Tài liệu: Accessibility (A11y) trong HTML và CSS

Mục tiêu

- Hiểu rõ **accessibility (A11y)** là gì, tầm quan trọng và cách áp dụng trong HTML và CSS.
- Nắm vững các kỹ thuật sử dụng HTML và CSS để tạo giao diện web dễ tiếp cận, tuân thủ **WCAG 2.1/2.2**.
- Cung cấp ví dụ minh họa và bài tập thực hành để áp dụng accessibility vào các dự án thực tế.

Đối tượng

- Sinh viên hoặc lập trình viên đã biết HTML và CSS cơ bản (tag, attribute, selector, box model).
- Người muốn xây dựng các trang web thân thiện với người khuyết tật (mù, điếc, hạn chế vận động, v.v.).
- Lập trình viên cần tuân thủ tiêu chuẩn accessibility cho các dự án thương mại hoặc công cộng (2025).

---

1. Định nghĩa

Accessibility (A11y) là việc thiết kế và phát triển các trang web, ứng dụng hoặc công cụ số để đảm bảo rằng mọi người, bao gồm cả những người có khuyết tật (như mù, điếc, hạn chế vận động, hoặc suy giảm nhận thức), đều có thể sử dụng một cách hiệu quả. Trong HTML và CSS, accessibility tập trung vào việc sử dụng các semantic tag, thuộc tính ARIA, và kiểu dáng CSS để hỗ trợ công nghệ hỗ trợ (assistive technologies) như trình đọc màn hình (screen readers), bàn phím điều hướng, hoặc công cụ phóng to.

Tiêu chuẩn WCAG

- **WCAG (Web Content Accessibility Guidelines)**: Bộ hướng dẫn quốc tế từ W3C, cung cấp các tiêu chí để đảm bảo accessibility. Các cấp độ tuân thủ: **A**, **AA**, **AAA** (AA là mức phổ biến nhất).
- Nguyên tắc chính (**POUR**):
    - **Perceivable**: Nội dung phải dễ nhận biết (ví dụ: văn bản thay thế cho hình ảnh, độ tương phản cao).
    - **Operable**: Giao diện phải dễ vận hành (ví dụ: điều hướng bằng bàn phím, thời gian đủ để đọc nội dung).
    - **Understandable**: Nội dung phải dễ hiểu (ví dụ: ngôn ngữ rõ ràng, cấu trúc logic).
    - **Robust**: Nội dung phải tương thích với nhiều công nghệ hỗ trợ (ví dụ: HTML semantic, ARIA).

---

2. Cách hoạt động

HTML Accessibility

- **Semantic HTML**: Sử dụng các thẻ HTML đúng ngữ nghĩa (như `<header>`, `<nav>`, `<main>`, `<article>`, `<footer>`) để cung cấp cấu trúc rõ ràng cho trình đọc màn hình.

- **Thuộc tính ARIA (Accessible Rich Internet Applications)**: Bổ sung thông tin ngữ nghĩa cho các phần tử không có semantic rõ ràng (ví dụ: `role`, `aria-label`, `aria-hidden`).
- **Thuộc tính chuẩn HTML**:
  - `alt` cho hình ảnh.
  - `lang` để chỉ định ngôn ngữ của trang.
  - `tabindex` để quản lý thứ tự điều hướng bàn phím.
- **Form accessibility**: Sử dụng `<label>`, `for`, và ARIA để đảm bảo form dễ sử dụng với bàn phím và trình đọc màn hình.

CSS Accessibility

- **Độ tương phản (Contrast)**: Đảm bảo văn bản có độ tương phản đủ với nền (tỷ lệ tối thiểu 4.5:1 cho văn bản thường, 3:1 cho văn bản lớn, theo WCAG AA).
- **Kích thước chữ**: Sử dụng đơn vị tương đối (`rem`, `em`) để hỗ trợ phóng to/thu nhỏ.
- **Focus styles**: Cung cấp kiểu dáng rõ ràng cho trạng thái `:focus` (ví dụ: viền nổi bật khi điều hướng bằng bàn phím).
- **Responsive design**: Đảm bảo bố cục hoạt động tốt trên mọi thiết bị, không che khuất nội dung khi phóng to.
- **Không phụ thuộc vào màu**: Không sử dụng màu sắc làm phương thức duy nhất để truyền tải thông tin (ví dụ: lỗi form cần văn bản, không chỉ màu đỏ).
- **Ẩn nội dung đúng cách**: Sử dụng kỹ thuật ẩn nội dung (`display: none`, `visibility: hidden`, hoặc clip-path) phù hợp để không làm rối trình đọc màn hình.

