3. 선택자(Selector)

기본 선택자

선택자(selector)는 CSS에서 **어떤 HTML 요소에 스타일을 적용할지 지정하는 도구**다.
기본 선택자는 가장 단순하고 자주 사용하는 선택자 유형으로, **태그 이름, 클래스, ID**를 기반으로 스타일을 지정한다.

---

1. 전체 선택자 (`*`)

모든 HTML 요소에 스타일을 일괄 적용한다.
단, 성능 저하와 의도치 않은 전역 스타일 오염에 주의해야 한다.

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

> 모든 요소의 기본 마진과 패딩을 제거하고, `box-sizing`을 초기화할 때 자주 사용됨 (reset CSS의 핵심).

---

2. 태그(요소) 선택자

특정 **HTML 태그**에 대해 스타일을 적용한다.

```
body {
  background-color: #f4f4f4;
}

h1 {
  font-size: 32px;
}
```

- 간단하지만 범용적이며, 요소 전체에 영향을 준다.
- 구체성이 낮기 때문에 클래스나 ID보다 우선순위는 낮다.

---

3. 클래스 선택자 (`.클래스명`)

HTML 요소의 `class` 속성과 연결되어 특정 그룹의 요소에 스타일을 지정할 수 있다.

```
.title {
  color: navy;
  text-align: center;
}
```

```
<h2 class="title">클래스 선택자 예제</h2>
```

- 같은 클래스를 여러 요소에 반복적으로 적용 가능
- **재사용성**이 매우 높아 가장 많이 사용됨

---

4. ID 선택자 (`#아이디명`)

HTML 요소의 `id` 속성과 연결되어 **고유한 요소 하나**에 스타일을 적용한다.

```
#main {
  width: 1000px;
  margin: 0 auto;
}
```

```
<div id="main">메인 컨테이너</div>
```

- 한 문서 내에서 **ID는 중복 없이 하나만 존재**해야 함
- 구체성(specificity) 점수에서 가장 강함 (클래스보다 우선)

---

5. 그룹 선택자 (`,`)

여러 선택자에 **동일한 스타일을 한 번에 적용**할 수 있다.

```
h1, h2, h3 {
  font-weight: bold;
  color: darkslategray;
}
```

- 중복 코드 방지
- 가독성과 유지보수에 유리

요약 표

선택자 유형	문법	설명	우선순위 점수
전체 선택자	`*`	모든 요소	0
태그 선택자	`p`, `div`, `h1` 등	해당 태그에 적용	1
클래스 선택자	`.box`, `.title`	클래스 속성 요소에 적용	10
ID 선택자	`#header`, `#main`	특정 ID 요소 1개에 적용	100
그룹 선택자	`h1, p, div`	여러 요소에 동일 스타일 적용	개별 계산

결론

> 기본 선택자는 **CSS 스타일링의 출발점**이며,
> 특히 **클래스 선택자**는 실무에서 가장 자주 사용된다.
> 적절한 선택자 조합과 우선순위 조절을 통해 코드 효율성과 명확성을 높일 수 있다.

결합자

결합자(Combinators)는 CSS에서 **두 개 이상의 선택자를 조합하여**
특정 관계를 가지는 요소에 스타일을 적용할 수 있게 해준다.
즉, 단순히 어떤 태그인지뿐만 아니라 **요소 간의 구조적 관계(계층, 형제 관계 등)**를 기준으로 선택할 수 있게 한다.

CSS에는 총 **4가지 결합자**가 있다:

---

1. 자손 결합자 (`선택자A 선택자B`)

의미

- A의 **모든 하위 요소 B**를 선택한다 (직계 자식 + 손자 + 그 아래까지 전부 포함).
- **공백**으로 결합한다.

예시

```
div p {
  color: red;
}
```

```
<div>
  <p>적용됨</p>
  <section>
    <p>이것도 적용됨</p>
  </section>
</div>
```

- `div` 안에 있는 **모든** `<p>` **요소**가 적용 대상

---

2. 자식 결합자 (`선택자A > 선택자B`)

의미

- A의 **직계 자식 B**만 선택한다.
- 한 단계 아래에 위치한 경우에만 적용됨.

