13. HTML 접근성과 웹 표준

13.1 웹접근성 개념 (WCAG)

— 모두를 위한 웹을 만드는 기본 원칙

웹접근성(Web Accessibility)이란 **장애가 있는 사용자**를 포함하여
모든 사용자가 동등하게 웹 콘텐츠에 접근하고 사용할 수 있도록 하는 것을 말한다.
이는 단순히 기술적인 요구사항이 아니라 **포용성과 평등권의 실현**이며,
국제 표준인 **WCAG(Web Content Accessibility Guidelines)** 가 이를 위한 핵심 지침을 제공한다.

✅ 웹접근성(Web Accessibility) 정의

> "Web accessibility means that websites, tools, and technologies are designed and developed so that people with disabilities can use them."
> — *W3C (World Wide Web Consortium)*

즉, 시각, 청각, 지체, 인지 등의 다양한 장애 유형을 가진 사용자도
웹 콘텐츠를 **지각하고(perceive), 운영하며(operate), 이해하고(understand), 접근할 수 있어야 함(access).**

✅ WCAG란?

WCAG (Web Content Accessibility Guidelines) 는
웹 접근성의 국제 표준 지침으로, W3C의 *WAI (Web Accessibility Initiative)*가 개발함.

- 최신 버전: **WCAG 2.1** (→ 2.2도 2023년 말 채택됨)
- 기술 중립적: HTML, CSS, JS 등 어떤 기술에도 적용 가능
- **4가지 원칙 (POUR)**에 따라 구성

✅ WCAG의 4대 원칙 (POUR)

원칙 (POUR)	설명
Perceivable	사용자에게 **정보가 인지 가능**해야 함 (예: 텍스트 대체, 명확한 색 대비 등)
Operable	사용자 인터페이스는 **모든 사용자 조작 가능**해야 함 (예: 키보드 사용 가능성)
Understandable	콘텐츠와 UI가 **이해 가능**해야 함 (예: 명확한 라벨, 예측 가능한 행동)
Robust	**다양한 기술 환경**에서도 콘텐츠가 견고하게 작동해야 함 (예: 스크린 리더 호환 등)

✅ WCAG 적합 수준

WCAG는 3단계의 **적합도 수준(Levels of Conformance)**을 정의함:

수준	설명	예시
A	필수적인 접근성 제공 (최소 요건)	키보드 접근 가능
AA	대부분의 사용자에게 충분한 접근성 제공 (권장)	색상 대비, 명확한 포커스
AAA	고급 수준 접근성 (선택 사항)	수화 영상 제공, 고급 설명

💡 많은 국가 및 공공기관은 **WCAG 2.1 Level AA**를 법적 기준으로 채택

✅ 웹접근성이 중요한 이유

구분	내용
▶️ 윤리적 이유	디지털 포용, 차별 없는 서비스
▶️ 법적 준수	미국 ADA, 한국 웹접근성 품질 인증 등
▶️ 사용자 확장	고령자, 저시력자 등 포함한 더 넓은 사용자
▶️ SEO 향상	명확한 구조와 텍스트가 검색엔진에 유리
▶️ 유지보수 향상	의미 있는 마크업 구조는 재사용성과 테스트 용이성 증가

✅ 실전 예시

항목	잘못된 사례	개선된 사례
이미지	`<img src="cat.jpg">`	`<img src="cat.jpg" alt="귀여운 고양이">`
색상 대비	회색 글자 + 흰 배경	검정 글자 + 흰 배경 (명도 대비 확보)
폼 요소	`<input>`	`<label for="name">이름</label><input id="name">`
버튼 텍스트	`<button>OK</button>`	`<button>신청 완료</button>`

✅ 접근성 검사 도구

도구	설명
Lighthouse	Chrome DevTools 내장 검사 도구
axe DevTools	브라우저 확장 프로그램 기반 검사
WAVE	WebAIM에서 제공하는 웹 접근성 평가 도구

도구	설명
NVDA, JAWS	스크린 리더 테스트 툴 (실사용자 시뮬레이션)

✅ 관련 법과 정책 (한국 기준)

법령	내용
장애인차별금지법 (2008)	웹 콘텐츠 접근성 보장 의무화
국가정보화 기본법 시행령	공공기관 웹사이트는 K-WCAG 준수
웹접근성 품질마크	심사 기준은 WCAG 2.1 AA 수준 기반

✅ 한 줄 요약

> 웹접근성은 **모두를 위한 웹**을 실현하기 위한 필수 요소이며,
> **WCAG는 그 원칙(P.O.U.R.)과 세부기준을 제공하는 국제 표준**이다.

