2주차 · 3시간 워크숍

HTML로 상품 소개 페이지 만들기

웹이 문서를 연결하는 방식부터 시작해, 스타일과 동작을 넣기 전의 탄탄한 정보 구조를 완성합니다.

02

이번 주에 만들 것: 상품 소개 페이지의 HTML 뼈대

완성하려는 화면을 먼저 보고 시작합니다. 아래는 **2주차 마지막에 브라우저에서 확인할 결과**입니다. 아직 CSS를 적용하지 않아 단순해 보이지만, 상품을 읽고 이해하는 순서는 HTML만으로 이미 만들어집니다.

1. 브라우저가 읽을 문서의 기본 골격을 만듭니다.
2. 상품 이미지·이름·설명·가격을 소개 영역에 넣습니다.
3. 상품 특징 목록과 다음 내용으로 이동하는 CTA 링크를 만듭니다.
4. FAQ와 바닥글을 더해 한 페이지의 정보 구조를 완성합니다.

이번 주의 완성 기준: 예쁜 디자인보다 내용의 의미, 읽는 순서, 부모·자식 구조가 코드에 드러나는 것입니다. 3주차에는 CSS로 디자인하고 4주차에는 JavaScript로 FAQ 동작을 연결합니다.

실습환경을 준비한 직후 실제 `index.html` 제작을 시작합니다. 뒤의 **6. 상품 소개 페이지 만들기**에서는 앞에서 만든 파일에 이미지·상품 정보·특징·FAQ를 일곱 단계로 이어 붙입니다.

실습 파일
시작본: starter/index.html · 2주차 결과: week-2-html/index.html

2주차 완성 화면. 디자인 전에도 상품 이미지, 제목, 설명, 가격, 버튼의 읽는 순서가 드러납니다.

이번 주에 할 수 있게 되는 것

- 인터넷과 웹, 웹페이지와 웹사이트를 구분해 설명합니다.
- HTML 요소의 태그·속성·중첩을 읽고 화면의 정보 구조와 연결합니다.
- 요구사항을 의미 있는 화면 영역으로 나누어 상품 소개 페이지를 만듭니다.
- AI의 코드 설명을 실제 코드와 브라우저 출력으로 검증합니다.

수업 흐름

1. 개념과 준비 40분
2. HTML 문서 구조 30분
3. 휴식 10분
4. 요구사항 분해와 누적 실습 70분
5. 휴식 10분
6. AI 코드 읽기와 최종 확인 20분

교재 목차

// 1. 인터넷과 웹

인터넷은 여러 컴퓨터 네트워크가 서로 데이터를 주고받을 수 있도록 연결된 기반입니다. **웹**은 그 인터넷 위에서 주소와 링크를 사용해 문서를 요청하고 공유하는 서비스입니다. 인터넷이 도로망이라면 웹은 그 도로를 이용해 오가는 문서 전달 체계에 가깝습니다.

하이퍼텍스트, 웹페이지, 웹사이트

하이퍼텍스트는 링크를 따라 다른 위치나 문서로 이동할 수 있는 비선형 문서입니다. 브라우저로 읽는 하나의 문서를 **웹페이지**라 하고, 서로 관련된 여러 웹페이지의 묶음을 **웹사이트**라 합니다. 흔히 말하는 홈페이지는 웹사이트의 시작점이 되는 대표 페이지입니다.

주소창에 로컬 파일을 열어도 웹페이지일까요?
네. 서버에 공개하지 않아도 브라우저가 HTML 문서를 해석하면 웹페이지입니다. 이번 실습은 인터넷 연결 없이 로컬 파일로 진행합니다.

// 2. HTML·CSS·JavaScript

HTML은 무엇인가요?

HTML(HyperText Markup Language)은 웹 문서의 내용에 역할을 표시하고 서로 연결하기 위한 **표준 마크업 언어**입니다. 계산 절차를 지시하기보다 "이것은 제목이고, 이것은 문단이며, 이것은 다른 문서로 가는 링크다"처럼 문서의 구조와 의미를 선언합니다.

- **HyperText**: 링크를 통해 한 문서에서 다른 문서나 위치로 이동할 수 있습니다.
- **Markup**: 콘텐츠를 `<h1>`, `<p>`, `<a>` 같은 태그로 감싸 역할을 표시합니다.
- **Language**: 태그, 속성, 중첩처럼 브라우저와 작성자가 함께 따르는 문법과 규칙이 있습니다.

왜 HTML을 쓰나요?

일반 텍스트만으로는 어느 문장이 제목인지, 무엇이 링크인지, 이미지가 어떤 의미인지 컴퓨터가 안정적으로 구분하기 어렵습니다. HTML로 역할을 표시하면 브라우저뿐 아니라 스크린 리더, 검색 엔진, 개발 도구가 같은 구조를 이해할 수 있습니다. CSS는 이 구조를 골라 디자인하고, JavaScript는 브라우저가 만든 요소를 찾아 동작을 연결합니다.

HTML은 어떻게 웹페이지가 되나요?

1. 작성자가 내용을 태그와 속성으로 표시하고 `.html` 파일로 저장합니다.
2. 브라우저가 파일을 위에서 아래로 읽고, 중첩 관계에 따라 요소가 연결된 **DOM(Document Object Model)**을 만듭니다.
3. 브라우저가 각 요소를 화면에 기본 모양으로 표시합니다. CSS가 있으면 모양을 바꾸고, JavaScript가 있으면 DOM의 내용·상태·동작을 바꿀 수 있습니다.

```
상품명
↓ 역할을 태그로 표시
<h1>무광 물병</h1>
↓ 브라우저가 해석
페이지에서 가장 중요한 제목 요소
```

DOM은 무엇인가요?

DOM은 브라우저가 HTML을 읽어 메모리에 만드는, 현재 웹페이지의 살아 있는 작업 모델입니다. HTML 파일이 설계도라면 DOM은 브라우저가 그 설계도를 보고 조립한 구조물에 가깝습니다.