예시

```
ul > li {
  list-style-type: square;
}
```

```
<ul>
  <li>적용됨</li>          <!-- 직계 자식 -->
  <div>
    <li>적용 안됨</li>     <!-- 손자: 무시됨 -->
  </div>
</ul>
```

---

3. 인접 형제 결합자 (`선택자A + 선택자B`)

의미

- A **바로 다음에 오는 형제 요소 B**를 선택한다.
- 둘은 같은 부모를 가져야 하며, **B는 A 다음에 1개만 존재 가능**

예시

```
h1 + p {
  font-style: italic;
}
```

```
<h1>제목</h1>
<p>이 단락만 적용됨</p>  <!-- h1 다음에 바로 오는 p -->
<p>이 단락은 적용 안됨</p>
```

---

4. 일반 형제 결합자 (`선택자A ~ 선택자B`)

의미

- A 이후에 나오는 **모든 형제 B 요소들**을 선택한다.
- A와 B는 같은 부모를 공유하고 있어야 함

예시

```
h1 ~ p {
  color: gray;
}
```

```
<h1>제목</h1>
<p>적용됨</p>
<p>이것도 적용됨</p>
```

- `h1` 이후의 **모든 형제** `<p>` **요소**가 적용 대상

결합자 비교 요약

결합자	의미	설명	예시
`A B`	자손	A 안에 포함된 모든 B	`div p`
`A > B`	자식	A의 바로 아래 자식 B	`ul > li`
`A + B`	인접 형제	A 다음의 형제 B 1개	`h1 + p`
`A ~ B`	일반 형제	A 이후의 형제 B 전부	`h1 ~ p`

실전 팁

- 레이아웃 구조가 복잡한 사이트에서 **선택 범위를 명확히 지정**하는 데 유용
- **불필요한 중첩 스타일 적용을 방지**할 수 있다
- 자식 결합자(`>`)와 자손 결합자(` `)는 혼동하기 쉬우므로 용도에 따라 구분해서 사용할 것

결론

> 결합자는 HTML 요소 간의 **계층 구조와 형제 관계**를 이용해 정밀한 선택을 가능하게 한다.
> 특히 컴포넌트 기반 UI에서 요소 범위를 제한하거나, 특정 구조 하의 요소만 스타일링할 때 매우 강력하다.

속성 선택자

속성 선택자는 HTML 요소의 특정 속성(attribute)을 기준으로 선택하는 CSS 선택자다.
단순히 태그나 클래스, ID가 아닌, 요소에 있는 **속성과 그 값**에 따라 스타일을 적용할 수 있다.

---

1. 기본 문법

```
[속성]              → 해당 속성을 가진 모든 요소
[속성="값"]          → 속성값이 정확히 일치하는 요소
[속성~="값"]         → 공백으로 구분된 여러 값 중 하나가 일치
[속성|="값"]         → 하이픈(-)으로 시작하거나 정확히 일치
[속성^="값"]         → 값이 특정 문자열로 **시작하는** 경우
[속성$="값"]         → 값이 특정 문자열로 **끝나는** 경우
[속성*="값"]         → 값이 특정 문자열을 **포함하는** 경우
```

---

2. 주요 예시

`[속성]` : 해당 속성을 가진 요소 모두 선택

```
input[required] {
  border: 2px solid red;
}
```

```
<input type="text" required>
<input type="text">
```

→ `required` 속성이 있는 첫 번째 `<input>`에만 스타일이 적용됨.

`[속성="값"]` : 정확히 일치하는 경우

```
input[type="checkbox"] {
  width: 20px;
  height: 20px;
}
```

→ `type` 속성 값이 정확히 `"checkbox"`인 요소만 선택

`[속성~="값"]` : 여러 값 중에 하나가 포함되어 있는 경우 (공백 기준)

```
[class~="featured"] {
  background-color: yellow;
}
```

```
<div class="card featured">적용됨</div>
<div class="card special">적용 안 됨</div>
```

→ `"card featured"` 중 `"featured"`가 **개별 단어로 존재**하므로 적용됨.

`[속성|="값"]` : 하이픈으로 시작하거나 동일한 값

```
[lang|="en"] {
  font-style: italic;
}
```

```
<p lang="en">영어</p>          <!-- 적용됨 -->
<p lang="en-US">미국 영어</p> <!-- 적용됨 -->
<p lang="ko">한국어</p>        <!-- 적용 안 됨 -->
```

→ `lang="en"` 또는 `lang="en-..."`인 경우만 적용

`[속성^="값"]` : 값이 특정 문자열로 시작할 때

```
a[href^="https://"] {
  color: green;
}
```

→ `href`가 https로 시작하는 모든 링크에 적용

`[속성$="값"]` : 값이 특정 문자열로 끝날 때

```
img[src$=".jpg"] {
  border: 1px solid black;
}
```

