

6. 텍스트 스타일링

글꼴 지정: font-family

CSS의 `font-family` 속성은 HTML 요소에 사용할 글꼴(폰트)을 지정하는 속성이다. 웹사이트의 가독성, 분위기, 디자인 톤앤매너를 결정짓는 중요한 스타일 속성 중 하나다.

1. 기본 문법

```
1 selector {  
2   font-family: [글꼴1], [글꼴2], [글꼴3], [글꼴 계열];  
3 }
```

- 쉼표(,)로 나열된 글꼴 리스트는 앞에서부터 순차적으로 적용되며,
- 브라우저가 해당 글꼴을 지원하지 않을 경우 다음 글꼴로 대체됨
- 마지막에는 반드시 글꼴 계열(generic family)을 지정하는 것이 안전함

2. 글꼴 이름 표기법

- 여러 단어로 이루어진 글꼴 이름은 따옴표(" ") 또는 작은따옴표(' ')로 묶음

```
1 font-family: "Times New Roman", Georgia, serif;
```

- 한 단어로 된 글꼴 이름은 따옴표 없이 사용 가능

```
1 font-family: Arial, sans-serif;
```

3. 주요 글꼴 계열(Generic Families)

계열	설명
<code>serif</code>	장식이 있는 전통적인 인쇄체 (ex: Times New Roman)
<code>sans-serif</code>	장식 없는 현대적 글꼴 (ex: Arial, Helvetica)
<code>monospace</code>	고정 폭 글꼴 (ex: Courier New, 개발자 코드용)
<code>cursive</code>	손글씨 스타일 (ex: Comic Sans)
<code>fantasy</code>	장식적 글꼴 (게임, 제목 등 제한적 용도)
<code>system-ui</code>	시스템 기본 UI 글꼴 (플랫폼별 다름)

4. 예시

```
1 body {  
2   font-family: "Helvetica Neue", Arial, sans-serif;  
3 }
```

- Helvetica Neue → 지원되면 사용
- 없으면 Arial 사용
- 둘 다 없으면 기본 sans-serif 계열 글꼴로 대체됨

5. 웹 안전 글꼴(Web-safe Fonts)

브라우저와 운영체제에 기본 탑재된 글꼴로, 거의 모든 환경에서 사용 가능

글꼴 이름	계열	비고
Arial	sans-serif	현대적, 가독성 좋음
Times New Roman	serif	전통적 문서 느낌
Courier New	monospace	코드나 타자기 스타일
Georgia	serif	Times보다 넓고 부드러운 느낌
Verdana	sans-serif	큰 글자에 최적화
Trebuchet MS	sans-serif	산뜻하고 가벼운 느낌
Comic Sans MS	cursive	장난스럽고 캐주얼한 글꼴

6. 웹 폰트 (Web Fonts) - @font-face, Google Fonts

사용자의 시스템에 없는 글꼴도 웹폰트로 불러올 수 있음

Google Fonts 사용 예

```
1 <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap"  
  rel="stylesheet">
```

```
1 body {  
2   font-family: 'Roboto', sans-serif;  
3 }
```

직접 선언

```
1 @font-face {  
2   font-family: 'MyFont';  
3   src: url('MyFont.woff2') format('woff2');  
4 }
```

7. 실무 팁

- 항상 여러 글꼴을 **웹표로 백업** 구성하자
- 가능한 경우 **rem**, **em**으로 **폰트 크기** 지정해 접근성과 반응형을 보장하자
- **한글 웹폰트**는 **Noto Sans KR**, **Spoqa Han Sans**, **Pretendard** 등이 자주 쓰임

```
1 body {  
2   font-family: 'Pretendard', 'Apple SD Gothic Neo', sans-serif;  
3 }
```

결론

font-family는 웹 페이지의 **시각적 정체성**과 **정보 전달력**을 좌우하는 핵심 속성이다.

여러 글꼴을 순서대로 지정해 폴백을 구성하고, **웹 폰트**와 **기본 글꼴**을 **조화롭게 조합**하는 것이 중요하다.

크기 조절: font-size

font-size는 CSS에서 **텍스트의 크기**를 지정하는 속성이다.