Document · 문서

브라우저에 열린 현재 웹 문서 전체를 뜻합니다.

Object · 객체

제목·문단·이미지 같은 각 요소를 프로그램이 찾고 다룰 수 있는 대상으로 표현합니다.

Model · 모델

요소가 누구 안에 들어 있는지 부모·자식 관계를 나무처럼 연결해 나타냅니다.

예를 들어 다음 HTML을 브라우저가 읽으면 요소가 위아래 목록이 아니라 **포함 관계가 있는 트리(tree)**로 연결됩니다.

```
HTML 파일                          브라우저가 만든 DOM
<body>                             document
  <main>                           └ body
    <h1>무광 물병</h1>             └ main
  </main>                                └ h1 "무광 물병"
</body>
```

> **HTML 파일**
>
> VS Code에 저장된 원본입니다. 파일을 직접 수정하고 저장하기 전까지 내용이 바뀌지 않습니다.

> **브라우저의 DOM**
>
> 현재 탭의 메모리에 있는 작업 상태입니다. JavaScript나 개발자 도구로 즉시 바꿀 수 있지만, 파일까지 자동으로 저장되지는 않습니다.

개발자 도구에서 직접 확인하기

1. 페이지에서 상품 제목을 오른쪽 클릭하고 **검사**를 선택합니다.
2. **Elements** 패널에서 선택된 `h1` 과 그 부모 요소를 봅니다. 이 패널이 현재 DOM 구조를 보여 줍니다.
3. Elements에서 제목 글자를 잠시 바꾸면 화면도 즉시 바뀝니다.
4. 페이지를 새로고침하면 VS Code에 저장된 HTML을 다시 읽어 DOM을 만들므로 원래 글자로 돌아옵니다.

HTML과 DOM은 같은 것이 아닙니다. 2주차에는 HTML 파일을 작성하고 브라우저가 만든 DOM과 화면을 확인합니다. 4주차에는 JavaScript가 이 DOM에서 요소를 찾고 상태를 바꾸는 방법을 배웁니다.

처음부터 접근성을 생각하기

웹 접근성은 장애가 있는 사람도 웹사이트와 도구를 인지하고, 이해하고, 탐색하고, 상호작용할 수 있게 설계하고 개발하는 일입니다. 이번 주에는 의미가 드러나는 제목과 영역, 이미지의 목적을 전하는 대체 텍스트, 마우스 없이도 작동하는 기본 버튼부터 적용합니다.

세 언어의 역할 나누기

브라우저는 세 파일을 함께 읽지만, 각 언어가 맡는 질문은 다릅니다.

HTML: 무엇인가?

제목, 문단, 이미지, 버튼처럼 콘텐츠의 의미와 구조를 표현합니다.

CSS: 어떻게 보이는가?

색, 글꼴, 여백, 크기, 배치와 화면 너비에 따른 변화를 정합니다.

JavaScript: 어떻게 반응하는가?

클릭과 키보드 입력 같은 사용자 행동에 반응하는 동작을 만듭니다.

이번 주에는 HTML로 읽을 수 있는 문서 구조를 완성합니다. CSS 파일과 JavaScript 파일은 폴더에 함께 두지만 아직 내용은 비워 둡니다. 다음 주와 그다음 주에 같은 페이지를 계속 발전시킵니다.

// 3. 실습환경 준비

1. 프로젝트 폴더

파일 탐색기에서 세 파일과 이미지 경로가 준비된 시작본을 확인합니다.

2. VS Code 설치와 폴더 열기

공식 데스크톱 앱을 설치한 뒤 **폴더 열기**로 시작본 폴더 전체를 엽니다. 파일 하나만 열면 상대 경로를 파악하기 어렵습니다.

3. Live Server와 웹 브라우저

`index.html` 을 Live Server로 열면 로컬 웹 주소가 생기고, 파일을 저장할 때 브라우저가 자동으로 새로고침됩니다. 확장을 설치할 수 없는 환경에서는 파일을 더블클릭해 열고 직접 새로고침해도 됩니다.

VS Code 설치 튜토리얼

Visual Studio Code(VS Code)는 HTML·CSS·JavaScript 파일을 작성하고 프로젝트 폴더를 한곳에서 관리하는 무료 코드 편집기입니다. 이번 과정은 브라우저용 `vscode.dev` 가 아니라 컴퓨터에 설치하는 **데스크톱 앱**을 사용합니다.

VS Code 공식 다운로드 페이지 열기

macOS에서 설치하기

1. 공식 다운로드 페이지에서 **macOS**를 선택합니다. 기종을 잘 모르겠다면 Intel과 Apple silicon에서 모두 실행되는 **Universal** 빌드를 선택합니다.
2. 다운로드 폴더에서 `.dmg` 파일을 더블클릭해 엽니다.
3. **Visual Studio Code.app** 아이콘을 **Applications(응용 프로그램)** 폴더로 드래그합니다.
4. Finder의 응용 프로그램 폴더에서 **Visual Studio Code**를 더블클릭합니다. 인터넷에서 받은 앱이라는 확인 창이 나오면 출처가 Microsoft의 공식 다운로드인지 확인한 뒤 **열기**를 선택합니다.
5. 자주 사용한다면 Dock의 VS Code 아이콘을 Control-클릭하고 **옵션** → **Dock에 유지**를 선택합니다.

Microsoft의 macOS 설치 안내

Windows에서 설치하기

1. 공식 다운로드 페이지에서 Windows용 **User Installer**를 선택합니다. 개인 계정에 설치되며 일반적으로 관리자 권한이 필요하지 않아 학습용으로 권장됩니다.
2. 다운로드한 `VSCodeUserSetup-{version}.exe` 파일을 실행합니다.
3. 사용권 계약을 확인하고, 설치 위치와 시작 메뉴 설정은 특별한 사유가 없다면 기본값으로 진행합니다.
4. **Install**을 누르고 설치가 끝나면 **Launch Visual Studio Code**를 선택해 실행합니다.
5. 설치 프로그램이 VS Code를 `PATH`에 추가하므로 나중에 새 터미널에서 `code .` 명령으로 현재 폴더를 열 수도 있습니다. 이번 실습에서는 메뉴의 폴더 열기만 사용해도 됩니다.

