- User Experience (UX)
  - How positive (or negative) the experience of using a computer system is
- Together these form the backbone of HCI



# Conceptual Models

- Abstract representation of the mental model the user forms around a system.
- When designing a UI, an understanding of the user's understanding is crucial.



# How can we be accessible?



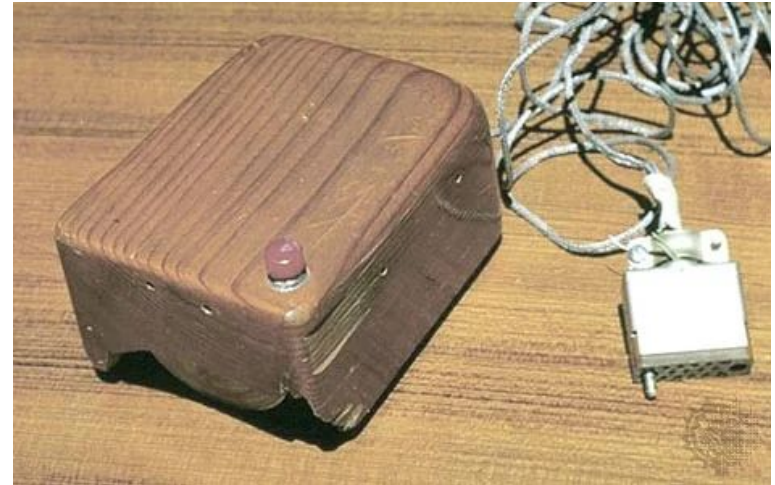
# “Design for all”

- A user interface can be used by all users, regardless of ability.
- Potential Considerations
  - Simple text
  - Easy/intuitive interface design
  - Use of appropriately contrasting colours
  - Navigable by assistive technologies
- Ignoring these considerations when designing a UI can make it completely unusable to many users.



# UI history

- Screen based user interfaces emerge in the 1960s.
- Originally text based terminals.
- 1964 - Douglas Engelbart creates first computer mouse
- Does not become standard peripheral until 1980s with wider adoption of graphical UIs.



# What's wrong with the mouse?

- Nothing (for most users).
- However, the mouse demands two expectations of its user:
  - Fine motor skill.
  - Ability to track the cursor.
- Excludes users with motor/vision impairments and cognitive difficulties



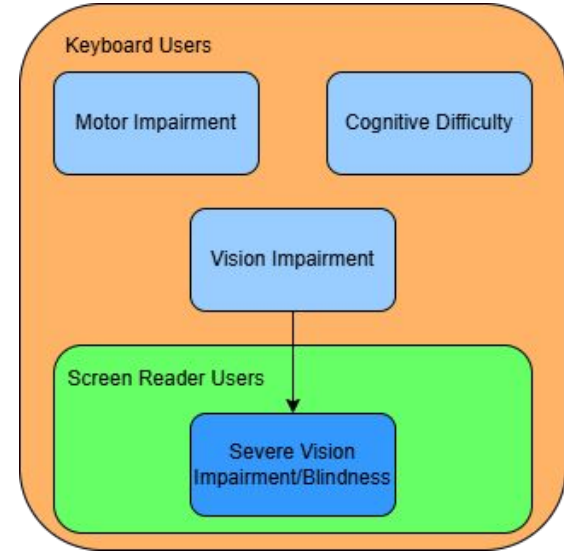
# Taxonomy of user types

- Keyboard users

- Motor impairments - cannot control mouse
- Cognitive difficulty - cannot follow cursor
- Vision impairment - cannot follow cursor

- Screen reader users

- Severe vision impairment/blindness - cannot see screen



$keyboard\ control\ users = \{motor\ impairments, vision\ impairments\}$

$screen\ reader\ users \subset keyboard\ control\ users$



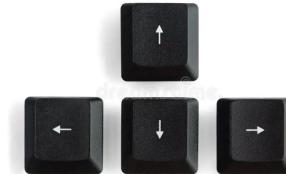
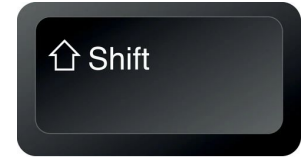
# What can be used in place of a mouse

- An alternative must be:
  - Discrete.
  - Visually bold.
- Keyboard fulfills this purpose well:
  - Key presses do not require high precision.



