2. CSS 적용 방법

인라인 스타일

인라인 스타일은 HTML 요소에 직접 `style` 속성을 사용하여 CSS를 작성하는 방식이다.
CSS의 3가지 적용 방식 중 가장 우선순위가 높으며, 가장 빠르게 적용할 수 있는 방법이다.

---

1. 기본 문법

```
<태그명 style="속성: 값; 속성: 값;">
```

예시:

```
<p style="color: blue; font-size: 18px;">
  인라인 스타일로 지정된 문장입니다.
</p>
```

이 문장은 다음과 같은 스타일을 갖는다:

- 글자 색: 파란색 (`color: blue`)
- 글자 크기: 18px (`font-size: 18px`)

---

2. 특징

항목	설명
적용 위치	HTML 요소 내부의 `style` 속성
적용 대상	해당 **요소 1개만** 적용
우선순위	외부/내부 CSS보다 **가장 높음** (단, `!important` 제외 시)
선언 방식	속성과 값들을 세미콜론(;)으로 구분하여 연속 작성

---

3. 장점

- 매우 **간단하고 빠르게** 적용 가능
- 외부 파일이 없어도 바로 적용됨
- JavaScript와 동적으로 쉽게 연동 가능

```
element.style.color = "red";
```

---

4. 단점

- **재사용 불가능**: 같은 스타일을 여러 요소에 적용하려면 반복해서 작성해야 함
- **유지보수 어려움**: 스타일이 HTML에 섞여 있어 코드가 복잡해지고, 일괄 수정이 어렵다
- **HTML과 CSS의 책임 분리 위반**: 구조와 표현이 섞여 웹 표준 철학에 어긋남
- **스타일 일관성 유지 어려움**: 대규모 페이지에서 혼란 초래

---

5. 인라인 스타일 vs 외부 스타일 예시 비교

인라인 스타일:

```
<button style="background-color: green; color: white;">확인</button>
<button style="background-color: green; color: white;">취소</button>
```

→ 두 버튼에 같은 스타일을 반복 작성해야 함

외부 CSS 사용 시:

```
<!-- HTML -->
<button class="btn">확인</button>
<button class="btn">취소</button>

<!-- CSS -->
.btn {
  background-color: green;
  color: white;
}
```

→ CSS 한 줄로 유지보수 가능

---

6. JavaScript로 인라인 스타일 조작

```
<div id="box">박스</div>

<script>
  const box = document.getElementById("box");
  box.style.width = "200px";
  box.style.backgroundColor = "lightgray";
</script>
```

→ JavaScript는 인라인 스타일 속성(`element.style`)을 통해 직접 스타일을 조작할 수 있다.

결론

> 인라인 스타일은 빠르게 테스트하거나 동적으로 스타일을 적용할 때 유용하지만,
> **재사용성과 유지보수성을 해친다**는 점에서 **실제 프로젝트에서는 제한적으로 사용하는 것이 좋다.**

내부 스타일시트 (`<style>`)

내부 스타일시트란, CSS 코드를 HTML 문서의 `<head>` 태그 내부에 `<style>` 요소로 직접 삽입하는 방식이다.
HTML 파일과 CSS 코드가 하나의 파일 안에 포함되어 있기 때문에 외부 파일을 따로 만들지 않고도 웹 페이지 전체에 스타일을 적용할 수 있다.

---

1. 기본 문법 구조

```
<!DOCTYPE html>
<html>
<head>
  <style>
    선택자 {
      속성: 값;
      속성: 값;
    }
  </style>
</head>
<body>
  <!-- HTML 요소들 -->
</body>
</html>
```

---

2. 예제

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: darkblue;
      font-family: Arial, sans-serif;
    }

    p {
      font-size: 16px;
      line-height: 1.6;
    }
  </style>
</head>
<body>
  <h1>내부 스타일시트 예제</h1>
  <p>이 문장은 내부 스타일시트에 의해 스타일이 적용되었습니다.</p>
</body>
```

