M03-01 · GIBALJA VISUAL MANUAL

화면을 HTML 구조로 읽기

> **한 문장 목표:** 눈에 보이는 웹 화면을 목적·의미 영역·제목·목록·폼·링크·버튼의 HTML 구조로 바꾸고, 현재 DOM 위치와 접근성 이름을 증거로 설명합니다.

난이도	개념	실습	셀프 테스트	최종 산출물
Level 1	55분	55분	15분	화면 구조표, DOM 구조도, 제목 개요, 시맨틱 HTML 점검 기록

화면을 “위쪽 메뉴, 왼쪽 박스, 파란 버튼”으로만 설명하면 디자인 좌표에 머뭅니다. 같은 화면을 `nav`, `main`, `h1`, `form`, `button`으로 읽으면 내용의 목적과 사용자 행동을 개발자·디자이너·테스터에게 같은 언어로 전달할 수 있습니다.

그림 1. 화면은 보이는 모양, HTML 의미, 현재 DOM 관계, 접근성 전달의 네 겹으로 읽습니다.


PIXELS
화면
무엇이 보이는가?
제목·검색·목록
MEANING
HTML
무엇을 뜻하는가?
h1·form·ul
구조를 읽으면 화면 정의·개발 요청·

TREE

DOM

지금 어떻게 연결됐나?

부모·자식·형제

A11Y

접근성

어떻게 전달되는가?

이름·역할·상태

접근성 검수가 같은 언어로 연결된다

1. 이 PDF를 공부하는 방법

1회차 · 네 겹만 구분하기 · 15분

그림 1을 보며 `화면 → HTML → DOM → 접근성`을 소리 내어 읽습니다. 각 겹이 대답하는 질문을 한 문장씩 말합니다.

2회차 · 영역과 계층 읽기 · 25분

`header` · `nav` · `main` · `aside` · `footer` 영역과 `h1` ~ `h6` 제목 계층을 그림에서 찾습니다. 태그를 외우기보다 각 영역이 맡은 내용을 설명합니다.

3회차 · 탐색기로 같은 모양 비교하기 · 30분

화면 구조 탐색기에서 의미 구조 정상과 `div`만 사용한 화면을 비교합니다. 모양은 비슷하지만 의미 영역·제목·상호작용 수가 달라지는 이유를 기록합니다.

4회차 · Chrome에서 현재 DOM 확인하기 · 40분

요소 선택 모드로 화면과 Elements의 노드를 연결합니다. 부모·자식·형제, DOM 경로, 접근성 이름·역할을 화면 구조표에 적습니다.

5회차 · 셀프 테스트 · 15분

정답을 가리고 10문제를 풉니다. 틀린 문제는 그림 12의 여섯 질문 중 어느 질문을 빠뜨렸는지 표시합니다.

학습 완료 기준
"검색창이 있는 화면"을 "하나의 main 안에 h1, 이름이 있는 search 폼, 결과 제목 h2, article 목록, 보조 aside가 있으며 검색 버튼의 접근성 이름은 '검색'이다"처럼 구조와 증거로 설명할 수 있습니다.

2. 화면·HTML·DOM·접근성 트리는 서로 다릅니다

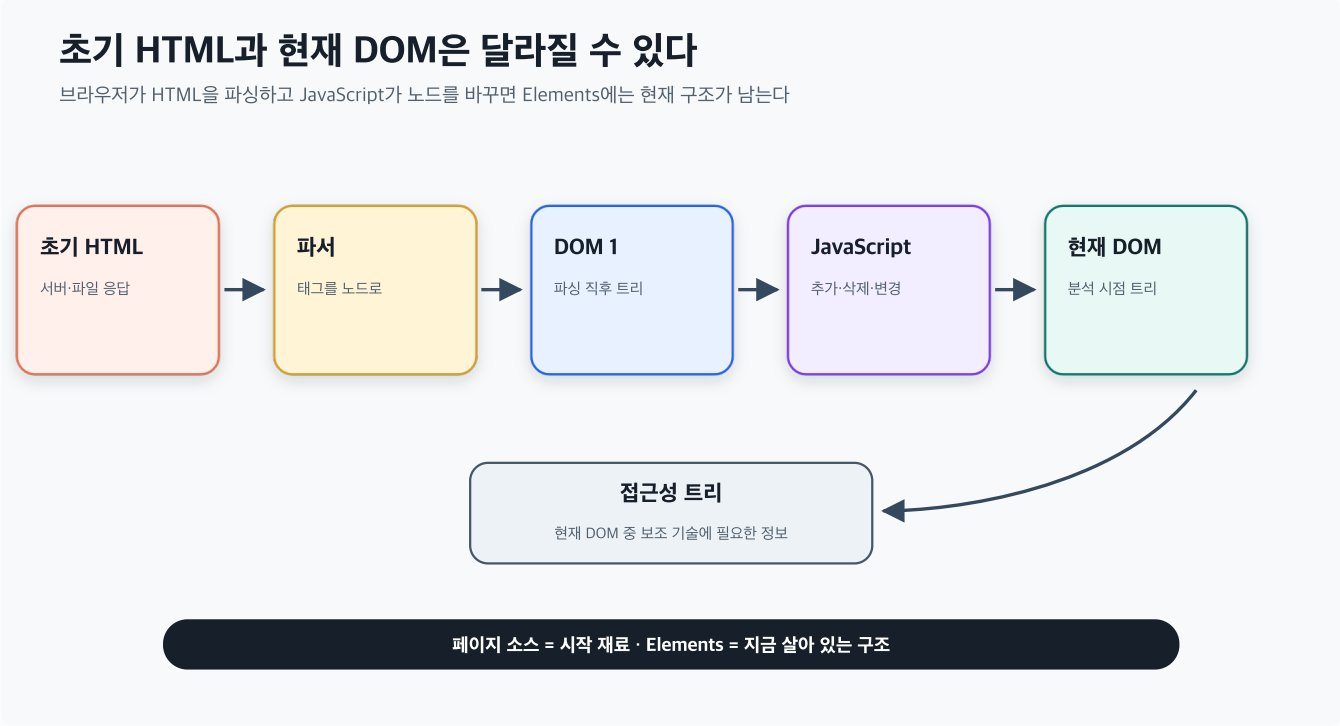
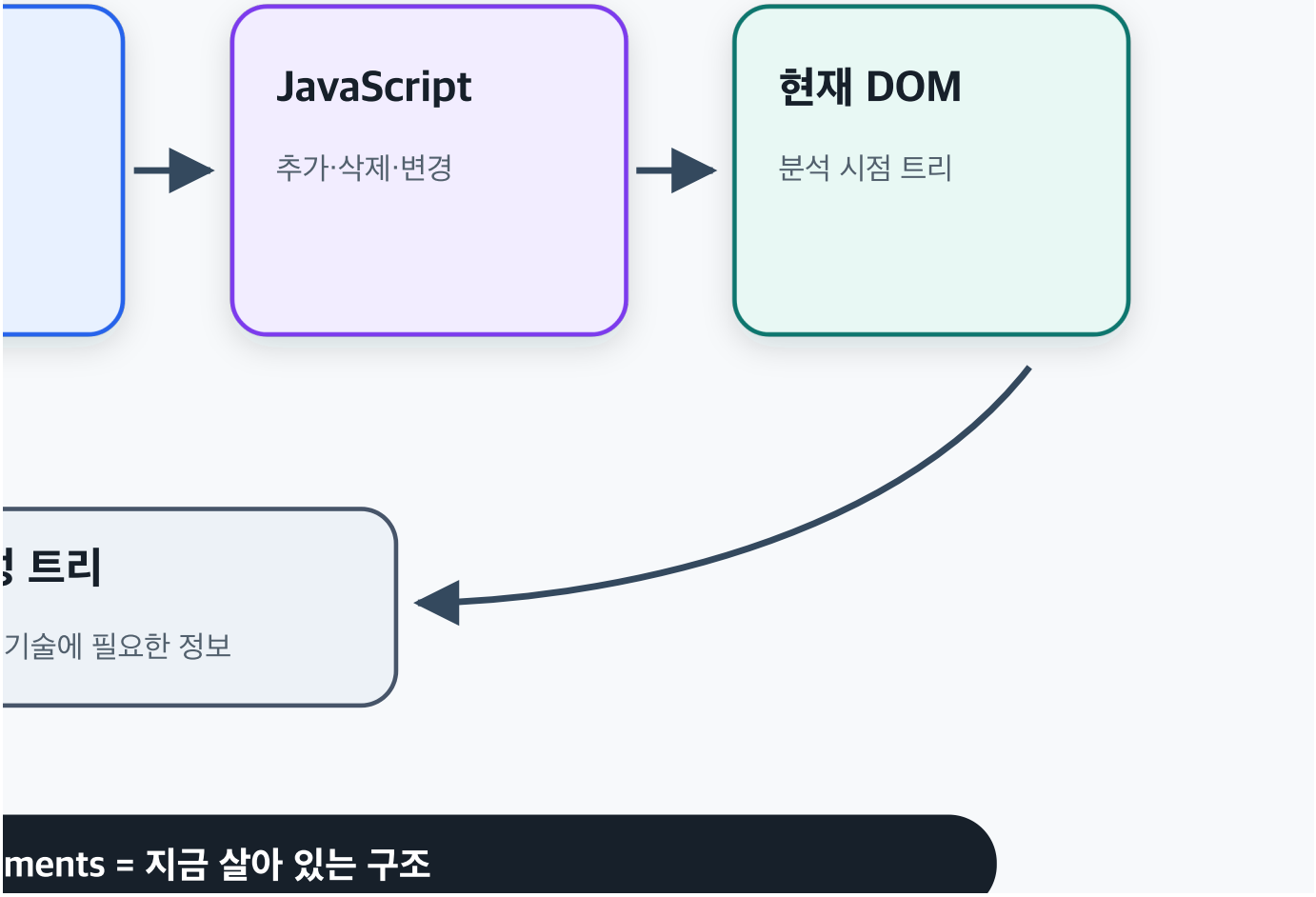


그림 2. 초기 HTML은 파싱되고 JavaScript로 바뀌며, Elements에는 분석 시점의 현재 DOM이 표시됩니다.


초기 HTML
서버·파일 응답
파서
태그를 노드로
DOM 1
파싱 직후 트리
접근성
현재 DOM 중 보조
페이지 소스 = 시작 재료 · Ele



ments = 지금 살아 있는 구조

2.1. 화면은 렌더링 결과입니다

화면은 브라우저가 HTML·CSS·이미지·글꼴·JavaScript 상태를 합쳐 그린 결과입니다. 사용자는 제목·버튼·목록처럼 보이는 모양을 먼저 봅니다.

하지만 큰 글자가 실제 `h1` 인지, 파란 박스가 실제 `button` 인지 화면만으로 확정할 수 없습니다.

2.2. HTML은 시작 구조와 의미를 표시합니다

하이퍼텍스트 마크업 언어(HyperText Markup Language, HTML)는 내용이 무엇인지 표시합니다.

```
<main>
  <h1>실무 자료 찾기</h1>
  <form role="search">
    <label for="query">검색어</label>
    <input id="query" type="search">
    <button type="submit">검색</button>
  </form>
</main>
```

이 코드는 글자의 크기나 좌표보다 다음 의미를 먼저 전달합니다.

- `main` : 이 문서의 중심 내용
- `h1` : 화면 전체를 대표하는 제목
- `form` : 함께 제출되는 입력 묶음
- `label` : 입력 목적을 알려 주는 글
- `input` : 사용자가 값을 입력하는 컨트롤
- `button` : 현재 화면에서 명령을 실행하는 컨트롤

2.3. DOM은 현재 살아 있는 트리입니다

문서 객체 모델(Document Object Model, DOM)은 브라우저가 문서를 객체와 노드의 트리로 표현한 것입니다. JavaScript는 노드를 추가·삭제·변경할 수 있습니다.

```
const message = document.createElement('p');
message.textContent = '검색 결과가 2건입니다.';
document.querySelector('main').appendChild(message);
```

초기 HTML에 없던 `p` 노드가 현재 DOM에 생깁니다. 따라서 동적 화면을 분석할 때는 **페이지 소스보다 Elements의 현재 DOM**을 기준으로 봅니다.

2.4. 접근성 트리는 DOM의 단순 복사본이 아닙니다

접근성 트리는 보조 기술이 페이지를 이해하고 조작하는 데 필요한 이름·역할·상태·값을 중심으로 구성됩니다. 장식용 요소나 의미 없는 컨테이너는 빠질 수 있습니다.

```
DOM:           form > label + input + button > span(장식 아이콘)
접근성 트리:   search > searchbox "검색어" + button "검색"
```

BIG IDEA 01

페이지 소스는 시작 재료이고, Elements는 지금의 구조이며, Accessibility는 그 구조가 보조 기술에 전달되는 방식입니다.

30초 확인

페이지 소스에는 검색 결과가 없지만 화면과 Elements에는 결과 카드가 있습니다. 어느 자료를 현재 상태의 구조 증거로 사용해야 할까요?

