3주차 · 3시간 워크숍

CSS로 상품 페이지 디자인하기

2주차에 만든 HTML은 그대로 두고, `style.css`에만 규칙을 더해 PC와 모바일에서 읽기 좋은 상품 페이지로 바꿉니다.

03

이번 주에 할 수 있게 되는 것

- CSS 선택자가 HTML 요소를 찾고 속성이 표현을 바꾸는 과정을 설명합니다.
- 박스 모델과 Flexbox로 크기, 여백, 배치를 조절합니다.
- 하나의 HTML이 화면 폭에 따라 가로와 세로 배치로 바뀌도록 만듭니다.
- AI가 제안한 CSS를 바로 붙이지 않고 변경 범위와 화면을 검증합니다.

수업 흐름

1. CSS 연결과 선택자 20분
2. 박스 모델과 기본 디자인 50분
3. 휴식 10분
4. Flexbox와 반응형 레이아웃 65분
5. 휴식 10분
6. AI 제한 수정과 화면 QA 25분

교재 목차

이번 실습에서 무엇을 만들까요?

이번 과정은 갑자기 새로운 사이트를 만드는 실습이 아니라, **하나의 작은 상품 소개 페이지를 2주차부터 4주차까지 누적해 완성**하는 프로젝트입니다. 물병은 구조가 단순해 웹의 공통 요소를 한 화면에서 연습하기 위한 샘플입니다.

1. **2주차 HTML:** 상품 이미지, 상품명, 설명, 가격, CTA, 특징과 FAQ의 의미와 순서를 만듭니다.
2. **3주차 CSS:** 같은 HTML에 글꼴, 색, 여백, 카드, Flexbox와 모바일 배치를 적용합니다.
3. **4주차 JavaScript:** FAQ 질문을 눌렀을 때 답변이 열리고 닫히는 동작을 연결합니다.

CTA(Call To Action, 행동 유도 요소)는 방문자에게 다음 행동을 제안하는 버튼이나 링크입니다. 이번 예제의 "자주 묻는 질문 보기"는 같은 페이지의 FAQ 위치로 이동하므로 `a` 를 사용합니다. 현재 화면의 값을 바꾸거나 메뉴를 여는 동작이라면 `button` 을 사용합니다. 먼저 아래 두 화면을 비교하고, CSS가 구조가 아니라 보이는 방식을 어떻게 바꾸는지 찾아봅니다.

이번 주 작업 파일

시작 상태: 2주차 HTML 완성본 열기 · 완성 상태: 3주차 CSS 완성본 열기

1. 2주차에 사용한 개인 작업 폴더를 그대로 엽니다.
2. 작업 폴더가 없다면 `week-2-html` 폴더를 같은 위치에 복사해 `my-product-page` 처럼 이름을 바꿉니다.
3. 제공된 `week-3-css` 는 정답 비교용이므로 직접 수정하지 않습니다.

두 예제를 먼저 비교해 보세요.

HTML과 동작은 같고 `style.css` 만 다릅니다. 이번 주의 핵심은 구조를 바꾸는 것이 아니라, 같은 구조를 어떻게 보여줄지 결정하는 것입니다.

CSS 적용 전: HTML 구조와 브라우저 기본 스타일만 보이는 2주차 화면.

CSS 적용 후: 같은 HTML이 여백, 글자 크기, 색, Flexbox가 적용된 768px 화면으로 바뀝니다.

실습 전에: 개발자 도구 한 번 열어보기

개발자 도구는 완성된 웹페이지의 HTML과 CSS를 브라우저 안에서 관찰하고 임시로 시험하는 도구입니다. 여기서 바꾼 값은 새로고침하면 사라지므로, 최종 수정은 반드시 VS Code의 파일에 저장합니다.

1. 페이지의 상품 제목을 오른쪽 클릭하고 **검사(Inspect)**를 선택합니다.
2. 단축키는 macOS **Option+Command+I**, Windows **Ctrl+Shift+I** 또는 **F12**입니다.
3. 왼쪽의 HTML에서 선택된 요소와 오른쪽 Styles의 CSS 규칙이 연결되는지 확인합니다.