13.2 ARIA 속성 개요

ARIA (Accessible Rich Internet Applications)는
웹 애플리케이션에서 **스크린 리더나 기타 보조 기술을 통해 의미를 부여하고 접근성을 향상시키는 역할**을 한다.
HTML만으로는 전달되지 않는 의미를 보완해, **장애가 있는 사용자도 완전한 사용자 경험**을 누릴 수 있도록 설계되었다.

✅ 왜 ARIA가 필요한가?

현대 웹은 단순 문서 기반이 아닌 동적 애플리케이션 중심이다.
하지만 `<div>`, `<span>` 같은 비시맨틱 태그는 **스크린 리더에 의미를 전달하지 못함**.
ARIA는 이러한 상황에서 **의미를 "명시적으로" 알려주는 도구**다.

📌 예시:

```
<div role="button" tabindex="0" aria-pressed="false">Play</div>
```

이 `<div>`는 **보조 기술에게 "이건 버튼이며, 현재 눌리지 않았어"**라고 알려준다.

✅ ARIA의 세 가지 핵심 속성군

구분	설명
`role`	요소의 의미/기능을 정의 (예: 버튼, 슬라이더)
`aria-*` 속성들	상태/속성을 설명 (예: `aria-checked`, `aria-expanded`)
`tabindex`	키보드 포커스 가능 여부 및 순서 지정 (HTML 속성이지만 자주 함께 사용됨)

✅ 1. `role` 속성

스크린 리더 등 보조 기술에 **요소의 의미와 사용 방법**을 알려준다.

역할	설명
`button`	클릭 가능한 버튼 역할
`checkbox`	체크 가능한 박스 역할
`dialog`	모달 창 역할
`navigation`	내비게이션 영역
`tab`	탭 패널 내의 개별 탭
`tabpanel`	탭과 연결된 콘텐츠 영역
`alert`	경고 메시지, 즉시 읽힘

📌 예시:

```
<div role="alert">이메일 주소가 잘못되었습니다.</div>
```

✅ 2. `aria-*` 속성

상태 및 속성을 보조 기술에게 전달한다.
많은 종류가 있으며, 일반적으로 `role`과 함께 사용한다.

📌 상태 관련 주요 속성

속성	설명
`aria-checked`	체크 여부 (`true`, `false`, `mixed`)
`aria-expanded`	확장 여부 (예: 드롭다운, 아코디언)
`aria-hidden`	화면에서 숨기고 스크린 리더도 무시
`aria-disabled`	비활성화 여부
`aria-pressed`	토글 버튼의 눌림 상태

📌 예시:

```
<button aria-pressed="false">Play</button>
```

✅ 3. `aria-label`, `aria-labelledby`, `aria-describedby`

속성	설명
`aria-label`	요소에 직접 라벨(이름) 부여
`aria-labelledby`	다른 요소의 ID로부터 라벨 연결
`aria-describedby`	보조 설명 연결 (툴팁, 설명 등)

📌 예시: 라벨 없는 아이콘 버튼

```
<button aria-label="검색">
  <img src="search-icon.png" alt="">
</button>
```

📌 스크린 리더는 `"검색"` 이라고 읽음

✅ 사용 시 주의사항

원칙	설명
✅ **시맨틱 태그가 우선**	가능한 경우 `<button>`, `<nav>` 등 기본 HTML 사용
✅ **의미 전달이 부족할 때만 사용**	ARIA는 "보완" 용도이며 "대체"가 아님
❌ **ARIA 속성이 있다고 동작하는 건 아님**	보조 기술에게만 정보를 제공 (기능은 JS로 처리해야 함)

✅ 접근성 예시 (비추천 vs 개선)

