

# HTML & CSS

참고:

<https://heropy.blog>



박영웅

# 학습 개요

웹 프론트엔드 개발의 **핵심 줄기**를 학습!

중요도가 높은 내용을 위주로 학습.

난이도가 높고 중요도가 낮은 내용은 가볍게 학습하거나 생략.

# 목차

1. 개요
2. HTML 기본 문법
3. CSS 기본 문법
4. CSS 속성

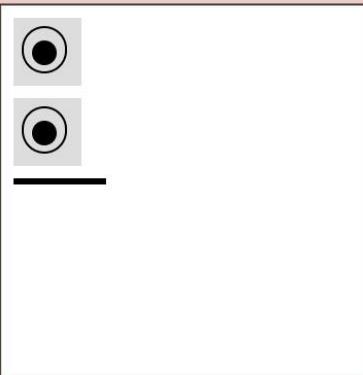
# 프론트엔드 개발

HTML, CSS, JS를 사용해 데이터를 그래픽 사용자 인터페이스(GUI)로  
변환하고, 그것으로 사용자와 상호 작용할 수 있도록 하는 것.

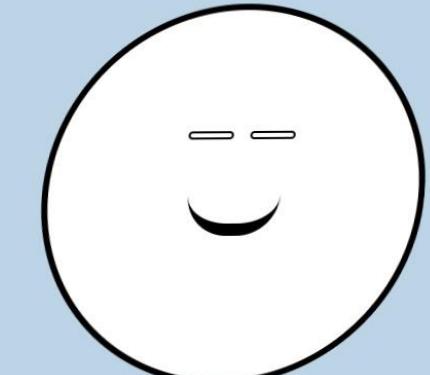




HTML

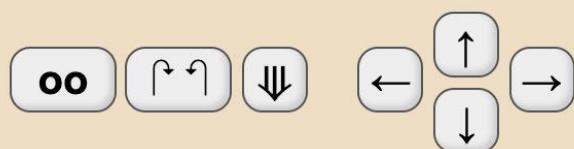


HTML + CSS



HTML + CSS + JS

Click here to PLAY!



# HTML (Hyper Text Markup Language)

페이지의 제목, 문단, 표, 이미지, 동영상 등 웹의 구조를 담당.

**HTML**



기획자

**CSS**

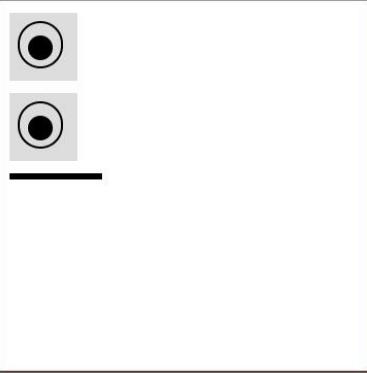


디자이너

**JS**

개발자

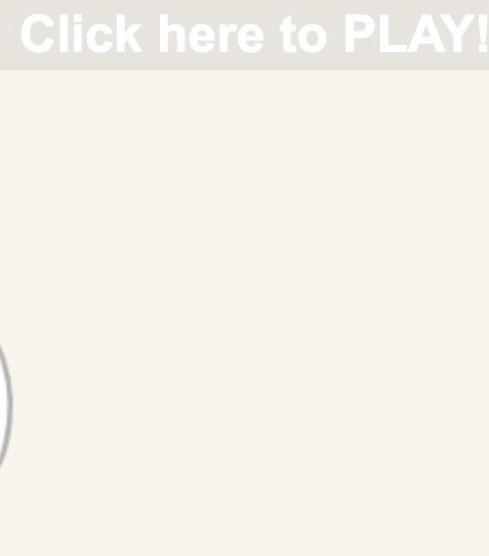
**HTML**



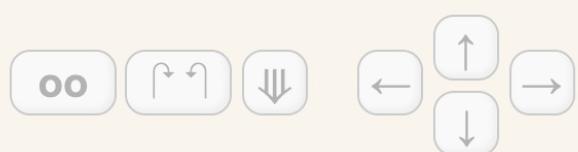
**HTML + CSS**



**HTML + CSS + JS**



Click here to PLAY!



# CSS (Cascading Style Sheets)

실제 화면에 표시되는 방법(색상, 크기, 폰트, 레이아웃 등)을 지정해 콘텐츠를 꾸며주는 시각적인 표현(정적)을 담당.