# Keyboard navigation

- Move linearly through page elements.
  - TAB - Forwards through page.
  - SHIFT + TAB - Back through page.
- Use arrow keys to move within elements.
- ENTER to select elements.
- Standard keyboard use of data input.
- Occasionally ESC may be required to exit an element



# Keyboard navigation - Login screen

## Sign in using Government Gateway

Government Gateway user ID  
This could be up to 12 characters.

Password

Show

Sign in

## Sign in using Government Gateway

Government Gateway user ID  
This could be up to 12 characters.

Password

Show

Sign in

## Sign in using Government Gateway

Government Gateway user ID  
This could be up to 12 characters.

Password

Show

Sign in

## Sign in using Government Gateway

Government Gateway user ID  
This could be up to 12 characters.

Password

Show

Sign in

# Keyboard navigation - UI

- Some considerations when designing a UI to support navigation:
  - Intuitive linear order of elements.
  - Clear element focus - bold, appropriately contrasting colours. (GET RATIO)
  - All available elements are focusable.
- Like TV/DVD menus the items will be individually focused, *but* in a linear order.

# Why is linear navigation so important?

- Short answer - universal support!
- There are a number of accessible technology systems we want to support
  - Screen Readers
  - Braille Readers
  - Sip and puff systems
  - etc.
- These need a simple linear navigation method and will integrate with keyboard support



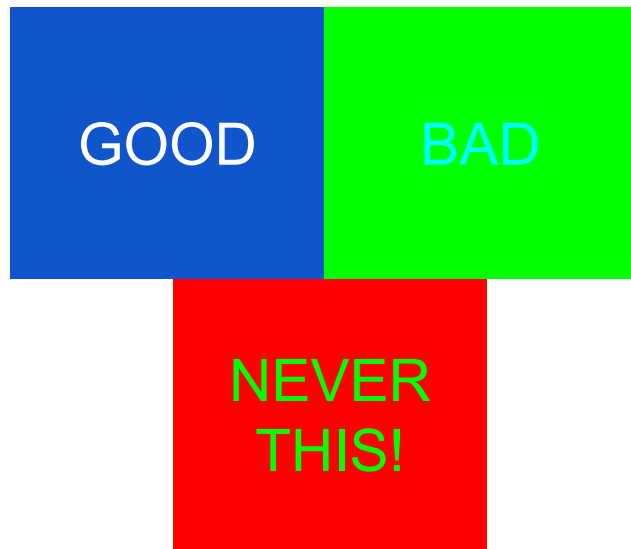
# Screen Readers

# We cannot rely on visual output alone

- So far we have covered the accessibility issues posed by common input methods, so what about output?
- There are two common output methods in modern computing
  - Visual - via the monitor/screen (generally the main output system)
  - Auditory - via speakers/headphones
- How do users with a vision impairment/blindness use visual UIs?

# Lower severity vision impairments

- Colour blindness
  - Normal text colour contrast 7:1
  - Large text colour contrast 4.5:1
  - UI components and graphics colour contrast 3:1 (WCAG 2.2 success criteria 1.4.6 and 1.4.11)
- Partial sight
  - Magnification - page elements must scale well
  - May use a screen reader in conjunction visual interface