Microsoft의 Windows 설치 안내

회사 컴퓨터에서 설치가 차단된다면: 보안 경고를 우회하거나 임의의 다른 사이트에서 설치 파일을 받지 않습니다. 공식 다운로드 주소와 이 교재를 조직의 IT 관리자에게 전달해 설치 승인을 요청합니다.

첫 실행 화면에서 네 영역 찾기

Activity Bar
왼쪽 가장자리의 세로 아이콘 모음입니다.

Explorer
프로젝트 폴더와 파일 목록을 보여 줍니다.

Editor
가운데에서 파일 내용을 작성하는 영역입니다.

Status Bar
아래쪽에서 파일 상태와 Live Server 실행 상태를 보여 줍니다.

화면을 한국어로 바꾸기 · 선택

1. 명령 팔레트를 엽니다. macOS는 ⇧⌘**P(Shift+Command+P)**, Windows는 **Ctrl+Shift+P**입니다.
2. **Configure Display Language**를 입력해 실행합니다.
3. **한국어(ko)**를 선택합니다. 필요하면 안내에 따라 Korean Language Pack을 설치합니다.
4. VS Code를 다시 시작해 메뉴가 한국어로 바뀌었는지 확인합니다. 이 교재는 찾기 쉽도록 주요 메뉴의 영문 이름도 함께 표시합니다.

`starter` 폴더 전체 열기

1. 상단 메뉴에서 **File → Open Folder…**를 선택합니다. 한국어 화면에서는 **파일 → 폴더 열기…**입니다.
2. 교재 폴더 안의 `course/examples/product-landing/starter` 를 선택하고 **Open(열기)**을 누릅니다.
3. 작업 영역을 신뢰할지 묻는 창이 나오면 경로가 배포받은 교재의 `starter` 폴더인지 확인한 뒤에만 **Yes, I trust the authors**를 선택합니다.
4. Explorer에 `index.html` , `style.css` , `script.js` 세 파일이 보이는지 확인합니다.
5. `index.html` 을 클릭해 Editor에 코드가 열리면 설치와 폴더 준비가 완료된 것입니다.

☐ **완료 기준: VS Code 창의 Explorer에서 `starter` 아래 세 파일이 보이고, Editor에서 `index.html` 을 열 수 있습니다. 아직 Live Server는 실행하지 않아도 됩니다.**

```
starter/
├── index.html    ← 구조와 콘텐츠
├── style.css     ← 다음 주에 작성
└── script.js     ← 4주차에 작성
```

Live Server 설치하고 실행하기

Live Server는 내 컴퓨터에서 정적 웹 서버를 잠시 실행하고, 저장된 변경을 브라우저에 자동으로 반영하는 VS Code 확장입니다.

1. VS Code 왼쪽의 **Extensions** 아이콘을 엽니다. 단축키는 macOS ⇧⌘**X(Shift+Command+X)**, Windows **Ctrl+Shift+X**입니다.
2. **Live Server**를 검색하고 제작자 **Ritwick Dey**, 확장 식별자 `ritwickdey.LiveServer` 를 확인한 뒤 **Install**을 누릅니다. 확장은 VS Code와 같은 권한으로 실행되므로 이름과 제작자를 확인하고 조직의 설치 정책을 따릅니다.
3. VS Code에서 파일 하나가 아니라 `starter` 폴더 전체를 엽니다.
4. Explorer의 `index.html` 을 오른쪽 클릭해 **Open with Live Server**를 선택합니다. 또는 `index.html` 을 연 상태에서 아래 상태 표시줄의 **Go Live**를 누릅니다.
5. 예를 들어 `http://127.0.0.1:5500/index.html` 같은 주소가 열립니다. 코드를 바꾸고 저장하면 브라우저가 자동으로 새로고침되는지 확인합니다.
6. 실습을 마치면 명령 팔레트에서 **Live Server: Stop Live Server**를 실행해 서버를 종료합니다.

주소에서 포트 읽기

`localhost` 와 `127.0.0.1` 은 모두 **내 컴퓨터**를 가리킵니다. 뒤의 `5500` 은 여러 프로그램 가운데 Live Server로 들어가는 **포트(port)**, 즉 번호가 붙은 출입구입니다. 마지막의 `/index.html` 은 서버 안에서 요청한 파일 경로입니다.

기본 포트는 보통 `5500` 이지만 이미 사용 중이면 다른 번호를 사용할 수 있으므로 실제로 표시된 주소를 따릅니다. 이 주소는 내 컴퓨터의 실습 서버이며 인터넷에 공개된 웹사이트가 아닙니다.

Visual Studio Marketplace에서 Live Server 확인하기

왜 이름이 `index.html` 일까요?
브라우저가 **파일명이 없는 주소**를 요청하면 웹 서버나 정적 호스팅은 설정된 **기본 문서**를 찾습니다. 많은 서비스가 그 기본 이름으로 `index.html` 을 사용합니다.

`https://example.com/` 을 요청했을 때 서버가 루트 폴더의 `index.html` 을 찾아 응답하므로, 이 파일이 웹사이트의 시작 페이지인 **홈페이지**로 활용됩니다. 하위 폴더에서도 같은 설정을 사용할 수 있습니다.

다만 `index.html` 은 HTML 표준이 강제하는 이름은 아니며 웹 서버나 정적 호스팅의 설정에 따라 달라질 수 있습니다. 로컬에서는 다른 이름의 `.html` 파일도 직접 열 수 있지만, 이번 과정에서는 배포할 때의 관례까지 익히기 위해 이름을 통일합니다.