Công nghệ hỗ trợ

- **Trình đọc màn hình**: JAWS, NVDA, VoiceOver, TalkBack.
- **Điều hướng bàn phím**: Người dùng chỉ sử dụng phím Tab, Enter, hoặc phím mũi tên.
- **Công cụ phóng to**: Zoom trình duyệt hoặc phần mềm phóng to màn hình.

---

3. Cú pháp và kỹ thuật

HTML Accessibility

```
<!-- Semantic HTML -->
<main>
  <article>
    <h1>Tiêu đề bài viết</h1>
    <p>Nội dung bài viết...</p>
  </article>
</main>

<!-- Hình ảnh với văn bản thay thế -->
<img src="example.jpg" alt="Mô tả ngắn gọn về hình ảnh">
```

```
<!-- Form với label và ARIA -->
<form>
  <label for="name">Họ và tên:</label>
  <input id="name" type="text" aria-required="true">
</form>

<!-- ARIA cho button không semantic -->
<button role="button" aria-label="Đóng cửa sổ">X</button>
```

CSS Accessibility

```
/* Độ tương phản cao */
body {
  font-size: 1rem; /* Đơn vị tương đối */
  color: #333; /* Tương phản tốt với nền trắng */
  background-color: #fff;
}

/* Focus styles rõ ràng */
button:focus {
  outline: 2px solid #007bff;
  outline-offset: 2px;
}

/* Ẩn nội dung chỉ cho trình đọc màn hình */
.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  border: 0;
}

/* Responsive font size */
h1 {
  font-size: 2rem; /* Tương đối với root */
}
```

---

4. Ví dụ minh họa

Ví dụ cơ bản: Form nhập liệu dễ tiếp cận

Tạo một form với label, văn bản lỗi, và focus styles.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Accessible Form</title>
  <style>
    .form-group {
      margin-bottom: 1rem;
    }
    label {
      display: block;
      font-size: 1rem;
      color: #333;
    }
    input {
      padding: 0.5rem;
      font-size: 1rem;
      border: 2px solid #ccc;
      border-radius: 4px;
      width: 100%;
      max-width: 300px;
    }
    input:focus {
      outline: 2px solid #007bff;
      outline-offset: 2px;
    }
    .error {
      color: #d32f2f;
      font-size: 0.875rem;
      margin-top: 0.25rem;
    }
    .error input {
      border-color: #d32f2f;
    }
  </style>
</head>
<body>
  <form>
    <div class="form-group">
      <label for="email">Email:</label>
      <input id="email" type="email" aria-required="true" aria-
describedby="email-error">
      <span id="email-error" class="error" hidden>Vui lòng nhập email hợp
lệ.</span>
    </div>
```

```
    <button type="submit">Gửi</button>
  </form>
</body>
</html>
```

Giải thích:

- **HTML**:
  - `lang="vi"`: Chỉ định ngôn ngữ tiếng Việt cho trình đọc màn hình.
  - `<label for="email">`: Liên kết label với input, giúp trình đọc màn hình đọc đúng.
  - `aria-required="true"`: Thông báo trường bắt buộc.
  - `aria-describedby="email-error"`: Liên kết input với thông báo lỗi.
- **CSS**:
  - `font-size: 1rem`: Đảm bảo kích thước chữ dễ đọc, tương đối với root.
  - `:focus`: Viền xanh nổi bật khi điều hướng bằng bàn phím.
  - `.error`: Văn bản lỗi màu đỏ, không chỉ dựa vào màu (có nội dung văn bản rõ ràng).

Ví dụ nâng cao 1: Navigation bar dễ tiếp cận

Tạo một menu điều hướng hỗ trợ bàn phím và trình đọc màn hình.

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Accessible Navigation</title>
  <style>
    nav {
      background-color: #333;
      padding: 1rem;
    }
    .nav-list {
      display: flex;
      list-style: none;
      margin: 0;
      padding: 0;
      gap: 1rem;
    }
    .nav-list a {
      color: #fff;
      text-decoration: none;
      padding: 0.5rem 1rem;
      font-size: 1rem;
      border-radius: 4px;
    }
```

```
    .nav-list a:hover,
    .nav-list a:focus {
      background-color: #555;
      outline: 2px solid #007bff;
      outline-offset: 2px;
    }
    .nav-list a[aria-current="page"] {
      background-color: #007bff;
      font-weight: bold;
    }
    @media (max-width: 600px) {
      .nav-list {
        flex-direction: column;
      }
    }
  </style>
</head>
<body>
  <nav aria-label="Điều hướng chính">
    <ul class="nav-list">
      <li><a href="#" aria-current="page">Trang chủ</a></li>
      <li><a href="#">Giới thiệu</a></li>
      <li><a href="#">Liên hệ</a></li>
    </ul>
  </nav>
</body>
</html>
```