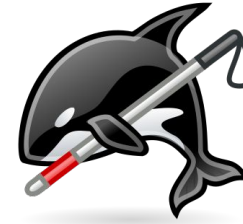


# What is a screen reader? (again)

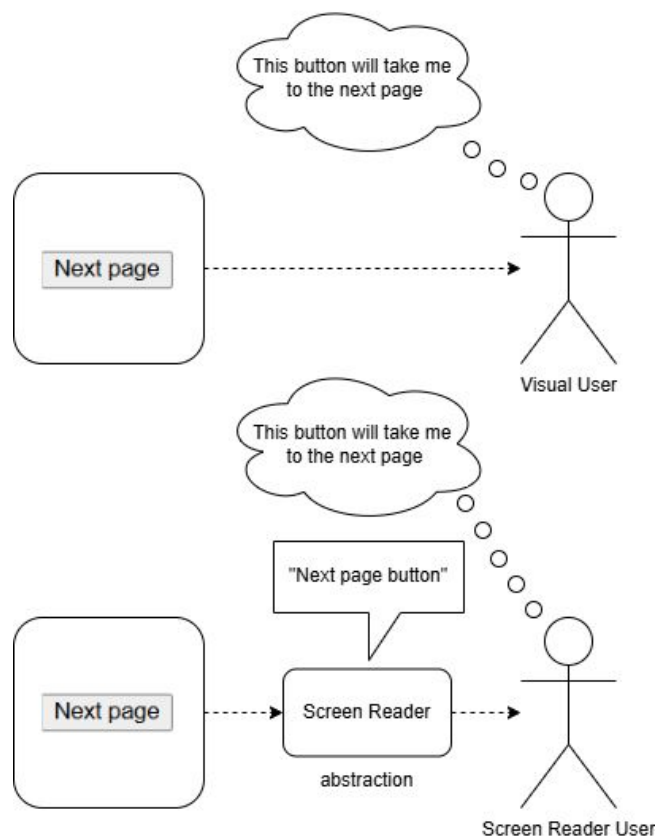
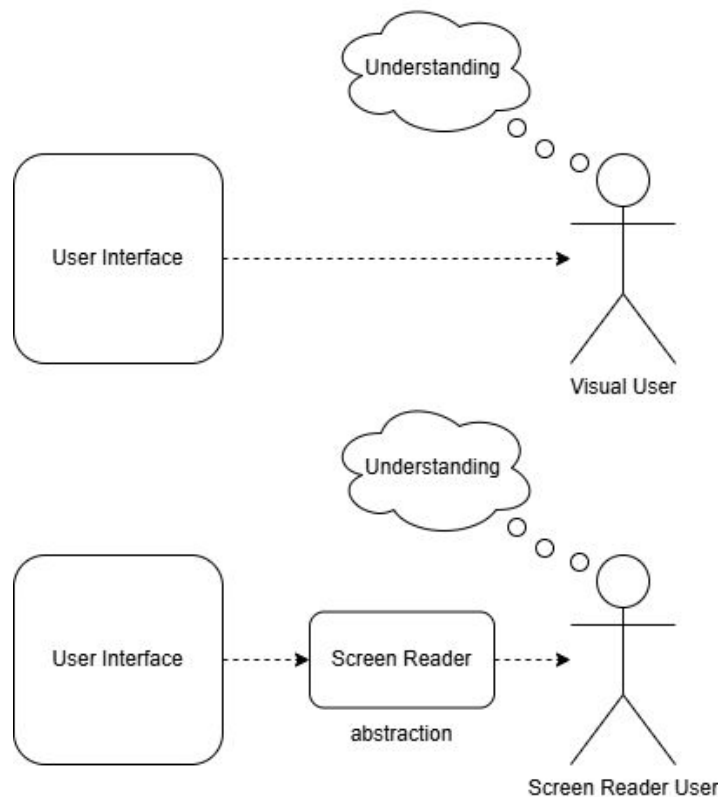
- Scans elements on the screen.
- Generates audio description of the UI.
- Describes elements as the user navigates through the UI.
- Informs the user of any dynamic element changes.
- Can be controlled with a keyboard, or another keyboard-navigation based device.

# Screen reader examples

- NVDA - Non-Visual Desktop Access
  - Widely used
  - Free and open-source
  - Arguably the industry standard
- JAWS - Job Access with speech
  - Proprietary - starting at \$85p/a for a home licence
  - JAWS Scripting Language - Allows for easier support of desktop applications
- Orca (Linux)
- Microsoft Narrator
  - Built into Windows
- VoiceOver
  - Built into Apple OSs




# Screen reader mental model



# Mental model - How would a screen reader user understand this?

[Home](#) [News](#) [Contact](#) [About](#)