지금 사용할 파일
로컬 시작본 열기 · 2주차 HTML 완성본 미리보기

PDF로 보고 있다면 파일 탐색기에서 `course/examples/product-landing/starter/index.html` 을 찾아 VS Code로 여세요.

저장-자동 새로고침 확인

1. VS Code에서 `index.html` 의 제목 한 글자를 바꿉니다.
2. 파일을 저장합니다.
3. Live Server로 연 브라우저가 자동으로 새로고침되고 변경된 글자가 보이는지 확인합니다.

변화가 없다면 Live Server 실행 여부, 파일 저장 여부, 현재 열어 둔 파일의 경로, 태그 오타를 순서대로 확인합니다.

10분 프로젝트 시작: 실제 페이지 골격 저장하기

긴 설명을 더 읽기 전에 이번 주에 계속 사용할 `index.html`을 실제로 고칩니다. 아직 태그를 외우지 않아도 되며, 아래 구조는 뒤에서 한 줄씩 설명합니다.

지금 따라 하기 · index.html

1. **열기:** VS Code에서 `starter/index.html`을 엽니다.
2. **위치:** 파일 전체를 선택합니다.
3. **행동:** 아래 코드로 바꾸고 저장합니다.
4. **확인:** 브라우저 탭 제목, 상단의 "첫 상품", 하단의 "첫 웹사이트 실습 예제"를 찾습니다.

저장할 코드 Console이 아니라 `index.html` 파일 전체에 입력합니다.

```
<!doctype html>
<html lang="ko">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>하루를 가볍게 만드는 첫 상품</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js" defer></script>
</head>
<body>
  <header class="site-header"><a href="#product">첫 상품</a></header>
  <main id="product">
    <section class="product-hero"></section>
    <section></section>
    <section></section>
  </main>
  <footer><p>첫 웹사이트 실습 예제</p></footer>
</body>
</html>
```

☐ **브라우저에 상단과 하단 문구가 보이고, 파일을 다시 열어도 코드가 남아 있으면 완료입니다. 실행 취소하지 말고 이 파일을 계속 사용합니다.**

관찰: 코드에 적은 글자와 브라우저에 보이는 글자가 같은가요? 비어 있는 세 `section`은 앞으로 어떤 내용을 담게 될까요?

// 4. HTML 문서 구조

HTML은 문서의 내용을 **요소(element)**로 나눕니다. 보통 요소는 여는 태그, 내용, 닫는 태그로 이루어집니다. 속성은 여는 태그 안에서 요소에 추가 정보를 줍니다.

```
<p class="price">29,000원</p>
     └ 속성       └ 내용
└──────── 하나의 p 요소 ────────┘
```

처음 헷갈리는 네 단어

- **태그(tag)**: `<p>`, `</p>` 처럼 꺾쇠로 표시한 시작과 끝의 표기입니다.
- **요소(element)**: 여는 태그부터 내용과 닫는 태그까지 합친 전체 단위입니다.
- **속성(attribute)**: `class="price"` 처럼 여는 태그 안에서 요소에 추가 정보를 붙입니다.
- **중첩(nesting)**: 한 요소 안에 다른 요소를 넣는 구조입니다. 먼저 연 바깥 요소는 안쪽 요소를 닫은 뒤에 닫습니다.

태그는 보통 이런 순서로 사용합니다

HTML을 작성할 때는 태그 이름부터 고르기보다, 먼저 **콘텐츠의 역할과 사용자가 할 행동**을 정합니다. 태그는 화면을 꾸미는 도구가 아니라 브라우저와 보조 기술에 "이 내용이 무엇인지" 알려 주는 표시입니다.

1. **내용을 평문으로 적습니다.** 예를 들어 "상품명, 한 줄 설명, 상세 페이지로 이동하는 링크"처럼 화면에 필요한 정보를 먼저 나열합니다.
2. **역할에 맞는 표준 요소를 고릅니다.** 제목은 `h1` ~ `h6`, 문단은 `p`, 다른 주소로 이동할 때는 `a`, 현재 페이지에서 동작을 실행할 때는 `button`, 이미지는 `img`, 목록은 `ul` 또는 `ol`과 `li`를 사용합니다.
3. **큰 부모 구조부터 자식 요소를 넣습니다.** 먼저 `header`, `main`, `section`, `footer` 같은 영역을 만들고, 그 안에 제목·문단·링크처럼 더 작은 요소를 중첩합니다.
4. **필요한 추가 정보는 속성으로 적습니다.** 속성은 여는 태그 안에 `이름="값"` 형태로 씁니다. `href`는 링크의 목적지, `src`와 `alt`는 이미지의 경로와 대체 텍스트, `id`와 `class`는 요소를 식별하거나 묶는 데 사용합니다. 필요한 경우 `aria-*` 속성으로 관계나 상태를 보완합니다.
5. **저장한 뒤 구조와 결과를 함께 확인합니다.** 여는 순서의 반대로 태그를 닫았는지, 들여쓰기로 부모와 자식이 구분되는지, 브라우저에서 읽는 순서와 링크·버튼의 동작이 의도와 같은지 확인합니다.

한 덩어리를 작성하는 예

```
<section>
  <h2>무광 물병</h2>
  <p>가볍고 매일 세척하기 쉬운 물병입니다.</p>
  <a href="products/bottle.html">상품 자세히 보기</a>
</section>
```

- `section`은 하나의 주제를 묶고, 바로 안의 `h2`가 그 주제의 제목을 알려 줍니다.
- `a`는 다른 HTML 문서로 이동하므로 알맞습니다. 클릭했을 때 현재 화면의 내용만 바꾼다면 `button`이 더 알맞습니다.

기억할 원칙: 기본 모양이 아니라 의미와 행동으로 태그를 선택합니다. 색상·크기·간격처럼 보이는 모양은 다음 주에 CSS로 바꿉니다.

