

## 웹 접근성과 웹 표준

### 1. 웹 접근성(Web Accessibility)

- **정의** : 장애 유무, 연령, 환경과 관계없이 모든 사용자가 웹 콘텐츠에 동등하게 접근할 수 있도록 보장하는 것.
- **예시** : 시각 장애인을 위한 대체 텍스트(alt), 키보드만으로도 웹사이트 탐색 가능, 색맹을 고려한 색상 대비 설계.
- **목표** : 정보 소외 계층을 줄이고, 더 많은 사용자가 웹을 활용할 수 있게 함.

### 2. 웹 표준(Web Standards)

- **정의** : **W3C(World Wide Web Consortium)**와 같은 국제 표준 기구가 제정한 웹 기술 규칙 및 가이드라인.
- **주요 내용** :
  - HTML, CSS, JavaScript의 문법과 구조 준수
  - 의미론적 마크업(Semantic Markup) 사용
  - 크로스 브라우저 호환성 확보

### 3. 웹 접근성과 웹 표준의 관계

- 웹 표준을 준수해야 웹 접근성을 확보할 수 있음.
- 웹 표준은 "기술적인 규칙", 웹 접근성은 "사람 중심의 사용성"에 초점.
- 두 가지가 함께 지켜져야 사용자 친화적이고 지속 가능한 웹 서비스 제공 가능.

### 4. 왜 필요한가?

- **사용자 측면** : 누구나 편리하게 웹을 이용 가능 → **정보 평등 보장.**

- **개발자/기업 측면** : 유지보수 비용 절감, 다양한 기기/브라우저 호환성 확보, 글로벌 서비스 확장성 강화.
- **사회적 측면** : 장애인·고령자 등 정보 소외 계층 해소 → 법적 규제 대응(장애인차별금지법, KWACG)

#### **KWCAG(Korea Web Content Accessibility Guidelines)**

**한국형 웹 접근성 지침으로** 장애인, 고령자 등 누구나 차별 없이 웹사이트를 이용할 수 있도록 보장하는 **국가 표준 지침**.

국제 지침인 WCAG (W3C의 Web Content Accessibility Guidelines)를 기반으로, 한국의 법·환경에 맞게 보완 및 제정.

**모든 공공기관 웹사이트**는 KWACG 준수 의무가 있음.

민간 기업도 서비스 품질 향상, 법적 리스크 예방 차원에서 적용 권장.

## **5. W3C(World Wide Web Consortium)**

- 설립 : 1994년, 팀 버너스리(Tim Berners-Lee) 가 설립한 **국제 웹 표준화 기구**.
- 역할 : 웹 기술 표준 개발 및 보급 (HTML, CSS, XML, WCAG 등).
- 목표 : 모든 사용자가 접근 가능한 웹 환경 조성, 기술 호환성과 개방성 보장.
- 주요 성과 : HTML5 제정, WCAG(Web Content Accessibility Guidelines) 제시.

**W3C**는 웹 기술의 “헌법”을 만드는 국제 표준 기구이며, 모든 사람이 차별 없이 웹을 이용할 수 있도록 웹의 규칙을 정하는 곳이다.

## **6. 웹 표준 참고 및 학습 사이트**

- <https://www.w3.org/> - **W3C 공식 사이트**
- <https://developer.mozilla.org/ko/> -MDN Web Docs
- <https://www.webstandards.org/> - Web Standards Project (**WaSP**) (역사적 자료)
-

## 📌 Web Standards Project (WaSP)

**WaSP** = 1998~2013년 활동한 웹 표준 운동 단체. 브라우저와 개발자가 표준을 지키도록 압력을 넣어 오늘날의 웹 표준 기반 환경을 만든 역사적 주역.

### 1. 정의

- Web Standards Project (WaSP)는 1998년에 시작된 웹 표준 준수 촉진 운동 단체.
- 웹 개발자와 디자이너들이 모여 만든 비영리 프로젝트.
- 목표: 웹 표준(W3C 권고안)을 지키도록 브라우저 업체와 개발자에게 압력을 주고 교육.

### 2. 등장 배경

- 1990년대 후반 웹은 브라우저 전쟁(Browser Wars) 시기.
  - Netscape, Internet Explorer가 서로 다른 방식으로 HTML/CSS를 해석.
  - 표준이 무시되면서 사이트가 특정 브라우저에서만 동작하는 문제가 심각.
- 이 문제를 해결하기 위해 "웹 표준 준수 운동"으로 WaSP가 결성됨.

### 3. 활동

- 브라우저 제조사 압박: 마이크로소프트, 넷스케이프 등에게 표준 준수를 요구.
- 교육 자료 제공: 웹 개발자와 디자이너들이 올바른 표준을 배울 수 있도록 캠페인, 가이드 제공.
- 테스트 프로젝트: Acid Test 같은 검사 도구를 만들어 브라우저의 표준 호환성 검증.