Giải thích:

- **HTML**:
  - `nav` với `aria-label="Điều hướng chính"`: Cung cấp ngữ cảnh cho trình đọc màn hình.
  - `aria-current="page"`: Đánh dấu trang hiện tại, giúp người dùng biết vị trí của họ.
  - `<ul>` và `<li>`: Semantic HTML cho danh sách điều hướng.
- **CSS**:
  - `display: flex`: Tạo bố cục ngang, responsive với `flex-direction: column` trên mobile.
  - `:focus`: Viền xanh cho điều hướng bàn phím.
  - Độ tương phản: Màu trắng (#fff) trên nền tối (#333) đạt tỷ lệ 12.5:1, vượt WCAG AA.

Ví dụ nâng cao 2: Modal dialog dễ tiếp cận

Tạo một modal dialog với focus management và hỗ trợ ARIA.

```
<!DOCTYPE html>
```

```
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Accessible Modal</title>
  <style>
    .modal {
      display: none;
      position: fixed;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      background-color: rgba(0, 0, 0, 0.5);
      justify-content: center;
      align-items: center;
    }
    .modal--open {
      display: flex;
    }
    .modal-content {
      background-color: #fff;
      padding: 2rem;
      border-radius: 8px;
      max-width: 500px;
      width: 90%;
    }
    .modal-content h2 {
      margin-top: 0;
      font-size: 1.5rem;
    }
    .modal-content button {
      padding: 0.5rem 1rem;
      font-size: 1rem;
      border: 2px solid #007bff;
      border-radius: 4px;
      cursor: pointer;
    }
    .modal-content button:focus {
      outline: 2px solid #007bff;
      outline-offset: 2px;
    }
    .sr-only {
      position: absolute;
      width: 1px;
      height: 1px;
```

```
      padding: 0;
      margin: -1px;
      overflow: hidden;
      clip: rect(0, 0, 0, 0);
      border: 0;
    }
  </style>
</head>
<body>
  <button onclick="openModal()">Mở Modal</button>
  <div class="modal" id="modal" role="dialog" aria-labelledby="modal-
title" aria-modal="true">
    <div class="modal-content">
      <h2 id="modal-title">Xác nhận</h2>
      <p>Bạn có muốn tiếp tục không?</p>
      <button onclick="closeModal()" autofocus>Đóng</button>
    </div>
  </div>
  <script>
    function openModal() {
      const modal = document.getElementById('modal');
      modal.classList.add('modal--open');
      modal.querySelector('button[autofocus]').focus();
    }
    function closeModal() {
      document.getElementById('modal').classList.remove('modal--open');
    }
  </script>
</body>
</html>
```

Giải thích:

- **HTML**:
  - `role="dialog"` và `aria-modal="true"`: Đánh dấu modal là một dialog.
  - `aria-labelledby="modal-title"`: Liên kết tiêu đề với modal.
  - `autofocus`: Tự động focus vào nút "Đóng" khi modal mở, hỗ trợ bàn phím.
- **CSS**:
  - `display: flex` với `justify-content` và `align-items`: Căn giữa modal.
  - `.sr-only`: Ẩn nội dung cho trình đọc màn hình mà không ảnh hưởng đến bố cục.
  - `:focus`: Viền nổi bật cho điều hướng bàn phím.

---

5. Ứng dụng thực tế

- **Website công cộng**: Đảm bảo các trang web chính phủ, giáo dục, hoặc thương mại điện tử tuân thủ WCAG để phục vụ tất cả người dùng.
- **Form nhập liệu**: Tạo form dễ sử dụng cho người dùng bàn phím hoặc trình đọc màn hình (ví dụ: form đăng ký, đăng nhập).
- **Navigation**: Xây dựng menu điều hướng hỗ trợ bàn phím và semantic HTML.
- **E-commerce**: Hiển thị sản phẩm với văn bản thay thế cho hình ảnh và độ tương phản cao.
- **Ứng dụng web**: Tạo các thành phần tương tác (modal, accordion) dễ tiếp cận với ARIA và focus management.

---

6. Ưu điểm và nhược điểm

Ưu điểm

- **Tiếp cận rộng rãi**: Cho phép mọi người, bao gồm người khuyết tật, sử dụng web.
- **Tuân thủ pháp luật**: Nhiều quốc gia yêu cầu accessibility (ví dụ: ADA ở Mỹ, EN 301 549 ở EU).
- **Cải thiện SEO**: Semantic HTML và văn bản thay thế hỗ trợ công cụ tìm kiếm.
- **Tăng trải nghiệm người dùng**: Giao diện dễ dùng hơn cho tất cả (ví dụ: chữ lớn, điều hướng bàn phím).