문서 전체의 기본 골격

```
<!doctype html>
<html lang="ko">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>하루를 가볍게 만드는 첫 상품</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js" defer></script>
</head>
<body>
  <main></main>
</body>
</html>
```

- `<!doctype html>` : 현대 HTML 문서임을 브라우저에 알립니다.
- `html` : 문서 전체를 감싸며 `lang="ko"` 로 주 언어를 알립니다.
- `head` : 문자 인코딩, 화면 설정, 제목, 연결 파일처럼 문서 정보를 둡니다.
- `body` : 브라우저 화면에 표시할 콘텐츠를 둡니다.
- `link` 와 `script` : 다음 주에 사용할 CSS와 4주차 JavaScript 파일을 미리 연결합니다. `defer` 는 HTML을 먼저 읽은 뒤 스크립트를 실행하라는 뜻이며 지금은 그대로 둡니다.

대표적인 HTML 태그와 기능

HTML 태그를 전부 외울 필요는 없습니다. 먼저 만들려는 콘텐츠의 역할을 정한 뒤, 아래처럼 범주별로 찾아 쓰면 됩니다. 같은 콘텐츠라도 어떤 태그를 선택하느냐에 따라 브라우저와 보조 기술이 이해하는 의미와 기본 동작이 달라집니다.

범주	대표 태그	기능과 선택 기준
문서 기본	`html`, `head`, `title`, `meta`, `body`	문서 전체, 문서 정보, 브라우저 탭 제목, 메타데이터, 화면에 보일 내용을 각각 나타냅니다.
제목과 문장	`h1` ~ `h6`, `p`, `strong`, `em`	제목의 단계, 문단, 중요한 내용, 문맥상 강조할 내용을 표현합니다. 단순히 글자를 크게·굵게 보이게 하려고 고르지는 않습니다.
문서 영역	`header`, `nav`, `main`, `section`, `article`, `footer`	머리말, 주요 이동 링크, 핵심 내용, 주제별 구역, 독립적으로 읽을 글, 바닥글처럼 문서의 큰 구조를 설명합니다.
일반 묶음	`div`, `span`	알맞은 의미의 태그가 없을 때 여러 요소 또는 문장 안의 일부를 묶습니다. 그 자체로는 콘텐츠의 역할을 설명하지 않습니다.
목록	`ul`, `ol`, `li`	순서가 중요하지 않은 목록, 순서가 있는 목록, 각 목록 항목을 나타냅니다. `li`는 `ul`이나 `ol` 안에 둡니다.
링크와 이미지	`a`, `img`, `figure`, `figcaption`	`a`는 `href` 주소로 이동하고, `img`는 `src` 이미지와 `alt` 대체 텍스트를 제공합니다. 이미지와 설명은 `figure`와 `figcaption`으로 묶을 수 있습니다.
동작과 입력	`button`, `form`, `label`, `input`	`button`은 현재 페이지의 동작을 실행합니다. `form`은 입력 묶음, `label`은 입력 이름, `input`은 실제 입력 칸을 나타냅니다.

태그는 모양이 아니라 역할로 선택합니다.
이번 주에 직접 쓰는 핵심: `html`, `head`, `body`, `header`, `main`, `section`, `footer`, `h1` ~ `h3`, `p`, `a`, `img`, `ul`, `li`, `article`입니다. 나머지는 외우는 목록이 아니라 필요할 때 찾아보는 참고 항목입니다.

- 다른 주소로 이동하면 `a`, 현재 화면에서 메뉴를 열거나 값을 바꾸면 `button`을 사용합니다.
- `strong`과 `em`은 중요도와 강조의 의미가 필요할 때 사용하고, 단순한 굵기와 기울기는 CSS로 표현합니다.
- `br`은 주소나 시처럼 실제 줄바꿈이 내용의 일부일 때만 사용합니다. 요소 사이의 여백을 만들기 위해 반복하지 않습니다.
- `meta`, `img`, `input`, `br`처럼 별도의 닫는 태그가 없는 요소도 있습니다.

중첩은 상자 안에 상자를 넣는 일

```
<main>
  <section>
    <h2>상품 특징</h2>
  </section>
</main>
```

안쪽에서 먼저 닫고 바깥쪽을 나중에 닫습니다. 들여쓰기는 화면에 직접 표시되지는 않지만 부모-자식 관계를 눈으로 확인하게 해줍니다. `img`처럼 별도의 닫는 태그가 없는 요소도 있습니다.

// 5. 요구사항을 화면으로 나누기

코드를 바로 쓰기 전에 “사용자가 무엇을 읽고 무엇을 할 수 있어야 하는가?”를 화면 영역과 HTML 요소로 번역합니다.

요구사항	화면 영역	HTML 후보	확인 방법
사이트의 시작점을 보여준다	상단	`header`, `a`	페이지 첫 부분에 이름이 보이는가?
상품을 한눈에 소개한다	상품 소개	`section`, `img`, `h1`, `p`	무엇을 소개하는지 바로 알 수 있는가?
핵심 특징을 훑어본다	특징	`h2`, `ul`, `li`	각 항목이 하나의 목록으로 묶였는가?
자주 묻는 내용을 찾는다	FAQ	`h2`, `article`, `h3`, `p`	질문과 답변의 관계가 드러나는가?
문서의 끝을 알린다	하단	`footer`	주요 내용 밖에 위치하는가?

시맨틱 요소를 먼저 선택합니다

시맨틱(semantic)은 "의미에 관한"이라는 뜻입니다. HTML에서 **시맨틱 요소**는 태그 이름만 보아도 콘텐츠의 역할을 알 수 있는 요소입니다. 예를 들어 `nav` 는 주요 이동 링크, `main` 은 문서의 핵심 내용, `button` 은 사용자가 실행할 동작임을 나타냅니다.

반대로 `div` 와 `span` 은 콘텐츠의 역할을 따로 설명하지 않는 **비시맨틱 요소**입니다. 영역을 묶거나 스타일을 적용할 때 유용하지만, 이름만으로는 그 안의 내용이 제목인지 버튼인지 알 수 없습니다.

왜 의미 있는 요소를 먼저 고를까요?