### 4. 성과

- 주요 브라우저들이 점차 W3C 표준 준수를 강화하게 만든 촉매제.
- 웹 개발 생태계가 크로스 브라우저 호환성을 개선하는 전환점 마련.
- HTML4, CSS2, DOM 등의 보급에 기여.

### 5. 현재

- WaSP는 2013년 공식 활동 종료.
- 하지만 그 정신은 W3C, **WHATWG**, 교육 커뮤니티(MDN 등)로 이어지고 있음.

## WHATWG (Web Hypertext Application Technology Working Group)

WHATWG = HTML Living Standard를 관리하는 국제 표준 그룹. 현재 HTML 표준은 사실상 WHATWG가 주도한다.

### 1. 정의

- WHATWG = *Web Hypertext Application Technology Working Group*
- 2004년에 \*\*애플(Apple), 모질라(Mozilla), 오페라(Opera)\*\*가 주도해 결성된 웹 표준화 단체.
- 목표: 웹을 현대 애플리케이션 플랫폼에 맞게 발전시키는 것.

### 2. 등장 배경

- 당시 W3C는 XHTML 2.0 개발에 집중 → 현실적인 웹 개발 요구(멀티미디어, 동적 웹앱)와 괴리.
- 이에 불만을 가진 브라우저 기업들이 모여 HTML을 계속 발전시키자고 만든 그룹이 WHATWG.

### 3. 주요 역할

- HTML Living Standard(살아있는 표준) 제정
  - 기존 W3C 방식 : "버전별로 발표(HTML4, HTML5...)"
  - WHATWG 방식 : "계속 업데이트되는 표준(HTML Living Standard)"
- 실용적이고 현실적인 웹 기술 발전
  - <video>, <audio>, <canvas> 등 멀티미디어 지원
  - 웹 애플리케이션 API 강화

### 4. W3C와의 관계

- 한동안 W3C와 WHATWG가 HTML5 표준을 따로 개발 → 중복과 혼란 발생.
- 2019년, W3C와 WHATWG가 협력 → HTML 표준은 WHATWG가 담당, W3C는 합의 기반으로 권고안 참고.
  - 즉, 현재 HTML 표준은 사실상 WHATWG가 관리.

### 5. 목표 & 철학

- 실용성 중시: "현실의 웹 개발자와 브라우저 제조사가 필요로 하는 것"
- 끊임없는 개선: "HTML은 버전이 아니라 살아있는 문서(Living Standard)"

### 6. 참고 사이트

- WHATWG 공식 사이트: <https://whatwg.org/>
- HTML Living Standard 문서: <https://html.spec.whatwg.org/>

### W3C vs WHATWG 비교표

구분	W3C (World Wide Web Consortium)	WHATWG (Web Hypertext Application Technology Working Group)
설립 연도	1994년	2004년
설립 주체	팀 버너스리(Tim Berners-Lee)	애플, 모질라, 오페라 (초기 주도)
성격	국제 웹 표준화 기구 (비영리 컨소시엄)	브라우저 제조사 중심의 실무 그룹
주요 목표	모든 사람이 접근 가능한 개방적 웹 표준 개발	현실적이고 현대적인 웹 애플리케이션 지원
표준 방식	버전 단위로 표준 발표 (HTML4, HTML5 등)	Living Standard (항상 최신으로 업데이트)
대표 성과	HTML4, HTML5, CSS, WCAG, XML 등	HTML Living Standard (video, audio, canvas 등 도입)
웹 접근성 기여	WCAG 제정 (웹 접근성 지침)	실용적 HTML 표준 제공, 접근성은 간접 기여
현재 관계 (2019 이후)	WHATWG의 HTML Living Standard를 공식 인정	HTML 표준 관리 주도 (W3C는 합의 기반 협력)
철학	"웹은 모두를 위한 것 (The Web for All)"	"웹은 살아있는 기술 (Living Standard)"

### 정리

- W3C: 국제 표준을 만드는 공식 기구, 접근성과 개방성에 초점.
- WHATWG: HTML을 실제로 관리·진화시키는 그룹, 실용성과 최신성에 초점.
- 현재 HTML 표준은 WHATWG가 주도, W3C는 협력하는 구조.