## Contact us



Full Name

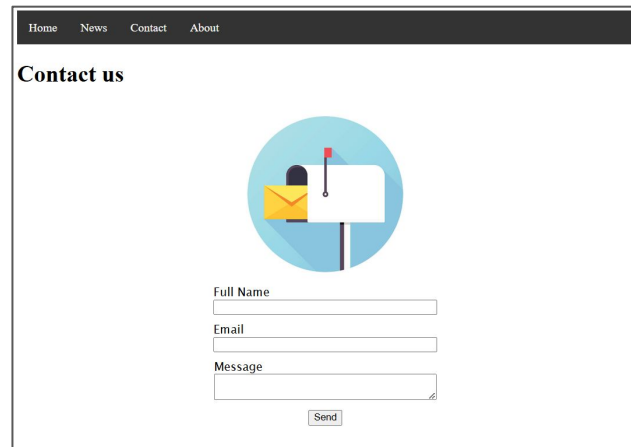
Email

Message

Send

# Mental model - How would a screen reader user understand this?

- [Name of page]
- “Navigation bar with four links: Home, News, Contact, About”
- “Heading level one: Contact us.”
- “Image: An American style mail box.”
- “Text edit: Full name, blank”
- “Text edit: Email, blank”
- “Text edit: Email, blank”
- “Text area: Message, blank”
- “Submit button”



The screenshot shows a web page with a dark navigation bar at the top containing links for Home, News, Contact, and About. Below the navigation bar is a heading 'Contact us'. Centered on the page is a circular graphic of a blue mailbox with a yellow envelope and a red flag. Below the graphic are three input fields: 'Full Name', 'Email', and 'Message'. The 'Full Name' and 'Email' fields are single-line text inputs, while the 'Message' field is a multi-line text area. A 'Send' button is located at the bottom right of the form.

# Easy avoidances when designing for screen readers

- What's wrong with this?

Name:

- Unclear
- Just first name?
- Full name?
- Really this is bad form design in general, not just for screen reader users

# A more relevant example for screen readers

- What's wrong with this?

Address:


## A much clearer version

- Fields **must** always have an associated label
- Label names must be unambiguous

Address Line 1:

Address Line 2:

Address Line 3:

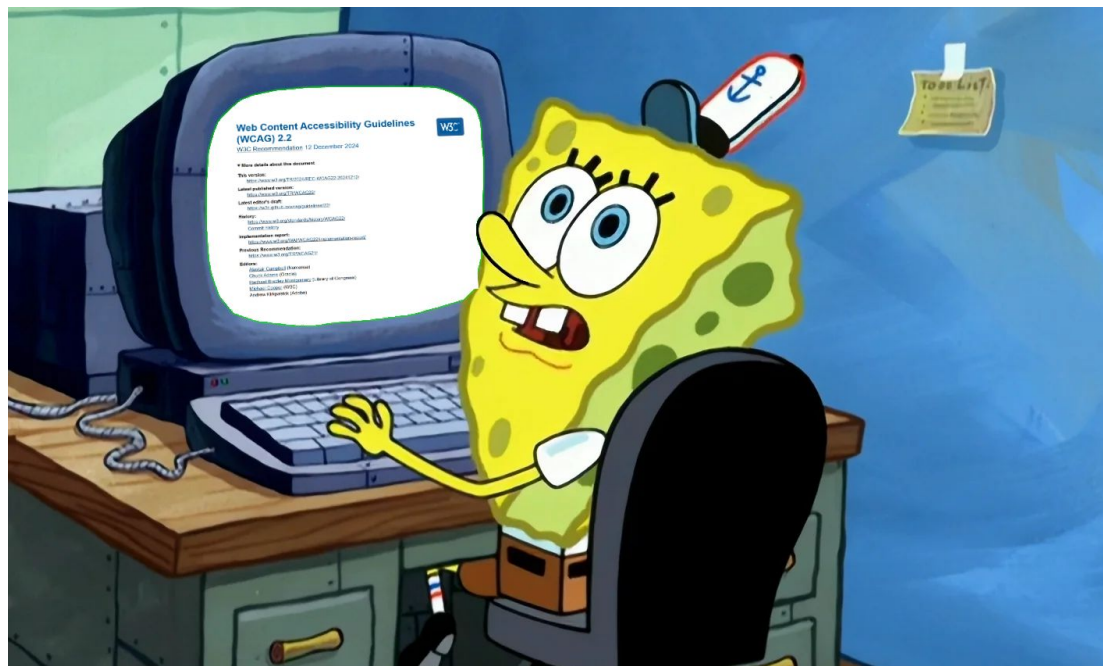
Address Line 4:



# Braille readers

- Works similarly to a screen reader
- Output UI description is given through a refreshable Braille display

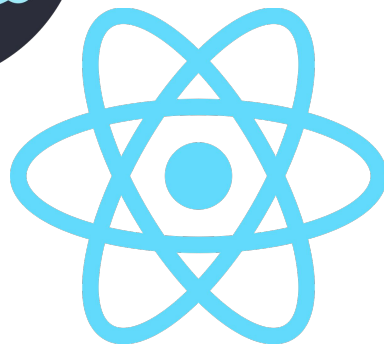




How can we implement accessibility features?

