1. HTML 개요

1.1 HTML이란?

정의

HTML (HyperText Markup Language)은
웹 페이지를 작성하고 구조를 정의하는 데 사용되는 마크업 언어이다.
브라우저가 HTML 문서를 해석해 텍스트, 이미지, 링크, 버튼 등 **화면에 보이는 웹 콘텐츠**를 구성하게 된다.

HTML은 프로그래밍 언어가 아니며, **문서의 내용과 구조를 '표현'** 하기 위한 **정적인 언어**이다.

용어 해설

- **HyperText (하이퍼텍스트)**
  클릭 가능한 링크로 문서 간 이동이 가능한 텍스트를 말한다.
  예: `<a href="page2.html">다음 페이지</a>`
  → HTML의 핵심 기능 중 하나가 바로 이 **문서 간 연결성**이다.
- **Markup Language (마크업 언어)**
  문서의 각 요소에 '의미'나 '역할'을 지정하는 언어.
  HTML에서는 `<h1>`, `<p>`, `<img>` 등 **태그(tag)** 를 통해 구조를 표시한다.

HTML의 역할

기능	설명
문서 구조화	제목, 본문, 사이드바, 푸터 등 콘텐츠의 전체 뼈대를 구성
콘텐츠 표현	텍스트, 이미지, 비디오 등 미디어 요소를 표시
의미 부여 (시맨틱)	각 콘텐츠가 무엇인지 명확히 정의 → 검색엔진, 스크린리더 등에 유리함
링크 연결	다른 웹페이지, 파일, 위치 등으로 이동하는 하이퍼링크 제공
폼 입력	사용자로부터 데이터 입력 받기 → 로그인, 검색 등 기능 수행 가능

