6. 이미지 및 미디어 삽입

6.1 이미지 태그 (`<img>`, `src`, `alt`, `title`)

HTML에서 `<img>` 태그는 **웹 페이지에 이미지를 삽입하는 기본 태그**이다.
이미지는 텍스트보다 시각적으로 더 많은 정보를 전달하므로,
웹 디자인, UI 구성, 정보 전달 등에서 핵심적인 요소로 사용된다.

✅ 기본 문법

```
<img src="이미지주소" alt="대체텍스트">
```

예시:

```
<img src="logo.png" alt="회사 로고">
```

- `src` : 실제 이미지 파일의 경로 (URL 또는 상대경로)
- `alt` : 이미지가 로딩되지 않을 때 표시할 **대체 텍스트**
  또는 **스크린 리더 사용자에게 설명 제공**

✅ 주요 속성 정리

속성	필수 여부	설명
`src`	✅ 필수	이미지 파일의 경로 (source)
`alt`	✅ 필수	대체 텍스트 (접근성, SEO 중요)
`title`	❌ 선택	툴팁 텍스트 (마우스를 올리면 표시됨)
`width`	❌ 선택	이미지 너비 (px 또는 %)
`height`	❌ 선택	이미지 높이 (px 또는 %)
`loading`	❌ 선택	lazy 로딩 등 최적화 설정 (HTML5+)

✅ 예제

```
<img src="/assets/profile.jpg" alt="프로필 사진" title="사용자 프로필" width="150">
```

- `title`: 마우스를 올리면 `사용자 프로필` 툴팁 표시
- `width`: 브라우저에서 가로 길이 150px로 표시

✅ alt 속성의 중요성

- 시각장애인을 위한 **스크린 리더 접근성 보장**
- SEO 최적화: 검색엔진은 `alt`를 기반으로 이미지 내용 파악
- 이미지가 깨졌을 때(404 등) **텍스트로 의미 전달 가능**

```
<img src="nonexistent.jpg" alt="실종된 고양이 사진">
```

🔽 이미지가 없다면 `"실종된 고양이 사진"`이라는 텍스트가 표시됨

✅ title vs alt 비교

항목	alt	title
역할	이미지의 **의미 설명**	마우스를 올리면 **툴팁 제공**
접근성	스크린 리더가 읽음	스크린 리더는 대부분 무시
SEO	검색 엔진이 인식함	검색 엔진 영향 적음
필요성	**반드시 입력**	선택 입력

✅ src 경로의 종류

유형	예시	설명
절대 경로	`https://example.com/logo.png`	외부 서버의 이미지
루트 기준 경로	`/images/logo.png`	사이트 루트 기준 경로
상대 경로	`./img/photo.jpg`, `../img/icon.png`	현재 HTML 문서 기준의 상대 위치

✅ Lazy Loading (지연 로딩)

```
<img src="photo.jpg" alt="자연 풍경" loading="lazy">
```

- 브라우저가 **스크롤될 때까지 이미지를 로딩하지 않음**
- **페이지 성능 개선**에 효과적 (특히 많은 이미지 사용 시)

✅ 이미지 크기 조절

```
<img src="map.png" alt="지도" width="300" height="200">
```

- 직접 지정하거나 CSS로 조절 가능
- 가로/세로 비율을 맞추지 않으면 **왜곡** 발생 주의

```
img {
  max-width: 100%;
  height: auto;
}
```

📌 반응형 웹을 위한 기본 스타일

✅ 웹 접근성 팁

이미지 유형	alt 속성 사용법
정보 전달 이미지	alt를 **의미 있게 설명**
장식용 이미지	`alt=""` (빈 문자열) 사용
텍스트 포함 이미지	alt에 **동일한 텍스트** 기입
클릭 유도 이미지 버튼	alt에 **행동 설명** ("검색", "보내기") 등