Elements	Styles	Computed	Console	Network
현재 HTML 구조와 선택된 요소	적용된 CSS와 임시 값 변경	최종 계산값과 박스 모델	메시지와 JavaScript 실행	HTML·CSS·이미지 요청 상태

첫 관찰: 제목을 선택하고 Styles에서 `font-size` 체크를 껐다 켜 보세요. 화면이 바뀌어도 파일은 아직 수정되지 않았습니다.

// 1. CSS 연결하기

CSS(Cascading Style Sheets)는 HTML 요소를 선택한 다음 `속성: 값;`으로 표현을 지정합니다. 이번 실습은 다른 파일이나 서비스 없이 로컬 폴더의 세 파일로 실행합니다.

---

1. CSS 파일 연결

> **지금 따라 하기 · style.css**
>
> 1. **열기:** 개인 작업 폴더의 `style.css`를 엽니다.
> 2. **위치:** 파일이 비어 있다면 첫 줄, 코드가 있다면 맨 위를 찾습니다.
> 3. **행동:** 색 변수와 `body` 규칙을 입력하고 저장합니다.

지금 할 일

`head`의 `link`가 `style.css`를 가리키는지 확인하고, 색 값을 재사용할 수 있게 CSS 변수를 준비한 뒤 본문에 바로 적용합니다. 2주차 완성본에는 연결이 이미 들어 있으므로 지우지 말고 경로를 확인합니다.

style.css에 저장할 코드

`index.html`의 `head`에서 다음 한 줄을 확인합니다.

```
<link rel="stylesheet" href="style.css">
```

`style.css`의 첫 부분에 다음을 작성합니다.

```
:root {
  --color-ink: #24221f;
  --color-muted: #6d6861;
  --color-surface: #f7f5f1;
  --color-card: #ffffff;
  --color-accent: #4f6656;
  --color-border: #ded9d1;
}

body {
  color: var(--color-ink);
  background: var(--color-surface);
}
```

저장 후 확인

페이지 배경이 밝은 베이지색으로 바뀌면 CSS 연결에 성공한 것입니다. `var(--color-surface)`는 `:root`에 저장한 색 값을 꺼내 씁니다.

한 줄 바꿔 보기

개발자 도구의 **Network**나 **Sources**에서 `style.css`가 읽혔는지 확인합니다. `href`를 잠시 다른 이름으로 바꾸고 새로고침한 뒤, 다시 `style.css`로 돌려 경로 연결을 확인합니다.

☐ **배경색이 바뀌고 `style.css` 요청 오류가 없으면 1단계 완료입니다.**

// 2. 선택자와 class

선택자(selector)는 CSS를 적용할 HTML 요소를 찾는 표현입니다. 태그 이름을 그대로 쓰거나, HTML의 `class` 앞에 점(`.`), `id` 앞에 샵(`#`)을 붙여 찾습니다.

CSS 선택자	읽는 방법	찾는 HTML	주로 언제 쓰나
`h1`	태그 선택자	`<h1>상품명</h1>`	같은 태그 전체에 공통 스타일을 줄 때
`.price`	`price` class 선택자	`<p class="price">29,000원</p>`	여러 요소에 반복해서 쓸 스타일을 묶을 때
`#sale-price`	`sale-price` id 선택자	`<p id="sale-price">29,000원</p>`	문서에서 하나인 요소를 정확히 가리킬 때
`.site-header a`	`site-header` 안의 `a`	`<header class="site-header">` 안의 링크	특정 영역 안에 있는 요소만 고를 때

점과 샵은 HTML 이름에 포함되지 않습니다.
HTML에는 `class="price"` 라고 쓰고 CSS에서 `.price` 로 찾습니다. HTML에는 `id="sale-price"` 라고 쓰고 CSS에서 `#sale-price` 로 찾습니다.

