1. CSS 개요 및 기본 개념

CSS란 무엇인가?

CSS(Cascading Style Sheets)는
웹 문서의 **스타일(Style)**, 즉 **디자인과 레이아웃**을 지정하는 **스타일 시트 언어**다.
HTML이 웹 페이지의 **구조와 콘텐츠를 정의**한다면,
CSS는 그 구조에 **색을 입히고 형태를 부여**하는 역할을 한다.

핵심 개념 요약

구분	내용
정의	HTML 등의 마크업 문서에 스타일을 적용하는 스타일 시트 언어
목적	웹 페이지의 **디자인, 레이아웃, 색상, 폰트** 등을 설정
사용 방식	HTML에 외부 또는 내부에서 연결하여 사용 (`<style>`, `<link>`)
기본 문법	`선택자 { 속성: 값; }` 형식
핵심 특징	**선택자 기반**, **계단식(Cascading)** 적용 원리

HTML과 CSS의 관계

- HTML: 웹 페이지의 뼈대와 내용 (예: 제목, 문단, 이미지 등)
- CSS: HTML 요소를 꾸미는 옷 (예: 글씨 크기, 색상, 배치 등)

```
<!-- HTML -->
<p class="highlight">이 문장은 중요한 문장입니다.</p>

<!-- CSS -->
<style>
  .highlight {
    color: red;
    font-weight: bold;
  }
</style>
```

Cascading(계단식) 스타일 규칙

"**계단식**"이라는 이름은 CSS가 **여러 스타일 규칙을 병합**하여 최종 스타일을 결정하기 때문임.

- **우선순위 순서**
    1. 브라우저 기본 스타일
    2. 외부 스타일시트

3. 내부 스타일(`<style>`)
4. 인라인 스타일 (`style=""`) ← 가장 우선

- **중복 적용 시 우선권**
  - 구체적인 선택자가 우선
  - 나중에 작성된 규칙이 우선
  - `!important` 사용 시 강제 우선

CSS로 할 수 있는 것들

분야	설명
글꼴 및 텍스트	글자 색, 크기, 정렬, 줄 간격, 글꼴 등 지정
배경	색상, 이미지, 반복, 고정 여부 등
레이아웃	박스 모델, Flexbox, Grid 등으로 배치
애니메이션	요소 이동, 페이드인/아웃, 회전 등
반응형 웹	화면 크기에 따라 유연하게 대응하는 레이아웃 구성

CSS 버전

버전	특징
CSS1 (1996)	기본적인 텍스트 스타일링
CSS2 (1998)	박스 모델, 포지셔닝 등 추가
CSS2.1	CSS2의 안정화 버전
CSS3 (2011~)	모듈화, 애니메이션, 그리드, 미디어 쿼리 등 현대적 기능

요약

> CSS는 웹 페이지를 **아름답게 꾸미고**, 다양한 **기기나 화면 해상도에 유연하게 대응**할 수 있도록 만드는 핵심 언어다.

HTML과 CSS의 관계

HTML과 CSS는 웹 페이지를 구성하는 두 가지 핵심 기술이다. 이 둘은 서로의 역할이 명확하게 분리되어 있으며, 상호 보완적으로 작동한다. 웹을 구성하는 "구조"와 "스타일"의 관계라고 이해하면 된다.

HTML: 웹의 구조와 내용

HTML(HyperText Markup Language)은 웹 페이지의 **기본 구조와 콘텐츠**를 정의한다. 다음과 같은 요소들이 HTML로 구성된다.

- 제목 (`<h1>`~`<h6>`)
- 문단 (`<p>`)
- 이미지 (`<img>`)
- 링크 (`<a>`)
- 리스트 (`<ul>`, `<ol>`)
- 표 (`<table>`)
- 입력 폼 (`<input>`, `<form>` 등)

HTML은 정보를 **표현하고 구조화**하는 언어로, 시각적인 디자인보다는 정보의 **의미와 계층 구조**에 중점을 둔다.

CSS: 웹의 시각적 표현

CSS(Cascading Style Sheets)는 HTML로 정의된 구조에 **스타일을 적용**한다. 즉, 사용자가 보는 웹 페이지의 **색상, 크기, 간격, 배치** 등을 담당한다. CSS를 통해 다음과 같은 스타일링이 가능하다.

- 텍스트 색상 및 크기 조절
- 글꼴 변경
- 요소의 위치 조정 (정렬, 배치)
- 배경 이미지 및 색상 설정
- 여백(padding, margin) 및 테두리(border) 설정
- 애니메이션 및 전환 효과 추가

역할 분리의 중요성

HTML과 CSS는 역할이 명확하게 분리되어 있기 때문에 유지보수가 용이하고, 콘텐츠와 디자인을 독립적으로 관리할 수 있다.