HTML 문서 예시

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>나의 첫 HTML 문서</title>
  </head>
  <body>
    <h1>안녕하세요!</h1>
    <p>이것은 HTML로 작성된 문서입니다.</p>
    <a href="https://example.com">링크</a>
  </body>
</html>
```

📌 위 문서는 브라우저에서 "안녕하세요!"라는 제목과 문단, 그리고 클릭 가능한 링크를 표시한다.

HTML이 왜 중요한가?

- 🌍 **웹의 기본 언어**: 모든 웹페이지는 HTML로 시작된다.
- 🧱 **웹의 구조**: CSS는 디자인을, JavaScript는 동작을 담당하지만, **기초 뼈대는 HTML**이 만든다.
- 👷 **개발의 출발점**: 웹개발, 프론트엔드, SEO, UI/UX 등 모든 웹 분야의 **공통 기반**

HTML vs 프로그래밍 언어

구분	HTML	프로그래밍 언어 (예: JavaScript)
실행 가능성	없음 (정적인 문서 구성만 가능)	있음 (명령어 수행, 변수, 조건문 등 가능)
목적	콘텐츠 구조 정의	동작 정의, 로직 수행
변수/반복문	없음	있음

한 줄 요약

> **HTML은 웹 콘텐츠의 '뼈대'를 만드는 언어이다.**
> 사용자가 브라우저에서 보는 거의 모든 시각 요소는 HTML로 시작된다.

1.2 HTML의 역사 및 버전 (HTML 4, XHTML, HTML5)

(HTML 4, XHTML, HTML5 중심으로 정리)

HTML은 단순한 문서 마크업에서 시작해, 지금은 **앱 수준의 웹 플랫폼**을 지원할 정도로 진화했다.
이 항목에서는 **HTML의 역사**, 그리고 주요 버전인 **HTML 4, XHTML, HTML5**의 특징과 차이점을 중심으로 설명한다.

(HTML 4, XHTML, HTML5 중심으로 정리)

HTML은 단순한 문서 마크업에서 시작해, 지금은 **앱 수준의 웹 플랫폼**을 지원할 정도로 진화했다.
이 항목에서는 **HTML의 역사**, 그리고 주요 버전인 **HTML 4, XHTML, HTML5**의 특징과 차이점을 중심으로 설명한다.

🔹 1. HTML의 기원과 초기 버전

버전	연도	설명
HTML 1.0	1991	팀 버너스 리(Tim Berners-Lee)가 개발. 단순한 텍스트, 링크 중심
HTML 2.0	1995	IETF에서 표준 제정. 기본 구조 (`<p>` , `<h1>` , `<a>` , `<img>` 등) 포함
HTML 3.2	1997	W3C 등장, 테이블/스크립트/스타일시트 최초 포함
HTML 4.0	1997	구조 중심 마크업, 폼 기능 확대. CSS 분리 강조
HTML 4.01	1999	HTML 4.0의 소폭 개정판, 오늘날 HTML4의 대표격

🔹 2. HTML 4.01 (1999)

✅ 주요 특징

- **문서 구조 강조**: 시맨틱 구조로 `div` , `span` , `id` , `class` 등 사용 확산
- **CSS 연동**: 디자인과 마크업 분리 강조 (`<style>` 태그, `link` 활용)
- **폼 요소 강화**: `input` , `textarea` , `select` 확장
- **3가지 DTD 버전**:
  - Strict: 의미만 정의, 표현용 태그 제거
  - Transitional: 과도기용, 표현용 태그 허용
  - Frameset: 프레임 페이지 용도

✅ 단점

- **엄격하지 않은 구조 허용** → 브라우저 호환성 문제
- **비표준 마크업 허용** → 유지보수 어려움
- **시맨틱 부족**: `<div>` 남용, 의미 없는 구조

🔹 3. XHTML (2000~2008)

✅ 개요

- **HTML을 XML 형식으로 재작성**한 버전
- **W3C에서 개발**, HTML 4.01 기반을 XML 문법으로 강제

✅ 특징

- **엄격한 문법 적용**
  - 모든 태그는 닫아야 함 (`<br>` → `<br />`)
  - 속성 값은 반드시 따옴표 사용 (`type="text"`)
  - 태그는 반드시 소문자
- **문서 타입: application/xhtml+xml**
- **문법 오류 = 렌더링 실패** → 무조건 정확해야 함
- **브라우저 호환성 이슈**: IE 등 구형 브라우저가 비협조적

✅ 평가

- **장점**: 명확한 문법, XML 기반 처리 가능
- **단점**: 지나치게 엄격해 개발 효율 저하
- **결과**: **대중화 실패**, HTML5로 흐름 전환됨

🔹 4. HTML5 (2014 정식 발표, 현재 기준 최신)

✅ HTML5의 등장 배경

- XHTML의 실패
- 모바일과 멀티미디어의 급격한 성장
- 브라우저 간 통일된 렌더링 방식 필요
- JavaScript의 폭발적 발전과 DOM 활용 증가

✅ HTML5의 핵심 특징

항목	내용
시맨틱 태그 도입	`<header>`, `<footer>`, `<section>`, `<article>` 등
멀티미디어 태그 지원	`<audio>`, `<video>` → 플러그인 없이 재생 가능