1. **구조가 전달됩니다.** 브라우저, 스크린 리더, 검색 엔진, 개발자가 같은 콘텐츠 역할과 읽는 순서를 이해하기 쉬워집니다.
2. **기본 동작을 활용할 수 있습니다.** 실제 `button` 은 키보드 포커스를 받고 Enter·Space 입력으로 실행되지만, 버튼처럼 꾸민 `div` 에는 이런 동작이 자동으로 생기지 않습니다.
3. **코드를 고치기 쉬워집니다.** 클래스 이름이나 디자인이 바뀌어도 `header`, `article`, `footer` 같은 태그가 문서의 의도를 계속 설명합니다.

같아 보이는 코드도 의미는 다릅니다

```
<!-- 모양만 이름으로 흉내 낸 경우 -->
<div class="title">상품 특징</div>
<div class="button">문의하기</div>

<!-- 콘텐츠의 실제 역할을 표시한 경우 -->
<h2>상품 특징</h2>
<button type="button">문의하기</button>
```

CSS를 사용하면 두 코드의 모양을 비슷하게 만들 수 있지만, HTML이 전달하는 의미와 기본 동작은 같아지지 않습니다.

선택 순서

1. “이 콘텐츠는 제목, 문단, 이동 링크, 실행 버튼, 목록, 독립 글, 문서 영역 중 무엇인가?”를 먼저 묻습니다.
2. 역할에 맞는 `h1` ~ `h6`, `p`, `a`, `button`, `ul`, `article`, `nav`, `main`, `footer` 같은 표준 요소를 선택합니다.
3. 맞는 의미가 없고 단순히 묶거나 꾸미기 위한 상자가 필요할 때 `div` 또는 `span`을 사용합니다.

> **`section` 사용 팁:** 하나의 주제를 묶을 때 사용하고, 그 주제를 설명하는 제목을 보통 함께 둡니다. 화면에 사각형 상자가 필요하다는 이유만으로 모든 영역을 `section`으로 만들지는 않습니다.

// 6. 상품 소개 페이지 만들기

아래 일곱 단계는 같은 `index.html`을 계속 고칩니다. 매 단계에서 앞의 코드를 지우지 말고 새 구조를 더합니다. 브라우저 화면이 단순해 보여도 지금의 목표는 디자인이 아니라 의미와 읽는 순서입니다.

1. 문서 기본 골격

지금 따라 하기 · index.html

1. **열기:** 실습환경 준비에서 저장한 `starter/index.html`을 엽니다.
2. **위치:** 파일 맨 위의 `doctype`과 `head`를 찾습니다.
3. **행동:** 아래 네 줄이 있는지 확인하고 빠진 줄만 보완합니다. 이미 같다면 다시 붙여 넣지 않습니다.

실행 화면

브라우저 탭에는 “하루를 가볍게 만드는 첫 상품”이 표시됩니다.

본문에는 실습환경 준비에서 넣은 상단과 하단 문구가 보입니다. 빈 `section`은 아직 화면에 표시할 내용이 없습니다.

index.html에서 확인할 코드

확인용 코드 이미 저장한 파일에서 다음 네 줄을 찾아 읽습니다.

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>하루를 가볍게 만드는 첫 상품</title>
<script src="script.js" defer></script>
```

자주 발생하는 실수

`head`와 `body`를 반대로 놓거나, 수정 뒤 저장하지 않는 경우가 많습니다. 탭 제목부터 확인하세요.

확인 질문

화면에 보일 콘텐츠는 `head`와 `body` 중 어디에 작성해야 할까요?

☐ **탭 제목이 “하루를 가볍게 만드는 첫 상품”이고 저장되지 않은 파일 표시가 없으면 1단계 완료입니다.**

2. 시맨틱 레이아웃

지금 따라 하기 · index.html

1. **위치:** `body` 안의 `header`, `main`, `footer`를 찾습니다.
2. **행동:** 실습환경 준비에서 넣은 구조와 아래 코드를 비교합니다.
3. **저장:** 세 개의 빈 `section`이 `main` 안에 있는 상태를 유지합니다.

실행 화면

"첫 상품"과 "첫 웹사이트 실습 예제"가 위아래로 보입니다.

가운데 `main`은 아직 비어 있어 두 문구 사이의 간격이 작습니다.

index.html에서 확인할 코드

기존 `body`를 아래 구조로 바꿉니다. `section` 세 개는 각각 소개, 특징, FAQ를 담당합니다.

```
<body>
  <header class="site-header"><a href="#product">첫 상품</a></header>
  <main id="product">
    <section class="product-hero"></section>
    <section></section>
    <section></section>
  </main>
  <footer><p>첫 웹사이트 실습 예제</p></footer>
</body>
```

자주 발생하는 실수

`main`을 여러 개 만들거나 `footer`를 `main` 안에 넣지 않았는지 닫는 태그와 들여쓰기를 확인합니다.

확인 질문

세 `section`의 역할은 각각 무엇이며, 문서의 핵심 콘텐츠를 감싸는 요소는 무엇인가요?

☐ **브라우저 상단과 하단 문구가 보이고 `main` 안에 빈 `section` 세 개가 있으면 2단계 완료입니다.**

3. 상품 이미지와 구체적인 대체 텍스트

지금 따라 하기 · index.html

1. **위치:** `main` 안의 첫 번째 `<section class="product-hero">`를 찾습니다.
2. **행동:** 빈 section의 여는 태그와 닫는 태그 사이에 아래 `img`를 넣습니다.
3. **저장:** 파일을 저장하고 브라우저에서 물병 이미지를 찾습니다.

실행 화면

상품 소개 영역에 밝은 회색 배경의 원통형 물병 이미지가 보입니다.

아직 CSS를 작성하지 않았으므로 HTML의 `width`와 `height` 속성으로 기본 표시 크기만 제한합니다.

index.html에 저장할 코드

첫 번째 `section` 안에 로컬 이미지를 추가합니다.