웹 페이지의 **가독성**, **시각적 계층 구조**, **반응형 대응**에 매우 중요한 역할을 한다.

1. 기본 문법

```
1 선택자 {  
2   font-size: 값;  
3 }
```

```
1 p {  
2   font-size: 16px;  
3 }
```

2. 사용 가능한 단위

A. 절대 단위

단위	설명
px	픽셀, 가장 많이 사용되는 고정 단위
pt	포인트, 인쇄 용도 (1pt = 1/72인치)
cm, mm, in	실측 단위, 인쇄물용 (웹에서는 거의 사용하지 않음)

B. 상대 단위

단위	기준	설명
%	부모 요소 크기	100%는 부모와 동일 크기
em	부모 font-size 기준	1em = 현재 글꼴 크기
rem	루트 <html>의 font-size 기준	1rem = 전역 기준
vw, vh	뷰포트 너비/높이 기준	반응형 글꼴에 사용

3. 예시

고정 크기

```
1 h1 {
2   font-size: 32px;
3 }
```

상대 크기 (em)

```
1 body {
2   font-size: 16px;
3 }
4 p {
5   font-size: 1.25em; /* → 20px */
6 }
```

루트 기준 (rem)

```
1 html {
2   font-size: 16px;
3 }
4 h2 {
5   font-size: 2rem; /* → 32px */
6 }
```

4. 키워드 값

CSS는 다음과 같은 미리 정의된 키워드도 지원한다:

```
1 font-size: small;  
2 font-size: medium; /* 기본값, 보통 16px */  
3 font-size: large;  
4 font-size: x-small;  
5 font-size: xx-large;
```

키워드는 브라우저, OS, 사용자 설정에 따라 결과가 달라질 수 있음

5. 계산 함수와 혼합 사용

```
1 font-size: calc(1rem + 0.5vw);
```

- 기본 크기 + 반응형 크기를 조합하여 유동적인 글꼴 크기를 만들 수 있음
-

6. 실무에서의 전략

기본 설정

```
1 html {  
2   font-size: 16px; /* 1rem = 16px 기준 */  
3 }
```

제목과 본문



```
1 h1 { font-size: 2.5rem; }  
2 h2 { font-size: 2rem; }  
3 p { font-size: 1rem; }
```

반응형 대응 (Clamp)

```
1 h1 {  
2   font-size: clamp(1.5rem, 2vw + 1rem, 3rem);  
3 }
```

→ 뷰포트 크기에 따라 동적으로 변화하되, 최소/최대값을 제한함

7. 접근성과 반응형 고려

- `rem`, `em`은 사용자의 브라우저 글꼴 설정을 **자연스럽게 반영함** →  접근성 우수
- `px`은 고정이라 사용자 확대 시 비례 확대되지 **않음** →  접근성 약함

결론

`font-size`는 텍스트의 계층 구조와 가독성, 반응성을 결정짓는 핵심 속성이다.

실무에서는 `rem` + `vw` 또는 `clamp()`와 같은 **반응형 전략**을 조합하여 사용하는 것이 표준이다.

굵기: `font-weight`

`font-weight`는 CSS에서 글꼴의 **굵기(두께)**를 지정하는 속성이다.

텍스트의 **강조 정도**, **시각적 계층 구조**, **가독성 향상** 등을 표현할 때 사용된다.

1. 기본 문법

```
1 selector {  
2   font-weight: 값;  
3 }
```

```
1 strong {  
2   font-weight: bold;  
3 }
```

2. 지정 가능한 값

A. 키워드 값

값	의미
<code>normal</code>	기본 굵기 (보통 400)
<code>bold</code>	굵게 (보통 700)
<code>lighter</code>	부모보다 얇게
<code>bolder</code>	부모보다 굵게

```
1 p {  
2   font-weight: normal;  
3 }  
4 strong {  
5   font-weight: bold;  
6 }
```

`lighter/bolder`는 부모 요소의 `font-weight`를 기준으로 상대적인 값을 적용

B. 숫자 값 (100 ~ 900)

- 100 단위로 9단계 제공
- **100은 가장 얇고, 900은 가장 두꺼움**
- 많이 사용되는 값: 400 (보통), 700 (굵게)

```
1 h1 {  
2   font-weight: 900;  
3 }  
4 p {  
5   font-weight: 300;  
6 }
```