예시	비추천 코드	개선된 코드
버튼 역할	`<div>Play</div>`	`<div role="button" tabindex="0" aria-pressed="false">Play</div>`
알림 메시지	`<span>Error</span>`	`<div role="alert">Error</div>`
토글 패널	`<div>▼ 메뉴</div>`	`<div aria-expanded="false">▼ 메뉴</div>`

✅ ARIA를 배우기 좋은 출처

- MDN ARIA 문서
- WAI-ARIA 공식 문서
- a11y 프로젝트
- Deque University

✅ 한 줄 요약

> ARIA 속성은 **스크린 리더 등 보조 기술에 HTML 요소의 역할과 상태를 명확히 전달하는 표준**이며,
> **시맨틱 태그를 보완하여 진정한 웹접근성을 구현하는 핵심 수단**이다.

13.3 키보드 네비게이션 구조

— 키보드만으로 웹사이트를 탐색할 수 있도록 만드는 설계 원칙

키보드 네비게이션(Keyboard Navigation)은
마우스를 사용할 수 없는 사용자(지체장애인, 시각장애인 등)가
웹 페이지를 **탐색하고 조작할 수 있도록 설계하는 방식**이다.

웹 접근성의 핵심은 **모든 요소에 키보드로 접근 가능해야 한다는 것**이며,
실제 웹 표준에서도 이를 강하게 요구한다.

✅ 왜 키보드 네비게이션이 중요한가?

대상 사용자	이유
시각장애 사용자 (스크린리더 사용)	마우스를 사용할 수 없음
손 떨림, 지체장애 등으로 마우스 제약 있음	키보드만 사용
파워 유저, 보안 환경의 사용자	키보드가 더 빠르고 안정적

✅ 기본 키보드 조작 키

키	동작 설명
`Tab`	다음 포커스 가능한 요소로 이동
`Shift + Tab`	이전 요소로 이동
`Enter`	버튼, 링크 등 활성화
`Space`	버튼/체크박스 등 토글
`Arrow keys`	리스트, 메뉴, 슬라이더 등 조작
`Esc`	닫기, 종료
`Home/End`	리스트 처음/끝으로 이동

✅ 포커스 가능한 기본 요소 (Tab으로 접근 가능)

HTML의 기본 태그 중 아래 요소들은 기본적으로 **Tab 키로 접근** 가능하다:

- `<a href="...">`
- `<button>`
- `<input>`, `<select>`, `<textarea>`
- `<summary>` (details 토글용)
- `<iframe>`
- `<object>`

✅ 포커스 순서 제어: `tabindex`

속성값	의미
없음	기본 순서대로 탐색
`tabindex="0"`	**기본 포커스 순서에 포함**
`tabindex="-1"`	**Tab으로는 접근 불가**, JS에서만 포커스 이동 가능
`tabindex="1"` ~	수동으로 순서 지정 (⚠️ 비권장)

📌 예시:

```
<div tabindex="0">커스텀 포커스 가능한 박스</div>
<div tabindex="-1" id="skipTarget">건너뛰기 대상</div>
```

✅ 키보드 네비게이션 UI 구성 전략

🔹 1. 스킵 네비게이션 제공

화면 맨 위에서 주요 콘텐츠로 건너뛰게 함

```
<a href="#main" class="skip-link">본문 바로가기</a>
<main id="main">여기에 콘텐츠 시작</main>
```

📌 키보드로 Tab 시, “본문 바로가기”가 먼저 나타나야 함

🔹 2. 커스텀 UI 컨트롤은 `tabindex`, `keydown` 처리 필요

예: `<div role="button">` → 키보드 접근 보장

```
<div role="button" tabindex="0" onkeydown="if(event.key === 'Enter') alert('클릭됨')">
  커스텀 버튼
</div>
```

🔹 3. 폼, 모달, 탭 등은 포커스 트랩(Focus Trap) 필요

모달 내에 Tab 키가 벗어나지 않도록 제한

```
// 기본 아이디어
modal.querySelectorAll('[tabindex], input, button, select').forEach(...);
```