**HTML**



기획자

**CSS**

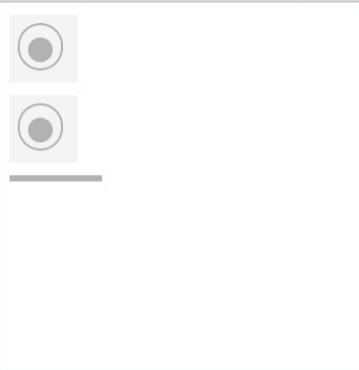


디자이너

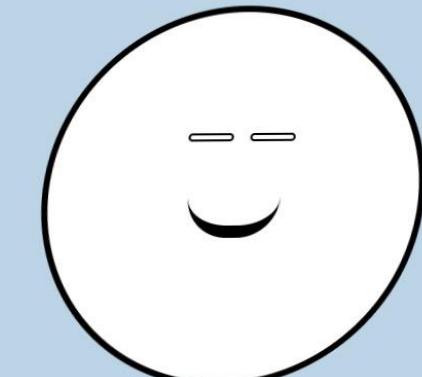


개발자

**HTML**

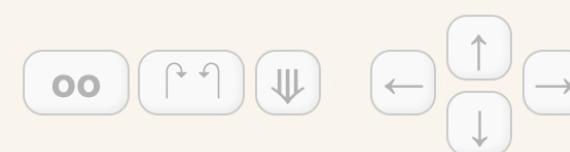


**HTML + CSS**



**HTML + CSS + JS**

Click here to PLAY!



# JS (JavaScript)

콘텐츠를 바꾸고 움직이는 등 페이지를 동작시키는 동적 처리를 담당.

HTML



기획자

CSS

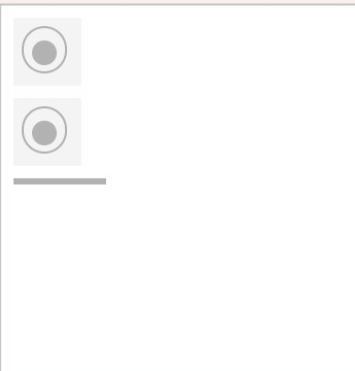


디자이너

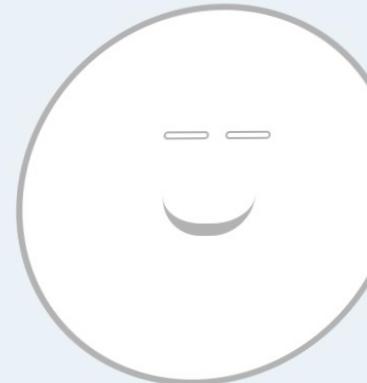
JS

개발자

HTML

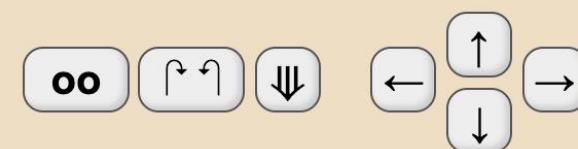


HTML + CSS



HTML + CSS + JS

Click here to PLAY!

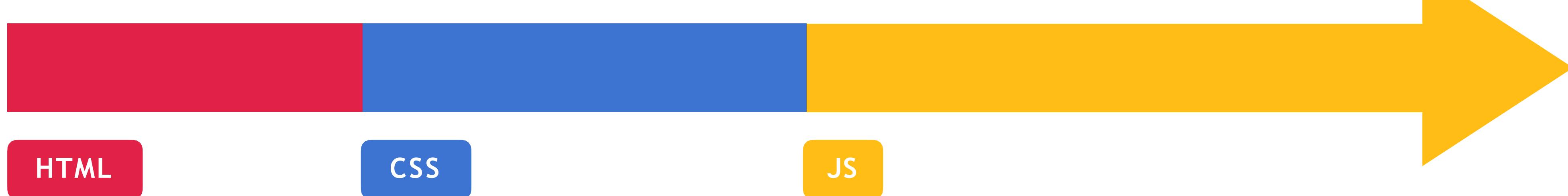
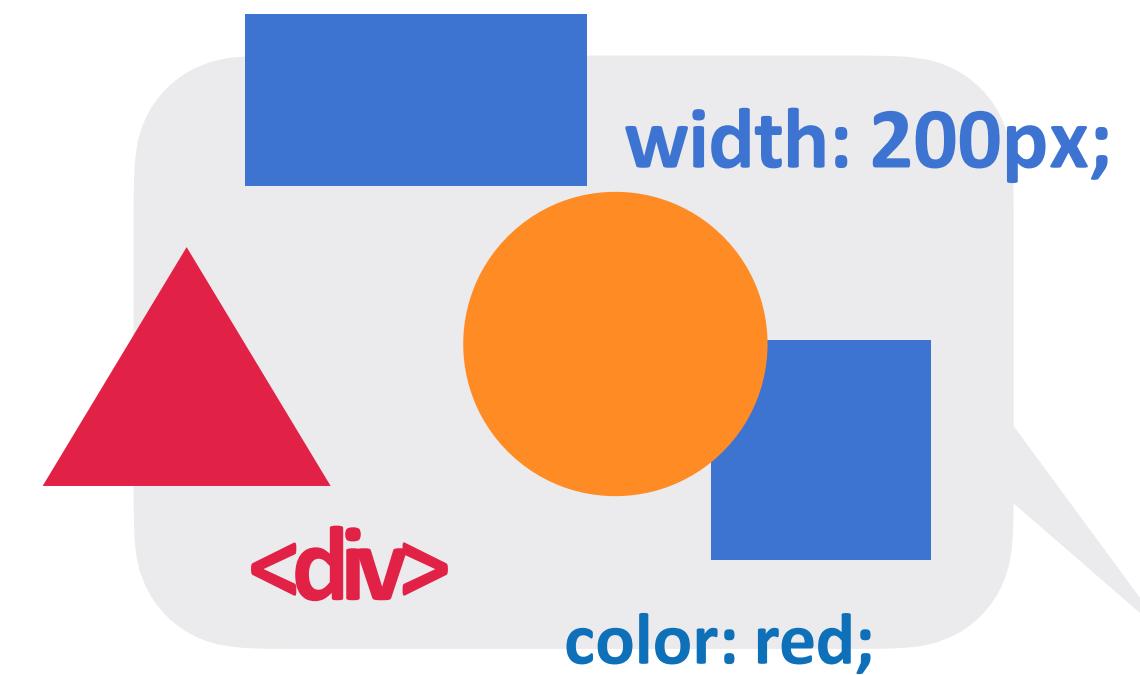