주의: 폰트 파일이 해당 굵기를 지원해야만 정확하게 반영됨
일부 브라우저는 중간값을 보간하거나 가장 가까운 굵기를 사용

3. 폰트 종류에 따른 차이점

- 고정 굵기(Font with fixed weights): 일부 폰트는 400, 700 만 지원 (예: 기본 시스템 폰트)
- 가변 굵기(Variable font): 최신 폰트는 100~900 사이 모든 수치를 지원 (예: Noto Sans, Pretendard)

4. 실무 예제

텍스트 강조

```
1 .highlight {  
2   font-weight: bold;  
3 }
```

다양한 계층 적용

```
1 h1 { font-weight: 900; }  
2 h2 { font-weight: 700; }  
3 h3 { font-weight: 500; }  
4 p { font-weight: 400; }
```

5. 글꼴과의 호환성 주의

- 모든 숫자 값이 모든 글꼴에서 지원되는 것은 아님
- 일부 글꼴은 `font-weight: 500`을 무시하고 400으로 렌더링할 수도 있음
- Google Fonts나 `@font-face`로 가져오는 웹 폰트는 반드시 `font-weight` 범위를 명시해야 함

```
1 <link href="https://fonts.googleapis.com/css2?
  family=Roboto:wght@300;400;700&display=swap" rel="stylesheet">
```

6. 브라우저 기본 스타일 예시

- ``: 기본적으로 `font-weight: bold;`
- ``: 시각적 효과는 동일하지만 의미는 없음 (접근성 측면에서 권장되지 않음)

결론

`font-weight`는 텍스트의 시각적 무게를 조절하여 **정보의 계층 구조와 강조 효과를 부여하는 핵심 도구**이다.

숫자 값(100~900)을 적극 활용하면 **더 정밀하고 유연한 스타일링**이 가능하며, 가변 폰트와의 조합으로 더욱 세밀한 제어가 가능하다.

스타일: font-style

`font-style` 속성은 CSS에서 **글꼴의 스타일(기울임 여부)**을 지정하는 데 사용된다.

주로 **이탤릭체(italic)** 혹은 **기울임체(oblique)** 효과를 줄 때 사용된다.

1. 기본 문법

```
1 selector {
2   font-style: 값;
3 }
```

```
1 em {
2   font-style: italic;
3 }
```

2. 사용 가능한 값

값	설명
<code>normal</code>	일반 스타일 (기본값)
<code>italic</code>	이탤릭체 (폰트가 직접 제공하는 이탤릭 글꼴)
<code>oblique</code>	기울임체 (보통 원래 글꼴을 인공적으로 기울임)
<code>inherit</code>	상위 요소의 값을 상속

A. normal

```
1 p {  
2   font-style: normal;  
3 }
```

- 특별한 기울임 없이 **기본 글꼴 형태로** 표시
-

B. italic

```
1 cite {  
2   font-style: italic;  
3 }
```

- 폰트가 제공하는 정식 이탤릭체(Italic face) 사용
 - 보통 `em`, `cite`, `dfn`, `i` 태그 등에 기본 적용됨
-

C. oblique

```
1 p {  
2   font-style: oblique;  
3 }
```

- 폰트에 이탤릭체가 없을 경우 **일반 글꼴을 기울여서 표현**
 - `italic` 과 비슷하지만, 타이포그래피상 정식 기울임이 아님
 - 일부 폰트에서는 `italic` 과 동일하게 처리되기도 함
-

3. 실무 예시

기울임 강조

```
1 .important-note {  
2   font-style: italic;  
3 }
```

조건부 기울임 처리

```
1 blockquote {  
2   font-style: oblique;  
3 }
```

4. HTML 기본 기울임 요소와 연관

태그	기본 스타일
<code><i></code>	시각적 기울임 (<code>font-style: italic</code>)
<code></code>	의미적 강조 + 기울임
<code><cite></code>	출처 표기 + 기울임