`#` 는 위치에 따라 뜻이 다릅니다. 규칙의 앞에 있는 `#sale-price` 는 id 선택자지만, `color: #4f6656;` 처럼 속성 값 자리에 있는 `#4f6656` 은 색상 코드입니다.

스타일은 여러 곳에 반복할 수 있는 `class` 를 중심으로 작성합니다. `id` 는 문서 안에서 하나만 사용하며 링크 목적지나 요소 관계처럼 정확한 식별이 필요할 때 주로 사용합니다.

Cascade는 무엇이 겹치고, 무엇이 이긴다는 뜻일까요?

Cascade는 여러 CSS 규칙이 한 요소에 흘러 들어왔을 때 브라우저가 속성별 최종값을 정하는 과정입니다. 서로 다른 속성은 함께 적용되고, **같은 속성이 충돌할 때만** 어느 선언을 사용할지 비교합니다.

가격 요소 하나에 세 규칙이 겹치는 예

```
<p id="sale-price" class="price">29,000원</p>
```

```
p { color: #444444; }
.price { color: #4f6656; }
#sale-price { color: #b33c34; }
```

세 선택자 모두 같은 가격 요소를 찾고 `color` 를 지정합니다. 이때 단순 선택자의 우선순위는 **id 선택자 > class 선택자 > 태그 선택자**이므로 최종 글자색은 `#sale-price` 의 `#b33c34` 가 됩니다.

확인 순서	브라우저가 묻는 질문	이번 예의 결과
1. 같은 대상·속성인가?	세 규칙이 같은 요소의 `color` 를 바꾸는가?	예. 충돌을 비교합니다.
2. 선택자가 더 구체적인가?	태그, class, id 중 어느 선택자가 더 구체적인가?	`#sale-price` 가 이깁니다.
3. 구체성이 같은가?	같은 우선순위라면 어느 규칙이 나중에 작성됐는가?	파일 아래쪽 규칙이 이깁니다.

```
.price { font-weight: 400; }
.price { font-weight: 700; }
```

두 규칙은 선택자가 완전히 같으므로 구체성도 같습니다. 그래서 나중에 작성한 `font-weight: 700` 이 적용됩니다. 반대로 한 규칙은 `color`, 다른 규칙은 `font-weight` 를 지정했다면 충돌하지 않고 두 속성이 모두 적용됩니다.

개발자 도구에서 승자 확인하기

Elements에서 가격을 선택하고 Styles를 보면, 같은 속성에서 진 선언은 취소선으로 표시되고 최종 적용된 선언은 남습니다. 먼저 선택자의 구체성을 비교하고, 같다면 작성 순서를 확인합니다.

`div`는 `product-copy`처럼 별도 의미 없이 여러 요소를 하나의 구역으로 묶을 때 유용합니다. `span`은 한 문단 안의 짧은 텍스트 범위를 묶을 때 쓰며, 기본 `display`가 `inline`입니다. 반면 `div`는 기본적으로 `block`처럼 작동합니다.

---

2. class 선택자와 cascade

> **지금 따라 하기 · style.css**
>
> 1. **위치:** 1단계 코드가 끝난 다음 줄, 즉 `style.css` 맨 아래로 이동합니다.
> 2. **행동:** 기존 코드를 지우지 않고 아래 선택자 규칙을 이어서 추가합니다.
> 3. **저장:** 상단 링크·보조 문구·가격의 글자 모양을 비교합니다.

지금 할 일

HTML에 이미 붙은 `site-header`, `eyebrow`, `price` class를 사용해 네비게이션, 보조 문구, 가격의 역할을 눈에 띄게 만듭니다.

style.css 맨 아래에 추가할 코드

`style.css`의 맨 아래에 다음 규칙을 추가합니다.

```
.site-header a {
  color: var(--color-ink);
  font-size: 1.1rem;
  font-weight: 700;
  text-decoration: none;
}

.eyebrow {
  color: var(--color-accent);
  font-weight: 700;
}

h1 {
  margin: 8px 0 20px;
  font-size: 3rem;
  line-height: 1.15;
}

h2 {
  font-size: 1.75rem;
}

.price {
  font-size: 1.35rem;
  font-weight: 700;
}
```

저장 후 확인

상단 링크와 가격은 굵게, 작은 보조 문구는 지정한 강조색으로 보입니다. `rem`은 문서의 기본 글자 크기를 기준으로 하고, `px`는 CSS 화면 픽셀 기준의 값입니다. 지금은 두 단위를 완벽히 계산하기보다 값을 바꿨을 때 보이는 차이를 관찰합니다.