```
<section class="product-hero">
  <img
    src="../../../assets/images/product-example.png"
    alt="밝은 회색 배경에 놓인 무광 회색 원통형 물병"
    width="320"
    height="400">
</section>
```

`src`는 현재 HTML 파일에서 이미지까지 가는 상대 경로입니다. `alt`는 파일명이 아니라 이 문맥에서 이미지가 전달하는 핵심 정보를 짧게 설명합니다. 원본은 1122×1402 픽셀이므로 같은 비율에 가까운 `320×400` 표시 크기를 지정해 작은 화면에서 가로로 넘치지 않게 합니다.

```
week-2-html/index.html
  └ ../../../        ← product-landing → examples → course로 세 단계 올라감
      └ assets/images/product-example.png
```

`../` 하나는 현재 폴더의 부모 폴더로 한 단계 올라간다는 뜻입니다. 경로가 헷갈리면 VS Code 탐색기에서 두 파일의 위치를 먼저 확인합니다.

자주 발생하는 실수

경로의 `../` 개수나 확장자가 다르면 이미지가 깨집니다. 의미 있는 이미지에 `alt="상품"` 처럼 모호한 설명을 쓰지 않습니다.

확인 질문

이미지를 보지 못하는 사람에게 이 `alt` 만 읽어주어도 페이지 맥락을 이해할 수 있을까요?

☐ **이미지가 깨지지 않고 보이며 `alt`, `width`, `height` 가 모두 있으면 3단계 완료입니다.**

---

4. 제목·설명·가격

지금 따라 하기 · index.html

1. **위치:** 3단계에서 넣은 이미지 바로 다음 줄을 찾습니다.
2. **행동:** 이미지가 있는 첫 번째 `product-hero` section 전체를 아래 코드와 같게 만듭니다.
3. **저장:** 상품명·설명·가격이 이미지 다음에 보이는지 확인합니다.

실행 화면

이미지 아래에 작은 소개 문구, 큰 상품명, 설명, 가격이 순서대로 보입니다.

제목 크기는 브라우저 기본 스타일이 적용됩니다.

index.html에 저장할 코드

이미지 다음에 상품 정보를 담는 묶음을 추가합니다. `div` 는 이곳에서 스타일을 위한 묶음 역할만 합니다.

```
<section class="product-hero">
  <img src="../../../assets/images/product-example.png"
       alt="밝은 회색 배경에 놓인 무광 회색 원통형 물병"
       width="320" height="400">
  <div class="product-copy">
    <p class="eyebrow">새로운 일상을 위한 도구</p>
    <h1>하루를 가볍게 만드는 첫 상품</h1>
    <p>간결한 설명과 명확한 정보 구조를 연습하기 위한 예시 상품입니다.</p>
    <p class="price">29,000원</p>
  </div>
</section>
```

자주 발생하는 실수

글자를 크게 만들기 위해 제목 단계를 고르지 않습니다. 페이지 전체 주제는 `h1`, 그 아래 영역 제목은 `h2`로 구조를 정합니다.

확인 질문

상품명은 왜 `p`가 아니라 `h1`이며, 가격은 왜 제목이 아니라 문단일까요?

☐ **상품명, 설명, 가격이 중복 없이 한 번씩 보이고 제목이 `h1` 이면 4단계 완료입니다.**

---

5. 특징 목록

지금 따라 하기 · index.html

1. **위치:** `main` 안의 두 번째 빈 `section`을 찾습니다.
2. **행동:** 빈 section 전체를 아래 특징 section으로 바꿉니다.
3. **저장:** “상품 특징”과 점이 붙은 세 항목을 확인합니다.

실행 화면

“상품 특징” 아래에 점이 붙은 세 항목이 보입니다.

항목 순서 자체가 중요하지 않으므로 번호 없는 목록을 사용합니다.

index.html에 저장할 코드

두 번째 `section`을 아래 내용으로 채웁니다.

```
<section aria-labelledby="features-title">
  <h2 id="features-title">상품 특징</h2>
  <ul>
    <li>간결한 사용법</li>
    <li>관리하기 쉬운 소재</li>
    <li>일상에 어울리는 디자인</li>
  </ul>
</section>
```

`aria-labelledby`의 값과 제목의 `id`가 같아 이 영역의 이름을 제목과 연결합니다.

자주 발생하는 실수

`li`를 `ul` 밖에 두거나, 같은 `id`를 다른 요소에 반복하지 않습니다.

확인 질문

세 특징의 순서를 바꾸어도 의미가 같다면 `ul`과 `ol` 중 무엇이 알맞을까요?

☐ **제목 아래에 세 특징이 목록으로 보이고 `li`가 모두 `ul` 안에 있으면 5단계 완료입니다.**

---

6. CTA 링크

지금 따라 하기 · index.html

1. **위치:** 첫 번째 상품 영역의 `<p class="price">` 바로 다음 줄을 찾습니다.
2. **행동:** 가격은 그대로 두고 아래 `a` 한 줄만 추가합니다.
3. **저장:** 링크가 보이는지 확인합니다. 목적지는 다음 단계에서 만듭니다.

실행 화면

가격 아래에 “자주 묻는 질문 보기” 링크가 보입니다.

누르면 같은 페이지의 FAQ 영역으로 이동합니다.

index.html에 추가할 코드

4단계의 `product-copy` 안에서 가격 다음 줄에 링크를 추가합니다.