→ `.jpg` 확장자로 끝나는 이미지에만 적용

`[속성*="값"]` : 값이 특정 문자열을 포함할 때

```
a[href*="naver"] {
  color: blue;
}
```

→ `href="https://www.naver.com"` 처럼 중간에 `"naver"`가 들어 있는 경우

---

3. 실무 활용 예시

- `input[type="password"]` : 비밀번호 필드에 특수 스타일 적용
- `[disabled]` : 비활성화된 버튼 스타일링
- `[data-role="modal"]` : 커스텀 속성을 기준으로 컴포넌트 선택
- `[aria-hidden="true"]` : 접근성 관련 스타일 처리
- 이미지 형식에 따라 처리: `img[src$=".png"]`, `img[src$=".svg"]`

---

4. 속성 선택자 우선순위

- 선택자 구체성 점수: **속성 선택자 = 10점** (클래스와 동일)
- 결합해서 사용할 수 있다:

```
div[class^="box-"] {
  padding: 20px;
}
```

---

5. 요약 정리

선택자	설명
`[attr]`	해당 속성이 존재하는 요소
`[attr="val"]`	정확히 일치
`[attr~="val"]`	공백으로 나뉜 값 중 일치
`[attr | ="val"]`	
`[attr^="val"]`	val로 시작
`[attr$="val"]`	val로 끝남
`[attr*="val"]`	val을 포함

결론

> 속성 선택자는 **정적 클래스 대신 동적 데이터 속성이나 구조 기반 요소 선택**이 필요할 때 유용하며,
> 특히 **컴포넌트 기반 UI**, **접근성 처리**, **사용자 입력에 따라 스타일링할 때 매우 강력한 도구**이다.

가상 클래스 선택자

가상 클래스 선택자는 HTML 문서에 명시적으로 존재하지 않는 **특정 상태나 조건**에 따라 요소를 선택할 수 있도록 해주는 CSS 선택자이다.
대표적으로 `:hover`, `:first-child`, `:nth-child()` 등이 있으며,
사용자 상호작용, 구조적 위치, 상태 등에 따라 **동적 스타일링**을 가능하게 한다.

---

1. 기본 문법

```
선택자:가상클래스 {
  속성: 값;
}
```

---

2. 사용자 상호작용 관련 가상 클래스

`:hover`

마우스를 올렸을 때 적용

```
a:hover {
  color: red;
  text-decoration: underline;
}
```

`:active`

클릭하는 순간 적용 (마우스 눌렀을 때)

```
button:active {
  transform: scale(0.95);
}
```

`:focus`

키보드로 포커스(예: Tab)되었을 때 적용

```
input:focus {
  outline: 2px solid royalblue;
}
```

`:visited`

방문한 적 있는 링크에 적용

```
a:visited {
  color: purple;
}
```

---

3. 구조적 위치 관련 가상 클래스

`:first-child`

부모 요소의 첫 번째 자식일 경우

```
li:first-child {
  font-weight: bold;
}
```

`:last-child`

부모 요소의 마지막 자식일 경우

```
li:last-child {
  color: gray;
}
```

`:nth-child(n)`

n번째 자식을 선택 (1부터 시작)

```
li:nth-child(2) {
  color: blue;
}

li:nth-child(odd) {
  background-color: #f0f0f0;
}
```

- `odd`, `even`, `2n`, `3n+1` 같은 수학 표현식 사용 가능

`:nth-last-child(n)`

뒤에서부터 n번째 자식을 선택

```
li:nth-last-child(1) {
  color: red; /* 마지막 요소 */
}
```

`:only-child`

부모 안에서 유일한 자식일 때

```
div:only-child {
  border: 1px solid green;
}
```

---

4. 상태 기반 가상 클래스

`:checked`

체크된 `<input type="checkbox">`, `<input type="radio">`에 적용

```
input:checked + label {
  color: green;
}
```

`:disabled`, `:enabled`

비활성화된/활성화된 `<input>`, `<button>` 등

```
input:disabled {
  background-color: #eee;
}
```

`:required`, `:optional`

폼 요소의 필수 입력 여부

```
input:required {
  border-color: red;
}
```

`:valid`, `:invalid`

유효성 검사 기준에 따라 스타일 적용

```
input:valid {
  border: 2px solid green;
}
input:invalid {
  border: 2px solid red;
}
```