한 줄 바꿔 보기

가격 요소를 검사해 `.price`의 `font-size` 체크를 끄면 기본 크기로 돌아가고, 다시 켜면 커지는지 확인합니다. Styles에서 선택자별 선언과 취소선을 비교하고, 같은 선택자의 같은 속성을 더 아래에 다른 값으로 임시 추가해 작성 순서도 관찰합니다.

☐ **가격이 굵고 크게 보이며 Styles에서 적용된 `.price` 규칙을 찾았으면 2단계 완료입니다.**

// 3. 박스 모델

브라우저는 모든 요소를 내용(content), 안쪽 여백(`padding`), 테두리(border), 바깥쪽 여백(`margin`)으로 이루어진 박스로 계산합니다. `width`와 `height`에 여백과 테두리를 어떻게 포함할지가 핵심입니다.

박스 모델은 바깥에서 안쪽으로 margin → border → padding → content 순서로 읽습니다.

속성	무엇을 조절하나	확인할 곳
`margin`	다른 박스와의 바깥쪽 간격	박스 밖의 색이 없는 영역
`padding`	내용과 테두리 사이 간격	배경색이 이어지는 안쪽 영역
`border`	박스의 경계선	개발자 도구의 Box model 도식

3. 전역 box-sizing과 본문 글꼴

지금 따라 하기 · style.css

1. **위치:** `style.css` 맨 아래로 이동합니다.
2. **행동:** 아래 `*`와 `body` 규칙을 이어서 추가합니다.
3. **저장:** 페이지 모서리의 기본 여백과 전체 글꼴이 달라지는지 봅니다.

지금 할 일

모든 요소의 선언한 크기 안에 padding과 border가 포함되게 하고, 본문의 기본 색·배경·글꼴·줄 간격을 지정합니다. 외부 웹폰트 없이 운영체제에 있는 글꼴만 사용합니다.

style.css 맨 아래에 추가할 코드

저장할 코드 기존 `body` 규칙을 지우지 말고 아래 선언을 추가합니다. 같은 속성이 있으면 한 `body` 규칙으로 합쳐도 됩니다.

```
* {
  box-sizing: border-box;
}

body {
  margin: 0;
  font-family: system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif;
  line-height: 1.6;
}
```

저장 후 확인

브라우저의 기본 바깥 여백이 사라지고 페이지 전체에 밝은 배경색과 일관된 글꼴이 적용됩니다.

한 줄 바꿔 보기

`body` 의 `margin: 0` 과 `line-height: 1.6` 을 각각 끄고 페이지 모서리 간격과 문장 사이 간격을 관찰합니다. 요소를 선택한 뒤 **Computed**의 박스 모델에서 margin과 padding도 확인합니다.

☐ **페이지 바깥 기본 여백이 사라지고 `line-height` 를 껐다 켠 차이를 설명할 수 있으면 3단계 완료입니다.**

// 4. 상품 영역 디자인

큰 화면에서 본문이 너무 넓게 퍼지면 읽기 어렵습니다. 컨테이너에 유동 너비와 최대 너비를 함께 주면 작은 화면에서는 줄어들고 큰 화면에서는 읽기 좋은 크기에서 멈춥니다.

4. 컨테이너 max-width

> **지금 따라 하기 · style.css**
>
> 1. **위치:** `style.css` 맨 아래로 이동합니다.
> 2. **행동:** 아래 너비·간격 규칙을 기존 코드 다음 줄에 추가합니다.
> 3. **저장:** 브라우저 폭을 넓혔다 줄이며 본문 너비와 가운데 정렬을 확인합니다.