✅ 잘못된 예시

```
<img src="photo.jpg">
```

- `alt` 속성이 없으면 접근성 및 SEO 저하

```
<img src="photo.jpg" alt="사진">
```

- alt 내용이 너무 추상적이면 의미 전달 실패

✅ 한 줄 요약

> **`<img>` 태그는 이미지 삽입을 위한 HTML 기본 태그이며, `alt`는 접근성과 SEO를 위해 필수적으로 작성해야 한다.**
> **`title`은 툴팁 기능만을 제공하는 보조적 속성이다.**

6.2 이미지 포맷 (JPG, PNG, SVG, WebP)

웹 페이지에서 이미지는 **시각적 전달력과 디자인 품질**을 결정짓는 요소다.
그러나 **이미지 포맷 선택**을 잘못하면 페이지 속도가 느려지고
화질이 손상되며 브라우저 호환성 문제가 생길 수 있다.

각 포맷은 **특정 목적에 최적화**되어 있으므로,
올바른 선택이 **퍼포먼스와 품질의 균형**을 좌우한다.

✅ 대표 포맷 요약

포맷	특징	용도 예시	투명도	애니메이션	브라우저 지원
JPG	손실 압축, 고화질 사진	인물, 풍경, 배경	❌	❌	✅ 전 브라우저
PNG	무손실, 투명도 지원	아이콘, UI 요소	✅	❌	✅
SVG	벡터 기반, 코드 작성 가능	로고, 아이콘, 일러스트	✅	✅ (CSS 활용)	✅
WebP	차세대 포맷, 고압축	모든 이미지 유형	✅	✅	✅ (Safari는 일부 제한)

🖼️ 1. JPG (JPEG)

✔ 특징

- **손실 압축**: 파일 크기를 줄이면서 일부 화질 손실
- **고해상도 이미지에 적합**
- 투명도 미지원

✔ 사용 예

```
<img src="photo.jpg" alt="풍경 이미지">
```

✔ 적합한 용도

- 사진, 제품 이미지, 배경 이미지
- 큰 이미지를 상대적으로 작은 용량으로 저장

🖼️ 2. PNG

✔ 특징

- **무손실 압축**: 화질 손실 없음
- **알파 채널(투명도)** 지원
- 배경이 투명한 이미지에 최적

✔ 사용 예

```
<img src="icon.png" alt="투명 아이콘">
```

✔ 적합한 용도

- UI 아이콘, 로고, 캡처, 텍스트 이미지
- 배경이 투명한 버튼/레이어

🖼️ 3. SVG (Scalable Vector Graphics)

✔ 특징

- **벡터 기반 포맷** (수학적 경로로 그려짐)
- 확대/축소 시 **화질 손실 없음**
- HTML 내에 **직접 코드 삽입 가능**
- CSS/JS와 연동하여 동적 애니메이션 구현 가능

✔ 사용 예

```
<img src="logo.svg" alt="로고">
```

✔ 적합한 용도

- 로고, 일러스트, 도형, 인터랙티브 애니메이션
- 고해상도 디스플레이 대응 (retina)

🖼️ 4. WebP

✔ 특징

- **Google에서 개발한 차세대 포맷**
- JPG보다 **~25% 더 작은 용량**으로 비슷한 품질
- 투명도, 애니메이션, 압축 모두 지원
- **브라우저 호환성**이 이슈 (일부 구버전 Safari, IE 미지원)

✔ 사용 예

```
<picture>
  <source srcset="image.webp" type="image/webp">
  <img src="image.jpg" alt="대체 이미지">
</picture>
```

🔽 지원 브라우저는 WebP를, 그 외는 JPG를 사용함

✔ 적합한 용도

- 대량 이미지가 필요한 웹사이트 (블로그, 쇼핑몰)
- 속도와 용량 최적화가 중요한 서비스

✅ 정리 비교표

포맷	압축 방식	투명도	애니메이션	장점	단점
JPG	손실	❌	❌	용량 작음, 사진 적합	화질 손상, 투명도 없음
PNG	무손실	✅	❌	화질 보존, 투명 지원	용량 큼
SVG	벡터	✅	✅ (CSS)	확대/축소 무제한, 인터랙티브	복잡한 이미지 부적합
WebP	손실/무손실	✅	✅	고효율, 모든 기능 지원	일부 브라우저 미지원