## 7. 웹 접근성 가이드 및 참고 자료

- <https://www.wah.or.kr/> - 한국형 웹 콘텐츠 접근성 지침(KWCAG)

<https://www.w3.org/WAI/standards-guidelines/wcag/> - W3C WCAG(Web Content Accessibility Guidelines)

## 8. 웹 표준 검증 및 마크업 확인 사이트

- <https://validator.w3.org/> : W3C Markup Validation Service → HTML/CSS 코드 검사
- <https://wave.webaim.org/> : WAVE (Web Accessibility Evaluation Tool) → 접근성 자동 검사
- <https://validator.w3.org/nu/> : Html Checker (WHATWG) → 최신 HTML5 마크업 검사

## 웹 표준에서 개발자가 꼭 지켜야 하는 것들

### 1. 올바른 문법 준수

- HTML, CSS, JavaScript 문법 오류를 최소화.
- 태그는 열고 반드시 닫기 (<p> ... </p>).
- 중첩 구조 올바르게 사용 (<ul><li></li></ul>).
- 속성값은 큰따옴표로 감싸기 (class="title").

### 2. 시맨틱 마크업(Semantic Markup)

- 의미에 맞는 HTML 태그 사용 → 구조적이고 의미 전달이 정확한 문서.
  - 제목 : <h1> ~ <h6>
  - 문단 : <p>
  - 목록 : <ul>, <ol>, <li>
  - 강조 : <strong>, <em>
  - 구획 : <section>, <article>, <nav>, <footer>
- ❌ div나 span만 남발하지 말 것.

### 3. 표준 속성 & 최신 기술 사용

- HTML5, CSS3 등 최신 표준 준수.
- Deprecated(폐지된) 태그/속성 사용 금지.
  - 예: <font>, <center>, <marquee> 등 사용 ❌
- 스타일은 CSS로, 구조는 HTML로, 동작은 JavaScript로 분리.

### 4. 접근성 고려

- 이미지에 alt 속성 반드시 추가.
- 폼 요소에는 label 연결 (<label for="id">).
- 색상만으로 정보 전달하지 말 것 (색약/색맹 고려).
- 키보드만으로도 조작 가능하게 설계(Tab, Enter).

#### “키보드만으로도 모든 기능을 사용할 수 있다”의 의미

##### 1. 정의

- 웹사이트의 모든 메뉴, 버튼, 입력 폼 등이 마우스를 쓰지 않고 키보드만으로도 조작 가능해야 한다는 뜻.
- 이는 웹 접근성의 핵심 원칙(운용의 용이성, Operable) 중 하나.

##### 2. 왜 필요한가?

- 시각장애인, 손떨림(파킨슨병), 또는 마우스 사용이 어려운 사용자는 화면 낭독기(Screen Reader) + 키보드만 사용.
- 임시적으로 마우스를 못 쓰는 상황(예: 노트북 터치패드 고장, 한 손만 자유로운 상황)에서도 접근 가능해야 함.

##### 3. 구체적인 예시

###### ✅ 좋은 예시 (키보드 접근 가능)

- Tab 키: 버튼/링크/폼 요소로 이동 가능
- Enter 키: 버튼 클릭 동작 실행
- Spacebar: 체크박스 선택/해제
- Arrow Keys: 드롭다운 메뉴 이동

- Skip Navigation 링크 제공 : 바로 본문으로 이동

#### ✗ 나쁜 예시 (마우스 전용 UI)

- 마우스 오버(hover)만으로 열리는 메뉴 (Tab 키로 접근 불가)
- JavaScript로 만든 커스텀 버튼인데, 키보드 이벤트 처리가 없음
- <div>만으로 만든 버튼 (role, tabindex 없음)

#### 4. 개발자가 지켜야 할 규칙

1. 모든 인터랙션에 키보드 이벤트 추가
  - onclick만 쓰지 말고, onkeypress 또는 onkeydown도 처리
2. Tab 순서(tabindex) 논리적 유지
  - 키보드 포커스가 시각적으로도 잘 보이게 :focus 스타일 제공
3. 폼 요소, 버튼은 시맨틱 태그 사용
  - <button>, <input> 태그는 기본적으로 키보드 접근 가능
  - <div>나 <span>을 버튼처럼 만들면 접근성 저하

#### 정리

"키보드만으로도 모든 기능을 사용할 수 있다"

= 마우스를 못 쓰는 사람도 Tab + Enter/Space만으로 사이트를 전부 탐색하고 조작할 수 있어야 한다는 뜻.

### 5. 크로스 브라우저 호환성 확보

- W3C 표준 문법을 기반으로 구현.
- 브라우저별 개발자도구에서 테스트 필수.
- 특정 브라우저 전용 코드(벤더 프리픽스 등)만 의존하지 말기

#### 벤더 프리픽스(Vendor Prefix)란?

- CSS 기능이 아직 표준으로 확정되지 않았거나 실험 단계일 때, 브라우저마다 독자적으로 제공하는 접두사(prefix).
- 브라우저별 접두사 예시:
  - -webkit- → 크롬(Chrome), 사파리(Safari)
  - -moz- → 파이어폭스(Firefox)
  - -ms- → 인터넷 익스플로러/엣지(구버전)



- -o- → 오페라(구버전)

```
/* 벤더 프리픽스 사용 */
.box {
  -webkit-border-radius: 10px; /* 크롬, 사파리 전용 */
  -moz-border-radius: 10px;    /* 파이어폭스 전용 */
  border-radius: 10px;         /* 표준 속성 */
}
```

예전에는 border-radius가 표준이 아니어서 브라우저별 프리픽스를 붙여야 동작했음.  
현재는 border-radius가 표준이 되었기 때문에 프리픽스 없이도 모든 브라우저에서 동작함.