```
</html>
```

3. 특징

항목	설명
적용 위치	`<head>` 태그 안 `<style>` 요소
적용 대상	HTML 문서 전체
사용 용도	단일 HTML 문서에서 CSS를 구조적으로 작성하고자 할 때
문법 형식	외부 스타일시트와 동일한 선택자 기반 구조 사용

4. 장점

- **재사용 가능**: 하나의 스타일 선언으로 여러 요소에 적용 가능
- **코드 분리 유지**: 인라인 스타일보다 구조와 표현이 더 잘 분리됨
- **디버깅 편의성**: 한 HTML 문서 내에서 모든 코드를 확인 가능
- **적은 파일 수**: 외부 파일 없이 하나의 파일로 구현 가능 (단일 문서 프로젝트에 적합)

5. 단점

- **재사용 어려움**: 다른 HTML 문서와 스타일을 공유할 수 없음
- **HTML과 CSS의 완전한 분리 불가**: 스타일이 HTML 파일 안에 존재
- **코드 양이 많아질수록 가독성 저하**: 큰 프로젝트에 부적합
- **캐싱 비효율**: 브라우저가 내부 스타일은 별도로 캐싱하지 못함

6. 내부 스타일시트 vs 인라인 스타일 vs 외부 스타일시트

구분	위치	적용 범위	재사용성	유지보수
인라인	요소 내부 `style` 속성	해당 요소 1개	없음	최악
내부	`<style>` 태그	문서 전체	없음	보통
외부	`.css` 파일	여러 문서	있음	우수

7. 실무에서의 활용 예시

- 프로토타입, 샘플 페이지 제작 시
- 외부 리소스를 불러오기 어려운 환경 (예: 이메일 템플릿)
- 빠르게 테스트하거나 코드 샌드박스에서 데모할 때

결론

> 내부 스타일시트는 **단일 문서용 웹 페이지**나 **개발 초기 테스트용**으로는 적합하지만,
> 규모가 커지면 반드시 **외부 스타일시트**로의 분리와 모듈화가 필요하다.

외부 스타일시트 (`<link rel="stylesheet">`)

외부 스타일시트는 CSS를 HTML 문서와 완전히 분리된 **별도의 `.css` 파일**로 작성하고,
이를 HTML 문서의 `<head>` 태그 안에서 `<link>` 요소를 통해 불러오는 방식이다.
웹 개발에서 가장 **권장되는 스타일 적용 방법**이며, 유지보수성과 확장성이 가장 뛰어나다.

---

1. 기본 문법

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

- `rel="stylesheet"`: 이 링크가 스타일시트임을 명시
- `href="..."`: CSS 파일의 경로 (상대경로 또는 절대경로)

---

2. 예제

HTML 파일 (`index.html`)

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>외부 스타일시트 예제</h1>
  <p>이 문장은 외부 CSS 파일에 의해 스타일이 적용됩니다.</p>
</body>
</html>
```

CSS 파일 (`style.css`)

```css
h1 {
  color: darkgreen;
  font-family: 'Segoe UI', sans-serif;
}

p {
  font-size: 17px;
  line-height: 1.8;
}
```

---

3. 특징

항목	설명
위치	HTML `<head>` 태그 내 `<link>` 요소
적용 대상	여러 HTML 파일에서 재사용 가능
문법	일반적인 CSS 문법 (`선택자 { 속성: 값; }`) 사용
파일 분리	HTML과 CSS가 완전히 분리되어 있음

---

4. 장점

- **재사용 가능**: 하나의 CSS 파일을 여러 HTML 문서에서 공유 가능
- **유지보수 용이**: 한 곳만 수정하면 모든 페이지에 자동 반영됨
- **캐싱 최적화**: 브라우저가 외부 CSS 파일을 **한 번 다운로드 후 재사용** 가능
- **가독성 향상**: HTML과 CSS가 분리되어 각각 명확하게 관리 가능
- **대규모 프로젝트 필수 구조**: 팀 개발 및 프레임워크에서도 기본 전제 방식

---

5. 단점

- **외부 파일 필요**: HTML 파일만으로 완전한 결과를 확인하기 어려움
- **초기 로딩 지연**: 외부 리소스를 불러오기까지 약간의 시간 소요
- **상대경로 실수 가능성**: `href` 경로가 틀리면 스타일이 적용되지 않음