---

5. 부정 조건 가상 클래스

`:not(selector)`

특정 조건을 **제외**하고 선택

```
div:not(.active) {
  opacity: 0.5;
}
```

- 다른 가상 클래스와 조합하여 정교한 선택이 가능함

---

6. 요약 정리표

선택자	의미
`:hover`	마우스가 올라갔을 때
`:active`	클릭 순간
`:focus`	포커스가 있을 때
`:visited`	방문한 링크
`:first-child`	첫 번째 자식 요소
`:last-child`	마지막 자식 요소
`:nth-child(n)`	n번째 자식 요소

선택자	의미
`:checked`	체크된 상태
`:disabled`, `:enabled`	비활성/활성 상태
`:not(X)`	X가 아닌 요소 선택

결론

> 가상 클래스 선택자는 **정적 구조를 넘어서 사용자 상태, 인터랙션, 위치 조건**에 따라
> **정교하고 반응적인 CSS 스타일링을 가능하게 해주는 핵심 도구**이다.

가상 요소 선택자

가상 요소 선택자는 HTML 문서 안에 **명시적으로 존재하지 않는 "가상의 요소"를 생성하여 스타일을 적용**할 수 있도록 하는 CSS 기능이다.
가상 클래스(`:hover`, `:nth-child` 등)가 **상태**나 **조건**을 의미하는 것이라면,
가상 요소(`::before`, `::after` 등)는 새로운 **"영역" 또는 "내용"을 생성**하는 데 사용된다.

---

1. 기본 문법

```
선택자::가상요소 {
  속성: 값;
}
```

- 최신 표준에서는 `::` (더블 콜론)을 사용하는 것이 권장됨
- 구버전 브라우저는 `:` 단일 콜론도 지원 (`:before`, `:after`)

---

2. 주요 가상 요소

`::before`

- 요소의 **내용 앞에** 가상 요소를 생성함
- **반드시** `content` **속성**이 있어야 동작함

```
p::before {
  content: "→ ";
  color: gray;
}
```

```
<p>리스트 아이템</p>
<!-- 화면 출력: → 리스트 아이템 -->
```

`::after`

- 요소의 **내용 뒤에** 가상 요소를 생성함
- 역시 `content` 속성 필수

```
a::after {
  content: " 🔗";
}
```

```
<a href="#">공식 사이트</a>
<!-- 화면 출력: 공식 사이트 🔗 -->
```

`::first-line`

- **첫 번째 줄에만** 스타일을 적용
- 줄은 화면 너비에 따라 달라지므로 반응형에 유의

```
p::first-line {
  font-weight: bold;
  color: darkred;
}
```

`::first-letter`

- **첫 글자에만** 스타일 적용

```
p::first-letter {
  font-size: 200%;
  float: left;
  color: navy;
}
```

→ **드롭캡(drop cap)** 효과에 자주 사용됨

---

3. 실무 활용 예시

버튼 뒤에 아이콘 추가

```
button.download::after {
  content: "⬇";
  margin-left: 8px;
}
```

공백 없는 구분선 만들기

```
h2::before {
  content: "";
  display: block;
  height: 1px;
  background: #ccc;
  margin-bottom: 10px;
}
```

강조 마크 붙이기

```
.notice::before {
  content: "[공지]";
  color: red;
  font-weight: bold;
}
```

---

4. content 속성 설명

가상 요소를 사용할 때는 반드시 `content` 속성이 필요하다.
이 속성은 다음을 포함할 수 있다:

형태	예시
텍스트	`"Hello"`, `"*"`, `"★"`
빈 문자열	`""` (주로 배경 요소 생성 시 사용)
속성값 삽입	`attr(title)` (해당 속성 값 출력)

---

5. 정리 표

선택자	설명
`::before`	요소의 **시작 앞부분에 내용 삽입**
`::after`	요소의 **끝부분 뒤에 내용 삽입**
`::first-line`	요소의 **첫 줄에만** 스타일 적용
`::first-letter`	요소의 **첫 글자에만** 스타일 적용

---

6. 결합 예시: 가상 클래스와 가상 요소

```
a:hover::after {
  content: " (링크)";
  font-size: 12px;
  color: gray;
}
```

→ 링크에 마우스를 올리면 뒤에 " (링크)" 텍스트가 추가됨

결론

> 가상 요소는 HTML을 직접 수정하지 않고도 **시각적 콘텐츠를 유연하게 삽입하거나 스타일링할 수 있는 강력한 도구**이며, 특히 **UI 장식, 마크업 최소화, 컴포넌트 디자인** 등에서 널리 활용된다.