#### 왜 의존하면 안 되나?

- 특정 브라우저에서만 동작 → 크로스 브라우저 호환성 깨짐
- 유지보수 어려움 → 표준 속성이 나오면 코드 중복 발생
- 사용자 경험 저하 → 다른 브라우저 사용자들이 정상 기능을 못 씀

#### 권장 방식

- 가능하면 표준 속성을 우선 사용.
- 필요하다면, 프리픽스 + 표준 속성 함께 작성 → 모든 브라우저 대응.
- CSS 자동화 도구(PostCSS, Autoprefixer)를 활용 → 필요한 경우 자동으로 프리픽스 추가.

## 6. 문서 구조 정의 & 선언

- 반드시 DOCTYPE 선언하기 (<!DOCTYPE html>).
- HTML 문서의 기본 구조를 정확히 지킬 것.

```

<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8">
    <title>웹 표준 예시</title>
  </head>
  <body>
    <h1>안녕하세요</h1>
  </body>
</html>

```

## 7. 유효성 검사 & 자동화 도구 활용

- 개발 중 정기적으로 W3C Validator로 검사.
- ESLint, Stylelint 같은 자동화 도구 활용.

### ESLint & Stylelint

#### 1. ESLint

- 정의: JavaScript(또는 TypeScript) 코드의 문법 오류, 코드 스타일, 잠재적 버그를 자동으로 검사해주는 도구.
- 역할:
  - 세미콜론 누락, 따옴표 규칙(작은따옴표 vs 큰따옴표), 들여쓰기, 사용하지 않는 변수 등 잡아줌.
  - 팀 규칙(코딩 컨벤션)에 맞춰 일관된 코드 작성 가능.
- 예시:

```

// ESLint 경고 예시
let name = "KOSTA"
console.log(name) // 세미콜론 누락 시 오류 발생

```

장점: 버그 예방, 협업 시 코드 품질 유지, 자동 수정(eslint --fix) 가능.

#### 2. Stylelint

- 정의: CSS코드의 문법 오류와 스타일 규칙을 자동 검사하는 도구.
- 역할:

- 잘못된 CSS 속성 사용, 중복 정의, 잘못된 단위(px, %, em 등) 체크.
- 팀의 CSS 컨벤션에 맞는 코드 유지.

- 예시:

```
/* Stylelint 경고 예시 */
body {
  color: #333;
  font-weight: bold;
  font-weigth: normal; /* 철자 오류 → 경고 */
}
```

### 3. 두 도구의 공통점

- 자동화된 코드 품질 검사기.
- 개발자가 실수하기 쉬운 부분을 빠르게 잡아줌.
- VS Code 같은 에디터에 플러그인으로 설치하면 저장할 때 자동 검사 가능.
- CI/CD(깃허브 액션, 젠킨스 등) 파이프라인에도 붙여서 팀 전체 코드 품질 유지 가능.

### 정리



- ESLint → JavaScript/TypeScript 검사기
- Stylelint → CSS 검사기


## 정리

👉 “구조는 HTML, 디자인은 CSS, 동작은 JS” 라는 원칙을 반드시 지켜야 웹 표준을 준수할 수 있다.

👉 시맨틱하게 마크업하고, 접근성을 고려하면 웹 접근성까지 자연스럽게 확보된다.

## 웹 표준 체크리스트 (개발자 자기 점검용)

/ 표시하면서 확인해보기.

1. 문서 맨 위에 `<!DOCTYPE html>`을 선언했다.
2. HTML 태그는 올바르게 열고 닫았다. (`<p> ... </p>`)
3. 의미에 맞는 시맨틱 태그를 사용했다. (`<header>`, `<nav>`, `<section>`, `<article>`, `<footer>`)
4. 이미지에는 항상 alt 속성을 넣었다.
5. `<font>`, `<center>`, `<marquee>` 같은 폐지된 태그를 사용하지 않았다.
6. 구조(HTML), 스타일(CSS), 동작(JavaScript)을 분리했다.
7. 색상만으로 정보를 구분하지 않았다. (예: 빨간색 글씨만으로 "필수 입력" 표시 )
8. 모든 폼 요소는 label과 연결되어 있다.
9. 키보드만으로도 모든 기능을 사용할 수 있다.
10. 코드 작성 후 W3C Validator로 검증했다.