지금 할 일

상단, 본문, 하단을 화면 너비의 90%로 맞추되 1120px보다 커지지 않게 하고 가운데에 배치합니다. 먼저 `width`, `max-width`, `margin: 0 auto` 세 역할만 구분합니다.

style.css 맨 아래에 추가할 코드

```
.site-header,
main,
footer {
  width: 90%;
  max-width: 1120px;
  margin: 0 auto;
}

.site-header {
  padding: 24px 0;
}

main > section {
  margin-bottom: 72px;
}

footer {
  padding: 24px 0 48px;
  color: var(--color-muted);
  border-top: 1px solid var(--color-border);
}
```

저장 후 확인

브라우저 창을 넓혔다 줄이면 본문도 함께 줄어들지만, 넓은 화면에서는 1120px 너머로 늘어나지 않습니다. `margin: 0 auto` 때문에 남은 좌우 공간이 똑같이 나뉩니다.

한 줄 바꿔 보기

`main` 을 검사해 `width`, `max-width`, `margin` 을 하나씩 끄고 켜면 본문 너비와 정렬이 바뀌는지 바로 비교할 수 있습니다.

☐ **넓은 화면에서 본문이 1120px 안에 가운데 정렬되고, 창을 줄이면 함께 줄어들면 4단계 완료입니다.**

// 5. Flexbox

컨테이너

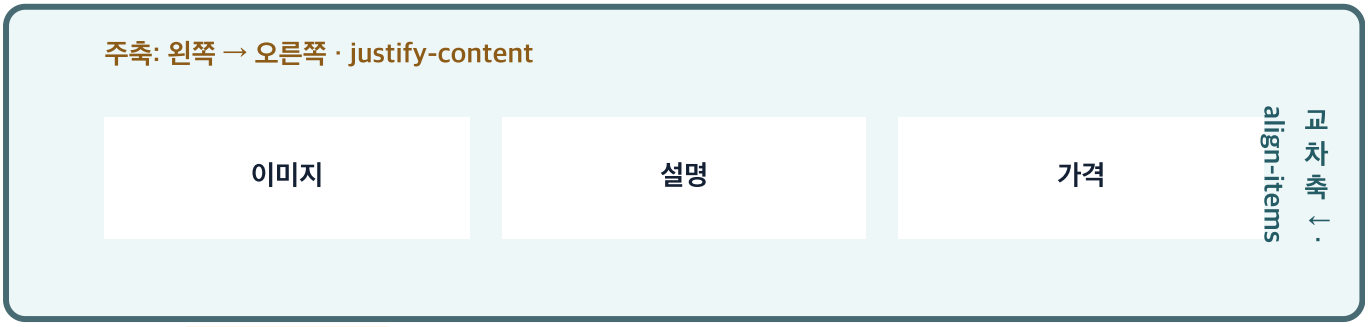
`display: flex` 를 받은 부모입니다. 자식 요소의 배치 규칙을 결정합니다.

주축과 교차축

`flex-direction` 이 주축을 정하고, `justify-content` 와 `align-items` 가 각 축의 정렬을 조절합니다.

자식과 간격

자식은 기본 주축인 가로로 나란히 배치되며, `gap` 이 자식 사이의 간격을 만듭니다.



`display: flex` 를 부모에 적용하면 자식이 주축을 따라 배치되고, 교차축에서 정렬됩니다.

---

5. 상품 hero Flexbox

지금 따라 하기 · style.css

1. **위치:** `style.css` 맨 아래로 이동합니다.
2. **행동:** 아래 Flexbox 규칙을 추가합니다.
3. **저장:** 이미지와 설명이 가로로 놓이는지 확인합니다.