정답 보기

Elements의 현재 DOM을 기준으로 사용합니다. JavaScript가 결과 노드를 추가했을 수 있으므로 화면 상태·확인 시각·사용자 행동도 함께 기록합니다.

3. DOM 트리의 관계를 읽습니다

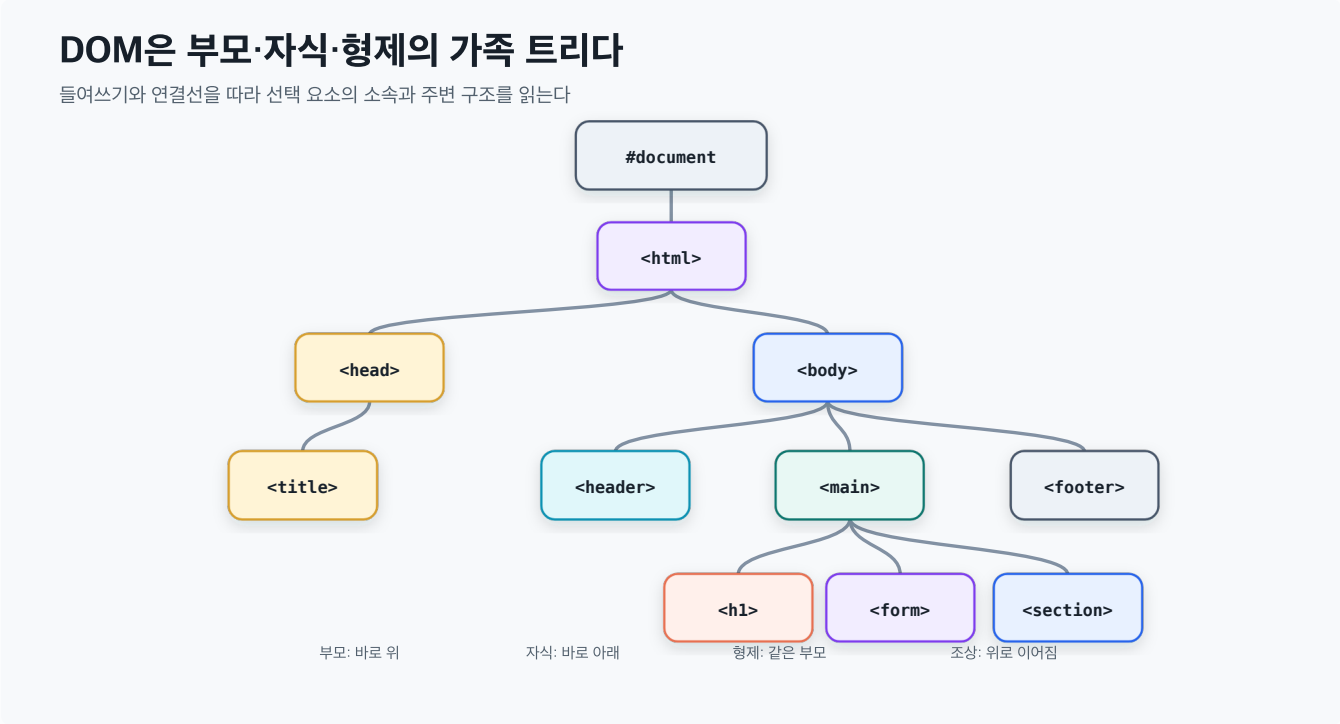
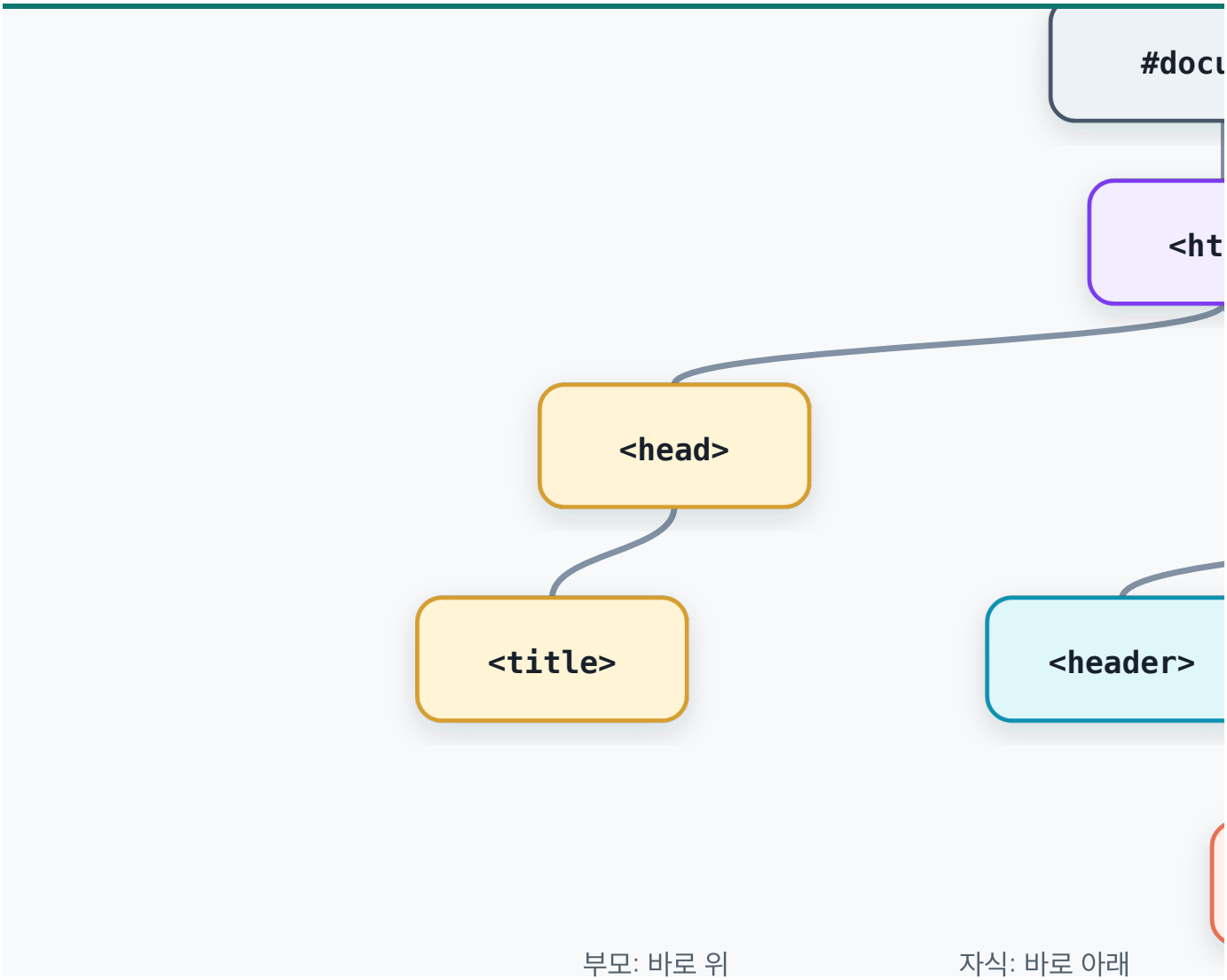
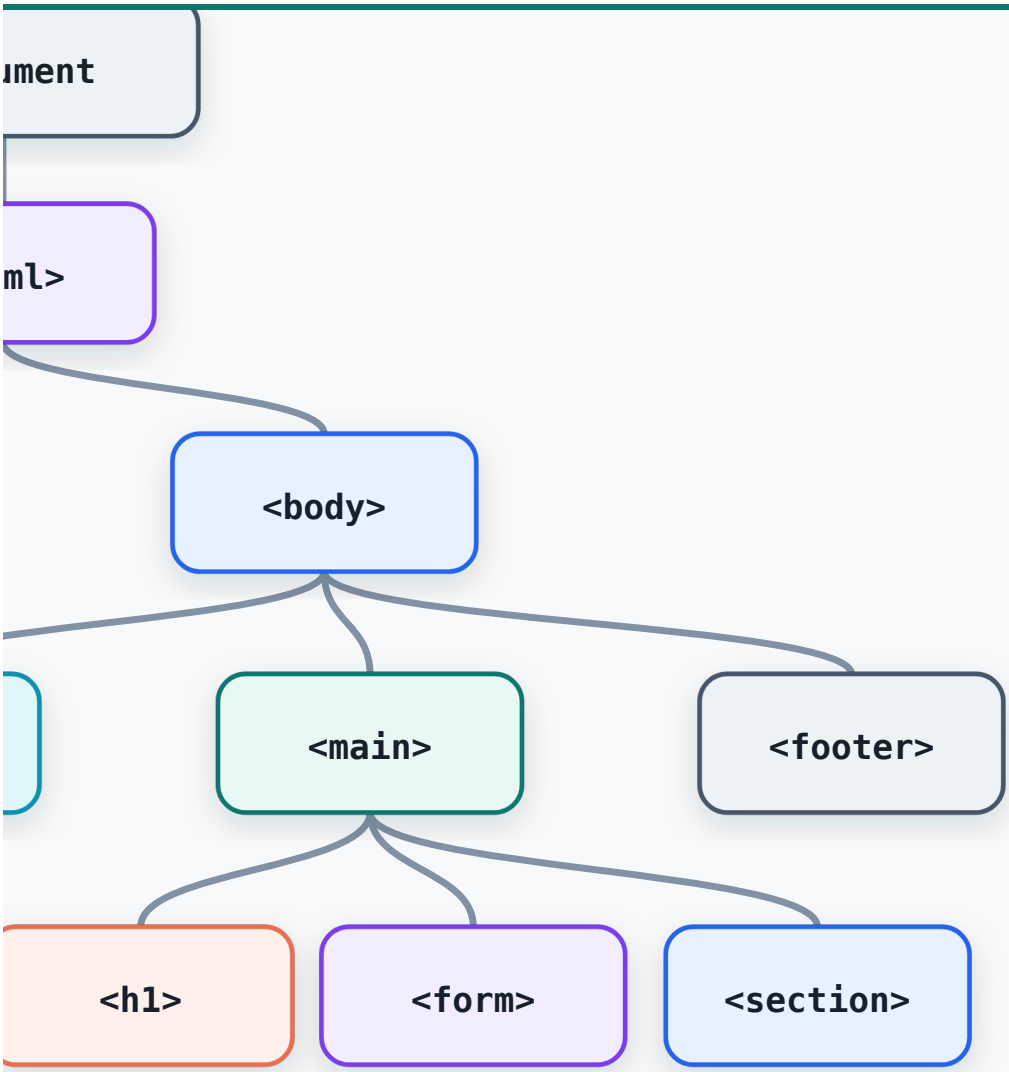


그림 3. DOM의 연결선과 들여쓰기는 부모·자식·형제·조상 관계를 보여 줍니다.


#docu
<ht
<head>
<title>
<header>
부모: 바로 위
자식: 바로 아래


ument
ml>
<body>
<main>
<footer>
<h1>
<form>
<section>
형제: 같은 부모
조상: 위로 이어짐

3.1. 노드와 요소를 구분합니다

DOM의 한 점을 노드(node)라고 합니다. 노드는 다음처럼 여러 종류입니다.

노드	뜻	예시
Document	전체 문서의 출발점	`document`
Element	HTML 요소	`main` , `h1` , `button`
Text	요소 안의 실제 글	`검색`
Comment	HTML 주석	`<!-- 검색 영역 -->`

모든 요소는 노드이지만 모든 노드가 요소는 아닙니다. `children`은 요소 자식만 보고, `childNodes`는 텍스트·주석을 포함한 노드를 봅니다.

3.2. 가족 관계로 위치를 읽습니다

```
<main>
  <h1>실무 자료 찾기</h1>
  <form>...</form>
  <section>...</section>
</main>
```

관계	이 예시에서 읽는 법
부모	`h1`의 부모는 `main`
자식	`main`의 자식은 `h1` · `form` · `section`
형제	`h1` · `form` · `section`은 같은 부모를 둔 형제
조상	`h1`의 조상에는 `main` · `body` · `html`이 있음
자손	`main` 안의 입력·버튼·결과 항목은 모두 자손

3.3. DOM 경로는 소속을 설명합니다

```
main.demo-main > form.demo-form > input#resource-query
```

이 경로는 검색 입력이 `main`의 검색 폼 안에 있음을 보여 줍니다. 단, 자동 생성된 긴 클래스나 위치 번호만으로 만든 경로는 화면 변경에 쉽게 깨집니다.

구조표에는 다음을 함께 적습니다.

- 보이는 글이나 업무 이름
- 의미 있는 태그와 역할
- 고유하고 안정적인 `id`
- 가장 가까운 의미 영역
- 분석한 화면 상태

3.4. DOM 순서와 화면 배치 순서는 다를 수 있습니다

CSS는 시각적 위치를 바꿀 수 있습니다. 화면에서 오른쪽에 보이는 요소가 DOM에서는 먼저 나올 수 있습니다. 키보드 초점과 화면 읽기 순서는 보통 DOM 순서의 영향을 받습니다.