✔ 의미 있는 기울임에는 `em`, `cite` 처럼 **시맨틱한 태그** 사용이 권장됨

5. 주의 사항

- `font-style` 은 글씨체의 굵기와는 관련 없음 → `font-weight` 로 제어해야 함
- 일부 글꼴은 **이탤릭체를 아예 제공하지 않을 수도 있음** → 이 경우 `italic`, `oblique` 모두 시각적으로 동일할 수 있음
- 이탤릭체 사용 시 **가독성 저하**가 발생할 수 있으므로 본문보다는 **강조, 인용** 등에 제한적으로 사용

결론

`font-style` 은 텍스트에 **기울임 효과**를 부여하여 강조나 인용 등의 의미를 시각적으로 표현한다.
의미 있는 강조에는 `italic` 을, 단순한 시각 효과에는 `oblique` 를 적절히 구분해서 사용하는 것이 바람직하다.

줄 높이: `line-height`

CSS의 `line-height` 속성은 **텍스트 줄 간의 세로 간격(줄 간격)**을 지정한다.
가독성 향상, 텍스트 정렬 보정, 디자인 밸런스 조절을 위해 매우 중요하게 사용된다.

1. 기본 문법

```
1 selector {
2   line-height: 값;
3 }
```

```
1 p {
2   line-height: 1.5;
3 }
```

2. 사용 가능한 값의 종류

값 종류	예시	설명
숫자 (unitless)	<code>1.5</code>	현재 글꼴 크기의 배수. 가장 권장됨

값 종류	예시	설명
길이 단위	24px, 1.5em, 2rem	절대 또는 상대 길이로 줄 높이를 지정
백분율	150%	글꼴 크기의 퍼센트
normal	기본값 (보통 1.2~1.4배)	브라우저와 글꼴에 따라 다름

3. 가장 많이 사용하는 방식

✅ 숫자 (단위 없는 배수) - 권장

```

1 | p {
2 |   font-size: 16px;
3 |   line-height: 1.5; /* 24px에 해당 */
4 | }
```

- 단위 없이 숫자만 쓴 경우 → 글꼴 크기의 배수
- 자식 요소에게 상속될 때 더 예측 가능하고 유연함

4. 길이 단위 사용 예

```

1 | p {
2 |   font-size: 16px;
3 |   line-height: 24px;
4 | }
```

- line-height 를 고정 px로 지정
- 상속되면 부모 폰트 크기에만 고정되어, 유연성이 떨어짐

5. 백분율 예

```

1 | p {
2 |   font-size: 16px;
3 |   line-height: 150%;
4 | }
```

- 150% → $16\text{px} \times 1.5 = 24\text{px}$

6. 실무 권장 값

문맥	권장 line-height 값
본문 (가독성 중시)	1.5 ~ 1.8

문맥	권장 line-height 값
제목 (간격 줄임)	1.2 ~ 1.4
UI 요소 (버튼, 네비게이션)	1.0 ~ 1.5

7. 예제

```
1 body {
2   font-size: 16px;
3   line-height: 1.6;
4 }
5
6 h1 {
7   font-size: 32px;
8   line-height: 1.3;
9 }
10
11 button {
12   font-size: 14px;
13   line-height: 1.2;
14 }
```

8. 시각적 차이 예시

font-size	line-height	설명
16px	16px	줄 간격 없음 (읽기 어려움)
16px	24px	적절한 간격 (가독성 좋음)
16px	32px	과도한 간격 (디자인 강조용)

9. 정렬과의 관계

- line-height는 텍스트 수직 정렬 시 정렬 기준 위치에 영향을 줌
- vertical-align, display: inline-block, text-align 등과 함께 조절하면 유용함

10. 상속과 사용 주의점

- line-height는 상속되는 속성
- 부모가 단위 없는 값으로 지정하면, 자식 요소의 font-size에 맞춰 자연스럽게 계산됨
- 반면, 부모가 px 단위로 지정한 경우 자식이 다른 폰트 크기를 가져도 줄 높이는 고정됨

결론

`line-height`는 웹 타이포그래피에서 가독성, 안정적인 레이아웃, 시각적 균형을 위한 핵심 속성이다.

특히 단위 없는 숫자 값 사용은 유연성, 접근성, 유지보수 측면에서 가장 이상적이다.