Nhược điểm

- **Phức tạp hơn**: Yêu cầu thêm thời gian để triển khai ARIA, focus management, và kiểm tra tương phản.
- **Khúc học tập**: Cần hiểu WCAG, ARIA, và công nghệ hỗ trợ.
- **Kiểm tra phức tạp**: Phải dùng công cụ như Lighthouse, WAVE, hoặc trình đọc màn hình để kiểm tra.

---

7. Kỹ thuật quan trọng

HTML

- **Sử dụng semantic HTML**: `<header>`, `<nav>`, `<main>`, `<footer>` thay vì `<div>` chung chung.
- **Văn bản thay thế**: Luôn cung cấp `alt` cho `<img>`, `aria-label` cho các phần tử không có văn bản rõ ràng.
- **Điều hướng bàn phím**: Đảm bảo mọi tương tác (button, link, form) có thể dùng phím Tab và Enter.
- **ARIA**: Sử dụng `role`, `aria-label`, `aria-hidden`, `aria-live` khi cần bổ sung ngữ nghĩa.

CSS

- **Độ tương phản**: Kiểm tra tỷ lệ tương phản với công cụ như WebAIM Contrast Checker (tối thiểu 4.5:1).
- **Đơn vị tương đối**: Sử dụng `rem`, `em` để hỗ trợ phóng to/thu nhỏ.
- **Focus styles**: Không xóa `outline` mặc định, thay bằng viền tùy chỉnh rõ ràng.
- **Ẩn nội dung**: Sử dụng `.sr-only` để ẩn nội dung chỉ dành cho trình đọc màn hình.
- **Không phụ thuộc màu**: Kết hợp biểu tượng, văn bản, hoặc viền với màu sắc để truyền tải thông tin.

---

8. Bài tập thực hành

1. **Form dễ tiếp cận**:
   - Tạo một form với 2 trường (email, password) và nút submit. Sử dụng `<label>`, `aria-required`, và văn bản lỗi rõ ràng.
   - Thêm CSS với focus styles và độ tương phản cao (4.5:1).
2. **Menu điều hướng**:
   - Tạo một menu ngang với 4 mục, sử dụng `<nav>` và `aria-current`.
   - Đảm bảo điều hướng bàn phím hoạt động và có focus styles rõ ràng.
3. **Gallery ảnh**:
   - Tạo một gallery với 4 hình ảnh, mỗi hình có `alt` mô tả rõ ràng.
   - Sử dụng Flexbox để sắp xếp responsive, đảm bảo độ tương phản cao.
4. **Modal dialog**:
   - Tạo một modal với nút mở/đóng, sử dụng `role="dialog"`, `aria-labelledby`, và `autofocus`.
   - Thêm focus management để giữ focus trong modal khi mở.

---

9. Tài nguyên bổ sung

- **WCAG**: Hướng dẫn chính thức (w3.org/WAI/standards-guidelines/wcag).
- **MDN Web Docs**: Accessibility trong HTML và CSS (developer.mozilla.org/en-US/docs/Web/Accessibility).
- **WebAIM**: Công cụ kiểm tra tương phản (webaim.org/resources/contrastchecker).
- **WAVE**: Công cụ đánh giá accessibility (wave.webaim.org).
- **Lighthouse**: Kiểm tra accessibility trong Chrome DevTools.
- **A11y Project**: Tài nguyên và checklist (a11yproject.com).
- **CodePen**: Thực hành và chia sẻ mã (codepen.io).

---

10. Lưu ý

- **Kiểm tra thực tế**: Dùng trình đọc màn hình (NVDA, VoiceOver) và điều hướng bàn phím để kiểm tra.
- **Tương thích trình duyệt**: Các kỹ thuật HTML/CSS accessibility được hỗ trợ tốt trên Chrome, Firefox, Edge, Safari (2025). Kiểm tra ARIA trên CanIUse.com nếu cần.
- **Không lạm dụng ARIA**: Chỉ dùng ARIA khi semantic HTML không đủ, vì ARIA sai có thể gây lỗi cho trình đọc màn hình.
- **Kết hợp với JavaScript**: Với các thành phần động (modal, accordion), cần JavaScript để quản lý focus và trạng thái ARIA.
- **Kiểm tra độ tương phản**: Sử dụng công cụ như WebAIM hoặc Lighthouse để đảm bảo tỷ lệ tương phản đạt chuẩn.