따라서 다음 세 순서를 비교합니다.

1. DOM 소스 순서
2. 화면에 보이는 시각 순서
3. `Tab` 키로 이동하는 초점 순서

> **공유 주의**
> Elements에는 숨김 입력·`data-*` 속성·주석·현재 입력값이 보일 수 있습니다. 화면 캡처나 HTML을 공유하기 전에 개인정보·토큰·내부 식별자를 제거합니다.

4. 페이지를 의미 영역으로 나눕니다

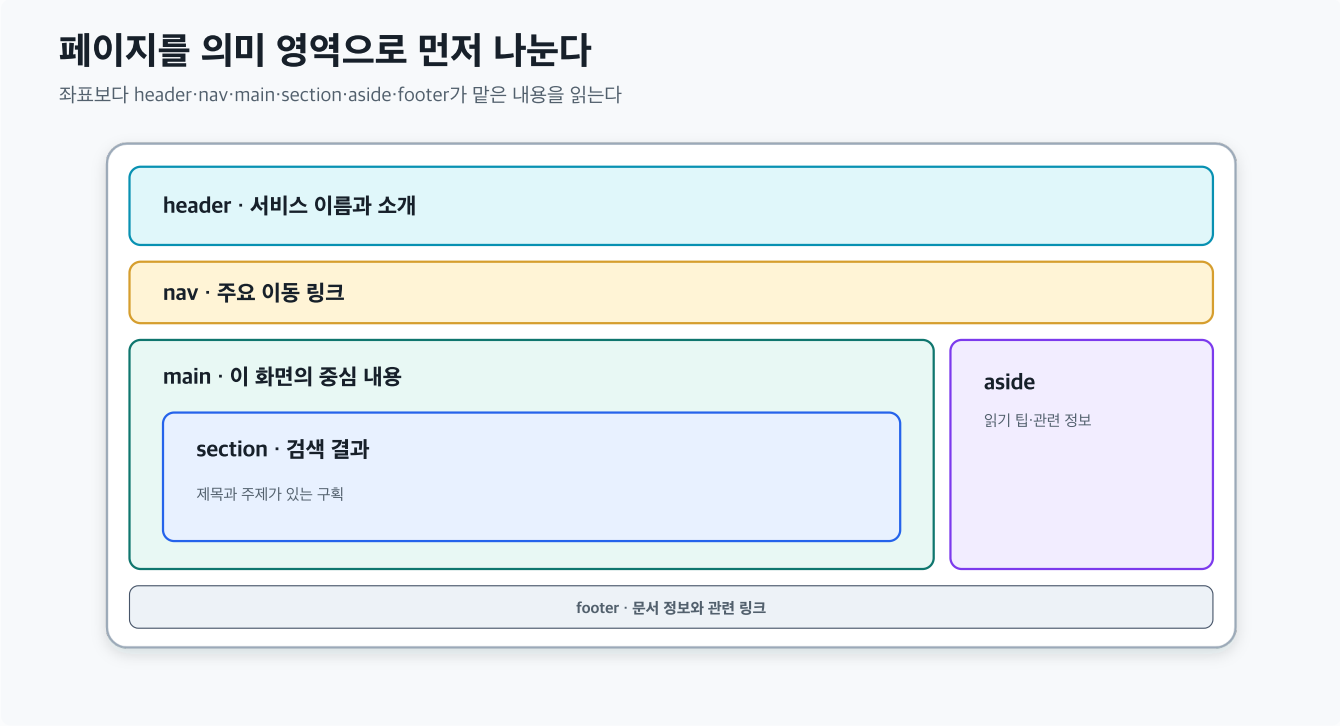
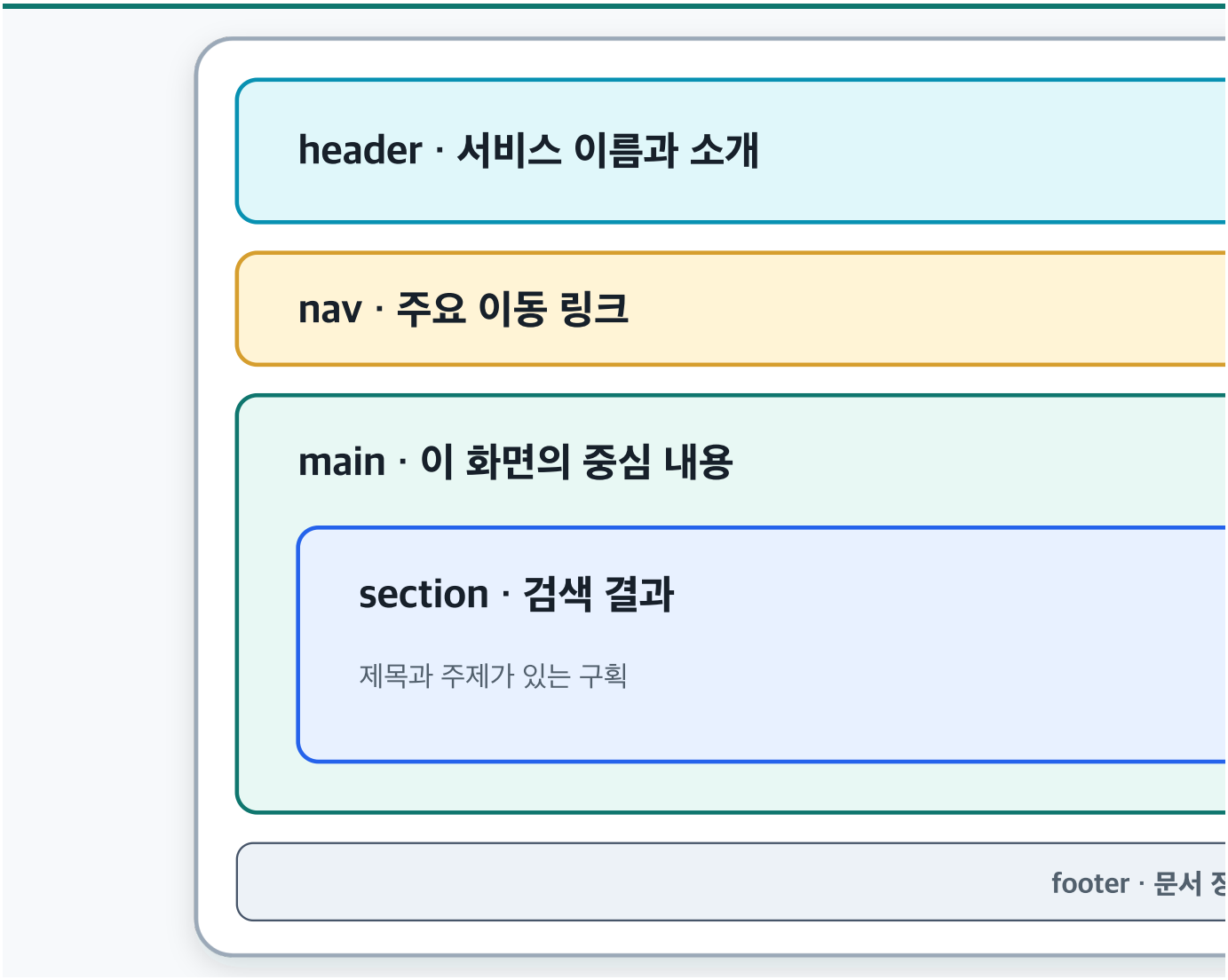


그림 4. 화면의 위·아래·좌·우 좌표 대신 각 영역이 맡은 의미와 내용을 표시합니다.


header · 서비스 이름과 소개
nav · 주요 이동 링크
main · 이 화면의 중심 내용
section · 검색 결과
제목과 주제가 있는 구획
footer · 문서 정


aside
읽기 팁·관련 정보
보와 관련 링크

4.1. `header`

`header`는 문서나 구획의 소개·탐색 보조 묶음입니다. 제목·로고·검색·목차·작성 정보가 들어갈 수 있습니다.

`header`가 항상 페이지 전체의 `banner` landmark가 되는 것은 아닙니다. `article`이나 `section` 안의 `header`는 그 구획의 머리말일 수 있습니다. 가장 가까운 의미 조상을 함께 봅니다.

4.2. `nav`

`nav`는 다른 페이지나 현재 페이지의 주요 부분으로 이동하는 링크 구획입니다. 모든 링크 묶음을 `nav`로 감쌀 필요는 없습니다.

한 페이지에 여러 `nav`가 있으면 목적을 구분합니다.

```
<nav aria-label="주요 메뉴">...</nav>
<nav aria-label="문서 목차">...</nav>
```

4.3. `main`

`main`은 문서의 중심 내용을 나타냅니다. 현재 표시되는 문서에는 보통 하나의 중심 `main`이 있어야 합니다. 단일 페이지 애플리케이션이 여러 `main`을 두면 현재 화면이 아닌 것은 `hidden` 처리해야 합니다.

4.4. `aside`

`aside`는 주변 내용과 간접적으로 관련되며 따로 보아도 되는 보조 내용입니다. 관련 링크·배경 설명·사이드바·보충 팁 등에 사용할 수 있습니다.

오른쪽에 있다는 이유만으로 `aside`가 되지는 않습니다. 중심 작업에 꼭 필요한 입력 폼이라면 `main` 안의 핵심 내용일 수 있습니다.

4.5. `footer`

`footer`는 가장 가까운 구획이나 전체 문서의 꼬리말입니다. 작성자·관련 문서·저작권·변경 정보 등이 들어갈 수 있습니다. 페이지 맨 아래에 보인다는 좌표보다 어느 구획의 꼬리말인지 확인합니다.

4.6. `section`과 `article`

요소	질문	적절한 예
`section`	제목이 붙을 만한 하나의 주제 구획인가	검색 조건·검색 결과·문의 안내

요소	질문	적절한 예
`article`	따로 배포·재사용해도 완결된 내용인가	게시물·상품·뉴스·검색 결과 항목
`div`	의미 없이 스타일·스크립트 묶음만 필요한가	레이아웃 열·간격 래퍼

이름 없는 `section`은 접근성 트리에서 별도 `region`으로 드러나지 않을 수 있습니다. 제목과 `aria-labelledby`로 실제 탐색 가치가 있는 구획만 이름을 제공합니다.

```
<section aria-labelledby="results-heading">
  <h2 id="results-heading">검색 결과 2건</h2>
  ...
</section>
```

영역 판독 순서
이 화면의 중심 내용은 무엇인가 → 주요 이동 구획은 어디인가 → 중심과 별도인 보조 내용은 무엇인가 → 주제별·독립형 내용은 어떻게 묶였는가.