요소	HTML	CSS
목적	콘텐츠와 구조 정의	스타일 및 레이아웃 지정
주요 기능	문서 구조, 의미, 내용	색상, 크기, 위치, 애니메이션
파일 확장자	`.html`	`.css`
문법 형식	`<태그 속성>`	`선택자 { 속성: 값; }`

예시: HTML과 CSS의 결합

```
<!-- HTML 문서 -->
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1 class="title">웹 페이지 제목</h1>
  <p>본문 내용입니다.</p>
</body>
</html>
```

```
/* styles.css */
.title {
  color: navy;
  font-size: 32px;
  text-align: center;
}
```

위 예시에서 HTML은 구조를 담당하고, CSS는 스타일을 분리된 파일로 정의해 외부에서 적용하고 있다.

결론

HTML은 웹의 **내용과 구조**를 담당하고,
CSS는 그 내용에 대한 **시각적 스타일링**을 적용한다.
이 둘을 함께 사용함으로써 정보성과 미적인 요소를 동시에 갖춘 웹 페이지를 만들 수 있다.

CSS의 등장 배경 및 필요성

1. HTML만으로는 표현의 한계가 있었다

웹의 초기 시대(1990년대 초중반)에는 HTML만으로 웹 페이지를 만들었으며, 웹은 단순한 문서 정보 전달 매체였다. 하지만 점점 더 많은 사람들이 웹을 활용하게 되면서 **정보의 시각적 표현**에 대한 수요가 증가하였다.

HTML의 한계점

- 글꼴 크기, 색상, 배치 등을 표현하기 위해 **태그 내에 직접 스타일 속성**을 사용해야 했다 (`<font>`, `bgcolor`, `align` 등).
- 디자인 요소가 HTML 코드에 뒤섞여서 문서 구조가 복잡해졌고, 유지보수가 어려워졌다.
- 여러 페이지에서 공통 디자인을 적용하려면 **페이지마다 스타일을 반복 작성**해야 했다.

---

2. 웹 표준의 필요성 대두

웹의 대중화와 함께 **통일된 디자인 규칙**과 **재사용 가능한 스타일 정의 방식**이 요구되었다. 특히 웹사이트가 수십~수백 페이지로 구성되면서 **디자인 일관성, 유지보수성, 효율성**이 중요한 과제가 되었다.

---

3. CSS의 도입과 역할

CSS는 1996년, W3C(World Wide Web Consortium)에 의해 제안되었고, **HTML의 표현적인 문제를 해결**하기 위한 공식적인 스타일링 언어로 등장했다.

CSS의 주요 목적

- HTML과 **스타일 정보를 분리**한다
- **재사용 가능한 스타일 정의**를 가능하게 한다
- 페이지 전체의 **디자인 일관성**을 유지할 수 있도록 한다
- 브라우저가 **빠르게 렌더링**할 수 있는 구조를 제공한다

---

4. 등장 당시의 문제 해결 사례

문제 상황	CSS 도입 전	CSS 도입 후
디자인 코드 중복	수십 페이지마다 `<font>` 반복	외부 CSS 한 줄로 전체 적용
유지보수	색상 바꾸려면 HTML 문서 전체 수정	CSS 파일만 수정
구조와 스타일 혼합	HTML 코드가 복잡해짐	역할 분리로 구조 명확화
접근성과 확장성	의미 없는 태그의 난무	시맨틱한 구조 유지 가능

---

5. CSS가 가져온 변화

- **웹 표준의 기반** 마련
- **반응형 웹 디자인**, **다크 모드**, **애니메이션**, **그리드/플렉스 레이아웃** 등 고급 기능이 가능해짐
- 다양한 장치(모바일, 태블릿, TV 등)에 적응 가능한 디자인 구현 가능
- HTML은 의미와 구조 중심, CSS는 표현 중심이라는 **역할 분담 확립**

요약

> CSS는 HTML 문서에 **디자인과 레이아웃을 체계적으로 적용하기 위해** 도입된 스타일 시트 언어로,
> 웹 페이지의 **일관성, 효율성, 유지보수성**을 향상시키기 위한 필수 기술이다.

CSS의 구조: 선택자 { 속성: 값; }

CSS는 매우 단순한 문법 구조를 가지고 있다.
기본 형태는 다음과 같다:

```
선택자 {
  속성: 값;
  속성: 값;
}
```

이 구조는 HTML 요소에 대해 "어떤 스타일을 적용할지"를 정의하는 방식이다.
아래에서 각각의 구성 요소를 구체적으로 설명한다.

---

1. 선택자(Selector)

선택자는 **스타일을 적용할 HTML 요소를 지정하는 부분**이다.

예시:

```
p {
  color: blue;
}
```

→ 이 코드는 모든 `<p>` 요소(문단 태그)에 글자 색상을 파란색으로 지정한다.