✅ 실전 선택 가이드

상황	추천 포맷
블로그 배경, 인물 사진	`JPG`
UI 아이콘, 투명한 배너	`PNG`
로고, 아이콘, 다이어그램	`SVG`
속도 중요 + 사진 품질 유지	`WebP`

✅ 한 줄 요약

> **이미지 포맷은 품질, 용량, 기능(투명도, 애니메이션), 호환성에 따라 적절히 선택되어야 하며, WebP와 SVG는 최신 웹 퍼포먼스 최적화에서 핵심적인 역할을 한다.**

6.3 오디오 삽입 (`<audio>`, `controls`, `loop`, `autoplay`)

HTML5부터 `<audio>` 태그를 통해 **브라우저 내에서 오디오 재생이 가능**해졌다.
이는 플러그인(예: Flash) 없이도 **음성, 효과음, 배경 음악 등을 직접 삽입하고 제어**할 수 있게 해주며,
게임, 교육 콘텐츠, 미디어 사이트 등에서 자주 사용된다.

✅ 기본 문법

```
<audio src="sound.mp3" controls></audio>
```

또는 `<source>`를 활용한 다중 포맷 방식:

```
<audio controls>
  <source src="sound.mp3" type="audio/mpeg">
  <source src="sound.ogg" type="audio/ogg">
  브라우저가 오디오를 지원하지 않습니다.
</audio>
```

✅ 주요 속성 정리

속성	설명
`controls`	기본 오디오 컨트롤러 표시 (재생, 정지, 볼륨 등)
`autoplay`	페이지 로드 시 자동 재생 (※ 브라우저 정책상 `muted` 필요할 수 있음)
`loop`	오디오 반복 재생
`muted`	기본 음소거 상태
`preload`	오디오 로딩 방식 (`auto`, `metadata`, `none`)
`src`	오디오 파일 경로 (간단 삽입용)

✅ 예제

```
<audio src="bgm.mp3" controls loop autoplay muted></audio>
```

- 자동 재생하며 반복(loop), 음소거 상태로 시작
- 브라우저에 따라 `muted` 없이 `autoplay`가 작동하지 않음

✅ `<audio>` + `<source>` 방식 권장 이유

```
<audio controls>
  <source src="voice.ogg" type="audio/ogg">
  <source src="voice.mp3" type="audio/mpeg">
  이 브라우저는 오디오를 지원하지 않습니다.
</audio>
```

- 여러 포맷을 지정해 다양한 브라우저 호환성을 확보할 수 있음
- `mp3`, `ogg`, `wav` 등 브라우저마다 지원 범위 다름

✅ `preload` 옵션

값	설명
`none`	미리 로딩하지 않음
`metadata`	메타데이터만 로딩 (길이, 포맷 등)
`auto`	브라우저가 판단하여 로딩

```
<audio src="intro.mp3" controls preload="metadata"></audio>
```

✅ JS 제어 (선택 사항)

```
const audio = document.querySelector("audio");

audio.play();     // 재생
audio.pause();    // 일시 정지
audio.volume = 0.5; // 볼륨 50%
```

✅ 접근성 및 사용자 경험 고려

- `controls` 는 **항상 넣는 것이 바람직**
- `autoplay` 는 **사용자 동의 없는 경우 지양**
- 무한 루프(`loop`)는 배경 음악 외에는 주의 필요

✅ 브라우저 호환성

브라우저	지원
Chrome	✅
Firefox	✅
Safari	✅ (단, autoplay 정책 존재)
Edge	✅
IE 9+	✅ (단, 제한적 포맷)