5. 제목 계층은 화면의 목차입니다

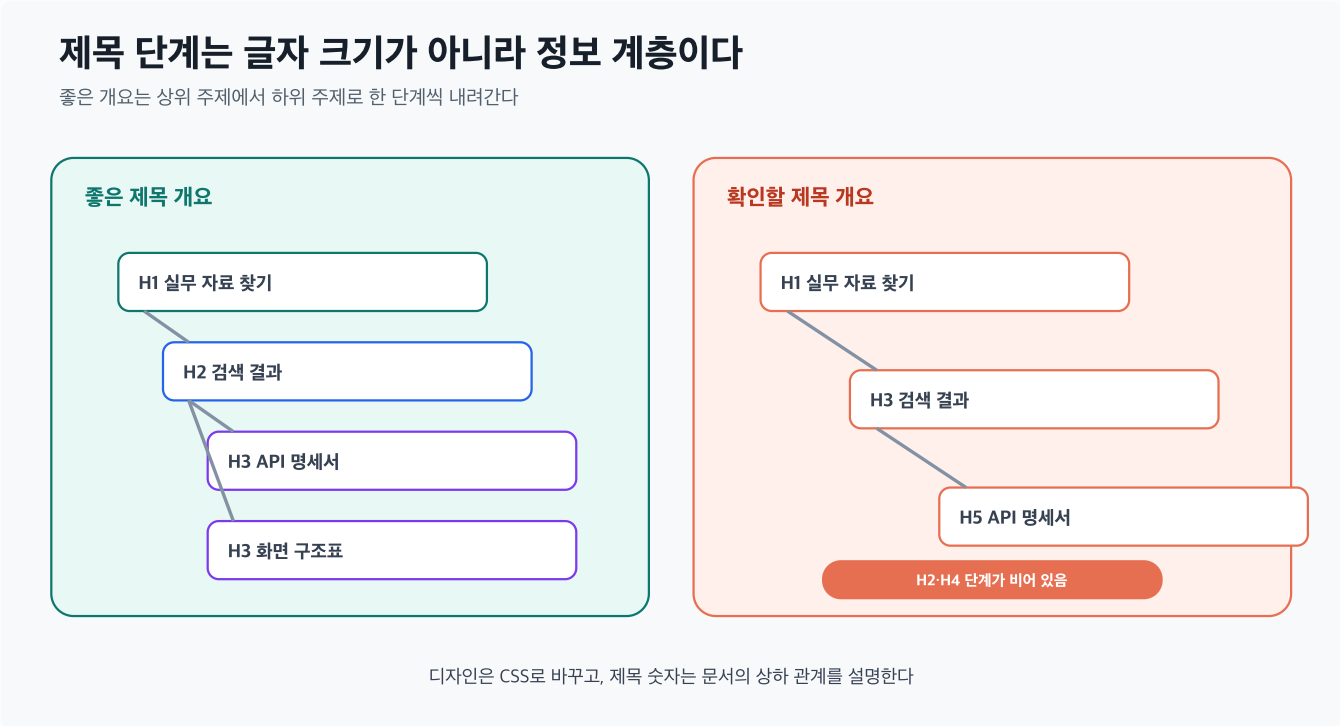
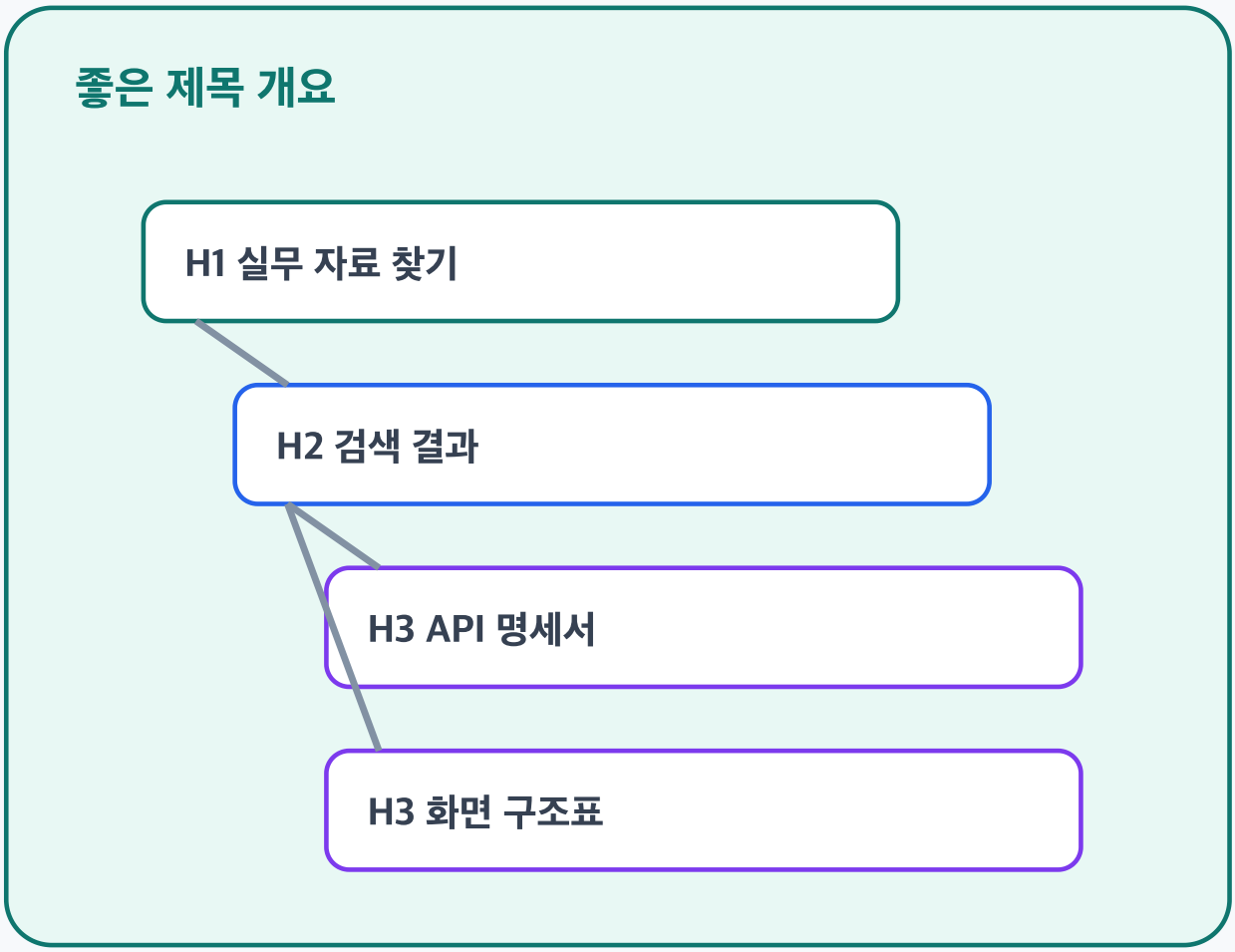
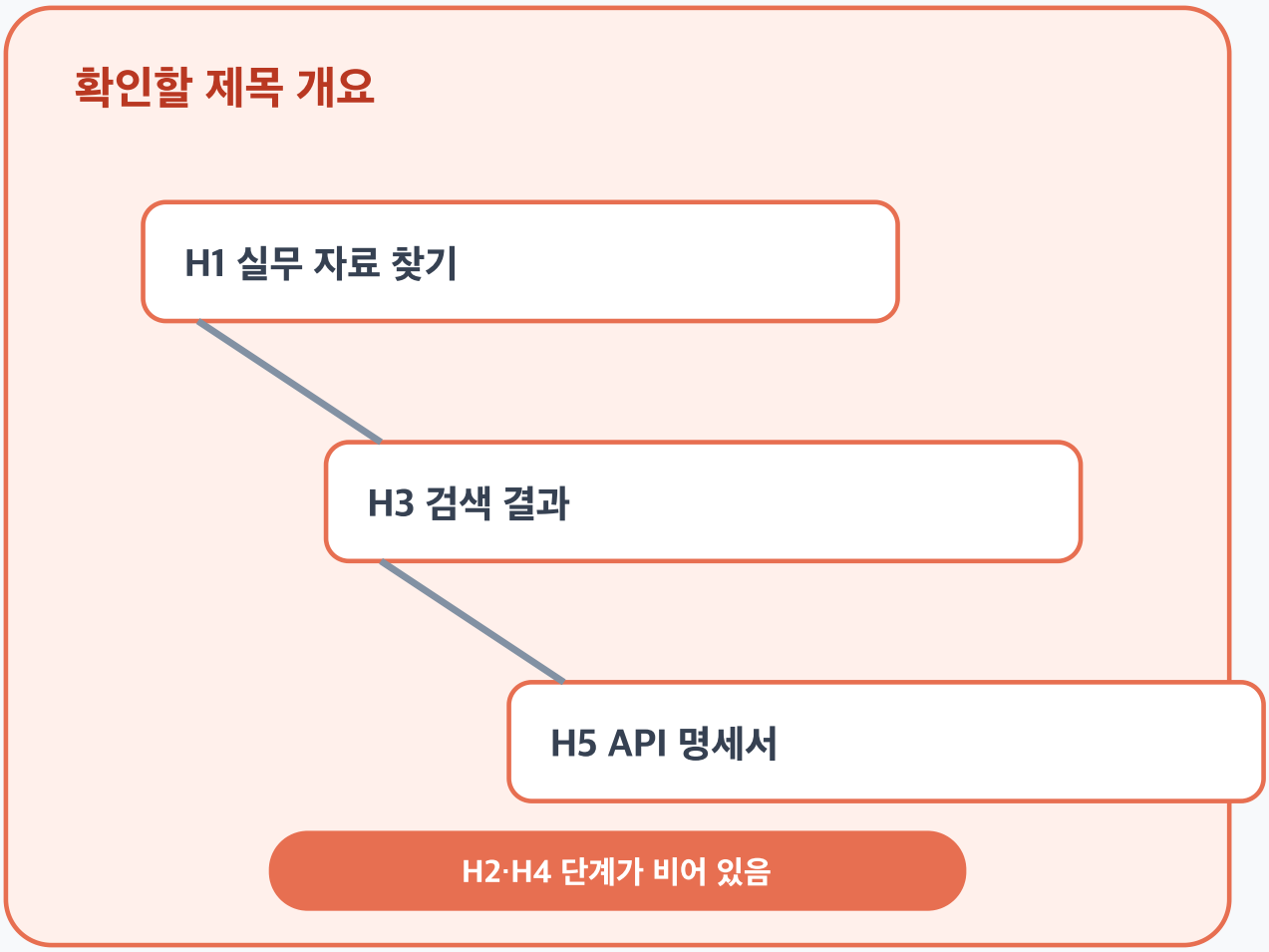


그림 5. 제목은 한 단계씩 내려가며 문서의 상하 관계를 설명해야 합니다.


좋은 제목 개요
H1 실무 자료 찾기
H2 검색 결과
H3 API 명세서
H3 화면 구조표


확인할 제목 개요
H1 실무 자료 찾기
H3 검색 결과
H5 API 명세서
H2·H4 단계가 비어 있음

5.1. `h1`~`h6`은 중요도 순위표가 아닙니다

제목 숫자는 내용의 계층을 나타냅니다.

```
H1 실무 자료 찾기
 ├─ H2 검색 조건
 ├─ H2 검색 결과
 │    ├─ H3 API 명세서 읽기
 │    └─ H3 화면 구조표
 └─ H2 이용 안내
```

`H3`은 글자가 작다는 뜻이 아니라 현재 `H2` 아래의 하위 주제라는 뜻입니다.

5.2. 제목 단계는 가능하면 건너뛰지 않습니다

W3C WAI는 문서 구조를 쉽게 탐색하도록 제목을 적절히 중첩할 것을 권합니다. `h1` 다음에 바로 `h3`가 나오면 빠진 `h2` 주제가 무엇인지 확인합니다.

제목이 새 상위 구획을 시작할 때는 낮은 단계에서 높은 단계로 돌아갈 수 있습니다.

```
H1 → H2 → H3 → H3 → H2     적절한 구조 가능
H1 → H3 → H5                중간 계층 누락 확인
```

5.3. 글자 크기와 제목 단계는 분리합니다

CSS로 `h2`를 작게, `h3`를 크게 보이게 만들 수 있습니다. 화면의 크기만 보고 제목을 추정하지 않고 Elements에서 태그를 확인합니다.

5.4. 명확한 페이지 제목을 둡니다

한 페이지의 중심 목적을 대표하는 명확한 `h1` 하나를 두는 방식은 학습자와 실무 도구에 이해하기 쉬운 기준입니다. 복잡한 문서 모델의 예외보다 현재 화면에서 무엇이 전체 제목인지 분명하게 만드는 일을 우선합니다.

5.5. 제목 글도 구체적이어야 합니다

모호한 제목	더 구체적인 제목
안내	결제 실패 해결 방법
목록	승인 대기 요청 12건

모호한 제목	더 구체적인 제목
정보	개인정보 수집 항목
결과	‘API’ 검색 결과 2건

올바른 `h2` 태그를 사용해도 제목 글이 모호하면 탐색에 도움이 되지 않습니다.

30초 확인

디자인 시안에서 “검색 결과”가 가장 큰 글자입니다. 무조건 `h1`로 요청해야 할까요?

정답 보기
아닙니다. 화면 전체 목적이 “실무 자료 찾기”라면 그것이 `h1`이고, “검색 결과”는 그 아래 구획의 `h2`가 적절할 수 있습니다. 글자 크기는 CSS로 결정합니다.