✅ 실전 고려 사항

체크리스트 항목	설명
모든 주요 요소에 `Tab`으로 접근 가능해야 함	숨겨진 요소는 제외 가능 (`tabindex="-1"`)
포커스 순서는 논리적이고 자연스러워야 함	시각적 구조와 일치
시각적으로 포커스가 표시되어야 함	`:focus` 스타일 반드시 필요
모달, 드롭다운 등은 열린 후 첫 포커스로 이동	`element.focus()` 사용
포커스가 벗어나지 않도록 제어 (트랩)	모달 내에서만 Tab 순환
접근성 테스트 도구 사용	`axe DevTools`, `NVDA`, `VoiceOver` 등

✅ 포커스 시각화 예제

```
:focus {
  outline: 2px dashed royalblue;
  outline-offset: 4px;
}
```

📌 포커스 상태가 명확하지 않으면 키보드 사용자는 **현재 위치를 알 수 없음**

✅ 접근성 위반 사례 (❌)

문제	설명
`<div onclick="...">`	`tabindex` 없으면 키보드 접근 불가
`display: none` / `visibility: hidden`	스크린 리더도 접근 불가
모달 열었는데 포커스가 외부에 있음	초점이 바깥으로 이동하면 접근성 심각 저하

✅ 한 줄 요약

> 키보드 네비게이션 구조는 **모든 인터랙티브 요소가 Tab으로 접근 가능**하고,
> **자연스러운 순서와 명확한 포커스 표시를 유지하는 것**이 핵심이다.

13.4 시각장애인을 위한 구조 설계

— 스크린 리더 사용자도 완전히 이해할 수 있는 HTML 구조 만들기

시각장애인을 위한 웹 구조 설계는 단순히 글자를 크게 하거나 색상을 조정하는 문제가 아니다.
이들은 주로 **스크린 리더(Screen Reader)** 를 통해 **소리로 웹페이지를 탐색**한다.
따라서 **명확한 시맨틱 구조, 라벨링, 포커스 흐름, 보조 텍스트 제공**이 핵심이다.

✅ 시각장애인을 위한 접근성 설계 목표

목표	설명
구조적 의미 전달	시맨틱 태그를 통해 구조를 "설명"해야 함
비시각 콘텐츠 대체 제공	이미지, 아이콘, 영상에 대체 텍스트 필수
키보드 탐색 보장	마우스 사용 없이도 모든 기능 가능해야 함
흐름 제어와 맥락 유지	페이지 내 내비게이션, 포커스 이동 순서 유지
인터랙션 의미의 명시적 표현	버튼, 알림 등 동작 요소에 의미와 상태를 알려줘야 함

✅ 1. 시맨틱 태그 구조 활용

스크린 리더는 태그에 따라 **"이건 제목입니다", "이건 섹션입니다"**라고 말해줌.

📌 예시: 기본 문서 뼈대

```
<header>
  <h1>서울시청 홈페이지</h1>
  <nav>
    <ul>
      <li><a href="/intro">기관 소개</a></li>
      <li><a href="/news">소식</a></li>
    </ul>
  </nav>
</header>

<main>
  <article>
    <h2>시각장애인을 위한 웹 개선</h2>
    <p>2025년까지 모든 공공 웹사이트를...</p>
  </article>
</main>
```

```
<footer>
  <p>&copy; 서울시청</p>
</footer>
```

태그	의미	스크린 리더가 읽는 방식
`<header>`	머리말	“헤더”
`<nav>`	내비게이션 영역	“내비게이션”
`<article>`	독립적 콘텐츠	“기사”
`<main>`	주요 콘텐츠 영역	“메인 콘텐츠”

✅ 2. 이미지와 아이콘: `alt`와 `aria-hidden`

경우	코드	설명
의미 있는 이미지	`<img src="graph.png" alt="2024년 매출 그래프">`	스크린 리더가 읽음
장식용 이미지	`<img src="dot.png" alt="" aria-hidden="true">`	시각장애인에게 숨김
아이콘 텍스트 대체	`<span class="icon" aria-hidden="true">🔍</span><span class="sr-only">검색</span>`	스크린 리더에게 “검색”만 노출

> 💡 `.sr-only`는 CSS로 화면에는 보이지 않지만 스크린 리더에만 읽히게 함

✅ 3. 폼 요소는 `label`, `aria-label` 필수

```
<label for="email">이메일 주소</label>
<input id="email" type="email" required>

<!-- 또는 라벨이 시각적으로 없을 경우 -->
<input type="search" aria-label="사이트 검색">
```