정렬: `text-align`, `vertical-align`

텍스트나 인라인 요소의 정렬 방식은 웹 디자인에서 매우 중요한 시각적 요소이다.

`text-align`은 수평 정렬, `vertical-align`은 수직 정렬을 담당하며,

둘은 전혀 다른 역할을 가진다.

1. `text-align`: 수평 정렬

1-1. 기본 문법

```
1 selector {  
2   text-align: 값;  
3 }
```

1-2. 주요 값

값	의미
<code>left</code>	왼쪽 정렬 (기본값)
<code>right</code>	오른쪽 정렬
<code>center</code>	가운데 정렬
<code>justify</code>	양쪽 정렬 (단어 간 간격을 늘려서 좌우 맞춤)
<code>start</code> / <code>end</code>	텍스트 방향에 따라 왼쪽/오른쪽 정렬 (LTR/RTL 대응)

1-3. 적용 대상

- `block`, `inline-block` 요소의 내부 인라인 콘텐츠에 적용됨
- 일반적으로 `p`, `h1`, `div` 등의 요소에 사용

1-4. 예시

```
1  p {
2    text-align: justify;
3  }
4
5  h1 {
6    text-align: center;
7  }
```

```
1  <div style="text-align: right;">
2    오른쪽 정렬된 텍스트입니다.
3  </div>
```

1-5. 사용 주의

- `justify`는 한글에는 가독성이 떨어질 수 있음 (띄어쓰기 간격이 과하게 벌어질 수 있음)
- `text-align: center;`는 부모 요소에 적용되어야 **자식 텍스트들이 가운데 정렬됨**

2. vertical-align: 수직 정렬

2-1. 기본 문법

```
1  selector {
2    vertical-align: 값;
3  }
```

2-2. 주요 값

값	설명
<code>baseline</code>	기준선 정렬 (기본값)
<code>top</code>	가장 위에 정렬
<code>middle</code>	가운데 정렬 (부모의 x-height 기준)
<code>bottom</code>	가장 아래 정렬
<code>text-top</code>	부모 텍스트 상단 기준
<code>text-bottom</code>	부모 텍스트 하단 기준
<code>sub</code>	아래 첨자 스타일 (ex: 화학식)
<code>super</code>	위 첨자 스타일 (ex: 제곱)

2-3. 적용 대상

- 인라인 요소 또는 inline-block 요소에만 적용됨
- 주로 이미지, 아이콘, 버튼 등 텍스트 옆에 놓이는 작은 요소들을 정렬할 때 사용

2-4. 예시

```
1 <p>
2   텍스트  가운데 정렬된 아이콘
3 </p>
```

```
1 span {
2   vertical-align: top;
3 }
```

2-5. display: table-cell 일 때 사용

```
1 .container {
2   display: table-cell;
3   vertical-align: middle;
4 }
```

- table, table-cell 레이아웃을 사용하면 vertical-align: middle 이 블록 수준에서도 작동함

3. 두 속성의 차이 요약

속성	방향	적용 대상	주요 사용 사례
text-align	수평(Horizontal)	블록 요소	제목, 본문 가운데 정렬
vertical-align	수직(Vertical)	인라인 또는 table-cell 요소	아이콘, 이미지, 수식 등 세로 정렬

결론

text-align은 텍스트 블록의 수평 정렬,
vertical-align은 인라인 요소의 수직 정렬을 위한 속성이다.
두 속성은 혼동되기 쉽지만 용도와 적용 대상이 명확히 다르므로 구분하여 사용해야 한다.

장식: text-decoration

text-decoration은 텍스트에 선(line)을 그어 시각적으로 강조하거나, 링크처럼 보이게 만드는 데 사용되는 CSS 속성이다.
밑줄, 윗줄, 가운데줄, 깜빡임 등 다양한 텍스트 장식 효과를 제어할 수 있다.