6. 내용의 관계에 맞는 요소를 고릅니다

그림 6. 화면의 박스 모양보다 내용 사이의 관계를 기준으로 HTML 구조를 선택합니다.

p

문장 묶음

하나의 문단

ul + li

순서 없는 항목

순서가 바뀌어도 됨

dl + dt + dd

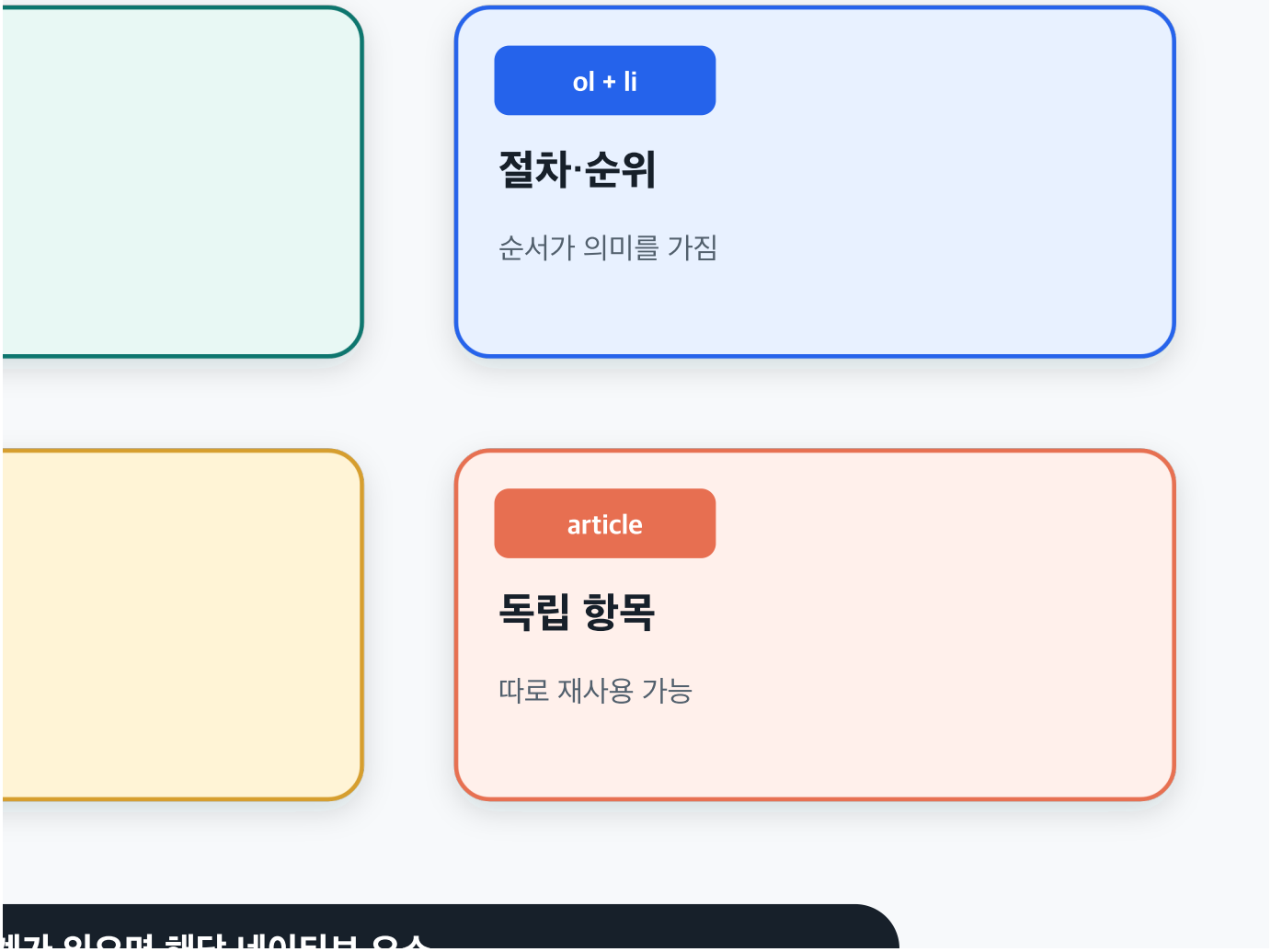
용어와 설명

이름과 풀이의 쌍

table

행·열 데이터

셀 관계가 핵심


ol + li
절차·순위
순서가 의미를 가짐
article
독립 항목
따로 재사용 가능

6.1. 문단 `p`

하나의 생각을 문장 묶음으로 설명할 때 사용합니다. 줄 간격을 만들기 위해 빈 `p`를 추가하지 않습니다. 간격은 CSS에서 다룹니다.

6.2. 순서 없는 목록 `ul`

항목 순서를 바꾸어도 의미가 크게 달라지지 않는 목록입니다.

```
<ul>
  <li>API 명세서 읽기</li>
  <li>화면 구조표</li>
</ul>
```

카드가 세로로 보이더라도 여러 결과가 같은 관계의 반복 항목이면 목록으로 표현할 수 있습니다.

6.3. 순서 있는 목록 `ol`

절차·순위·단계처럼 순서가 의미를 바꾸는 목록입니다.

```
<ol>
  <li>요소 선택</li>
  <li>DOM 위치 확인</li>
  <li>접근성 이름 기록</li>
</ol>
```

6.4. 설명 목록 `dl`

용어와 설명, 이름과 값처럼 짝을 이루는 정보를 나타냅니다.

```
<dl>
  <dt>역할</dt><dd>button</dd>
  <dt>접근성 이름</dt><dd>검색</dd>
</dl>
```

6.5. 데이터 표 `table`

행과 열의 관계를 함께 읽어야 하는 데이터에 사용합니다. 단순한 2열 화면 배치를 만들기 위해 표를 사용하지 않습니다.

확인할 요소:

- 표의 목적을 알려 주는 `caption`
- 행·열 헤더인 `th`
- 헤더와 데이터의 범위인 `scope`
- 작은 화면에서 정보가 사라지지 않는 표현

6.6. 독립 항목 `article`

검색 결과·상품·게시물처럼 하나를 떼어 내도 완결된 항목에 적절합니다. 모든 카드 모양을 `article`로 만들 필요는 없습니다.

6.7. 그림과 설명 `figure` · `figcaption`

도표·사진·코드 예시처럼 본문에서 하나의 단위로 참조하는 내용을 묶습니다. 장식 이미지는 `figure`가 아닐 수 있습니다.

BIG IDEA 02

CSS 박스는 보이는 묶음이고, HTML 요소는 내용의 관계입니다. 둘은 같을 수도 있고 다를 수도 있습니다.

7. 링크·버튼·폼은 결과로 구분합니다

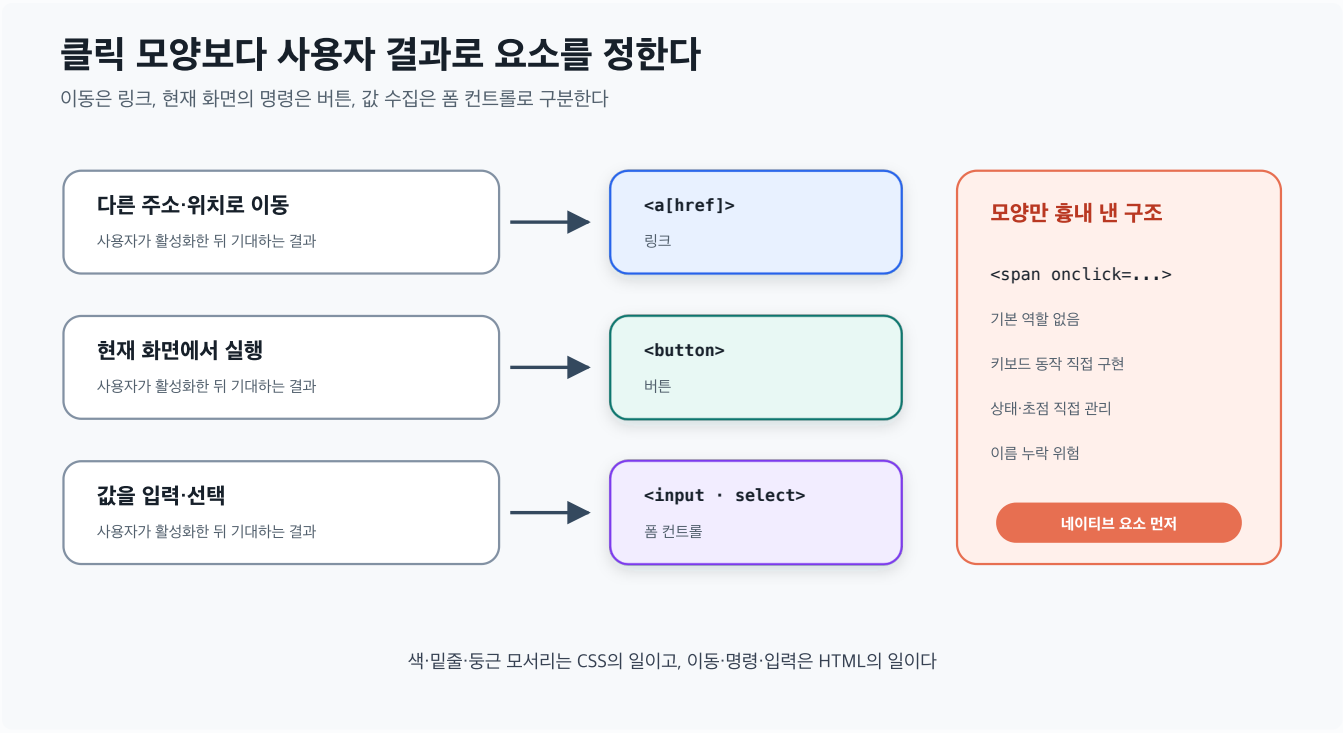


그림 7. 클릭 뒤 사용자가 기대하는 결과에 따라 링크·버튼·폼 컨트롤을 구분합니다.

새.민줄.두근 모서리는 CSS의 일이고

href]>

tton>

put · select>

트롤

모양만 흉내 낸 구조

기본 역할 없음

키보드 동작 직접 구현

상태·초점 직접 관리

이름 누락 위험

네이티브 요소 먼저

이동·명령·입력은 HTML 이 있어야

실행 주의
운영 화면을 구조 분석할 때 삭제·승인·결제·제출 버튼을 시험 삼아 누르지 않습니다. Elements와 Accessibility에서 구조만 확인하거나 별도 시험 환경을 사용합니다.

7.1. 링크는 이동합니다

`a` 요소에 `href`가 있으면 다른 주소나 문서 위치로 이동하는 하이퍼링크입니다.

```
<a href="/manuals/api-spec">자료 열기</a>
```

새 페이지, 상세 화면, 다운로드 주소, 같은 페이지의 다른 구획으로 이동하는 목적에 사용합니다.

7.2. 버튼은 현재 화면에서 명령을 실행합니다

```
<button type="button">필터 열기</button>
<button type="submit">검색</button>
```

버튼은 대화상자 열기, 목록 정렬, 저장, 삭제, 폼 제출처럼 현재 인터페이스에서 동작을 실행합니다.

`button`의 기본 `type`은 폼 안에서 제출로 동작할 수 있습니다. 제출이 목적이 아니면 `type="button"`을 분명히 적습니다.

7.3. 폼 컨트롤은 값을 입력·선택합니다

요소	대표 목적
`input`	짧은 글·검색어·날짜·수치·체크 값
`select`	정해진 선택지 중 선택
`textarea`	여러 줄 글 입력
`button`	제출·초기화·폼 관련 명령

7.4. 보이는 라벨과 컨트롤을 연결합니다

```
<label for="resource-query">검색어</label>
<input id="resource-query" name="query" type="search">
```

`label`의 `for`와 입력의 `id`를 같게 하면 관계를 프로그램으로 판단할 수 있습니다. 라벨을 누르면 입력이 활성화되어 클릭 영역도 넓어집니다.

`placeholder`는 입력 예시나 힌트이며 지속적으로 보이는 라벨을 대신하기 어렵습니다. 사용자가 값을 입력하면 사라지고, 브라우저의 이름 계산에서 마지막 수단으로 쓰일 수 있습니다.

7.5. `div` · `span`으로 흉내 내면 기본 기능을 잃습니다

```
<span onclick="search()">검색</span>
```

이 요소는 모양을 버튼처럼 꾸밀 수 있지만 기본 버튼 역할·키보드 활성화·초점·사용 불가 상태·폼 동작이 없습니다. 이를 모두 스크립트와 ARIA로 다시 구현하기 전에 네이티브 `button`을 먼저 사용합니다.