목적	설명
`label`	사용자에게 항목 설명을 연결
`aria-label`	스크린 리더 전용 설명
`aria-describedby`	보조 설명 연결 (힌트, 오류 등)

✅ 4. ARIA 속성을 통한 명확한 상태 전달

📌 드롭다운 예시

```
<button aria-expanded="false" aria-controls="menu">메뉴</button>
<ul id="menu" hidden>
  <li><a href="/profile">프로필</a></li>
</ul>
```

스크린 리더는 버튼을 "메뉴, 펼쳐짐 여부: false"로 읽음.
사용자가 행동한 결과에 따라 `aria-expanded="true"`로 업데이트 필요.

✅ 5. 알림, 변경 사항은 `aria-live`로 읽힘 알림

```
<div aria-live="polite" id="notify">
  <!-- 동적으로 메시지 들어올 예정 -->
</div>
```

스크린 리더는 이 영역에 텍스트가 생기면 자동으로 읽어줌.

✅ 6. 포커스 이동 관리

모달창, 스킵링크, 팝업 등에서는 초점을 명확히 이동시켜야 함:

```
document.querySelector('#modal').focus();
```

또는 HTML에서:

```
<div id="modal" tabindex="-1" aria-modal="true" role="dialog">
  <h2>신청 완료</h2>
  ...
</div>
```

✅ 실전 접근성 예시: 접근 가능한 버튼

```
<div role="button" tabindex="0" aria-pressed="false" onclick="...">
  <span aria-hidden="true">▶</span> <span class="sr-only">재생</span>
</div>
```

역할	설명
`role="button"`	스크린 리더에게 "버튼"이라고 인식시킴
`tabindex="0"`	키보드 포커스 허용
`aria-pressed`	상태 전달

역할	설명
`aria-hidden`, `.sr-only`	시각적 요소와 보조 텍스트 분리

✅ 툴 추천: 시각장애인 관점 테스트

도구	설명
NVDA (Windows)	무료 스크린 리더, 실제 사용자 경험 테스트
VoiceOver (macOS/iOS)	Apple 기기 내장 스크린 리더
axe DevTools, **WAVE**	자동 진단 + 키보드/스크린리더 조건 시뮬레이션 가능

✅ 요약 정리

> 시각장애인을 위한 구조 설계는 **시맨틱 HTML, 대체 텍스트, 명확한 상태 표시, 키보드 흐름 유지**의 총합이다.
> 접근성이 뛰어난 사이트는 **스크린 리더가 읽기 쉬운 HTML 구조**로 구성되어야 하며,
> **ARIA 속성과 포커스 제어**는 그 핵심 기술이다.

13.5 스크린리더 테스트 방법

— 스크린 리더 사용자도 완전히 이해할 수 있는 HTML 구조 만들기

시각장애인을 위한 웹 구조 설계는 단순히 글자를 크게 하거나 색상을 조정하는 문제가 아니다.
이들은 주로 **스크린 리더(Screen Reader)** 를 통해 **소리로 웹페이지를 탐색**한다.
따라서 **명확한 시맨틱 구조, 라벨링, 포커스 흐름, 보조 텍스트 제공**이 핵심이다.

✅ 시각장애인을 위한 접근성 설계 목표

목표	설명
구조적 의미 전달	시맨틱 태그를 통해 구조를 "설명"해야 함
비시각 콘텐츠 대체 제공	이미지, 아이콘, 영상에 대체 텍스트 필수
키보드 탐색 보장	마우스 사용 없이도 모든 기능 가능해야 함
흐름 제어와 맥락 유지	페이지 내 내비게이션, 포커스 이동 순서 유지
인터랙션 의미의 명시적 표현	버튼, 알림 등 동작 요소에 의미와 상태를 알려줘야 함

✅ 1. 시맨틱 태그 구조 활용

스크린 리더는 태그에 따라 **"이건 제목입니다", "이건 섹션입니다"**라고 말해줌.