폼 기능 강화	`<input type="email">`, `<datalist>`, `<output>` 등
Canvas 및 SVG 통합	2D/3D 그래픽 표현 가능
스토리지 API	`localStorage`, `sessionStorage` 지원
웹 앱 지원 강화	`offline`, `web workers`, `geolocation`, `WebSocket`
브라우저 간 일관성 강조	비표준 마크업 무시, 단순한 파서 채택
Doctype 간소화	`<!DOCTYPE html>` 하나로 통일

✅ HTML5의 장점

- **모바일 친화적**
  → 반응형 웹, 앱 수준의 기능 제공
- **플러그인 제거**
  → Flash, Silverlight 등 필요 없음
- **시맨틱 구조 개선**
  → 접근성, SEO, 유지보수성 향상
- **빠른 브라우저 파싱**
  → 유연한 문법 해석

✅ HTML5 이후

- WHATWG(웹 기술 표준화 그룹)가 HTML "Living Standard"로 지속 업데이트 중
- HTML은 더 이상 버전 넘버를 사용하지 않고 **진화하는 플랫폼**으로 간주됨

🔸 주요 버전 비교 요약표

항목	HTML 4.01	XHTML 1.0	HTML5
발표 시기	1999	2000	2014 (지속 업데이트 중)
문법	느슨한	매우 엄격함	유연하고 관대한
문서 타입 선언	복잡 (`3가지`)	XML 방식	단순 (`<!DOCTYPE html>`)
시맨틱 태그	없음	없음	있음 (`<section>`, `<nav>` 등)
멀티미디어 지원	거의 없음	없음	`<video>`, `<audio>` 내장
파서	SGML 기반	XML 파서	HTML 파서
모바일 대응	미흡	미흡	우수

✅ 한줄 요약

> **HTML은 "문서 중심의 마크업"에서 "앱 플랫폼을 위한 언어"로 진화해왔다.**
> 그 핵심 분기점은 **XHTML의 엄격함과 HTML5의 유연함의 대비**에 있다.

1.3 HTML과 웹의 관계

HTML은 단순히 웹페이지를 만드는 도구가 아니라, **웹 자체의 근간을 이루는 핵심 기술 중 하나**다.
이 장에서는 **HTML이 웹에서 어떤 역할을 하며, 웹의 다른 구성 요소들과 어떤 방식으로 협력하는지**를 설명한다.

✅ 웹(Web)이란?

- **Web (World Wide Web)**: 인터넷 위에서 작동하는 **문서 중심의 하이퍼텍스트 시스템**
- HTML 문서들이 **링크로 연결되어 있으며**, 사용자는 브라우저를 통해 문서 간을 이동함
- 웹의 기반 기술은 세 가지로 구성됨:

기술 구성요소	설명
HTML	문서의 구조와 콘텐츠를 정의
CSS	문서의 시각적 표현을 담당
JavaScript	문서의 동작 및 상호작용을 담당

✅ HTML의 역할: 웹 콘텐츠의 "뼈대"

HTML은 웹에서 다음과 같은 역할을 수행한다:

1. **문서 구조 제공**
   - 제목, 본문, 리스트, 링크, 이미지 등 웹페이지 구성 요소의 **계층 구조** 정의
   - 예: `<header>`, `<main>`, `<footer>`, `<section>` 등
2. **콘텐츠 의미 전달**
   - `<h1>`은 문서 제목, `<p>`는 문단, `<a>`는 링크로 해석됨
   - 검색 엔진, 스크린 리더, 봇 등이 의미를 이해할 수 있도록 도움
3. **브라우저와 통신**
   - 브라우저는 HTML을 해석하여 DOM(Document Object Model)을 생성
   - DOM은 JavaScript에 의해 조작 가능

✅ HTML과 브라우저의 상호작용

1. 사용자가 URL을 입력하면:
   - 브라우저가 HTTP 요청을 서버에 전송
   - 서버는 HTML 파일을 반환함
2. 브라우저는 HTML 파일을 파싱하고 DOM 트리 생성
3. HTML에서 참조한 CSS, JS, 이미지 파일 등을 추가로 요청
4. 최종적으로 렌더링 엔진이 시각적으로 페이지를 그림

📌 브라우저의 핵심 기능:
HTML → DOM → 렌더 트리 → 화면 표시

✅ HTML과 다른 웹 기술 간의 관계

기술	HTML과의 관계
CSS	HTML의 태그에 시각적 스타일을 입힘 (`class`, `id`로 연결됨)
JavaScript	HTML 문서의 구조나 콘텐츠를 동적으로 조작
HTTP	HTML 문서를 주고받는 통신 프로토콜
웹 서버	HTML 문서를 저장하고 클라이언트 요청에 따라 전송
SEO	HTML 구조(특히 시맨틱 마크업)는 검색 엔진 최적화에 핵심적
웹 접근성(Accessibility)	HTML의 시맨틱 구조와 ARIA 속성은 장애인 접근성 향상에 기여
SPA / PWA	HTML은 초기 뼈대 제공 + 이후 JS가 동적으로 변경 (SPA 구조)

✅ HTML이 없는 웹은?

- 웹페이지가 존재하지 않음
- 검색, 쇼핑, 블로그, 지도 등 대부분의 기능이 표현될 수 없음
- CSS와 JavaScript는 HTML이 있어야만 의미가 있음
- 요약하자면:

> **HTML이 없다면 웹도 없다.**
> HTML은 웹을 구성하는 가장 기초적이고 핵심적인 요소이다.

✅ 한눈에 보는 구성 요소 관계도