8. 이름·역할·상태·값을 확인합니다

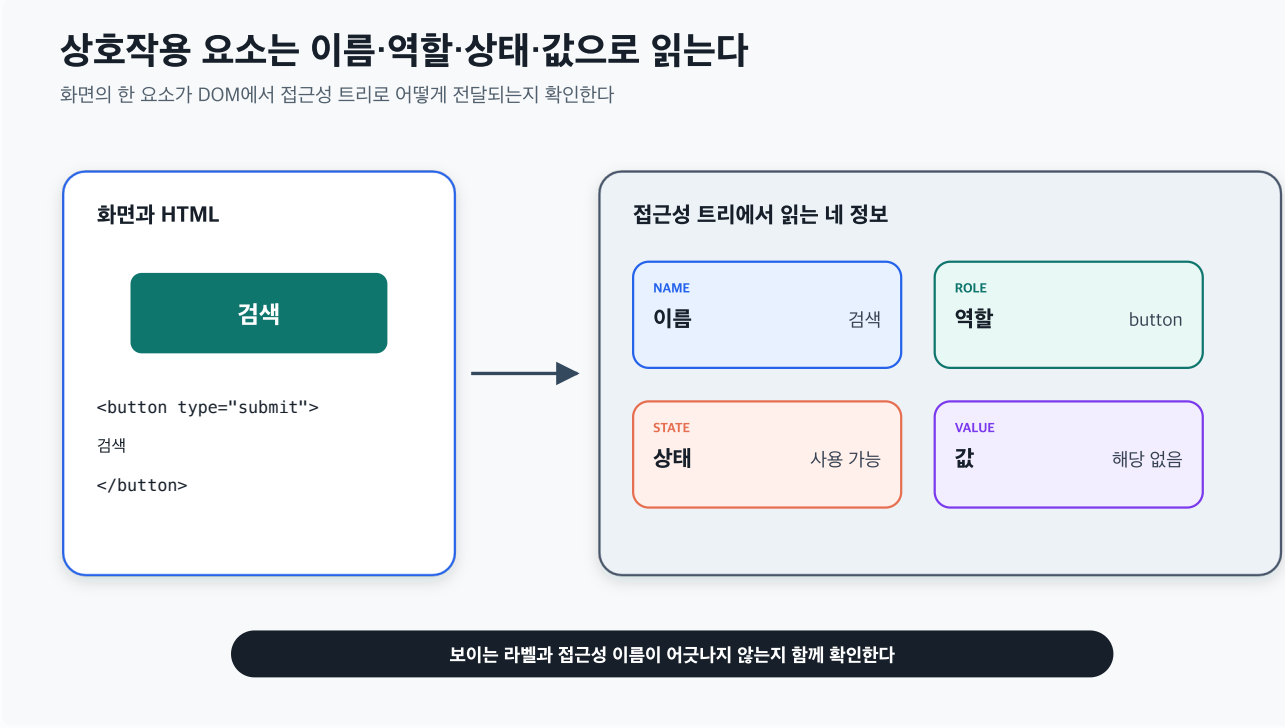
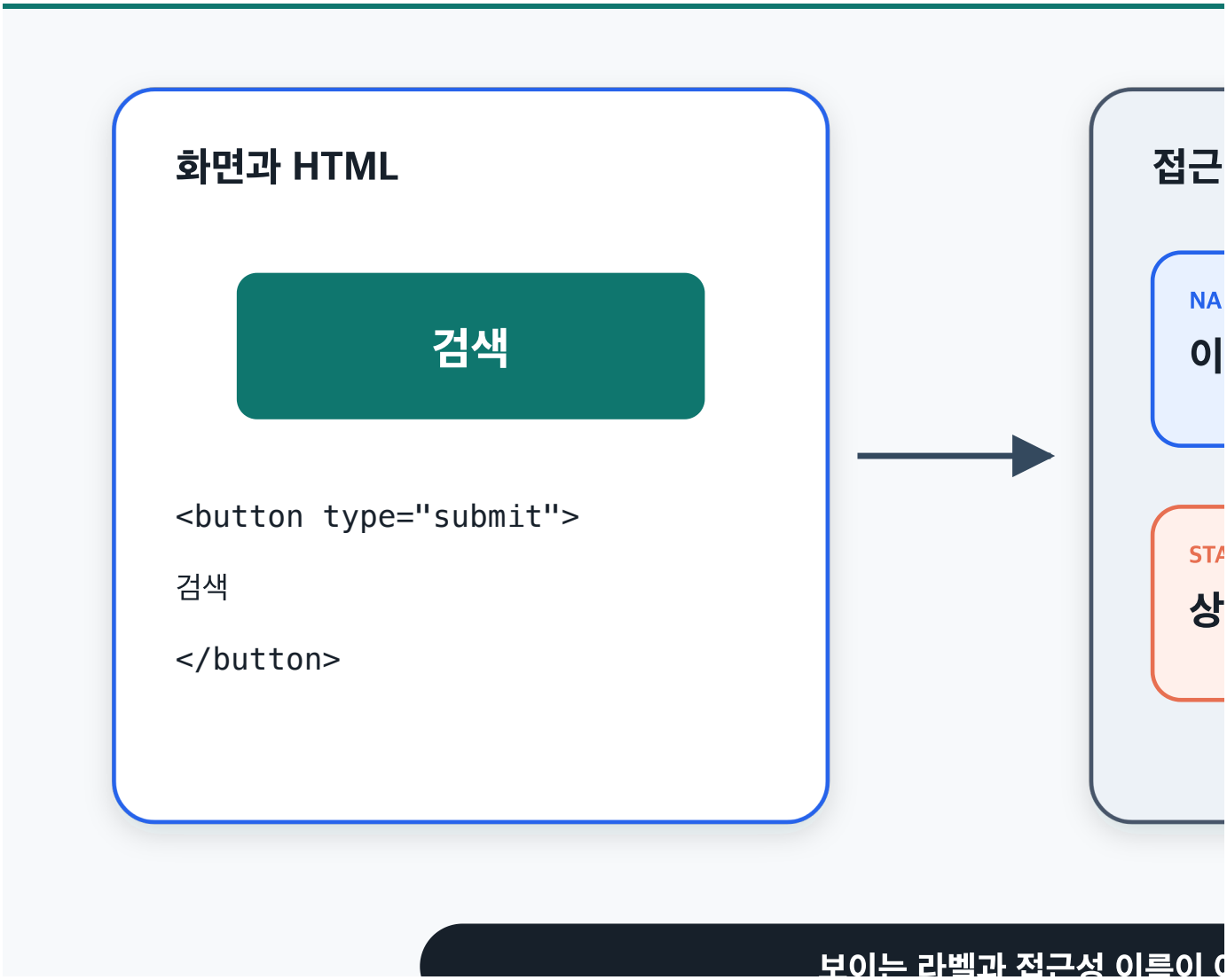


그림 8. 상호작용 요소는 보이는 글과 함께 접근성 이름·역할·상태·값으로 전달됩니다.


화면과 HTML
검색
<button type="submit">
검색
</button>
접근
NA
이
STA
상
보이는 라벨과 접근성 이름이 C

어군나지 않는지 함께 확인하다

8.1. 접근성 이름

접근성 이름(accessible name)은 보조 기술이 요소를 구별하고 목적을 전달하는 짧은 이름입니다.

```
검색어  searchbox
검색    button
자료 열기  link
```

이름은 요소에 따라 다음에서 올 수 있습니다.

- 연결된 `label`
- 버튼·링크 안의 보이는 글
- `aria-labelledby`가 가리키는 글
- `aria-label`
- 이미지의 `alt`
- 기타 브라우저의 대체 계산

정확한 계산 규칙은 복잡합니다. Chrome Accessibility의 Computed Properties에서 최종 이름과 출처를 확인합니다.

8.2. 역할

역할(role)은 요소가 링크·버튼·제목·탐색·검색 상자 중 무엇인지 알려 줍니다. 네이티브 HTML은 기본 역할과 키보드 동작을 함께 제공합니다.

```
<a href="/guide">      → link
<button>검색</button>   → button
<input type="search"> → searchbox
<nav>                  → navigation
```

8.3. 상태와 속성

상호작용 요소는 현재 조건을 함께 전달해야 합니다.

상태·속성	질문	예
disabled	사용할 수 있는가	저장 버튼 사용 불가
expanded	펼쳐졌는가	도움말 닫힘·열림
selected	선택됐는가	현재 탭 선택됨

상태·속성	질문	예
checked	체크됐는가	알림 수신 켬
required	반드시 입력해야 하는가	이메일 필수
invalid	현재 값이 유효한가	형식 오류

화면의 아이콘·색만 바꾸고 접근성 상태를 갱신하지 않으면 보조 기술에는 이전 상태가 남을 수 있습니다.

8.4. 값

입력·선택·범위 요소는 현재 값을 전달합니다. 기획자는 기본값·빈 값·허용 범위·오류 값·변경 뒤 상태를 정의합니다.

8.5. 보이는 글을 이름의 기준으로 우선합니다

W3C WAI는 가능하면 보이는 글과 네이티브 HTML 이름 방식을 우선하도록 권합니다. 보이지 않는 `aria-label`로 보이는 글을 덮으면 두 이름이 어긋나거나 번역에서 누락될 수 있습니다.

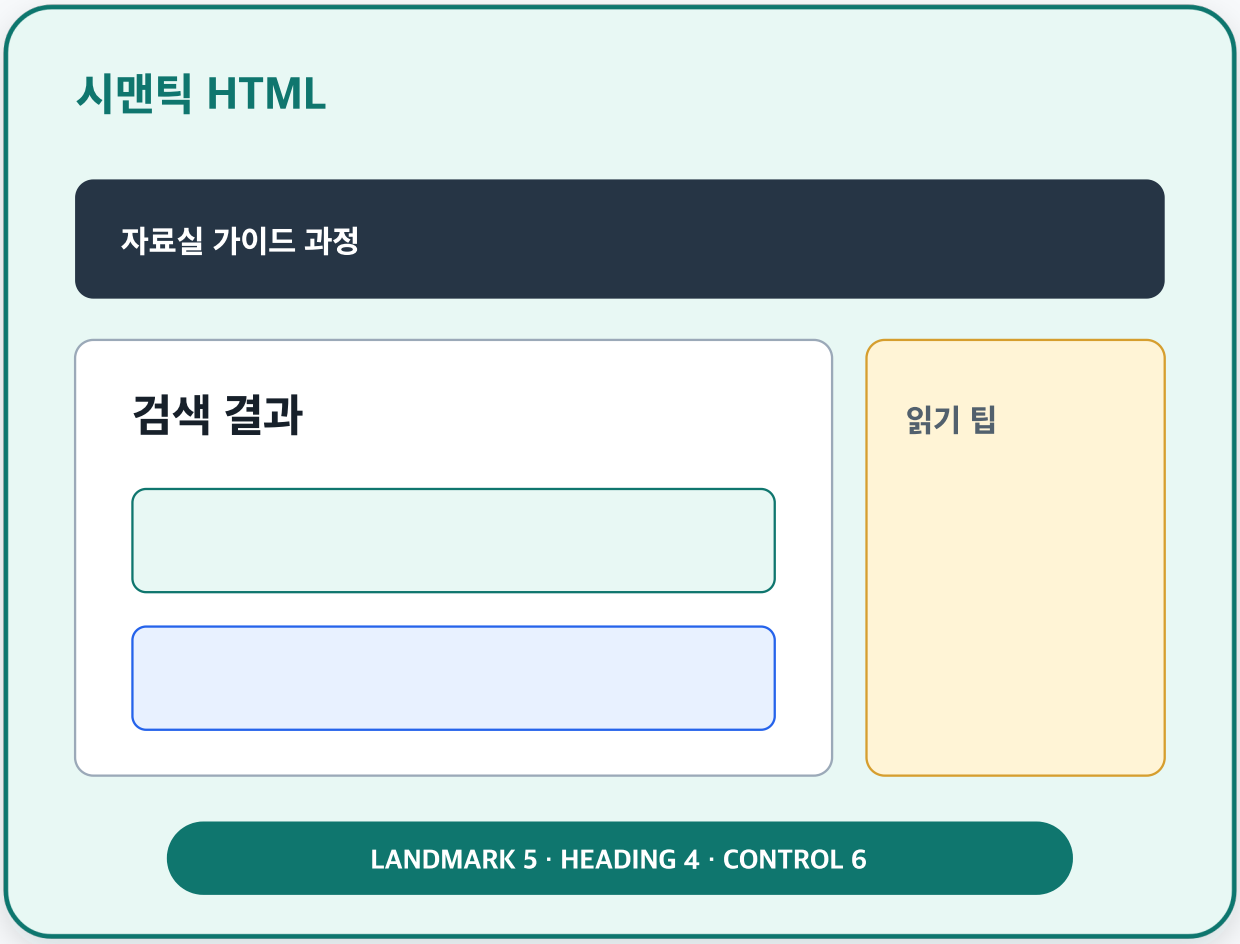
```
<!-- 일치 -->
<button>검색</button>

<!-- 검토 필요: 화면에는 검색, 보조 기술에는 실행 -->
<button aria-label="실행">검색</button>
```

BIG IDEA 03

ARIA는 HTML의 의미와 동작을 보완하지만, 잘못 고른 요소의 모든 기본 기능을 자동으로 만들어 주지는 않습니다.

9. 같은 모양과 같은 의미를 구분합니다

그림 9. CSS가 같은 화면을 그려도 HTML 의미 구조와 키보드·보조 기술 결과는 달라질 수 있습니다.


시맨틱 HTML
자료실 가이드 과정
검색 결과
읽기 팁
LANDMARK 5 · HEADING 4 · CONTROL 6


div 묶음
자료실 가이드 과정
검색 결과
읽기 팁
LANDMARK 0 · HEADING 0 · CONTROL 0

9.1. 스크린샷으로 알 수 있는 것

- 보이는 글과 아이콘
- 상대적 배치와 강조
- 현재 화면 상태의 일부
- 클릭할 것처럼 보이는 시각 단서

9.2. 스크린샷만으로 확정할 수 없는 것

- 실제 태그와 DOM 관계
- 제목 단계
- 링크의 `href`
- 버튼의 `type`
- 폼 라벨 연결
- 접근성 이름·역할·상태
- 키보드 초점 순서
- 숨김·동적 노드

9.3. `div` 자체가 나쁜 것은 아닙니다

`div`는 의미 없는 일반 컨테이너가 필요할 때 적절합니다. 문제는 제목·목록·버튼·탐색처럼 이미 알맞은 HTML 요소가 있는데도 모양만 `div`로 흉내 내는 것입니다.

