5. 색상 및 단위

색상 지정 방식

CSS에서는 **텍스트, 배경, 테두리 등 시각적 요소의 색상**을 다양한 방식으로 지정할 수 있다.
각 방식은 **표현력, 정밀도, 투명도, 가독성** 등에 차이가 있으며, 상황에 맞게 선택하는 것이 중요하다.

---

1. 색상 이름 (Color Keywords)

문법

```
color: red;
background-color: black;
```

설명

- CSS가 정의한 **147개 표준 색상 이름** 중 하나를 사용
- 예: `black`, `white`, `red`, `blue`, `gray`, `lightblue`, `tomato`, `gold`

장점

- 짧고 직관적임
- 빠르게 테스트하거나 프로토타입 만들 때 유용

단점

- **색상 선택 범위가 제한됨**
- 디자이너 입장에선 구체적인 색상 제어가 어려움

---

2. HEX (16진수 색상 코드)

문법

```
color: #RRGGBB;
color: #RGB;
```

설명

- 16진수로 **빨강(R), 초록(G), 파랑(B)**의 값을 각각 지정 (00~~FF = 0~~255)
- `#ffffff` = 흰색, `#000000` = 검정
- 축약형 `#abc`는 `#aabbcc`와 동일

예시

```
color: #3498db;     /* 파란색 계열 */
background: #ffcc00; /* 노란색 계열 */
```

장점

- 웹 전통 방식
- 짧고 읽기 쉬움

단점

- **가독성 떨어짐**
- **투명도 표현 불가능**

---

3. `rgb()` 함수

문법

```
color: rgb(255, 0, 0);
```

설명

- `rgb(R, G, B)` 형식으로 0~255 범위의 정수값으로 색상 지정
- 정확한 색상 조절이 가능함

장점

- 시각적으로 명확하게 구성됨
- JavaScript와의 연동 시 용이

단점

- 긴 코드
- **투명도 표현 불가**

---

4. `rgba()` 함수 (투명도 지원)

문법

```
color: rgba(255, 0, 0, 0.5);  /* 빨간색, 50% 투명 */
```

설명

- `a`는 **alpha(알파 채널, 투명도)**: 0(완전 투명) ~ 1(불투명)
- 요소의 색상이 **배경을 통해 비쳐 보이게** 만들 수 있음

---

5. `hsl()` 함수

문법

```
color: hsl(120, 100%, 50%);
```

설명

- 색조(Hue): 0~360도 (0 = 빨강, 120 = 초록, 240 = 파랑)
- 채도(Saturation): %로 지정 (0% = 회색, 100% = 선명한 색)
- 명도(Lightness): %로 지정 (0% = 검정, 100% = 흰색)

장점

- **디자이너 친화적**
- 색상 톤 조절이 쉬움

---

6. `hsla()` 함수 (투명도 포함)

```
color: hsla(240, 100%, 50%, 0.5);
```

- `hsl()` 방식에 `alpha`를 추가한 형태
- 디자인 시스템 기반 색상 조절에 유용

---

7. 최신 CSS 기능: `color-mix()`, `color()` (CSS4)

```
color: color-mix(in srgb, red 40%, blue);
```

- 두 색상을 **혼합(mix)**하거나, **sRGB 외의 색 공간**을 지정할 수 있음
- 현재는 최신 브라우저에서만 부분적으로 지원됨

비교 요약

방식	형식	투명도 지원	장점	단점
키워드	`red`	X	쉽고 빠름	범위 제한
HEX	`#ff0000`	X	웹 전통, 짧음	불투명

방식	형식	투명도 지원	장점	단점
RGB	`rgb(255,0,0)`	X	명확한 수치	투명도 없음
RGBA	`rgba(255,0,0,0.5)`	O	투명도 포함	코드 길음
HSL	`hsl(0,100%,50%)`	X	톤 조절 용이	초기 학습 필요
HSLA	`hsla(0,100%,50%,0.3)`	O	디자인 유리	가독성 낮음
color-mix	`color-mix(...)`	O	혼합 가능	실험적, 지원 제한

결론

> CSS 색상 지정은 목적에 따라 다양한 방식으로 제공되며,
> **간단한 경우엔 키워드나 HEX**,
> **정확한 색 조절에는 RGB/HSL**,
> **투명도 표현에는 RGBA/HSLA**,
> **고급 조합에는 color-mix** 등을 선택적으로 사용할 수 있다.