1. 기본 문법

```
1 selector {  
2   text-decoration: 값;  
3 }
```

```
1 a {  
2   text-decoration: underline;  
3 }
```

2. 주요 값

값	의미
<code>none</code>	장식 없음 (선 제거)
<code>underline</code>	밑줄
<code>overline</code>	윗줄
<code>line-through</code>	가운데 줄 (취소선)
<code>blink</code>	깜빡이는 텍스트 (거의 모든 브라우저에서 지원되지 않음)

3. 예시

```
1 h1 {  
2   text-decoration: overline;  
3 }  
4  
5 p {  
6   text-decoration: line-through;  
7 }  
8  
9 a {  
10  text-decoration: none;  
11 }
```

4. 링크 밑줄 제거 예시

HTML 기본 링크는 밑줄이 기본 적용되어 있음.
이를 제거하고 싶을 때는 다음처럼 지정한다:

```
1 a {  
2   text-decoration: none;  
3 }
```

5. 여러 값의 조합

CSS3부터 `text-decoration` 속성은 단일 속성처럼 보이지만, **복합 속성**이 되었다.

아래 속성들을 개별적으로도 지정할 수 있다:

- `text-decoration-line`
- `text-decoration-style`
- `text-decoration-color`
- `text-decoration-thickness`

예:

```
1 a {  
2   text-decoration-line: underline;  
3   text-decoration-style: wavy;  
4   text-decoration-color: red;  
5   text-decoration-thickness: 2px;  
6 }
```

결과 → 붉은색 물결 모양의 두꺼운 밑줄이 생김.

6. 자주 쓰이는 실무 패턴

링크에서 밑줄 제거 + 호버 시 다시 표시

```
1 a {  
2   text-decoration: none;  
3   color: inherit;  
4 }  
5  
6 a:hover {  
7   text-decoration: underline;  
8 }
```

취소선 스타일

```
1 .del-text {  
2   text-decoration: line-through;  
3 }
```

7. 상속 여부

- `text-decoration` 은 기본적으로 **상속된다**.
- 자식 요소가 밑줄을 원하지 않으면 반드시 `text-decoration: none;` 을 설정해야 한다.

```
1 | <a href="#"><span style="text-decoration: none;">밑줄 없음</span></a>
```

결론

`text-decoration`은 텍스트를 강조하거나 링크로 식별시키기 위해 사용하는 시각적 장식 속성이다. CSS3부터는 세부 속성(`line`, `style`, `color`, `thickness`)으로 세밀한 제어가 가능하므로 사용자 정의 디자인에 적극 활용할 수 있다.

변환: `text-transform`

`text-transform` 속성은 텍스트의 대소문자 형식을 변경하는 데 사용된다. HTML 콘텐츠 자체를 수정하지 않고도 시각적으로 대문자/소문자/첫 글자 대문자 등을 표현할 수 있다.

1. 기본 문법

```
1 | selector {
2 |   text-transform: 값;
3 | }
```

```
1 | h1 {
2 |   text-transform: uppercase;
3 | }
```

2. 주요 값

값	설명	예시 (hello world)
<code>none</code>	변환 없음 (기본값)	<code>hello world</code>
<code>capitalize</code>	각 단어의 첫 글자만 대문자	<code>Hello world</code>
<code>uppercase</code>	모든 문자를 대문자로	<code>HELLO WORLD</code>
<code>lowercase</code>	모든 문자를 소문자로	<code>hello world</code>
<code>full-width</code>	문자들을 전각 문자로 변환 (일본어 등에서 사용)	<code>h e l l o w o r l d</code>
<code>inherit</code>	부모 요소로부터 상속	(부모 속성 따름)

3. 예시

```
1 .title {
2   text-transform: uppercase;
3 }
4
5 .subtitle {
6   text-transform: capitalize;
7 }
8
9 .note {
10  text-transform: lowercase;
11 }
```

```
1 <h1 class="title">spring boot</h1>      <!-- 출력: SPRING BOOT -->
2 <h2 class="subtitle">spring boot</h2>    <!-- 출력: Spring Boot -->
3 <p class="note">Spring Boot</p>          <!-- 출력: spring boot -->
```

4. 실무 팁

입력값을 변경하지 않고 시각적으로만 표현하고 싶을 때

- 데이터는 `hello world`로 저장해두고,
- 화면에서는 `HELLO WORLD`처럼 출력할 수 있음

접근성 측면

- `text-transform`으로 변경된 대소문자는 **스크린리더가 원본 그대로 읽음**
- 즉, `Spring Boot` → `uppercase` → `SPRING BOOT`로 보여도, 음성 출력은 원래 발음 그대로