```
사용자
  ↓
브라우저
  ↓
HTML  ↔  CSS (디자인)
   ↑       ↓
 JavaScript (동작 제어)
```

- HTML은 모든 웹 기술의 "중심 축"이다.

✅ 정리 요약

역할	HTML이 하는 일
구조	콘텐츠의 구획을 나누고 계층화
의미	각 요소의 용도와 목적을 표현

역할	HTML이 하는 일
인터페이스	사용자와 상호작용할 수 있도록 구성
연결성	다른 문서나 리소스와의 연결 허용
기반	CSS/JavaScript가 작동할 수 있는 토대 제공

1.4 HTML 파일 구조

HTML 문서는 단순한 텍스트 파일이지만, 브라우저가 이해하고 렌더링하기 위해서는 **정해진 구조와 규칙**을 따라야 한다.
 이 항목에서는 HTML 파일이 어떻게 구성되는지 **전체 구조**, **각 부분의 역할**, **기본 예시**, **실수 방지 포인트**까지 상세하게 설명한다.

✅ 기본 HTML 문서 구조

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <title>문서 제목</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1>안녕하세요!</h1>
    <p>이것은 HTML 문서의 예시입니다.</p>
  </body>
</html>
```

✅ 구성 요소 상세 설명

🔸 `<!DOCTYPE html>`

- 문서가 **HTML5 표준을 따름**을 브라우저에 선언
- 문법 파서가 HTML5 규칙에 따라 문서를 해석하도록 함
- **필수 요소**이며, 문서 맨 위에 있어야 함

```
<!DOCTYPE html>
```

🔸 `<html>`

- HTML 문서의 **루트(root) 요소**
- `lang` 속성은 문서의 기본 언어를 명시 (검색엔진, 스크린리더 참고)

```html
<html lang="ko">
  ...
</html>
```

🔸 <head>

- 문서의 **메타 정보, 설정, 외부 자원 로딩** 등 브라우저에게 필요한 정보를 포함
- 직접 사용자에게 보여지는 콘텐츠는 아님

주요 태그들:

태그	설명
`<meta charset="UTF-8">`	문자 인코딩을 UTF-8로 설정 (한글 포함 모든 언어 표현 가능)
`<title>`	브라우저 탭 제목 표시
`<link>`	외부 CSS 파일 불러오기
`<script>`	외부 JS 파일 연결
`<meta name="viewport">`	모바일 대응을 위한 뷰포트 설정
`<meta name="description">`	SEO 검색 설명에 사용

```html
<head>
  <meta charset="UTF-8" />
  <title>페이지 제목</title>
  <link rel="stylesheet" href="main.css" />
</head>
```

🔸 <body>

- 사용자가 **브라우저에서 실제로 보는 모든 콘텐츠**가 들어가는 영역
- 텍스트, 이미지, 버튼, 폼 등 UI 요소가 모두 이곳에 위치함

```html
<body>
  <h1>메인 제목</h1>
  <p>본문 내용입니다.</p>
</body>
```

✅ 전체 구조 계층 (DOM 관점에서)

```
HTML
├── HEAD
│   ├── META
│   ├── TITLE
│   └── LINK / SCRIPT 등
└── BODY
    ├── H1
    ├── P
    └── 기타 UI 요소들
```

✅ 실수 방지 체크리스트

체크 항목	설명
`<!DOCTYPE html>`을 안 쓰면?	브라우저가 호환 모드로 렌더링하여 레이아웃 깨짐
`<meta charset="UTF-8">` 누락 시	한글, 특수문자 깨짐 발생
`<html>`, `<head>`, `<body>` 생략	일부 브라우저는 자동 보정하지만 권장되지 않음
태그 짝 누락	닫히지 않은 태그는 렌더링 오류 발생 가능
중첩 구조 오류	`<p><div></div></p>` → 문법상 불가능 (블록 요소 중첩 주의)

✅ 실전 예제: 모바일 대응 포함한 기본 HTML 문서

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="HTML 파일 구조 설명 페이지" />
  <title>HTML 구조 예시</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <header>
    <h1>나의 첫 HTML</h1>
  </header>
  <main>
    <p>이 문서는 HTML의 구조를 설명하는 예시입니다.</p>
  </main>
  <footer>
    <p>&copy; 2025 Jeongseok All rights reserved.</p>
  </footer>
</body>
</html>
```

✅ 요약

> HTML 문서는 항상
> `<!DOCTYPE>` → `<html>` → `<head>` → `<body>`
> 순서로 구성되며, 각각의 요소는 **정확한 역할**을 갖는다.
> 이 구조를 이해하는 것이 HTML의 출발점이다.

1.5 브라우저의 역할과 HTML 렌더링

웹 브라우저는 사용자가 입력한 URL을 바탕으로 **HTML을 포함한 웹 콘텐츠를 가져와 시각적으로 렌더링**하는 도구다.
단순한 문서 보기 도구가 아니라, HTML, CSS, JavaScript를 해석하여 **웹 애플리케이션 수준의 복잡한 UI와 인터랙션을 제공**하는 핵심 플랫폼이다.

이 항목에서는 브라우저가 **HTML을 처리하고 화면에 출력하는 전 과정을 상세하게 설명**한다.

✅ 웹 브라우저란?

- 사용자가 웹사이트에 접근할 수 있게 해주는 **클라이언트 측 소프트웨어**
- HTML, CSS, JavaScript, 이미지 등 웹 문서를 해석하고, **시각적 결과물로 렌더링**
- 대표적인 브라우저: Chrome, Firefox, Safari, Edge 등

✅ 브라우저의 주요 구성 요소

구성 요소	설명
사용자 인터페이스	주소창, 뒤로가기, 즐겨찾기 등 UI 구성 요소
브라우저 엔진	사용자 인터페이스와 렌더링 엔진 사이의 조정자
렌더링 엔진	HTML, CSS 파싱 → DOM 트리 & 렌더 트리 생성 → 실제 화면 출력 (ex: Blink, Gecko)
자바스크립트 엔진	JavaScript 코드 실행 (V8, SpiderMonkey 등)
네트워킹 모듈	HTTP/HTTPS 프로토콜로 서버와 통신
데이터 저장소	쿠키, localStorage, IndexedDB, 세션 등

✅ 브라우저 렌더링 전체 흐름

📌 단계 요약