지금 할 일

상품 이미지와 `product-copy` div를 가로로 배치하고, 세로 가운데 정렬과 48px 간격을 적용합니다. 이미지는 컨테이너의 46%를 사용하되 너무 커지지 않게 480px에서 멈춥니다.

style.css 맨 아래에 추가할 코드

```
.product-hero {
  display: flex;
  align-items: center;
  gap: 48px;
}

.product-hero img {
  width: 46%;
  max-width: 480px;
  height: auto;
  border-radius: 24px;
}

.product-copy {
  flex: 1;
}
```

저장 후 확인

이미지와 설명이 가로로 배치됩니다. `gap: 48px`은 두 자식 사이의 간격이고, `height: auto`는 너비가 바뀌어도 원본 종횡비를 유지합니다.

한 줄 바꿔 보기

`.product-hero`의 `display: flex`, `align-items`, `gap`을 순서대로 끄고 켜며 세로 배치, 정렬, 간격의 차이를 관찰합니다. 개발자 도구의 Flex 배지를 켜면 주축과 gap을 눈으로 확인할 수 있습니다.

☐ **이미지와 설명이 가로로 놓이고 `gap` 값을 바꾸면 두 영역의 간격이 달라지면 5단계 완료입니다.**

---

6. CTA 링크와 FAQ 카드

지금 따라 하기 · style.css

1. **위치:** `style.css` 맨 아래로 이동합니다.
2. **행동:** CTA 링크와 FAQ 카드 규칙을 추가합니다.
3. **저장:** 링크와 FAQ의 모양이 각각 달라졌는지 확인합니다.

지금 할 일

FAQ로 이동하는 CTA 링크는 강조색과 넓은 padding으로 버튼처럼 보이게 하고, FAQ는 border·배경·안쪽 여백을 더해 하나의 질문이 하나의 카드로 읽히게 합니다. 모양은 버튼 같아도 HTML 역할은 이동 링크입니다.

style.css 맨 아래에 추가할 코드

```
.product-cta {
  display: inline-block;
  padding: 14px 20px;
  border-radius: 999px;
  color: white;
  background: var(--color-accent);
  font-weight: 700;
  text-decoration: none;
}

.faq-item {
  margin-top: 16px;
  padding: 20px 24px;
  border: 1px solid var(--color-border);
  border-radius: 16px;
  background: var(--color-card);
}

.faq-item h3,
.faq-item p {
  margin: 0 0 8px;
}
```

저장 후 확인

FAQ 이동 링크는 둥근 강조색 버튼처럼, FAQ는 흰 배경의 개별 카드로 보입니다. 모양이 바뀌어도 HTML의 역할과 읽는 순서는 유지됩니다.

한 줄 바꿔 보기

첫 FAQ의 `padding` 과 `margin-top` 을 각각 끄고 안쪽·바깥쪽 여백의 차이를 비교합니다. `border` 와 `background` 도 끄고 켜며 카드의 경계가 어떻게 만들어지는지 확인합니다.

☐ **CTA가 둥근 버튼처럼 보이고 각 FAQ가 흰 카드로 구분되면 6단계 완료입니다.**

// 6. 반응형 화면

반응형 디자인은 같은 HTML을 화면 폭에 맞게 배치하는 방식입니다. media query는 지정한 조건이 맞을 때만 안쪽 CSS를 추가로 적용합니다. 이 예제에서는 720px보다 넓은 768px 화면까지 가로 배치를 유지하고, 390px 화면에서 세로로 바꿉니다.

7. 720px media query

> **지금 따라 하기 · style.css**
>
> 1. **위치:** `style.css`의 가장 마지막 줄로 이동합니다.
> 2. **행동:** 아래 media query를 통째로 추가합니다.
> 3. **저장:** 개발자 도구에서 화면 폭을 768px과 390px로 바꿔 봅니다.

지금 할 일

화면이 720px 이하일 때 상품 영역의 주축을 세로로 바꾸고, 이미지가 컨테이너 너비를 전부 사용하게 합니다.

style.css 맨 아래에 추가할 코드

```
@media (max-width: 720px) {
  .product-hero {
    flex-direction: column;
    align-items: stretch;
  }

  .product-hero img {
    width: 100%;
    max-width: none;
  }

  h1 {
    font-size: 2.25rem;
  }
}
```

저장 후 확인

개발자 도구의 기기 모드에서 768px을 입력하면 가로 배치, 390px을 입력하면 이미지 아래에 설명이 오는 세로 배치가 되어야 합니다.