---

6. 사용 팁

- CSS 파일은 일반적으로 `/css/style.css`처럼 폴더에 정리한다.
- 여러 개의 외부 스타일시트를 연결할 수도 있다:

```
<link rel="stylesheet" href="reset.css">
<link rel="stylesheet" href="main.css">
```

- `media` 속성을 활용해 조건부 스타일 적용 가능:

```
<link rel="stylesheet" href="print.css" media="print">
```

---

7. 외부 스타일시트 권장 상황

- **반복되는 디자인 패턴이 많은 웹사이트**
- **SPA/MPA, CMS 기반 사이트, 쇼핑몰 등 규모 있는 프로젝트**
- **다국어 페이지에서 스타일을 공유하는 경우**
- **프레임워크 기반(React, Vue, Spring 등) 개발 시 전제 조건**

결론

> 외부 스타일시트는 **웹 개발의 표준 방식**으로,
> 코드 재사용, 성능 최적화, 유지보수 효율성 측면에서 **가장 이상적인 CSS 적용 방법**이다.

`@import` 방식

`@import`는 CSS 파일 안에서 **다른 CSS 파일을 불러올 때 사용하는 지시문**이다.
HTML에서 직접 `<link>`를 사용하는 대신, CSS 내부에서 **다른 CSS 파일을 연결**하는 방식이다.

---

1. 기본 문법

```
@import url("파일경로.css");
```

또는

```
@import "파일경로.css";
```

예시:

```
/* main.css */
@import url("reset.css");

body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
}
```

위 코드는 `main.css`를 불러올 때 `reset.css`도 함께 불러오게 된다.

---

2. 사용 위치

- `@import`는 **CSS 파일의 가장 첫 줄에 위치**해야 한다.
- `@charset` **다음에만 올 수 있고**, 다른 규칙보다 먼저 나와야 한다.

```
@charset "UTF-8";
@import url("theme.css");

/* 이 아래부터 일반 CSS 작성 */
```

---

3. HTML에서 직접 사용하는 경우

CSS를 HTML에서 직접 작성할 때도 `@import`를 사용할 수 있다:

```
<head>
  <style>
    @import url("style.css");
  </style>
</head>
```

하지만 이는 **외부 스타일시트를 직접 `<link>`로 연결하는 것보다 덜 권장**된다.

---

4. `@import`와 `<link>`의 차이점

항목	`<link>`	`@import`
선언 위치	HTML `<head>` 안	CSS 파일 내부 또는 `<style>` 안
실행 시점	**HTML 파싱 시 동기적으로 로드**	**CSS 파싱 후 비동기적으로 로드**
속도	빠름 (병렬 다운로드)	느림 (병렬 처리 안됨, 렌더링 지연 가능성)
조건부 로딩	`media` 속성 지원	일부 제한적 지원
IE 호환성	완전 지원	일부 구버전 IE에서 문제 발생 가능

---

5. 장점

- **모듈화에 유리**: CSS 파일을 논리적으로 분리해서 불러올 수 있음
- **컴포넌트 기반 CSS 구성 가능**: reset, layout, theme 등 역할에 따라 분리 가능
- **간단한 구문으로 여러 CSS 연결 가능**

---

6. 단점 및 주의사항

- **렌더링 지연**: 외부 CSS를 병렬 로딩하지 못하고, 순차적으로 처리 → 성능 저하
- **브라우저 캐싱 비효율**: `<link>` 보다 캐싱 활용도가 떨어질 수 있음
- **우선순위 모호**: 여러 개의 `@import`를 순서 없이 나열하면 예측이 어려움
- **실무에서는 권장되지 않음**: 특히 초기 로딩 속도에 민감한 프로젝트에서는 피해야 함

---

7. 실제 사용 예시 (모듈화 목적)

```
/* style.css */
@import "reset.css";
@import "layout.css";
@import "colors.css";

/* 실제 스타일 정의 */
body {
  font-size: 16px;
  line-height: 1.5;
}
```

결론

> `@import`는 CSS 파일 간의 **의존성 분리와 코드 모듈화**에는 유리하지만,
> **성능 측면에서는 외부 `<link>` 방식보다 떨어지며**,
> **실무에서는 대체로 제한적이거나 빌드 도구(SASS, PostCSS 등)를 통해 처리**하는 것이 일반적이다.