`capitalize` 주의점

- 모든 단어의 **첫 글자만 대문자**이므로, 고유명사 스타일을 쓸 때 적합
- 단, 언어에 따라 정확하지 않을 수 있음 (예: 독일어의 명사 대문자 규칙 등)

5. 브라우저 호환성

속성 값	주요 브라우저 지원
<code>none</code> , <code>capitalize</code> , <code>uppercase</code> , <code>lowercase</code>	✅ 전부 지원
<code>full-width</code>	❖ 일부 브라우저에서만 지원 (특수한 상황에만 사용)

결론

`text-transform`은 HTML의 콘텐츠를 수정하지 않고도 대소문자 표현을 제어하는 데 효과적인 속성이다.

특히 디자인 일관성 유지, 시각적 강조, 다국어 지원 등의 상황에서 유용하게 쓰인다.

간격: `letter-spacing`, `word-spacing`

`letter-spacing`과 `word-spacing`은 텍스트의 가독성, 디자인 스타일, 레이아웃 미세 조정을 위해 자주 사용되는 CSS 속성이다.

각각 글자 간 간격, 단어 간 간격을 제어한다.

1. `letter-spacing` — 글자 간 간격

1.1 문법

```
1 selector {  
2   letter-spacing: 값;  
3 }
```

```
1 p {  
2   letter-spacing: 0.05em;  
3 }
```

1.2 값 종류

- 길이 단위 사용 가능: `px`, `em`, `rem` 등
- 음수 값도 가능 → 글자를 더 가깝게 붙임
- 기본값: `normal` (브라우저 기본 간격 사용)

1.3 예시

```
1 h1 {  
2   letter-spacing: 0.1em; /* 넓게 띄움 */  
3 }  
4  
5 .logo {  
6   letter-spacing: -1px; /* 좁게 붙임 */  
7 }
```

1.4 실무 활용

- 로고 텍스트나 헤더 등에서 시각적 강조를 위해 조절
- 자간이 너무 좁은 글꼴에선 `letter-spacing`을 조절해 가독성을 높임
- 한글의 경우 너무 좁히면 글자가 겹칠 수 있어 주의

2. word-spacing — 단어 간 간격

2.1 문법

```
1 selector {
2   word-spacing: 값;
3 }
```

```
1 p {
2   word-spacing: 8px;
3 }
```

2.2 값 종류

- 길이 단위만 허용: px, em, rem, % 등
- 기본값: normal (브라우저 기본 간격)

2.3 예시

```
1 p {
2   word-spacing: 0.5em;
3 }
4
5 blockquote {
6   word-spacing: 10px;
7 }
```

3. 비교 요약

속성	적용 대상	기본값	효과
letter-spacing	문자(글자) 사이	normal	글자 간격 조절
word-spacing	단어(띄어쓰기 기준) 사이	normal	단어 간격 조절

4. 시각적 예시 (개념적으로 설명)

설정	결과
letter-spacing: 2px	S P A C E
word-spacing: 1em	this is spaced

5. 상속 여부

- 둘 다 상속된다.
- 자식 요소가 다른 폰트 크기를 가지면 **상대 단위(em)**를 사용한 간격은 재계산됨

6. 권장 단위

목적	추천 단위
웹 타이포그래피 기본 설정	em 또는 rem
픽셀 정밀 조정	px
반응형 디자인	em 선호

결론

letter-spacing과 word-spacing은 텍스트의 가독성, 타이포그래피 스타일링, 시각적 정렬을 조정하는 데 핵심적인 역할을 한다.
단어 또는 글자 사이의 간격을 조절함으로써 사용자의 **시각적 경험을 세밀하게 제어**할 수 있다.

그림자: text-shadow

접근성(Accessibility)은 장애를 가진 사용자도 웹 콘텐츠에 **동등하게 접근하고 이해할 수 있도록** 만드는 것을 말한다.
특히 시각 장애인을 위한 **스크린 리더(Screen Reader)**는 테이블의 구조를 이해하기 위해 **시맨틱한 HTML 마크업**과 **명확한 관계 지정**을 요구한다.
복잡한 표일수록 접근성을 고려한 마크업이 필수이다.