📌 예시: 기본 문서 뼈대

```
<header>
  <h1>서울시청 홈페이지</h1>
  <nav>
    <ul>
      <li><a href="/intro">기관 소개</a></li>
      <li><a href="/news">소식</a></li>
    </ul>
  </nav>
</header>

<main>
  <article>
    <h2>시각장애인을 위한 웹 개선</h2>
    <p>2025년까지 모든 공공 웹사이트를...</p>
  </article>
</main>

<footer>
  <p>&copy; 서울시청</p>
</footer>
```

태그	의미	스크린 리더가 읽는 방식
`<header>`	머리말	"헤더"
`<nav>`	내비게이션 영역	"내비게이션"
`<article>`	독립적 콘텐츠	"기사"
`<main>`	주요 콘텐츠 영역	"메인 콘텐츠"

✅ 2. 이미지와 아이콘: `alt`와 `aria-hidden`

경우	코드	설명
의미 있는 이미지	`<img src="graph.png" alt="2024년 매출 그래프">`	스크린 리더가 읽음
장식용 이미지	`<img src="dot.png" alt="" aria-hidden="true">`	시각장애인에게 숨김
아이콘 텍스트 대체	`<span class="icon" aria-hidden="true">🔍</span><span class="sr-only">검색</span>`	스크린 리더에게 "검색"만 노출

> 💡 `.sr-only`는 CSS로 화면에는 보이지 않지만 스크린 리더에만 읽히게 함

✅ 3. 폼 요소는 `label`, `aria-label` 필수

```
<label for="email">이메일 주소</label>
<input id="email" type="email" required>

<!-- 또는 라벨이 시각적으로 없을 경우 -->
<input type="search" aria-label="사이트 검색">
```

목적	설명
`label`	사용자에게 항목 설명을 연결
`aria-label`	스크린 리더 전용 설명
`aria-describedby`	보조 설명 연결 (힌트, 오류 등)

✅ 4. ARIA 속성을 통한 명확한 상태 전달

📌 드롭다운 예시

```
<button aria-expanded="false" aria-controls="menu">메뉴</button>
<ul id="menu" hidden>
  <li><a href="/profile">프로필</a></li>
</ul>
```

스크린 리더는 버튼을 "메뉴, 펼쳐짐 여부: false"로 읽음.
사용자가 행동한 결과에 따라 `aria-expanded="true"`로 업데이트 필요.

✅ 5. 알림, 변경 사항은 `aria-live`로 읽힘 알림

```
<div aria-live="polite" id="notify">
  <!-- 동적으로 메시지 들어올 예정 -->
</div>
```

스크린 리더는 이 영역에 텍스트가 생기면 자동으로 읽어줌.

✅ 6. 포커스 이동 관리

모달창, 스킵링크, 팝업 등에서는 초점을 명확히 이동시켜야 함:

```
document.querySelector('#modal').focus();
```

또는 HTML에서:

```html
<div id="modal" tabindex="-1" aria-modal="true" role="dialog">
  <h2>신청 완료</h2>
  ...
</div>
```

✅ 실전 접근성 예시: 접근 가능한 버튼

```html
<div role="button" tabindex="0" aria-pressed="false" onclick="...">
  <span aria-hidden="true">▶</span> <span class="sr-only">재생</span>
</div>
```

역할	설명
`role="button"`	스크린 리더에게 “버튼”이라고 인식시킴
`tabindex="0"`	키보드 포커스 허용
`aria-pressed`	상태 전달
`aria-hidden`, `.sr-only`	시각적 요소와 보조 텍스트 분리

✅ 툴 추천: 시각장애인 관점 테스트

도구	설명
NVDA (Windows)	무료 스크린 리더, 실제 사용자 경험 테스트
VoiceOver (macOS/iOS)	Apple 기기 내장 스크린 리더
axe DevTools, WAVE	자동 진단 + 키보드/스크린리더 조건 시뮬레이션 가능

✅ 요약 정리

> 시각장애인을 위한 구조 설계는 **시맨틱 HTML, 대체 텍스트, 명확한 상태 표시, 키보드 흐름 유지**의 총합이다.
> 접근성이 뛰어난 사이트는 **스크린 리더가 읽기 쉬운 HTML 구조**로 구성되어야 하며,
> **ARIA 속성과 포커스 제어**는 그 핵심 기술이다.