```
1. HTML 요청 및 수신
2. HTML 파싱 → DOM 트리 생성
3. CSS 파싱 → CSSOM 생성
4. DOM + CSSOM → 렌더 트리 구성
5. 레이아웃 계산
6. 페인팅(Painting)
```

```
7. 화면 출력 (Compositing)
```

---

1. HTML 파싱 → DOM 트리 생성

- HTML 문서를 위에서부터 읽어 내려가며 **DOM(Document Object Model)** 이라는 트리 구조 생성
- 각 HTML 태그가 노드로 변환되어 **계층 구조**를 이루며, 자바스크립트에서 접근 가능

```
<body>
  <h1>Hello</h1>
  <p>World</p>
</body>
```

→ DOM 구조

```
BODY
├── H1
└── P
```

---

2. CSS 파싱 → CSSOM 생성

- `<style>`, `<link>` 등을 통해 로딩된 CSS를 파싱하여 **CSSOM(CSS Object Model)** 트리 생성
- DOM과 별도로 존재하며, 스타일 규칙들이 트리 구조로 정리됨

---

3. DOM + CSSOM → 렌더 트리(Render Tree) 생성

- DOM 노드와 CSS 스타일을 결합하여 **화면에 실제로 표시할 요소들만 포함하는 렌더 트리** 생성
- `display: none` 요소는 렌더 트리에 포함되지 않음

---

4. 레이아웃(Layout, Reflow)

- 렌더 트리를 기반으로 각 요소의 **위치, 크기, 간격** 등을 계산
- 부모 요소의 크기에 따라 자식 요소의 위치가 결정됨

---

5. 페인팅(Paint)

- 각 노드의 시각적 요소 (색상, 그림자, 폰트 등)를 픽셀 단위로 계산
- 실제 **화면에 그릴 준비가 완료됨**

---

6. 합성(Compositing)

- 여러 레이어를 병합하여 최종 화면을 구성
- GPU를 활용해 빠르게 렌더링 → 스크롤, 애니메이션 등 최적화

✅ 자바스크립트의 개입 시점

- `<script>`가 DOM 파싱 중간에 등장하면 **파싱이 일시 중단**됨
- `defer`, `async` 속성으로 파싱과 병렬 실행 가능
- DOMContentLoaded 이벤트: DOM 생성 완료 시점 감지

✅ 실제 흐름 예시

1. 사용자가 주소창에 `https://example.com` 입력
2. DNS 조회 → 서버 접속 → HTML 파일 수신
3. 브라우저가 HTML 파싱 → DOM 트리 생성
4. CSS 로딩 → CSSOM 생성
5. DOM + CSSOM → 렌더 트리 생성
6. 레이아웃 계산 → Paint → 합성 → 출력

✅ 성능에 영향을 주는 요소

요소	영향
`<script>` 위치	DOM 파싱 차단 여부
CSS 파일 개수/크기	렌더링 지연
이미지 크기	페인팅 비용 증가
리플로우 발생 요소	DOM 구조 변경, 폰트 변경 등
브라우저 캐시 활용 여부	네트워크 지연 최소화

✅ 요약 다이어그램