스타일 우선순위와 적용 순서

CSS에서는 하나의 요소에 여러 스타일이 적용될 수 있다.
이때 브라우저는 **어떤 스타일을 우선적으로 적용할지**를 판단해야 하며, 이 과정은 **계단식(Cascade)**이라는 이름으로 정의된다.

이 우선순위는 다음 세 가지 원칙을 통해 결정된다:

---

1. 출처(Source) 우선순위

CSS는 적용 위치에 따라 다음과 같은 **출처 계층**을 가진다:

우선순위	출처 종류	설명
1순위	인라인 스타일 (`style=""`)	HTML 요소에 직접 지정
2순위	내부 스타일시트 (`<style>`)	HTML 문서 안의 `<head>`에서 지정
3순위	외부 스타일시트 (`<link>`)	별도의 `.css` 파일
4순위	브라우저 기본 스타일	브라우저가 제공하는 초기값 (user agent stylesheet)

❗ 단, `!important`가 사용되면 위 규칙보다 **더 강력한 우선순위**를 가진다 (아래에서 설명).

---

2. 선택자 구체성(Specificity)

같은 요소에 여러 규칙이 적용되더라도 **선택자가 더 구체적일수록 우선 적용**된다.

구체성 점수 계산 방식

선택자 종류	점수 방식 (숫자형)
인라인 스타일	1000점
ID 선택자	100점
클래스/속성/가상 클래스	10점
태그/가상 요소	1점
전체 선택자 `*`	0점

예시 비교

```
/* 구체성: 1점 */
p { color: black; }

/* 구체성: 10점 */
.text { color: blue; }

/* 구체성: 100점 */
#main { color: green; }

/* 구체성: 1000점 */
<p style="color: red;">
```

적용 결과: **빨간색(`style=""`)이 최종 적용됨**

---

3. 선언 순서(Order of Appearance)

동일한 구체성을 가진 규칙일 경우 **나중에 선언된 스타일이 우선** 적용된다.

```
p {
  color: blue;
}
p {
  color: green;  /* 이 값이 적용됨 */
}
```

---

4. `!important`의 역할

`!important`는 모든 우선순위 계산을 무시하고 **강제 적용**된다.

```
p {
  color: red !important;
}
```

이 선언은 **구체성이 낮더라도**, 어떤 다른 스타일보다 강하게 적용된다.

단, 아래와 같은 경우 **두 개 모두** `!important`일 때는 다시 구체성 비교를 따진다.

---

5. 종합 우선순위 판단 흐름

브라우저는 스타일 충돌 시 다음과 같은 절차로 적용할 스타일을 결정한다:

```
1. !important 여부 확인 → 있으면 무조건 우선
2. 구체성 점수 계산 → 점수 높은 선택자가 우선
3. 동일 점수면 → 나중에 작성된 것이 우선
```

---

6. 우선순위 예제 정리

```
<!DOCTYPE html>
<html>
<head>
  <style>
    /* 구체성 1 */
    p {
      color: blue;
    }

    /* 구체성 10 */
    .highlight {
      color: green;
    }

    /* 구체성 100 */
    #special {
      color: purple;
    }
  </style>
</head>
<body>
  <p id="special" class="highlight" style="color: red;">
    이 텍스트의 색은 무엇일까요?
  </p>
</body>
</html>
```

- `style="color: red;"`: 1000점 (가장 강함) → **빨간색이 적용됨**

요약 정리표

규칙	내용
1. 출처 우선순위	인라인 > 내부 > 외부 > 브라우저 기본
2. 선택자 구체성	ID > 클래스 > 태그
3. 선언 순서	나중에 작성된 규칙이 우선
4. !important	모든 규칙보다 우선 적용 (단, 동일한 경우 구체성 비교)