다양한 선택자 예시:

선택자 형식	의미	예시
태그 선택자	특정 태그 전체에 적용	`h1`, `p`, `div`
클래스 선택자	특정 class 속성에 적용	`.box`, `.title`
ID 선택자	특정 id 속성에 적용	`#header`, `#main`
복합 선택자	조합된 요소에 적용	`div.box`, `ul li`, `#nav .item`

---

2. 중괄호 { }

선택자 다음에는 중괄호를 열어 해당 요소에 적용할 스타일들을 기술한다.
중괄호 내부에는 **속성: 값** 쌍들이 포함된다.

```
.box {
  width: 300px;
  height: 150px;
}
```

---

3. 속성(Property)

속성은 **변경하고자 하는 스타일의 항목**이다. 예: 색상, 너비, 글꼴 등

- `color`: 글자 색상
- `font-size`: 글자 크기
- `margin`: 바깥 여백
- `padding`: 안쪽 여백
- `border`: 테두리
- `background-color`: 배경색

---

4. 값(Value)

각 속성에는 **적용할 구체적인 값**을 지정한다.

- `color: red;` → 글자를 빨간색으로
- `width: 100px;` → 요소의 너비를 100픽셀로
- `text-align: center;` → 텍스트를 가운데 정렬

---

5. 세미콜론(;)

각 `속성: 값` 쌍은 **세미콜론(;)으로 구분**된다.
마지막 속성이라도 세미콜론을 붙이는 것이 유지보수에 유리하다.

예시 전체 구조

```
/* 선택자가 p인 경우 */
p {
  color: #333;
  font-size: 16px;
  line-height: 1.6;
}
```

이 코드는 모든 `<p>` 태그에 대해:

- 글자 색상: 짙은 회색
- 글자 크기: 16픽셀
- 줄간격: 1.6배

를 적용한다.

CSS 문법 정리

요소	설명
선택자	스타일을 적용할 HTML 요소 지정
속성	스타일 항목 이름 (color, width 등)
값	속성에 설정할 구체적인 값 (red, 100px 등)
중괄호	스타일 영역을 묶는 구분자
세미콜론	각 속성 선언의 마침표 역할

브라우저 렌더링 원리 간략 이해

1. HTML 파싱 → DOM 트리 생성

브라우저는 HTML 파일을 받아들여 **파싱(parser)**을 시작한다.

- HTML 문서를 위에서부터 차례로 해석한다
- 각 요소(`<div>`, `<p>`, `<img>`, 등)를 **노드(node)**로 바꾸고
- 이를 **트리 구조(DOM 트리)**로 구성한다

📌 DOM (Document Object Model): HTML 문서의 구조를 계층적으로 표현한 객체 트리

---

2. CSS 파싱 → CSSOM 트리 생성

HTML 문서 안에서 CSS를 발견하면 브라우저는 이를 파싱해 **CSSOM(CSS Object Model)**이라는 트리를 만든다.

- 외부 CSS 파일, `<style>`, 인라인 `style` 등을 모두 수집
- 선택자와 스타일 규칙을 해석하여 각 노드에 적용 가능한 스타일 트리를 구성

📌 CSSOM: CSS의 규칙들을 객체 구조로 표현한 트리

---

3. 렌더 트리(Render Tree) 생성

이제 브라우저는 DOM과 CSSOM을 합쳐서 실제로 화면에 보여줄 정보를 담은 **렌더 트리(render tree)**를 만든다.

- 보이지 않는 요소 (`display: none`)는 제외
- 각 노드에 어떤 스타일이 적용될지를 계산

📌 렌더 트리 = "무엇이", "어떻게" 보여질지를 결정한 트리

---

4. 레이아웃(Layout) 계산 (리플로우)

렌더 트리가 완성되면, **각 요소의 위치와 크기**를 계산한다.

- 요소 간의 상대 위치, 뷰포트에 따른 크기, 폰트 크기, 마진/패딩 등 고려
- 이 과정을 **리플로우(reflow)**라고도 부른다

---

5. 페인팅(Painting)

계산된 레이아웃을 바탕으로 각 요소에 **색상, 텍스트, 이미지, 배경 등**을 실제로 그린다.

- 이 작업은 픽셀 단위로 진행된다
- 글자 색상, 배경색, 테두리, 그림자, 이미지 등 시각적 표현 처리

---

6. 합성(Compositing)

페인팅된 각 레이어를 **최종적으로 병합하여 화면에 출력**한다.