```
<p class="price">29,000원</p>
<a class="product-cta" href="#faq-title">자주 묻는 질문 보기</a>
```

CTA(Call To Action)는 사용자가 다음 행동을 선택하도록 돕는 요소입니다. 여기서는 FAQ라는 위치로 **이동**하므로 `a`를 사용합니다. 7단계에서 추가할 제목의 `id="faq-title"`이 목적지입니다.

자주 발생하는 실수

`href="#faq-title"`의 값과 목적지 제목의 `id="faq-title"`이 다르면 이동하지 않습니다. 링크와 버튼은 모양이 아니라 역할로 구분합니다.

확인 질문

현재 화면의 값을 바꾸거나 메뉴를 연다면 `a` 대신 어떤 요소가 알맞을까요?

☐ **가격 다음에 CTA 링크가 한 번만 보이고 `href="#faq-title"` 이면 6단계 완료입니다.**

---

7. FAQ 제목과 article

지금 따라 하기 · index.html

1. **위치:** `main` 안의 세 번째 빈 `section`을 찾습니다.
2. **행동:** 빈 section 전체를 아래 FAQ section으로 바꿉니다.
3. **저장:** CTA를 눌러 “자주 묻는 질문” 위치로 이동하는지 확인합니다.

실행 화면

“자주 묻는 질문” 아래에 질문 두 개와 각 답변이 보입니다.

이번 주에는 모두 펼쳐 둡니다. 열고 닫는 동작은 4주차에 추가합니다.

index.html에 저장할 코드

세 번째 `section`을 채웁니다. 각 질문과 답변은 독립적으로 이해할 수 있는 `article`입니다.

```
<section aria-labelledby="faq-title">
  <h2 id="faq-title">자주 묻는 질문</h2>
  <article class="faq-item">
    <h3>언제 배송되나요?</h3>
    <p>결제 확인 후 순차적으로 발송됩니다.</p>
  </article>
  <article class="faq-item">
    <h3>교환할 수 있나요?</h3>
    <p>수령 후 안내된 기간 안에 접수할 수 있습니다.</p>
  </article>
</section>
```

자주 발생하는 실수

FAQ 영역 제목을 건너뛰고 바로 질문부터 쓰면 전체 구조를 파악하기 어렵습니다. `h1 → h2 → h3`의 관계를 확인합니다.

확인 질문

FAQ 전체 제목과 개별 질문의 제목 단계가 다른 이유를 문서 구조로 설명할 수 있나요?

☐ **질문 두 개와 답변 두 개가 보이고 CTA가 FAQ 제목으로 이동하면 7단계 완료입니다.**

최종 브라우저 확인

1. 저장한 뒤 새로고침합니다.
2. 문서를 위에서 아래로 읽으며 상단, 상품, 특징, FAQ, 하단 순서를 확인합니다.
3. 이미지가 보이지 않으면 경로를 완성본과 한 글자씩 비교합니다.
4. 브라우저 창을 좁혀도 콘텐츠 순서와 문장이 유지되는지 확인합니다.
5. 로컬 완성본과 태그·텍스트를 비교합니다.

// 7. AI와 코드 읽기

AI에게 먼저 전체 코드를 다시 만들게 하지 않습니다. 내가 작성한 코드를 붙여 넣고, 설명 범위와 수정 범위를 제한합니다. 아래 프롬프트의 “새로 작성하지 말고”, “해당 줄만”, “전체 코드는 다시 작성하지 마”가 그 경계입니다.

예시 프롬프트
아래 HTML을 새로 작성하지 말고, 위에서 아래 순서대로 각 요소가 화면에서 어떤 역할을 하는지 설명해줘. 잘못된 중첩이나 접근성 문제가 있다면 해당 줄만 지적하고 전체 코드는 다시 작성하지 마.

AI 설명을 검증하는 네 가지 대조

1. **실제 코드의 태그명**과 AI가 말한 태그명이 같은지 확인합니다.
2. **실제 코드의 중첩**과 AI가 설명한 부모-자식 관계를 들여쓰기로 대조합니다.
3. **브라우저의 화면 출력**에서 설명한 콘텐츠와 순서가 실제로 보이는지 확인합니다.
4. **AI 설명**이 맞다고 바로 가정하지 말고, 지적한 줄만 바꾸기 전후로 비교합니다.

AI가 이렇게 말한다면	직접 확인할 것
“이미지 대체 텍스트가 없습니다.”	실제 `img` 여는 태그 안에 `alt`가 있는가?
“FAQ가 `main` 밖에 있습니다.”	들여쓰기와 닫는 `</main>`의 위치가 어디인가?
“버튼이 동작합니다.”	직접 클릭했을 때 변화가 있는가? 아직 스크립트가 비어 있지는 않은가?

이번 주 마무리 체크

- 한 문서에 `h1`이 하나 있고 하위 제목의 순서가 자연스럽습니다.
- 모든 여는 태그의 닫는 위치와 들여쓰기를 확인했습니다.
- 상품 이미지에 문맥을 설명하는 대체 텍스트가 있습니다.
- 시작본에서 HTML 완성본까지 일곱 변경을 설명할 수 있습니다.
- AI 설명과 실제 코드, 중첩, 브라우저 출력을 대조했습니다.

// 출처와 더 읽을 자료

아래 링크는 개념 설명을 검증한 공식 출처입니다. 실습 파일은 외부 자료를 불러오지 않으므로 인터넷 연결 없이도 실행됩니다.

- Visual Studio Code, Download - 운영체제별 공식 데스크톱 앱 다운로드
- Visual Studio Code, macOS setup - DMG와 응용 프로그램 폴더를 이용한 설치 절차
- Visual Studio Code, Windows setup - User Setup과 System Setup의 차이 및 설치 절차
- Visual Studio Code, Change the display language - 표시 언어 변경과 Language Pack 안내
- Visual Studio Code, Use extensions - Extensions 화면에서 확장을 찾고 설치·관리하는 공식 절차
- Visual Studio Marketplace, Live Server - 설치 식별자, 실행·종료 방법, 포트 설정 안내
- WHATWG, HTML Living Standard - HTML 문서 구조, 요소 의미, DOM 관련 기준
- W3C WAI, Introduction to Web Accessibility - 웹 접근성의 정의와 기본 원칙
- W3C WAI, Images Tutorial - 이미지 목적에 맞는 대체 텍스트 작성 지침

다음 주에는 같은 HTML에 CSS를 연결해 글꼴, 색상, 여백, Flexbox와 반응형 배치를 적용합니다.