10. HTML 속성 정리

10.1 전역 속성 (`id`, `class`, `style`, `title`, `lang`, `hidden`)

HTML의 전역 속성은 **모든 HTML 요소에서 공통적으로 사용할 수 있는 속성들**이다.
이들은 콘텐츠의 **식별, 스타일링, 접근성, 동작 제어, 메타 정보 부여** 등에 활용되며,
웹 개발의 기본 구조를 잡는 데 핵심적인 역할을 한다.

✅ 1. `id`

📌 목적:

- 문서 내에서 **유일한 식별자**를 부여
- **자바스크립트로 DOM 조작**하거나 **CSS 스타일 적용**, **앵커 이동** 등에 사용

```
<h2 id="intro">소개</h2>
<a href="#intro">소개 섹션으로 이동</a>
```

📌 한 문서 내에 동일한 `id`가 **두 번 이상 사용되면 안 됨** (유일해야 함)

✅ 2. `class`

📌 목적:

- 여러 요소에 **공통 스타일 또는 기능을 적용**하기 위한 **분류자**
- 하나의 요소에 여러 개의 클래스를 **공백으로 구분하여 지정 가능**

```
<p class="highlight important">중요 문장입니다.</p>
```

📌 `class`는 **중복 가능**하며, CSS/JS에서 다중 요소를 대상으로 작동시킬 때 유용

✅ 3. `style`

📌 목적:

- **인라인 스타일**을 직접 지정할 때 사용

```
<p style="color: red; font-weight: bold;">경고 메시지</p>
```

📌 유지보수성 및 재사용성을 위해 **가급적 외부 CSS 사용 권장**,
인라인 `style`은 **최후의 수단**

✅ 4. `title`

📌 목적:

- 요소에 **마우스를 올렸을 때 보여줄 설명 툴팁**을 제공

```
<button title="저장합니다">💾 저장</button>
```

📌 스크린 리더 사용자에게도 읽힐 수 있으나,
접근성 보조 수단으로만 사용하고 핵심 정보는 본문에 직접 노출해야 함

✅ 5. `lang`

📌 목적:

- 요소 또는 문서의 **언어를 명시**
- **스크린 리더**나 검색 엔진, 브라우저 번역 기능이 적절히 작동하도록 돕는다

```
<p lang="en">Hello, world!</p>
<p lang="ko">안녕하세요</p>
```

📌 전체 문서에는 `<html lang="ko">` 같이 루트 수준에 지정하는 것이 표준

✅ 6. `hidden`

📌 목적:

- 요소를 **문서에서 시각적으로/논리적으로 숨김**
- **렌더링되지 않으며** 스크린 리더에서도 기본적으로 무시됨

```
<div hidden>이 내용은 보이지 않습니다.</div>
```

📌 자바스크립트로 `element.hidden = false` 와 같이 **동적 제어** 가능
📌 시각적으로만 숨기고 접근성은 유지하려면 `aria-hidden="true"` 등 대안 고려

✅ 요약 표

속성	설명	사용 예시
`id`	고유 식별자	`id="main-title"`
`class`	분류 및 그룹	`class="nav active"`
`style`	인라인 스타일	`style="color: blue;"`
`title`	툴팁 텍스트	`title="더 많은 정보"`

속성	설명	사용 예시
`lang`	언어 정보	`lang="en"`
`hidden`	표시하지 않음	`hidden`

✅ 한 줄 요약

> 전역 속성은 **모든 HTML 요소에서 공통적으로 사용되며**,
> 웹 콘텐츠의 **식별, 표현, 설명, 제어, 국제화**를 담당하는 기본 도구이다.

10.2 데이터 속성 (`data-*`)

— HTML 요소에 사용자 정의 데이터를 안전하게 저장하는 표준 방식

`data-*` 속성은 HTML5에서 도입된 기능으로,
HTML 요소에 커스텀 데이터를 임베드할 수 있게 해주는 구조화된 방법이다.
JavaScript와의 연동, **UI 상태 추적**, **DOM 기반 로직 구현** 등에 매우 자주 사용된다.

✅ 1. 문법 및 기본 구조

```
<div data-user-id="12345" data-role="admin">홍길동</div>
```

- `data-` 접두어 뒤에 원하는 이름을 붙여 정의
- HTML 파서와 브라우저는 무시하고, **개발자가 사용 목적에 맞게 해석**

📌 속성 이름 규칙:

- 반드시 `data-`로 시작해야 하며,
- 이후에는 **소문자, 숫자, 하이픈(-)** 조합 가능 (`data-user-id`, `data-index`, `data-type` 등)

✅ 2. JavaScript로 접근하기

📌 `dataset` 속성 사용

```
const div = document.querySelector("div");
console.log(div.dataset.userId);   // "12345"
console.log(div.dataset.role);     // "admin"
```

> 하이픈(`-`)은 camelCase로 자동 변환됨 → `data-user-id` → `dataset.userId`

📌 설정하기

```
div.dataset.status = "active";
```

📌 이는 `<div data-status="active">` 와 동일하게 적용됨

✅ 3. 실전 활용 예제

✅ 예제: 버튼 클릭 시 ID 참조

```
<button data-id="789" onclick="handleClick(this)">삭제</button>

<script>
  function handleClick(elem) {
    const id = elem.dataset.id;
    alert(`ID ${id}를 삭제합니다.`);
  }
</script>
```

✅ 4. 왜 `data-*`를 써야 할까?