한 줄 바꿔 보기

390px에서 `flex-direction: column`을 끄면 두 영역이 좁은 폭에 억지로 나란히 남습니다. 다시 켜고 세로 배치를 확인한 뒤, 화면 아래에 가로 스크롤이 생기지 않는지 확인합니다.

☐ **768px에서는 가로, 390px에서는 세로 배치가 되고 두 화면 모두 가로 스크롤이 없으면 7단계 완료입니다.**

완성 뒤 찾아볼 심화 문법

`clamp()`는 최솟값과 최댓값 사이에서 크기를 유동적으로 만들고, `min()`은 여러 후보 중 작은 값을 고릅니다. `margin-block`과 `padding-block`은 글의 진행 방향을 고려한 논리 속성입니다. 모두 유용하지만 이번 주 필수 암기 항목은 아니며, 먼저 위의 고정값과 기본 속성으로 배치 원리를 이해합니다.

// 7. AI와 변경 전후 비교

AI에게 전체 CSS를 다시 쓰게 하면 원하지 않는 선택자까지 바뀔 수 있습니다. 현재 파일, 바꿀 화면, 수정 범위, 먼저 받을 설명을 명시합니다.

제한 수정 프롬프트

현재 HTML은 변경하지 말고 style.css에서 상품 이미지와 설명 사이 간격을 넓히는 최소 수정만 제안해줘. 변경하는 선택자와 속성, 변경 이유를 먼저 설명하고 전체 CSS는 다시 작성하지 마.

1. 예상 수정 범위

- HTML은 바꾸지 않는다.
- `.product-hero`의 간격을 담당하는 `gap`만 후보다.
- 이미지 너비나 텍스트 크기는 바꿀 이유가 없다.

2. AI 결과 비교

- 제안한 선택자가 `.product-hero`인지 본다.
- 변경 속성이 `gap`인지, 기존 반응형 값과 충돌하지 않는지 본다.
- 전체 CSS나 HTML을 다시 작성했다면 적용하지 않고 범위를 다시 요청한다.

파일 수정 전, 개발자 도구에서 먼저 시험

1. `.product-hero` 를 검사합니다.
2. Styles에서 AI가 제안한 `gap` 값을 임시로 입력합니다.
3. 1280px, 768px, 390px에서 간격과 넘침을 확인합니다.
4. 예상한 변화만 있을 때 `style.css` 에 직접 반영하고 저장합니다.

// 8. 화면 QA

완성은 CSS 코드가 저장되었다는 뜻이 아니라, 요구한 화면이 실제 브라우저에서 재현된다는 뜻입니다. 개발자 도구의 기기 모드에 정확한 너비를 입력하고 새로고침한 뒤 확인합니다.

1280px

상품 이미지와 설명이 가로로 배치되고, 본문이 1120px 안에 정렬되는지 확인합니다.

768px

720px보다 넓으므로 가로 배치가 유지되는지, 텍스트와 이미지가 겹치지 않는지 봅니다.

390px

이미지 아래에 설명이 오는 세로 배치인지, 이미지가 본문 너비를 넘지 않는지 봅니다.

390px 최종 확인 화면. 같은 HTML이 세로 흐름으로 바뀌고 이미지가 컨테이너 너비를 사용합니다.

- 세 화면 폭 모두에서 가로 스크롤이 없는가?
- 이미지 종횡비가 원본처럼 유지되는가?
- 버튼 텍스트와 FAQ가 배경에서 충분히 잘 보이는가?
- 브라우저 콘솔에 예상하지 않은 오류가 없는가?
- 2주차 HTML의 제목, 이미지 대체 텍스트, 버튼, FAQ 순서가 유지되는가?

이번 주의 완성 상태

HTML은 내용과 구조, CSS는 모양과 배치를 담고 있습니다. 다음 주에는 같은 페이지에 JavaScript를 연결해 FAQ가 사용자 행동에 반응하게 만듭니다.