9.4. 구조 품질은 자동 점수 하나로 끝나지 않습니다

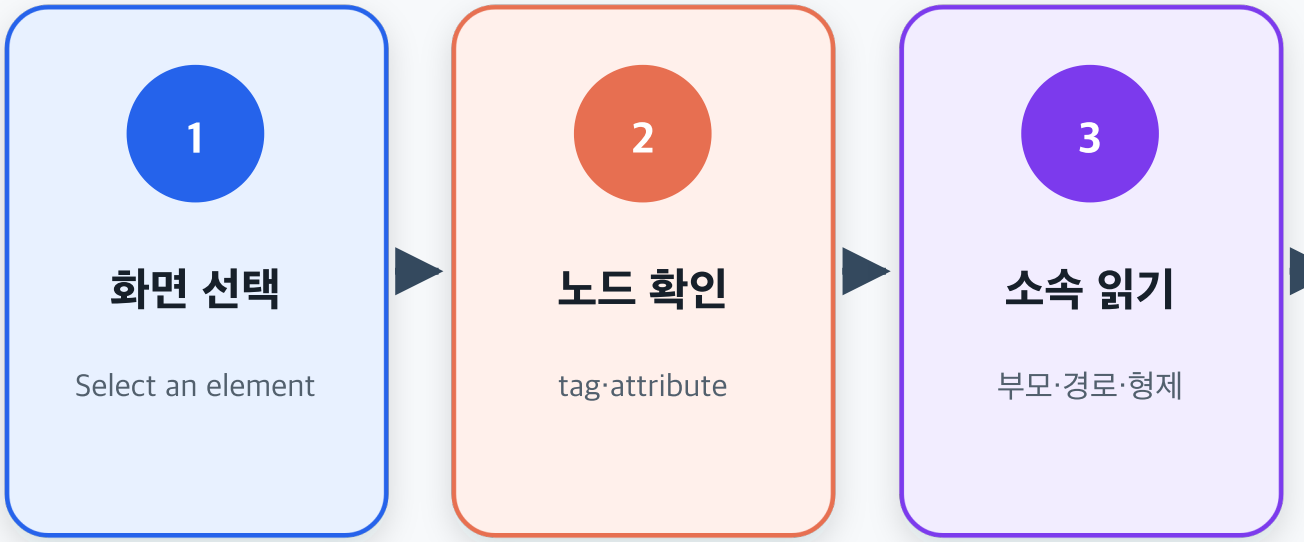
자동 도구는 이름 누락·잘못된 속성·일부 계층 문제를 찾는 데 유용합니다. 그러나 화면의 업무 목적, 제목 글의 적절성, 링크와 버튼의 의도, 상태 변화의 이해 가능성은 사람이 확인해야 합니다.

검증 층:

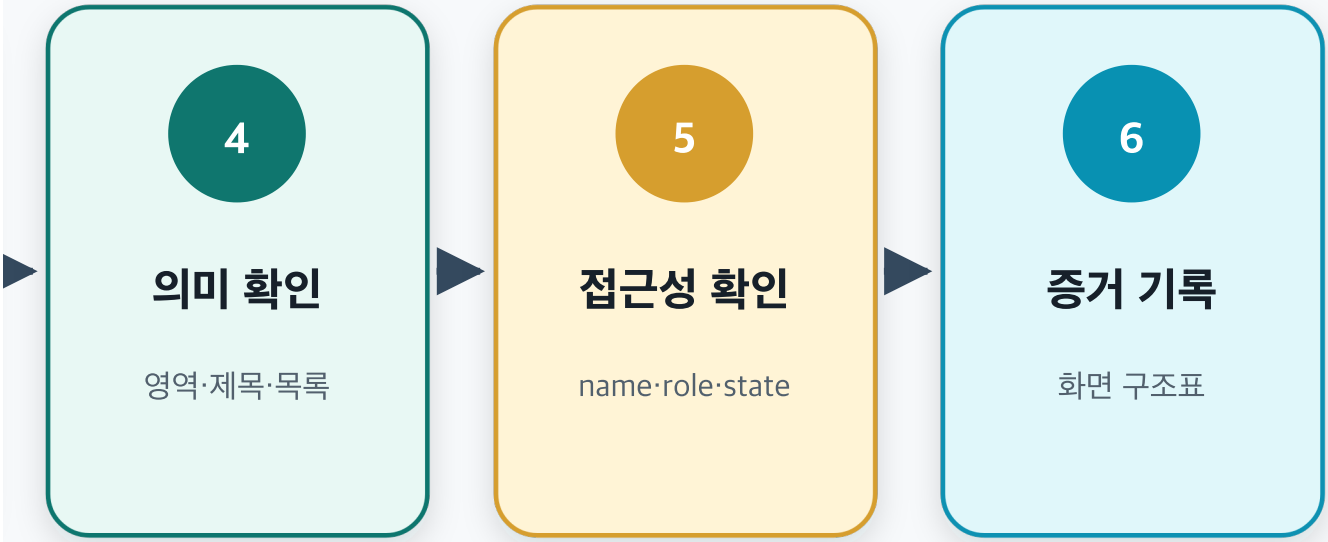
1. HTML·DOM 구조 검토
2. 브라우저 접근성 트리 확인
3. 키보드만으로 사용
4. 자동 접근성 점검
5. 실제 보조 기술과 사용자 시험

10. Chrome Elements로 구조를 확인합니다

그림 10. 화면 선택에서 DOM·접근성 증거와 화면 구조표까지 여섯 단계로 왕복합니다.



기록 단위 화면 상태 · 보이는 글 · 태그 · 역할 · (



이름 · DOM 경로 · 사용자 영향

10.1. 화면 요소 선택

1. Chrome 개발자 도구를 엽니다.
2. **Select an element**를 누릅니다.
3. 화면에서 분석할 제목·버튼·입력을 누릅니다.
4. Elements에서 강조된 노드를 확인합니다.

단축키:

- macOS: `Command` + `Option` + `C`
- Windows·Linux·ChromeOS: `Control` + `Shift` + `C`

10.2. 노드와 속성 확인

```
<input id="resource-query" name="query" type="search">
```

확인 항목:

- 태그 이름
- `id` · `class` · `name`
- `href` · `type` · `value`
- `aria-*` · `role`
- 숨김·사용 불가·필수 상태

10.3. 소속과 경로 확인

DOM 트리의 들여쓰기와 하단 breadcrumb로 부모·조상을 읽습니다. `main` · `nav` · `form` · `article` 같은 가장 가까운 의미 조상을 찾습니다.

10.4. Console의 `$0` 사용

Elements에서 현재 선택한 노드는 Console에서 `$0`으로 참조할 수 있습니다.

```
$0.tagName
$0.textContent.trim()
$0.parentElement
$0.children
$0.closest('main, nav, header, footer, aside, section, article, form')
```

DOM을 바꾸는 코드는 운영 화면에서 실행하지 않습니다. 조회 결과만 사용합니다.

10.5. Accessibility 확인

Elements에서 노드를 선택하고 Accessibility 탭의 Computed Properties를 봅니다.

확인	기록 예
Role	searchbox
Name	검색어
Focusable	true
Disabled	false
Value	API
Name source	label `for="resource-query"`

접근성 트리는 DOM의 관련 노드만 보여 줍니다. DOM 트리와 접근성 트리를 전환하며 같은 선택 요소가 어떻게 매핑되는지 비교합니다.

10.6. Elements 수정은 저장이 아닙니다

Elements에서 글·태그·속성을 임시로 바꿀 수 있습니다. 이 변경은 현재 브라우저 세션의 DOM에만 적용되며 새로 고치면 사라집니다.

임시 수정은 다음 질문을 확인할 때 사용합니다.

- 이 요소가 `h2`이면 개요가 나아지는가
- 라벨을 연결하면 접근성 이름이 생기는가
- `button`으로 바꾸면 기본 키보드 동작이 생기는가

검증 뒤에는 원본 코드·컴포넌트·요구사항의 수정 요청으로 남깁니다.

11. 구조 분석에서 자주 하는 오해

오해	왜 틀렸는가	바꿀 질문
큰 글자는 모두 `h1` 이다	CSS 크기와 정보 계층은 다름	이 제목의 상위·하위 주제는 무엇인가
화면 위쪽은 모두 `header` 다	`header` 는 좌표가 아니라 소개·탐색 묶음	어느 문서·구획의 머리말인가
오른쪽 박스는 모두 `aside` 다	중심 작업일 수도 있음	주변 내용과 분리해도 되는가
클릭되면 모두 버튼이다	이동과 현재 동작은 다름	실행 뒤 주소가 바뀌는가, 화면 상태가 바뀌는가
`placeholder` 가 라벨이다	입력하면 사라지고 이름 품질이 낮음	지속적으로 보이는 라벨과 연결됐는가
ARIA를 추가하면 접근성이 해결된다	역할만으로 키보드·초점·상태 동작이 생기지 않음	네이티브 HTML로 먼저 구현할 수 있는가
페이지 소스가 현재 화면 구조다	JavaScript 뒤 DOM이 달라질 수 있음	Elements의 현재 DOM은 무엇인가
자동 검사 100점이면 완료다	업무 의도와 실제 사용성은 자동 판정 한계가 있음	키보드·보조 기술·사용자 시험 결과는 무엇인가

오류 1 · 제목처럼 보이는 `div`

```
<div class="page-title">실무 자료 찾기</div>
```

화면에서는 제목처럼 보여도 제목 탐색에 잡히지 않습니다. 정보 계층에 따라 `h1` ~`h6`을 사용합니다.

오류 2 · `button` 안의 아이콘만 있음

```
<button><svg aria-hidden="true">...</svg></button>
```

아이콘이 장식으로 숨겨졌고 다른 글이 없으면 버튼 이름이 비어 있을 수 있습니다. 보이는 글을 제공하거나 꼭 필요한 아이콘 버튼에 간결한 이름을 제공합니다.

```
<button aria-label="검색"><svg aria-hidden="true">...</svg></button>
```

오류 3 · CSS 순서로만 재배치

화면에서는 제목 → 입력 → 버튼 순서지만 DOM과 키보드 초점은 버튼 → 입력 → 제목일 수 있습니다. 의미 있는 읽기 순서를 DOM에 먼저 두고 CSS로 배치합니다.

오류 4 · 이름 없는 여러 탐색 영역

페이지에 `nav`가 여러 개지만 모두 같은 “탐색”으로만 전달되면 구분이 어렵습니다. 주요 메뉴·문서 목차·관련 자료처럼 목적을 이름으로 구분합니다.

12. 화면 구조표는 일곱 증거를 묶습니다

화면 구조표의 한 행은 다음을 연결합니다.

증거	질문	예
화면 상태	언제 보였는가	검색 완료·결과 2건
보이는 글	사용자가 무엇을 읽는가	자료 열기
HTML	어떤 요소인가	`a[href]`
역할	무엇으로 전달되는가	link
이름	어떻게 구별되는가	자료 열기
DOM 위치	어디에 속하는가	`main > section > article > a`
사용자 결과	실행 뒤 무엇이 일어나는가	상세 자료로 이동

기획자가 결정할 사항

1. 화면의 중심 목적과 `main`
2. 전체 제목과 하위 제목 개요
3. 반복 항목의 목록·독립 항목 관계
4. 이동·명령·입력의 구분
5. 기본·로딩·빈 결과·오류·완료 상태의 구조 변화

6. 상호작용 요소의 보이는 라벨·이름·상태·값
7. 키보드 초점과 오류 안내 위치

개발자에게 확인할 질문

- 이 요소는 초기 HTML에 있는가, JavaScript가 나중에 만드는가
- 같은 컴포넌트가 어느 화면에서 반복되는가
- 목록 항목과 제목 단계는 데이터가 늘어도 유지되는가
- 링크의 실제 `href`와 버튼의 `type`은 무엇인가
- 입력 라벨·오류 메시지·도움말은 어떻게 연결되는가
- 로딩·빈 결과·오류가 DOM과 접근성 트리에 어떻게 추가되는가
- 화면 배치 순서와 DOM·초점 순서가 같은가

전달용 문장
"결과 카드가 이상합니다" 대신 "검색 완료 상태에서 `main > section[aria-labelledby=results-heading] > ul > li > article` 구조이며, 두 '자료 열기' 요소는 이동 목적이지만 현재 `span`이라 link 역할과 키보드 초점이 없습니다"라고 기록합니다.