기존 방식	문제점
`class`, `id`를 남용하여 데이터 저장	의미와 역할이 섞여 구조 혼란
`title`이나 `value`를 변형 사용	표준 위반, 접근성 저하
JS에서만 변수로 저장	HTML과 JS 간 분리, 추적 어려움

✅ `data-*`는 **표준 준수 + 구조 명확성 + 유지보수 용이성** 모두 만족시킨다.

✅ 5. CSS에서 `data-*` 활용

CSS에서는 선택자는 가능하지만 **속성 값 조건은 제한적**이다.

```
div[data-role="admin"] {
  background-color: lightyellow;
}
```

📌 `data-*`는 JavaScript에서의 활용이 중심이나,
CSS에서 단순 **속성 유무/값**으로 스타일 조건 지정도 가능

✅ 6. 데이터 속성과 접근성(보조 기술)

- `data-*`는 **접근성 API나 스크린 리더에 직접 노출되지 않음**
- 시각적 정보나 의미 전달이 필요한 경우는 `aria-*`, `alt`, `label` 등을 활용해야 함
- `data-*`는 **프로그래밍 로직 기반 상태 추적**에만 사용하는 것이 바람직

✅ 한 줄 요약

> `data-*` 속성은 HTML 요소에 **구조적이고 의미 없는 사용자 정의 데이터를 저장할 수 있는 표준 방식**으로,
> **JavaScript와의 통신, UI 상태 관리, DOM 기반 애플리케이션 설계**에 핵심적인 역할을 한다.

10.3 `tabindex`, `accesskey`, `contenteditable`

— `tabindex`, `accesskey`, `contenteditable` 완전 정리

이 항목은 **웹 접근성(A11y)** 및 **사용자 편의성 향상**을 위해 사용되는 HTML 속성 중,
다음 3가지를 집중적으로 다룬다:

- `tabindex`: **탭 순서 조정**
- `accesskey`: **단축키 부여**
- `contenteditable`: **실시간 콘텐츠 편집 가능 여부 지정**

이들은 모두 **전역 속성(global attributes)**이며,
특히 키보드 기반 네비게이션 설계 시 매우 중요하다.

✅ 1. `tabindex`

— 키보드 탭 이동 순서를 제어하는 속성

📌 기본 설명:

HTML 요소들이 기본적으로 `Tab` 키로 포커스를 받을 수 있지만,
`tabindex` 속성을 사용하면 **탭 이동 순서를 명시적으로 제어**할 수 있다.

📌 사용 값 의미:

값	설명
`0`	**기본 순서에 따라 포커스 가능**하게 함
`-1`	**포커스는 가능하나 Tab 키로는 이동 불가**, JavaScript로만 포커스 가능
`1~n`	**사용자 지정 순서**, 숫자가 작을수록 먼저 포커스됨 (권장 안됨)

✅ 예제:

```
<a href="#">링크1</a>
<a href="#" tabindex="2">링크2</a>
<a href="#" tabindex="1">링크3</a>
```

> `링크3 → 링크2 → 링크1` 순으로 탭 이동됨 (숫자 기준)

📌 **접근성 권장사항:**
숫자 기반 `tabindex="1"` 이상의 사용은 **의도치 않은 순서 혼란**을 유발할 수 있어,
보통은 `tabindex="0"` 또는 `-1`만 사용하는 것이 바람직하다.

✅ 2. `accesskey`

— 키보드 단축키 지정

📌 설명:

- `accesskey` 속성은 요소에 **단축키를 부여**하여 키보드만으로 빠르게 접근할 수 있게 한다.
- 운영체제/브라우저별 조합 키(Alt, Ctrl 등)와 함께 사용된다.