```
HTML → DOM 트리
CSS  → CSSOM
       ↓
DOM + CSSOM → Render Tree
                    ↓
          Layout → Paint → Screen
```

✅ 한 줄 요약

> **브라우저는 HTML을 파싱해 구조화(DOM), 스타일(CSSOM), 위치(Layout), 픽셀(Paint) 단계를 거쳐 화면에 출력한다.**
> 효율적인 HTML 작성은 브라우저 렌더링 속도와 직결된다.

1.6 W3C, WHATWG, MDN 등의 표준화 기관

✅ 왜 표준화 기관이 중요한가?

- **웹 기술은 전 세계 수십억 사용자에게 동시 제공됨**
- 브라우저(Chrome, Firefox, Safari 등)가 HTML을 **일관되게 해석하고 동작**해야 함
- 웹 개발자는 브라우저별 동작 차이를 걱정하지 않고 **하나의 명세만 참고**하면 됨
- 이를 위해 **표준화 기관이 명세(specification)를 작성하고 관리**함

🔹 1. W3C (World Wide Web Consortium)

🌐 개요

- **1994년 팀 버너스 리**(HTML 발명자)가 설립
- HTML, CSS, XML, SVG 등 웹 전반의 표준을 담당
- 웹의 **개방성, 접근성, 국제화, 보안** 등을 목표로 함

📌 주요 특징

- **합의 기반(Consensus-driven)** 방식 → 기업, 단체, 개인이 참여
- 표준 단계: *Working Draft* → *Candidate Recommendation* → *Recommendation*
- HTML 4, XHTML 등의 사양을 주도

📌 한계

- XHTML 실패 이후 느린 대응, 복잡한 승인 프로세스 등으로 **WHATWG에 주도권을 넘김**

🔗 공식 사이트

https://www.w3.org

🔹 2. WHATWG (Web Hypertext Application Technology Working Group)

🌐 개요

- 2004년 **Apple, Mozilla, Opera, Google**이 중심이 되어 설립
- 기존 W3C의 XHTML 추진에 반발하여 **HTML5 개발 주도**
- 현재는 **HTML의 유일한 표준 제정 기구로 자리잡음**

📌 주요 특징

- **Living Standard** 방식:
  - 버전 5, 5.1 같은 구분 없이 **계속 진화하는 표준**
  - 항상 최신 브라우저 기술 반영
- 주요 브라우저 벤더들이 직접 참여 → **현실적이고 빠른 반영**

📌 현재 역할

- HTML, DOM, Fetch API, URL API 등 **현대 웹의 핵심 기술 대부분을 관리**

🔗 공식 사이트

https://whatwg.org

🔹 3. MDN Web Docs (Mozilla Developer Network)

🌐 개요

- **Mozilla**가 주도하는 웹 개발자 중심의 기술 문서 사이트
- W3C/WHATWG의 공식 문서는 너무 딱딱하므로, **실전 예제와 해설 중심의 친절한 문서** 제공

📌 특징

항목	설명
사용자 참여형	GitHub을 통해 누구나 기여 가능
방대한 예제	HTML, CSS, JS 태그별 예제 및 실습
다국어 지원	한국어를 포함한 여러 언어로 번역됨
브라우저 지원표	각 태그/속성의 브라우저별 지원 여부 확인 가능
실전 문서 중심	"어떻게 쓰는지"에 집중 (명세보다는 활용 설명 위주)

🔗 공식 사이트

https://developer.mozilla.org

🔹 4. 기타 주요 기관 및 커뮤니티

이름	설명
ECMA (Ecma International)	JavaScript 표준(ECMAScript)을 관리
IETF (Internet Engineering Task Force)	HTTP, URI, TCP/IP 등 인터넷 핵심 프로토콜 관리
WHATWG GitHub	HTML 명세 이슈와 수정을 관리하는 저장소
Can I Use	https://caniuse.com — 기능별 브라우저 호환성 조회 사이트

✅ 표준화 흐름 요약

```
과거:
  W3C → HTML, CSS 등 전통적인 명세 작성

현재:
  WHATWG → HTML, DOM, Fetch 등 핵심 API 주도
  MDN → 실무 중심의 문서화 및 튜토리얼 제공
```

📌 요즘은 공식 명세는 WHATWG에서 관리하고, **실전 참고 문서는 대부분 MDN을 사용**함

✅ 개발자 입장에서 요약 정리

목적	참고할 사이트
공식 HTML 문서 구조/정의	WHATWG HTML Spec
실전 개발 방법/예제	MDN Web Docs
표준 이슈/진행 상황	WHATWG GitHub
기술 역사/철학적 기반	W3C

✅ 한 줄 요약

> **WHATWG가 HTML의 살아있는 표준을 관리하고, MDN이 실무용 문서를 제공하며, W3C는 웹의 철학과 국제화를 이끈다.**