✔ 접근성 향상을 위한 핵심 요소

요소	설명
<caption>	표의 제목을 지정해줌 (스크린 리더에서 유용)
scope	헤더 셀(th)이 어떤 셀을 설명하는지 지정
headers	각 데이터 셀(td)이 어떤 헤더와 연관되는지 ID로 지정
id	헤더 셀의 식별자 정의 (headers 속성과 연결)
summary	예전 HTML4의 설명 속성 (HTML5에서는 폐지됨)
<thead>, <tbody>, <tfoot>	구조를 명확히 나눠줌

✓ 예제 1: 기본적인 접근성 마크업

```
1 <table>
2   <caption>학생 성적표</caption>
3   <thead>
4     <tr>
5       <th scope="col">이름</th>
6       <th scope="col">수학</th>
7       <th scope="col">영어</th>
8     </tr>
9   </thead>
10  <tbody>
11    <tr>
12      <td>홍길동</td>
13      <td>90</td>
14      <td>85</td>
15    </tr>
16    <tr>
17      <td>김영희</td>
18      <td>95</td>
19      <td>92</td>
20    </tr>
21  </tbody>
22 </table>
```

✓ `scope="col"` 은 해당 `<th>` 가 열 제목이라는 뜻 → 스크린 리더가 셀을 읽을 때 연관된 헤더를 같이 안내함.

✓ 예제 2: 복잡한 표에서 `headers` 와 `id` 사용

```
1 <table>
2   <caption>병원 진료 기록</caption>
3   <thead>
4     <tr>
5       <th id="name">환자명</th>
6       <th id="date">진료일</th>
7       <th id="dept">진료과</th>
8       <th id="diag">진단명</th>
9     </tr>
10  </thead>
11  <tbody>
12    <tr>
13      <td headers="name">이철수</td>
14      <td headers="date">2025-05-01</td>
15      <td headers="dept">내과</td>
16      <td headers="diag">감기</td>
17    </tr>
18    <tr>
19      <td headers="name">박민지</td>
20      <td headers="date">2025-05-02</td>
21      <td headers="dept">정형외과</td>
```

```

22     <td headers="diag">골절</td>
23   </tr>
24 </tbody>
25 </table>

```

🔴 이 구조는 특히 **행/열 병합**이 많은 복잡한 표에서 사용됨.
스크린 리더가 `headers` 속성을 따라가 해당 `td`가 어떤 `th`에 대응하는지 정확히 안내함.

✅ `caption` 태그의 활용

- 표의 제목/설명을 의미하며, 스크린 리더는 가장 먼저 이 요소를 읽음
- 디자인 상 보이지 않게 할 수도 있음 (`.sr-only` 클래스 등)

```

1 <caption class="sr-only">2025년 1학기 중간고사 성적표</caption>

```

✅ `scope` 속성 값 종류

값	의미
<code>col</code>	열 방향으로 적용되는 헤더
<code>row</code>	행 방향으로 적용되는 헤더
<code>colgroup</code>	열 그룹에 대한 헤더 (복잡한 구조용)
<code>rowgroup</code>	행 그룹에 대한 헤더

✅ ARIA 접근성 보조 속성 (선택적)

속성	설명
<code>role="table"</code>	사용자 정의 구조에서 명시적 역할 부여 가능
<code>aria-label</code>	<code>caption</code> 대체 텍스트
<code>aria-describedby</code>	표 전체에 설명 연결 가능 (ID 기반)

✅ 스크린 리더 동작 예시

화면에 표시된 내용	스크린 리더가 읽는 방식
85	영어, 홍길동, 85 점
내과	진료과, 이철수, 내과

→ 이것이 가능하려면 정확한 `scope`, `headers`, `id` 구조가 필요

✔ 한 줄 요약

테이블에서의 접근성은 `caption`, `scope`, `headers`, `id`를 통해 셀과 헤더의 논리적 관계를 명확히 지정함으로써 스크린 리더 사용자에게 올바른 정보를 제공하는 데 핵심이다.