✅ 예제:

```
<button accesskey="s">저장하기</button>
```

📌 사용 방법:

플랫폼	단축키 사용법
Windows (Chrome)	`Alt + accesskey`
Windows (Firefox)	`Alt + Shift + accesskey`
macOS (Chrome/Safari)	`Control + Option + accesskey`

📌 브라우저마다 다르게 작동하며, **키 충돌 위험**이 있으므로 신중하게 사용해야 함
📌 `<kbd>` 또는 `<small>`로 사용자에게 **명시적으로 안내**하는 것이 중요함

✅ 3. `contenteditable`

— HTML 요소를 브라우저에서 실시간으로 편집 가능하게 만듦

📌 설명:

- `contenteditable="true"`를 설정하면 해당 요소의 콘텐츠가 **직접 편집 가능**해진다.
- 사용자는 **텍스트를 입력하거나 삭제**할 수 있고, 내부에서 스타일이나 링크도 수정 가능

✅ 예제:

```
<div contenteditable="true">
  여기를 클릭하고 내용을 편집하세요.
</div>
```

📌 기본 값은 `inherit`이며, 상위 요소가 `true`인 경우 하위 요소도 편집 가능해짐
📌 `contenteditable="false"`를 통해 하위 요소만 비활성화하는 것도 가능

✅ 세 속성 비교 요약

속성	설명	사용 목적
`tabindex`	요소의 탭 이동 순서를 정의	**접근성 네비게이션 제어**
`accesskey`	요소에 키보드 단축키 지정	**빠른 접근 경로 제공**
`contenteditable`	사용자가 직접 내용을 편집 가능하게 함	**실시간 콘텐츠 입력 기능**

✅ 한 줄 요약

> `tabindex`, `accesskey`, `contenteditable`은 모두 **접근성과 사용자 경험 향상을 위한 HTML 전역 속성**으로,
> **키보드 네비게이션, 단축키 제공, 콘텐츠 편집 인터페이스 구현**에 핵심적인 역할을 한다.

10.4 `draggable`, `spellcheck`

이 항목에서는 HTML5에서 도입된 전역 속성 중
사용자 상호작용을 제어하거나 브라우저의 기본 기능을 활성화/비활성화하는 속성들인
`draggable`과 `spellcheck`에 대해 심층적으로 설명한다.

✅ 1. `draggable`

— 요소를 드래그할 수 있게 만들지 여부를 지정

📌 기본 설명:

`draggable` 속성은 **요소가 마우스로 드래그될 수 있는지 여부**를 지정한다.

```
<p draggable="true">이 문장은 드래그할 수 있습니다.</p>
```

📌 사용 가능한 값:

값	설명
`"true"`	드래그 가능 (사용자 정의 drag 시작 가능)
`"false"`	드래그 불가능
(없음)	요소 종류에 따라 기본값 결정됨 (이미지는 기본적으로 `true`)

✅ 브라우저 동작 예:

```
<img src="logo.png" draggable="false">
```

👉 브라우저 기본 이미지 드래그를 비활성화

✅ JavaScript와 연계 (Drag & Drop API)

```
<div id="box" draggable="true">Box</div>

<script>
  const box = document.getElementById("box");
  box.addEventListener("dragstart", (e) => {
    e.dataTransfer.setData("text/plain", "Box 드래그됨!");
  });
</script>
```

📌 `dragstart`, `dragover`, `drop` 이벤트와 함께 사용됨
📌 주로 **파일 업로드 인터페이스**, **드래그 정렬 UI** 등에 활용됨

✅ 2. `spellcheck`

— 요소 내 텍스트의 철자 검사 여부를 브라우저에 알림

📌 기본 설명:

`spellcheck` 속성은 브라우저가 **사용자 입력 내용에 대해 맞춤법 검사**를 수행할지를 제어한다.
주로 `input`, `textarea`, `contenteditable` 요소와 함께 사용된다.

```
<textarea spellcheck="true">기본값 텍스트</textarea>
```

📌 사용 가능한 값:

값	설명
`"true"`	브라우저가 **맞춤법 검사를 수행**함
`"false"`	맞춤법 검사를 **수행하지 않음**
(없음)	기본값은 요소 유형/브라우저 설정에 따라 다름

✅ 실전 예제:

```
<input type="text" spellcheck="false" value="Hello wrld">
```

👉 맞춤법 오류로 밑줄이 표시되지 않음

✅ 브라우저 지원:

- 대부분의 최신 브라우저(Chrome, Firefox, Edge, Safari)는 `spellcheck`를 지원
- **한국어, 영어 등 주요 언어의 사전 기반 맞춤법 검사**가 작동함

📌 유의사항:

- `<input type="email">`, `<input type="password">` 등 일부 필드에서는 자동으로 `spellcheck`가 비활성화됨
- 맞춤법 검사 기능은 **브라우저 설정**에 따라 꺼져 있을 수도 있음

✅ 요약 비교

속성	설명	주요 사용 목적
`draggable`	요소를 드래그할 수 있는지 여부 제어	사용자 정의 Drag & Drop 인터페이스
`spellcheck`	맞춤법 검사를 활성화할지 여부 제어	입력 필드에서의 오류 감지 UX 개선

✅ 한 줄 요약

> `draggable`은 **드래그 인터페이스 제어**, `spellcheck`는 **입력 필드 맞춤법 검사 제어**를 위한 HTML 전역 속성으로, 웹 애플리케이션의 **사용자 상호작용을 강화하거나 제약할 수 있는 중요한 도구**이다.

