**Lab Exercise 2- Implementing WCAG 2.x Operable Features in a Web Page**

**Objective**

To implement and validate accessibility features that ensure users can interact with a website **without relying on a mouse**, **follow logical focus navigation**, and **avoid flashing content** that may trigger seizures.

**Pre-requisites**

* Basic HTML, CSS, and JavaScript knowledge.
* Familiarity with browser DevTools and screen reader simulation (optional).
* Access to accessibility testing tools (e.g., Keyboard-only testing, Lighthouse, Axe).

**Part 1: Keyboard Accessibility (2.1.1)**

**Goal**

Ensure all interactive elements (links, buttons, forms) are fully usable with a keyboard.

**Steps**

1. Create a simple interactive HTML interface:

<button onclick="alert('Submitted!')">Submit</button>

<a href="#info">Go to Info Section</a>

<input type="text" placeholder="Enter your name">

1. Make custom components (like menus or modals) keyboard-accessible:

<div role="button" tabindex="0" onclick="openMenu()" onkeypress="if(event.key==='Enter'){openMenu()}">

Open Menu

</div>

**Explanation**

* All elements must be focusable using the Tab key.
* tabindex="0" adds keyboard focus to non-focusable elements.
* onkeypress ensures functionality with Enter/Space keys.

**Test**

* Navigate using Tab, Shift+Tab, Enter, and Space.
* Confirm all functionality works without a mouse.

**Part 2: Focus Order (2.4.3)**

**Goal**

Ensure users navigate in a logical, predictable order when using the keyboard.

**Steps**

1. Structure your form or navigation in logical source order:

<h2>Login</h2>

<form>

<label for="username">Username:</label>

<input id="username" type="text"><br>

<label for="password">Password:</label>

<input id="password" type="password"><br>

<button type="submit">Login</button>

</form>

1. Avoid disrupting natural tab order using negative or high tabindex values.
2. Highlight focus with CSS:

:focus {

outline: 2px solid #007acc;

}

**Explanation**

* Logical tab order helps screen reader and keyboard users move predictably.
* Improper ordering confuses users and breaks accessibility.

**Test**

* Press Tab repeatedly and follow the visual focus.
* Use DevTools “Accessibility” pane to inspect tab order.

**Part 3: Avoiding Seizures (2.3.1)**

**Goal**

Prevent flashing or strobing content that may trigger seizures in users with photosensitive epilepsy.

**Steps**

1. Avoid using animations or flashing content more than **three times per second**.
2. If using animation, ensure it is subtle and not rapidly blinking.

**Bad Example** (DO NOT USE):

<!-- Avoid this type of flashing -->

<div style="animation: flash 0.2s infinite;">

Warning!

</div>

<style>

@keyframes flash {

0% { background-color: red; }

50% { background-color: white; }

100% { background-color: red; }

}

</style>

**Good Example**:

<div style="background-color: red; color: white; padding: 10px;">

⚠️ Important Notice: Please read carefully.

</div>

**Explanation**

* WCAG 2.3.1 prohibits content that flashes more than 3 times per second.
* Flashing content must be tested with a tool like the **Photosensitive Epilepsy Analysis Tool (PEAT)** if needed.

**Use Case Scenario**

**Use Case**: A public health portal is accessed by users of all ages and abilities. It must be fully accessible without a mouse and ensure no visuals pose health risks.

**Implementation**:

* All forms and menus accessible via Tab and Enter.
* Logical focus order maintained.
* Focus indicators styled for visibility.
* No blinking banners or rapidly changing backgrounds used.

**Outcome**:

* Fully compliant with WCAG 2.1 Level A and AA requirements.
* Lower bounce rate and higher accessibility rating in audits.

**Accessibility Testing Checklist (Operable)**

| **Feature** | **Test Description** | **Status** |
| --- | --- | --- |
| Keyboard Navigation | Use Tab to navigate through all controls | ✅ |
| Keyboard-Only Functionality | Submit buttons and menus work via keyboard | ✅ |
| Logical Focus Order | Tab moves in reading order (top to bottom) | ✅ |
| Visible Focus Indicators | Focus is clearly outlined when tabbing | ✅ |
| No Flashing Elements | No element flashes more than 3 times per second | ✅ |

**Conclusion**

This lab exercise demonstrates the application of **Operable** accessibility principles, helping developers and designers ensure that users can interact with web content using **keyboards** and **assistive technologies**, and without risk to health. Ensuring operability increases inclusivity and compliance with laws like **ADA** and **EN 301 549**.