✅ 예제: 효과음 버튼

```
<button onclick="document.getElementById('clickSound').play()">눌러보세요</button>
<audio id="clickSound" src="click.mp3" preload="auto"></audio>
```

📌 버튼을 누르면 효과음 재생

✅ 한 줄 요약

> **`<audio>` 태그는 HTML5에서 기본 지원되며, `controls`, `loop`, `autoplay`, `muted` 등의 속성을 통해 사용자에게 다양한 오디오 재생 경험을 제공할 수 있다. 브라우저 호환성과 사용자 경험을 고려하여 사용해야 한다.**

6.4 비디오 삽입 (`<video>`, `source`, `poster`, `controls`)

HTML5에서는 `<video>` 태그를 통해 **브라우저 자체에서 비디오 재생**이 가능하다.
별도의 플러그인 없이도 동영상 강의, 제품 소개, 광고 영상 등을
웹페이지에 자연스럽게 삽입할 수 있으며, 다양한 속성을 통해 재생을 제어할 수 있다.

✅ 기본 문법

```
<video src="movie.mp4" controls></video>
```

또는 다중 포맷 지원을 위해 `<source>` 태그 사용:

```
<video controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  이 브라우저는 비디오를 지원하지 않습니다.
</video>
```

✅ 주요 속성 정리

속성	설명
`controls`	기본 컨트롤 UI 표시 (재생, 일시정지, 볼륨 등)
`autoplay`	페이지 로드 시 자동 재생
`loop`	반복 재생
`muted`	음소거 시작 (autoplay와 병행 시 필요)
`poster`	비디오가 재생되기 전 표시할 썸네일 이미지
`preload`	비디오 로딩 방식 (`auto`, `metadata`, `none`)
`width`, `height`	가로/세로 크기 설정 (픽셀 또는 %)

속성	설명
`src`	단일 비디오 소스 경로

✅ 예제: 실전 코드

```
<video controls width="640" height="360" poster="thumbnail.jpg">
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  <p>브라우저가 비디오 태그를 지원하지 않습니다.</p>
</video>
```

📌 `poster`는 비디오가 재생되기 전 썸네일 역할

✅ 자동 재생 예제

```
<video src="intro.mp4" autoplay muted loop></video>
```

> `autoplay`는 대부분의 브라우저에서 **음소거(muted)** 상태일 때만 허용됨

✅ preload 옵션

값	설명
`none`	미리 로딩하지 않음 (대역폭 절약)
`metadata`	길이, 포맷 등 메타 정보만 로딩
`auto`	브라우저가 필요에 따라 로딩

```
<video src="sample.mp4" controls preload="metadata"></video>
```

✅ 브라우저별 지원 포맷

포맷	확장자	MIME 타입	지원 범위
MP4	`.mp4`	`video/mp4`	✅ 전 브라우저
WebM	`.webm`	`video/webm`	✅ 대부분 지원
Ogg	`.ogv`	`video/ogg`	🔶 일부 지원 (Safari 제외)

✅ JS를 통한 제어 (선택)

```
const video = document.querySelector("video");

video.play();    // 재생
video.pause();   // 정지
video.currentTime = 0;  // 처음으로 되감기
video.volume = 0.5;     // 볼륨 조절
```

✅ 실전 활용 예: 배경 비디오

```
<video autoplay muted loop playsinline>
  <source src="background.mp4" type="video/mp4">
</video>
```

- `playsinline`: 모바일 사파리에서 전체화면 없이 재생
- 주로 **홈페이지 배경 영상**, **인트로 효과** 등에 사용

✅ 접근성 및 UX 팁

- `controls`는 항상 고려할 것 (재생을 사용자가 제어할 수 있도록)
- `poster`를 제공해 사용자에게 콘텐츠 예고
- 자동 재생은 **음소거(muted)**, **짧은 클립**일 때만 권장
- `track` 태그로 자막을 제공하면 접근성 개선 가능