10.5 `role`, ARIA 속성 소개

— HTML 접근성(A11y)의 핵심 기초 설계 요소들

현대 웹에서 "접근성(Accessibility)"은
모든 사용자, 특히 시각장애인이나 키보드 사용자 등 비표준 환경의 사용자도
콘텐츠에 접근할 수 있도록 보장하는 것을 의미한다.

이를 위해 WAI-ARIA (Web Accessibility Initiative – Accessible Rich Internet Applications)는
표준 HTML만으로는 표현하기 어려운 의미나 상태를 명확하게 설명할 수 있는 속성군을 정의한다.

✅ 1. `role` 속성

— 요소의 의미/역할을 명시적으로 지정

📌 기본 설명:

`role` 속성은 HTML 요소가 **어떤 기능/역할을 수행하는지** 스크린 리더 등 보조 기술에게 알려준다.

```
<div role="button" tabindex="0">클릭</div>
```

➡️ `<div>`는 시맨틱 요소가 아니므로, `role="button"`을 통해 **버튼임을 명시**

✅ 주요 `role` 값 예시

역할 이름	설명
`button`	클릭 가능한 버튼 역할
`link`	하이퍼링크 역할 (`<a>`가 아닌 경우)
`navigation`	내비게이션 영역
`banner`	사이트의 헤더 역할 (`<header>`에 해당)
`main`	페이지의 메인 콘텐츠 (`<main>` 요소와 동일)
`alert`	긴급하게 주의를 끌어야 하는 경고 메시지
`dialog`	모달 대화 상자
`progressbar`	진행 상태 바
`tooltip`	툴팁 설명

✅ role이 필요한 경우

- `<div>`, `<span>` 등 **비시맨틱 요소**로 의미 있는 인터페이스를 만들 때
- JS 프레임워크로 구성된 UI (ex: React, Vue)에서 **사용자 정의 컴포넌트** 활용 시
- HTML5 태그로 표현되지 않는 구조를 **스크린 리더에게 설명하고자 할 때**

📌 이미 시맨틱 태그(`<button>`, `<nav>`)를 사용한 경우에는 `role`을 **명시할 필요 없음**

✅ 2. ARIA 속성 소개 (Attribute Categories)

ARIA 속성은 총 3종류로 나뉜다.

📘 2.1 상태(State) 속성: 현재 상태 표현

속성	설명	예시
`aria-checked`	체크 여부 (`true`, `false`)	체크박스
`aria-expanded`	펼침 여부 (`true`, `false`)	아코디언, 드롭다운
`aria-hidden`	보조 기술에서 숨김 처리	스크린 리더에서 무시할 요소
`aria-selected`	선택 여부 (`true`, `false`)	탭 UI, 선택 목록 등

```
<div role="checkbox" aria-checked="false">옵션</div>
```

📘 2.2 속성(Property) 속성: 본질적 특성 설명

속성	설명
`aria-label`	요소의 텍스트 라벨 직접 제공
`aria-labelledby`	다른 요소의 ID를 통해 라벨 지정
`aria-describedby`	설명용 텍스트 ID 참조
`aria-controls`	연관된 컨트롤 대상 ID

```
<button aria-label="검색 실행">🔍</button>
```

📘 2.3 관계(Relationship) 속성: 요소 간 의미 연결

- `aria-owns`: 시각적 DOM 순서와 논리적 소유권이 다를 때
- `aria-activedescendant`: 키보드 포커스가 하위 항목 중 어디에 있는지 지정

✅ 3. 실전 예시: 커스텀 아코디언

```
<button aria-expanded="false" aria-controls="panel1">자세히 보기</button>
<div id="panel1" hidden>
  <p>숨겨진 내용입니다</p>
</div>
```

📌 JS로 `aria-expanded`, `hidden` 값을 함께 토글하면
보조 기술과 시각적 사용자 모두에게 일관된 피드백 제공 가능

✅ 4. ARIA 사용 시 주의사항

- **기본 HTML 시맨틱 요소를 우선 사용**해야 한다 (ex: `<button>` > `<div role="button">`)
- ARIA는 **보완용**이며, HTML의 **기본 접근성 기능을 대체하지 않는다**
- 불필요하게 남용하면 **오히려 혼란을 줄 수 있음**

✅ 한 줄 요약

> `role`과 ARIA 속성은 **스크린 리더 및 키보드 사용자에게 UI의 의미와 상태를 명확히 전달**하기 위한 **웹 접근성의 핵심 도구**이며, 특히 커스텀 컴포넌트를 구성할 때 필수적이다.