# Focus on the web

- We will focus on web technologies for a few reasons:
  - Prevalence of web applications in everyday life
  - Desktop applications moving to web based frameworks - Electron, React Native, etc.
  - There is an extensive and universally accepted set of accessibility guidelines



# Web Content Accessibility Guidelines (WCAG)

- Version 1 released 1999
- Focuses on HTML features that can be utilised to support accessible technology
- Three levels of compliance: A, AA, AAA
  - Must meet all relevant guidelines to achieve compliance level



# WCAG 2.0

- Released in 2008
- Introduced four principles
  - Perceivable
  - Operable
  - Understandable
  - Robust
- Guidelines are grouped by these principles



# WCAG 2.1 & 2.2

- 2.1
  - 2018
  - Update to better reflect how web use has changed since 2008
  - More dynamic webpages, mobile browsing, etc.
- 2.2
  - 2023
  - Minor update similar to 2.1



# WCAG 3

- Announced in 2021
- Complete overhaul
- Most recent working draft December 2024
  - Barebones overview of guidelines
  - Most branded with “Needs addition research”



# How can we achieve WCAG compliance?

- Levels A, AA, AAA
- Website must satisfy all level A success criteria to achieve level A compliance





# How can we achieve WCAG compliance?

## § Guideline 2.1 Keyboard Accessible

Make all functionality available from a keyboard.

[Understanding Keyboard Accessible](#)

[How to Meet Keyboard Accessible](#)

## § Success Criterion 2.1.1 Keyboard

(Level A)

All [functionality](#) of the content is operable through a [keyboard interface](#) without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.

[Understanding Keyboard](#)

[How to Meet Keyboard](#)

...

## § Success Criterion 2.1.3 Keyboard (No Exception)

(Level AAA)

All [functionality](#) of the content is operable through a [keyboard interface](#) without requiring specific timings for individual keystrokes.

[Understanding Keyboard \(No Exception\)](#)

[How to Meet Keyboard \(No Exception\)](#)

# Accessibility statement

- State compliance level
- List areas of non-compliance
- Example (left)
  - gov.uk
  - Regarded as a gold standard
  - Yet only achieves partial level AA

## Compliance status

This website is partially compliant with the [Web Content Accessibility Guidelines version 2.2](#) AA standard, due to the non-compliances and exemptions listed below.

## Non-accessible content

The content listed below is non-accessible for the following reasons.

### Non-compliance with the accessibility regulations

1. Images on some pages do not always have suitable image descriptions. Users of assistive technologies may not have access to information conveyed in images. This fails WCAG 2.2 success criterion 1.1.1 (Non-text Content).
2. Some tables do not have table row or column headers. This means assistive technologies will not read the tables correctly. This fails WCAG 2.2 success criterion 1.3.1 (Info and Relationships).
3. Some tables are structured incorrectly, so screen readers cannot understand the relationships between information in the table. This fails WCAG 2.2 success criterion 1.3.1 (Info and Relationships).

# WAI-ARIA

- Web Accessibility Initiative
  - Oversees the WCAG standard
- Accessible Rich Internet Applications
  - Collection of roles and attributes extending standard HTML to support assistive technology



# The first rule of ARIA



You do **NOT** use ARIA

(unless absolutely necessary)

# Using ARIA Rule 1

*Do not use ARIA unless absolutely necessary - always use standard HTML/CSS features where possible.*

# Using ARIA Rule 2

*Do not change native semantics, unless you really have to - avoid making completely custom elements with ARIA attributes.*

# Using ARIA Rule 3

*All interactive ARIA controls must be usable with the keyboard.*

# Using ARIA Rule 4

*Do not use `role="presentation"` or `aria-hidden="true"` on a focusable element - these attributes will cause the element to be completely ignored by assistive technologies.*