단위

CSS에서 **길이, 크기, 위치, 여백 등 수치적 스타일을 지정할 때**는 반드시 **단위**가 함께 사용되어야 한다.
단위는 크게 **절대 단위**와 **상대 단위**로 나뉘며, 각각의 특성과 사용 목적에 따라 선택해야 한다.

---

1. 절대 단위 (Absolute Units)

화면 크기나 글꼴 크기와 **무관하게 고정된 크기**를 의미한다.
인쇄물이나 픽셀 정확도가 필요한 경우 주로 사용된다.

단위	의미	비고
`px`	픽셀	가장 널리 사용됨. 화면에서 고정된 크기
`pt`	포인트 (1pt = 1/72인치)	인쇄물 기준
`cm`	센티미터	인쇄용
`mm`	밀리미터	인쇄용
`in`	인치	1in = 2.54cm
`pc`	파이카 (1pc = 12pt)	드물게 사용됨

예시

```
p {
  font-size: 16px;
}
```

> `px`은 스크린에 고정된 크기이므로 디바이스 해상도나 줌 레벨에 따라 상대적으로 다르게 보일 수 있음

---

2. 상대 단위 (Relative Units)

상대 단위는 **부모 요소, 루트 요소, 뷰포트** 등을 기준으로 크기를 계산한다.
반응형 웹, 접근성 향상, 유연한 디자인에 필수적이다.

A. `em`

- **부모 요소의 `font-size`를 기준**으로 배율을 지정
- `1em`은 현재 요소의 폰트 크기와 동일한 크기

```
body {
  font-size: 16px;
}

p {
  font-size: 1.5em; /* → 24px */
}
```

B. `rem` (Root EM)

- **루트 요소(`<html>`)의 폰트 크기를 기준**으로 배율 지정
- 부모의 영향을 받지 않음

```
html {
  font-size: 16px;
}

h1 {
  font-size: 2rem; /* → 32px */
}
```

✅ 실무에서 `rem`은 **일관된 크기 제어와 접근성 대응에 매우 선호됨**

C. `%` (퍼센트)

- **부모 요소의 해당 속성을 기준**으로 비율을 지정
- `width`, `height`, `margin`, `padding` 등에서 자주 사용됨

```
.container {
  width: 80%;
}
```

D. 뷰포트 단위 (`vw`, `vh`, `vmin`, `vmax`)

단위	기준	설명
`vw`	viewport width	브라우저 너비의 1%
`vh`	viewport height	브라우저 높이의 1%
`vmin`	작은 쪽 기준	`min(vw, vh)`
`vmax`	큰 쪽 기준	`max(vw, vh)`

예시

```
section {
  height: 100vh; /* 화면 전체 높이 */
  font-size: 5vw; /* 화면 너비 기준 글자 크기 */
}
```

→ 반응형 페이지에서 매우 유용

---

3. 계산 함수: `calc()`

다양한 단위를 연산으로 혼합할 수 있음

```
width: calc(100% - 50px);
font-size: calc(1rem + 0.5vw);
```

→ 실무에서 동적인 레이아웃 계산에 자주 사용됨

---

4. 정리 표

단위	설명	기준
`px`	고정 픽셀	절대 단위
`em`	부모 폰트 크기 기준	상대 단위
`rem`	루트 폰트 크기 기준	상대 단위
`%`	부모 요소 기준 비율	상대 단위
`vw`	뷰포트 너비 1%	상대 단위
`vh`	뷰포트 높이 1%	상대 단위
`vmin`	뷰포트의 작은 축 1%	상대 단위
`vmax`	뷰포트의 큰 축 1%	상대 단위

단위	설명	기준
`pt`, `cm`, `mm`, `in`, `pc`	인쇄 단위	절대 단위

결론

> CSS 단위는 **디자인의 유연성과 정밀도 모두에 직결되는 요소**이며,
> 고정된 레이아웃에는 `px`, 반응형/적응형 레이아웃에는 `rem`, `vw`, `%` 등을 혼합 사용하는 것이 일반적이다.