- CSS 애니메이션, 3D 효과, z-index 등의 이유로 레이어가 분리될 수 있음
- GPU를 활용해 여러 레이어를 병합

브라우저 렌더링 요약 순서

```
HTML → DOM 트리
CSS → CSSOM 트리
DOM + CSSOM → 렌더 트리
렌더 트리 → 레이아웃 계산 (reflow)
레이아웃 → 페인팅 (paint)
페인팅 결과 → 합성 → 최종 화면 출력
```

렌더링 최적화와 성능 이슈

- DOM이나 CSSOM이 **변경되면 전체 렌더링 작업이 재실행**될 수 있음
- 레이아웃/페인트/합성이 많이 발생하면 성능 저하로 이어짐
- 특히 JavaScript에서 DOM을 자주 조작하거나, CSS를 빈번히 바꾸면 문제 발생

결론

> 브라우저 렌더링은 HTML과 CSS를 파싱하고, 이를 화면에 시각적으로 출력하기 위한 복잡한 절차다.
> 이 과정을 잘 이해하면, 더 성능 좋은 웹 페이지를 만들고 디버깅에도 유리하다.

CSS의 장단점

CSS는 웹 디자인에서 **필수 불가결한 기술**이다.
HTML로는 구조를, CSS로는 스타일을 정의하며,
두 기술의 분리를 통해 웹의 유지보수성과 표현력을 극대화할 수 있다.

아래는 CSS의 대표적인 **장점과 단점**을 체계적으로 정리한 내용이다.

✅ CSS의 장점

1. 표현력 있는 디자인

- 색상, 레이아웃, 폰트, 그림자, 애니메이션 등 시각 요소를 자유롭게 제어할 수 있다.
- Flexbox, Grid 등의 레이아웃 시스템은 복잡한 구조도 쉽게 표현 가능.

2. 구조와 표현의 분리

- HTML은 구조, CSS는 디자인을 담당하여 역할이 분명하다.
- 이는 코드 가독성과 유지보수성을 높인다.

3. 코드 재사용 및 일관성 유지

- 외부 CSS 파일을 여러 HTML 문서에 공통 적용할 수 있다.
- 하나의 CSS 변경만으로 전체 사이트의 디자인을 일괄 수정 가능.

4. 반응형 웹 디자인 지원

- 미디어 쿼리 등으로 화면 크기별 대응이 가능하여 모바일, 태블릿, PC 모두 지원 가능.
- 다양한 단위(`vw`, `vh`, `%`, `em`, `rem`)를 활용하여 유연한 레이아웃 설계가 가능하다.

5. 웹 표준 기반의 브라우저 지원

- 대부분의 모던 브라우저에서 지원됨.
- 벤더 프리픽스 없이도 최신 CSS 기능이 점점 표준화되고 있다.

6. 속도 향상

- 외부 CSS를 캐싱하면 페이지 로딩 속도가 빨라진다.
- `<style>`이 아닌 링크 방식 사용 시 CSS 분리로 HTML 파일이 가벼워진다.

❌ CSS의 단점

1. 복잡한 우선순위 및 상속 규칙

- 동일한 요소에 여러 스타일이 적용될 경우 우선순위 판단이 어렵다.
- `!important`, 상속, 계단식 적용 등으로 인해 예측하지 못한 결과가 나올 수 있다.

2. 대규모 프로젝트에서의 유지 어려움

- 클래스명이 충돌하거나 스타일이 꼬일 수 있다.
- 컴포넌트 기반 개발에서 전역 스타일이 문제를 일으킬 수 있다.

3. 논리 기반 조건 처리가 어렵다

- if/else, 반복문 같은 프로그래밍적 로직이 없다.
- 스타일 조건 분기를 위해 JS 또는 CSS preprocessor(SASS 등)에 의존해야 한다.

4. 브라우저 간 호환성 이슈

- 같은 CSS라도 브라우저별로 표현이 다를 수 있다.
- 예전 버전의 브라우저에서 일부 속성이 지원되지 않음 (ex. `grid`, `clamp()` 등)

5. 디버깅이 까다로움

- 어떤 CSS가 어떤 요소에 영향을 주는지 추적하기가 쉽지 않다.
- 개발자 도구 없이는 분석이 어렵고, 복잡한 계층 구조에서는 더 복잡해진다.

요약: CSS 장단점 표

구분	장점	단점
구조	HTML과 분리되어 구조화 우수	우선순위 규칙이 복잡함
재사용성	외부 스타일시트로 재사용 가능	전역 스타일로 충돌 가능
표현력	다양한 레이아웃과 애니메이션 지원	논리 처리 불가능
반응형	다양한 디바이스 대응 가능	미디어 쿼리 유지 어려움
성능	브라우저 캐시 활용 가능	브라우저 호환성 문제 존재

결론

> CSS는 웹 디자인의 강력한 도구로써 다양한 표현과 유연한 구조를 제공하지만,
> 규모가 커질수록 **관리와 설계의 체계화가 매우 중요**하다.
> 이를 해결하기 위해 BEM, SCSS, CSS-in-JS, Tailwind 등의 보완 기술이 등장하게 되었다.