# Using ARIA Rule 5

*All interactive elements must have an accessible name - this name must provide all context derivable from the visual interface.*

# ARIA - role attribute

- Used to break the page into sections
- This creates landmarks for element focus

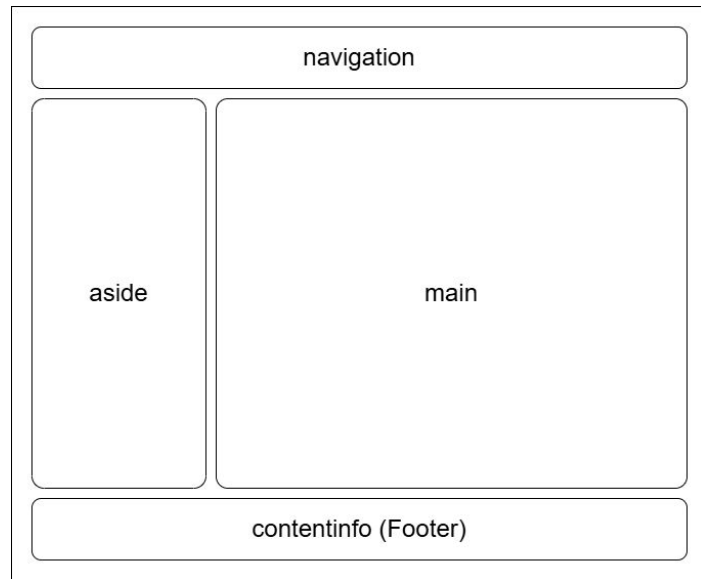
```
<nav role="navigation">  
  ...  
</nav>
```

```
<div role="main">  
  ...  
</div>
```

```
<footer role="contentinfo">  
  ...  
</footer>
```

# Breaking things down

- We can break the page into clear sections to add landmarks
- A possible layout is:
  - Navbar - role="navigation"
  - Side menu - role="aside"
  - Main content - role="main"
  - Footer - role="contentinfo"
- The landmarks aid in description generation as well as keyboard navigation



# ARIA - aria-expanded

- Used to indicate the state of an expandable element
- Attribute will need to be updated with JavaScript
- Should be supported by default for standard `<select>` element

`aria-expanded="false"`

WCAG version: WCAG 1 ▼

`aria-expanded="true"`

WCAG version: WCAG 1 ▼

- WCAG 1
- WCAG 2.0
- WCAG 2.1
- WCAG 2.2

# ARIA - aria-labelledby

- Used for selecting the relevant label element
- Can be used to label non-form elements if appropriate
- Can be used to reference multiple labels
- Generally used as well as the *for* attribute to guarantee connection for accessible technologies

```
<label for="username-input"
id="username-label">
    Username
</label>
```

```
<input id="username-input"
aria-labelledby="username-label">
```

# ARIA - aria-label

- Used to define an explicit label description for the element
- Use with caution - always reference an actual label element where possible

```
<label for="price-input"
aria-hidden="true">
    Price
</label>
<small aria-hidden="true">
    (GBP)
</small>

<input id="price-input"
aria-label="Price in GBP.">
```

# More ARIA attributes

- aria-autocomplete
- aria-checked
- aria-disabled
- aria-errormessage
- aria-haspopup
- aria-hidden
- aria-invalid
- aria-level
- aria-modal
- aria-multiline
- aria-multiselectable
- aria-orientation
- aria-activedescendant
- aria-colcount
- aria-colindex
- aria-colspan
- aria-controls
- aria-details
- aria-errormessage
- aria-flowto
- aria-labelledby
- aria-owns

And many, many more...

<https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Reference/Attributes>

# Further considerations - Skip to content button

- Skip to content button allows user to skip the consistent elements at the top of every page
- First element in the page
- Hidden unless focused
- Jumps focus to “main” landmark





## Further considerations - tabindex

- Used to set precedence of focus
- Elements with `tabindex="1"` will be focused before elements with `tabindex="2"`...
- Can be useful if the DOM is unable to determine an intuitive tab order from the order of elements
- `tabindex="0"` can be used to make a non-focusable element focusable

## Quiz - Which of the following is an effective use of WAI-ARIA?

- a. 