✅ 한 줄 요약

> **`<video>` 태그는 HTML5에서 동영상을 직접 삽입하고 제어할 수 있는 강력한 요소이며, `controls`, `poster`, `autoplay`, `loop` 등의 속성을 조합해 사용자 경험을 유연하게 설계할 수 있다.**

6.5 유튜브 iframe 삽입

HTML에서는 `<iframe>` 태그를 사용해 외부 콘텐츠(예: 다른 웹사이트, 비디오 등)를 **현재 웹 페이지에 포함**할 수 있다.
유튜브 영상도 `<iframe>`을 통해 간편하게 삽입할 수 있으며, 이 방식은 공식적으로 유튜브에서 권장하는 방법이다.

✅ 기본 문법

```
<iframe
  width="560"
  height="315"
  src="https://www.youtube.com/embed/영상ID"
  title="YouTube video player"
  frameborder="0"
  allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
  allowfullscreen>
</iframe>
```

✅ 실제 예시

```
<iframe
  width="640"
  height="360"
  src="https://www.youtube.com/embed/dQw4w9WgXcQ"
  title="Rick Astley - Never Gonna Give You Up"
  frameborder="0"
  allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
  allowfullscreen>
</iframe>
```

📌 위의 `src` URL에서 `dQw4w9WgXcQ`는 해당 유튜브 영상의 고유 ID이다.

✅ 주요 속성 설명

속성명	설명
`src`	유튜브 임베드용 URL (`https://www.youtube.com/embed/영상ID`)
`width`, `height`	iframe의 가로·세로 크기 지정
`frameborder`	테두리 제거 (`0` 권장)
`allow`	다양한 기능 허용 설정 (autoplay, fullscreen 등)
`allowfullscreen`	전체 화면 모드 사용 가능하게 함
`title`	접근성과 SEO 개선 (영상 제목 제공)

✅ 유튜브 공유 링크와의 차이점

공유 링크 (X)	iframe URL (O)
`https://www.youtube.com/watch?v=ID`	`https://www.youtube.com/embed/ID`

`embed` 형식을 반드시 사용해야 iframe 삽입이 정상 동작함.

✅ 자동 재생, 시작 시간 등 옵션 추가

🎵 자동 재생:

```
src="https://www.youtube.com/embed/영상ID?autoplay=1&mute=1"
```

- 브라우저 정책상 **자동 재생은 반드시 mute 상태에서만 가능**

⏱️ 특정 시점부터 재생:

```
src="https://www.youtube.com/embed/영상ID?start=60"
```

- 초 단위로 지정 (위 예시는 1분 후부터 재생)

🔁 반복 재생:

```
src="https://www.youtube.com/embed/영상ID?playlist=영상ID&loop=1"
```

- `playlist=영상ID`를 반드시 함께 명시해야 `loop=1`이 작동함

✅ 반응형 유튜브 iframe 처리 (CSS)

```
<div class="video-container">
  <iframe src="https://www.youtube.com/embed/영상ID" frameborder="0" allowfullscreen>
</iframe>
</div>

<style>
.video-container {
  position: relative;
  padding-bottom: 56.25%; /* 16:9 비율 */
  height: 0;
  overflow: hidden;
}
.video-container iframe {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
```

```
18 }
19 </style>
```

📌 이 방식은 다양한 화면에서도 자동으로 크기를 조절해주는 **반응형 유튜브 임베드 레이아웃**이다.

✅ 접근성 팁

- `<iframe>`에 `title` **속성 필수**
- 동영상 내용이 중요할 경우 **자막 또는 텍스트 요약 제공**
- 다크 모드 테마에서는 iframe 주변 배경 색 대비도 고려

✅ 한 줄 요약

> **유튜브 영상은 `<iframe>` 태그를 통해 쉽게 삽입할 수 있으며, `embed` URL, autoplay, loop 등 다양한 파라미터 조합으로 기능을 확장할 수 있다. 반응형 웹에서는 CSS로 비율 기반 컨테이너를 만드는 것이 중요하다.**