13. 실습 · 같은 화면의 다른 구조 찾기

연결 파일: L03-01 화면을 HTML·DOM 구조로 읽기

탐색기: L03-01 화면 구조 탐색기

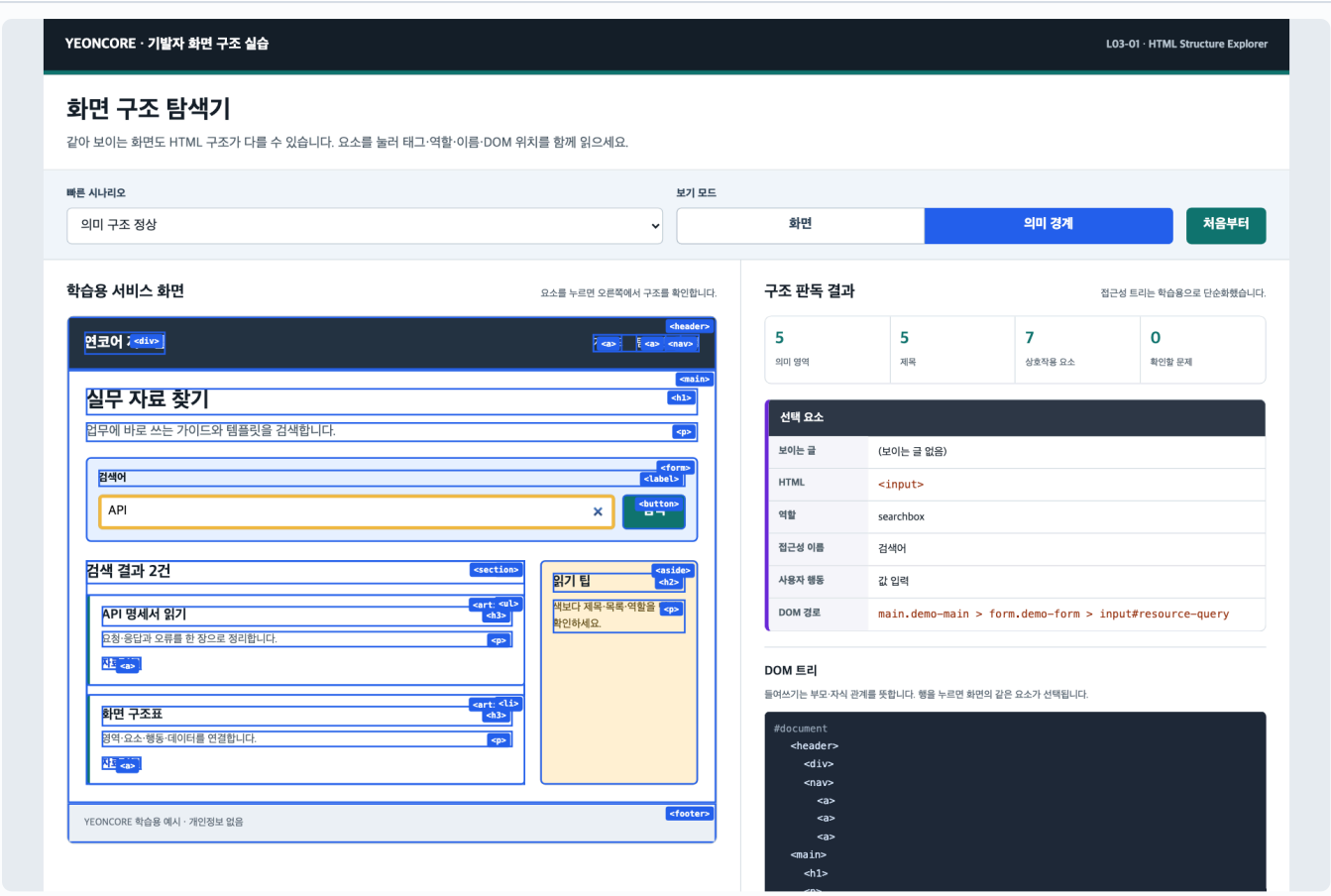


그림 11. 탐색기는 실제 화면 요소와 단순화한 DOM 트리·접근성 정보를 같은 화면에서 연결합니다.

실습 A · 다섯 시나리오 비교

시나리오	예상 확인
의미 구조 정상	의미 영역 5·제목 5·상호작용 7·즉시 문제 0
같은 화면· `div` 만 사용	중심·탐색·제목·기본 상호작용 의미 없음
제목 단계 건너뜀	`h1` 다음 단계 누락 경고
검색 입력·아이콘 이름 없음	입력과 버튼의 판별 가능한 이름 누락
링크와 버튼의 목적 뒤바뀜	이동을 버튼으로, 현재 동작을 링크로 표시

실습 B · 현재 DOM과 초기 HTML 비교

탐색기의 초기 파일에는 `#sample` 이 비어 있고 JavaScript가 현재 시나리오의 노드를 만듭니다. 페이지 소스와 Elements를 비교해 차이를 기록합니다.

실습 C · Elements와 Accessibility

요소 선택 모드로 검색 입력을 잡고 다음을 확인합니다.

- HTML: `input`
- 역할: `searchbox`
- 이름: `검색어`
- 행동: 값 입력
- DOM 경로: `main.demo-main > form.demo-form > input#resource-query`

실습 D · 화면 구조표

T03-01 화면 구조표에 의미 영역·제목 개요·상호작용 요소·문제와 개선 요청을 작성합니다.

완료 산출물

- 화면 구조표 1개
- DOM 구조도 1개
- 제목 개요 1개
- 상호작용 요소 8행 이상
- 구조 개선 요청 3건 이상

14. 인쇄용 화면 구조 학습지

A. 화면 목적과 상태

항목	기록
화면 이름	
사용자	
중심 작업	
현재 상태	기본·로딩·빈 결과·오류·완료
분석 시각	

B. 의미 영역

순서	보이는 영역	HTML·역할	구분 이름	핵심 내용
1				
2				
3				
4				
5				

C. 제목 개요

```
H1 _____________________________________
 ├ H2 _____________________________________
 │    ├ H3 _____________________________________
 │    └ H3 _____________________________________
 └ H2 _____________________________________
```

D. 상호작용

보이는 글	목적	HTML	역할	이름	상태·값	판정
	이동·동작·입력					

E. DOM 증거

```
선택 요소: _____________________________________________
부모: _____________________________________________
주요 자식: _____________________________________________
가장 가까운 의미 조상: _____________________________
DOM 경로: _____________________________________________
```

F. 개선 요청

문제	사용자 영향	근거	개선	완료 기준

15. 셀프 테스트

문제 1 · 네 겹

화면·HTML·DOM·접근성 트리가 각각 대답하는 질문을 한 문장씩 쓰세요.

문제 2 · 현재 구조

JavaScript로 목록 항목을 추가한 뒤 페이지 소스와 Elements가 다릅니다. 현재 화면의 목록 구조는 어디를 기준으로 분석해야 합니까?

문제 3 · DOM 관계

`main > section > ul > li > article > a`에서 `article`의 부모·자식·조상을 하나씩 쓰세요.

문제 4 · 의미 영역

화면 오른쪽에 있는 결제 입력 폼을 위치만 보고 `aside`로 판단해도 됩니까?

문제 5 · 제목

`h1 → h3 → h5` 순서에서 가장 먼저 확인할 문제는 무엇입니까?

문제 6 · 목록

검색 결과 카드 12개가 같은 관계로 반복됩니다. 화면의 `div` 12개 외에 검토할 HTML 구조는 무엇입니까?

문제 7 · 링크와 버튼

“자료 열기”는 상세 페이지로 이동하고 “필터 열기”는 현재 화면의 패널을 엽니다. 각각 어떤 기본 요소가 적절합니까?

문제 8 · 폼 라벨

`<label for="email">이메일</label>`과 연결할 입력에 반드시 맞아야 하는 속성과 값은 무엇입니까?

문제 9 · 접근성 이름

화면에는 “검색”이라고 쓰였지만 버튼에 `aria-label="실행"`이 있습니다. 어떤 문제를 확인해야 합니까?

문제 10 · 구조 설명

다음 관찰을 개발자에게 전달할 한 문장으로 바꾸세요.

```
화면: 자료 검색 완료
보이는 글: 자료 열기
현재 HTML: span class="link"
사용자 목적: 상세 페이지 이동
키보드 Tab: 초점 없음
DOM: main > section > div.result > span.link
```

16. 정답과 해설

문제 1

- 화면: 사용자에게 무엇이 보이는가
- HTML: 내용이 무엇을 뜻하도록 표시됐는가
- DOM: 분석 시점에 노드가 어떤 부모·자식 관계로 연결됐는가
- 접근성 트리: 보조 기술에 어떤 이름·역할·상태·값으로 전달되는가

문제 2

Elements의 현재 DOM을 기준으로 분석합니다. 페이지 소스는 초기 HTML이고 JavaScript 변경을 반영하지 않을 수 있습니다. 화면 상태와 확인 시각도 함께 기록합니다.

문제 3

- 부모: `li`
- 자식: `a`
- 조상 예: `li`, `ul`, `section`, `main`

바로 위는 부모이고 위로 이어지는 모든 요소는 조상입니다.

문제 4

안 됩니다. `aside`는 좌우 위치가 아니라 중심 내용과 간접적으로 관련된 별도 내용입니다. 결제가 화면의 중심 작업이면 `main` 안의 핵심 폼이 적절할 수 있습니다.

문제 5

`h2`와 `h4`에 해당하는 정보 계층이 빠졌는지 확인합니다. 디자인 크기가 아니라 상위·하위 주제 관계로 제목 단계를 다시 정합니다.

문제 6

검색 결과 전체는 `ul`과 각 `li`의 목록 구조를 검토합니다. 각 결과가 따로 재사용해도 완결된 내용이면 `li` 안의 `article`도 검토할 수 있습니다.

문제 7

상세 페이지로 이동하는 “자료 열기”는 `a[href]`, 현재 화면에서 패널을 여는 “필터 열기”는 `button type="button"`이 적절합니다.

문제 8

입력에 `id="email"`이 있어야 합니다. `label`의 `for` 값과 컨트롤의 `id` 값이 일치해야 명시적으로 연결됩니다.

문제 9

보이는 라벨 “검색”과 접근성 이름 “실행”이 어긋납니다. 음성 입력 사용자와 보조 기술 사용자가 같은 요소를 다른 이름으로 인식할 수 있습니다. 보이는 글을 이름으로 사용하거나 최소한 이름이 보이는 라벨을 포함하도록 수정합니다.

문제 10

예시 답안:

```
자료 검색 완료 상태의 “자료 열기”는 상세 페이지 이동 목적이지만 현재
main > section > div.result > span.link 구조라 link 역할과 Tab 초점이 없습니다.
실제 목적지 href가 있는 a 요소로 변경하고 키보드 이동과 접근성 이름이
“자료 열기”로 확인되는 것을 완료 기준으로 제안합니다.
```

17. 한 장 요약

그림 12. 목적·영역·계층·행동·접근성·DOM 증거의 여섯 질문으로 화면 구조를 복습합니다.

1 **목적**

이 화면의 중심 작업은?

2 **영역**

header·nav·main은?

4 **행동**

이동·동작·입력 중?

5 **전달**

이름·역할·상태는?

최종 설명 = 화면 목적 + 의미 영역 + 정

3 **계층**

제목·목록 관계는?

6 **증거**

현재 DOM 경로는?

보 계층 + 행동 + 접근성 + DOM 증거

단계	질문	산출물
1	이 화면의 중심 작업은 무엇인가	화면 목적·상태 한 문장
2	어떤 의미 영역으로 나뉘는가	`header` · `nav` · `main` · `aside` · `footer` 지도
3	제목·목록·표·독립 항목의 관계는 무엇인가	제목 개요·내용 구조표
4	사용자는 이동·동작·입력 중 무엇을 하는가	링크·버튼·폼 구분
5	이름·역할·상태·값은 어떻게 전달되는가	접근성 판독 기록
6	현재 DOM에서 어디에 있는가	부모·자식·경로 증거

처음에는 태그를 모두 외우지 않아도 됩니다. **목적 → 영역 → 계층 → 행동 → 전달 → 증거**의 순서를 지키면 필요한 태그와 질문이 따라옵니다.

18. 다음 학습과 출처

18.1. 다음 매뉴얼

M03-02 「CSS 레이아웃과 반응형 판단하기」에서는 HTML 구조가 화면의 크기와 배치에 따라 어떻게 그려지는지 확인합니다. 박스 모델·Flexbox·Grid·중단점·넘침을 구조와 연결해 반응형 점검표를 만듭니다.

18.2. 함께 사용할 자료

- L03-01 화면을 HTML·DOM 구조로 읽기
- L03-01 화면 구조 탐색기
- T03-01 화면 구조표
- HTML·DOM 화면 구조 용어집

18.3. 출처와 확인일

아래 공식 자료는 모두 2026. 7. 15.에 확인했습니다.

- WHATWG HTML Living Standard · Sections: `article` · `section` · `nav` · `aside` ·제목· `header` · `footer`
- WHATWG HTML Living Standard · Grouping Content: 문단·목록·설명 목록·그림· `main` · `div`
- WHATWG HTML Living Standard · Forms: 폼·라벨·입력·버튼

- WHATWG HTML Living Standard · Text-level Semantics: `a` 링크
- WHATWG DOM Living Standard: 노드 트리와 DOM 관계
- W3C WAI Page Structure Tutorial: 의미 영역·제목·내용 구조
- W3C WAI Labeling Controls: 폼 컨트롤과 라벨 연결
- W3C WAI WCAG 2.2 · Info and Relationships: 시각적 관계의 프로그램 판독
- W3C WAI WCAG 2.2 · Name, Role, Value: 상호작용 요소의 이름·역할·상태·값
- W3C WAI · Providing Accessible Names and Descriptions: 보이는 글·네이티브 이름 방식 우선
- Chrome for Developers · View and change the DOM: Elements·현재 DOM· `$0`
- Chrome for Developers · Inspect mode: 화면 요소와 DOM 노드 선택
- Chrome for Developers · Accessibility reference: 접근성 트리·계산된 속성·소스 순서