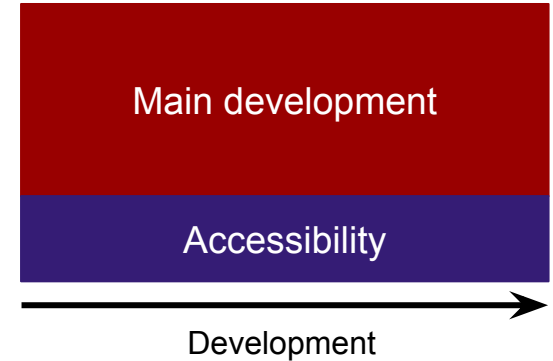
```
<label id="name-field-label" for="name-field">Full Name:</label>  
<input type="text" id="name-field">
```
- b. 

```
<label_id="name-field-label" for="name-field">Full Name:</label>  
<input type="text"_id="name-field" aria-label="Full Name">
```
- c. 

```
<label_id="name-field-label" for="name-field">Full Name:</label>  
<input type="text"_id="name-field" aria-labelledby="name-field-label">
```

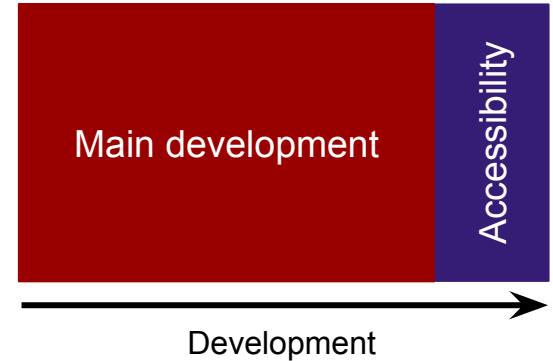
# Testing and evaluation - Ad-hoc

- Implementing accessibility features through development
- Requires deep knowledge of WCAG
- Slower initial development, but potentially more effective



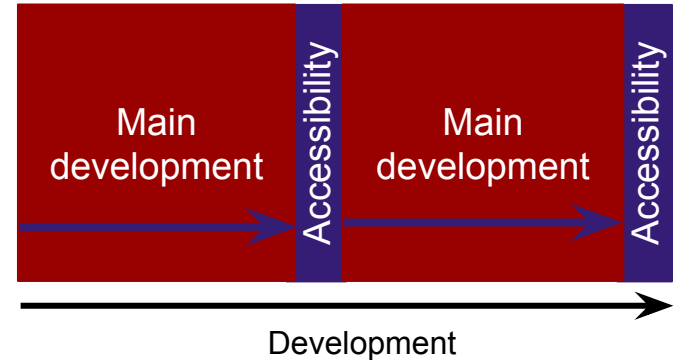
# Testing and evaluation - Post-hoc

- Implementing accessibility features as a secondary development phase
- Can use browser extensions to isolate issues
- There are services which regularly test live pages and report any issues
- Possibly faster in the long run depending on how many issues arise



# A third approach - Accessibility by design

- Most realistic
- Make primary and secondary development phases more granular
  - i.e. break the main development up with a regular accessibility pass
- Accessible by design systems
  - Well Formed for creating web form elements
  - For web application frameworks such as React or Angular, element libraries will generally design their elements to be WCAG compliant



# Summary

- How we interact with computers - Problems with the mouse
- Accessibility considerations
  - Keyboard control
  - Linear navigation
- How screen readers work
  - Generates an audio description of the UI
  - Describes interactive elements as they are selected
- How we can implement accessibility support
  - Use WCAG for requirements
  - Use WAI-ARIA (responsibly) when developing assistive technology support
  - Find the right methodology

# Further Reading

User Interface Design for Vision Impaired Users: Screen Readers and Other Assistive Technologies (companion to this lecture) - James McLean

W3Cx: Introduction to Web Accessibility

<https://www.edx.org/learn/web-accessibility/the-world-wide-web-consortium-w3c-introduction-to-web-accessibility>

WCAG 2.2 (latest published version)

<https://www.w3.org/TR/WCAG22/>

WCAG 3 (latest published version)

<https://www.w3.org/TR/wcag-3.0/>

A Generator for Accessible HTML Forms - James McLean

<https://www.jmcl.xyz/dissertation>