HTML

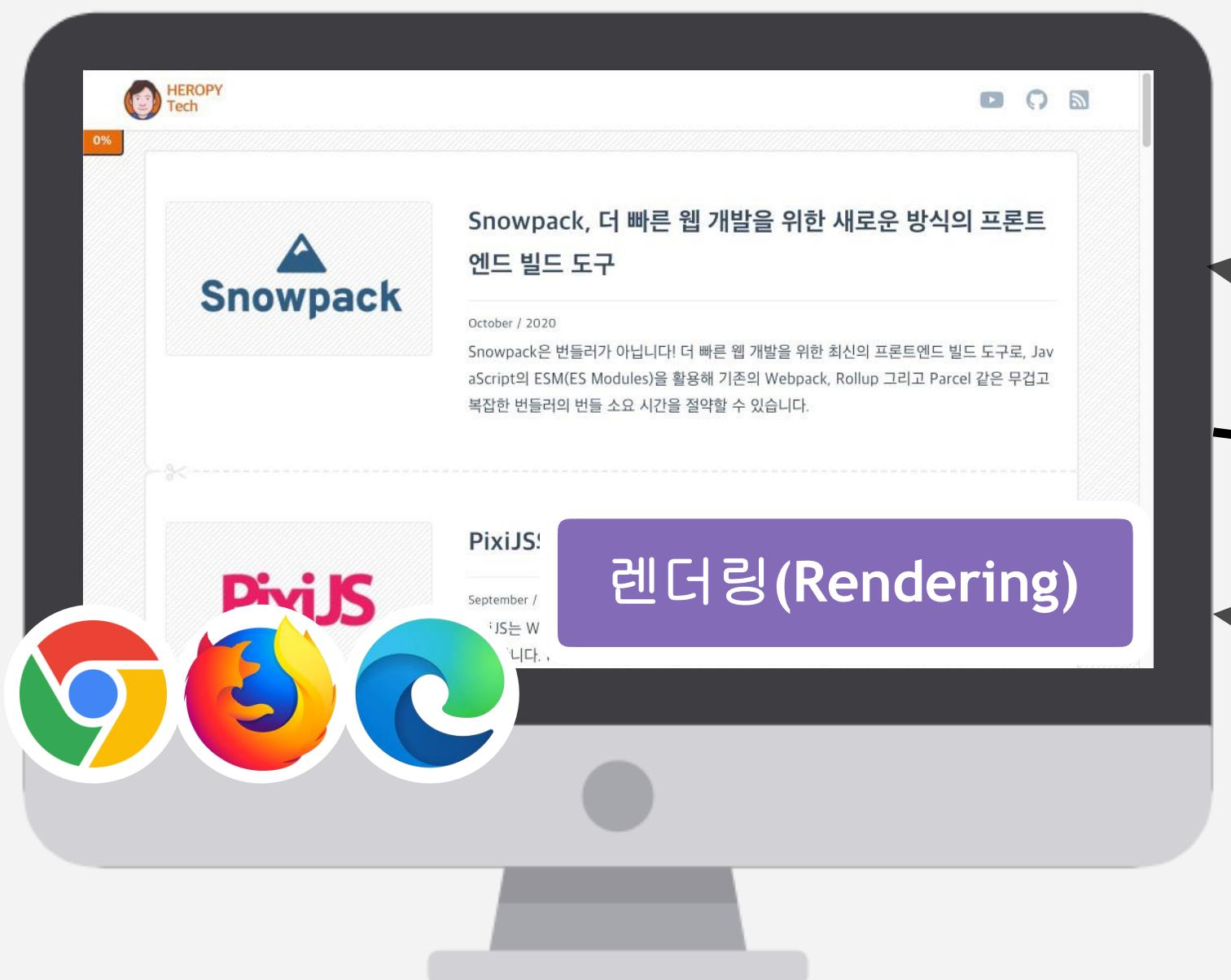
CSS

JS



# <https://heropy.blog>

주소창에 페이지 주소 입력!



사용자 컴퓨터(브라우저)

1

최초 요청  
(Request)



2

최초 응답  
(Response)

3

추가 요청



4

추가 응답



...

서버

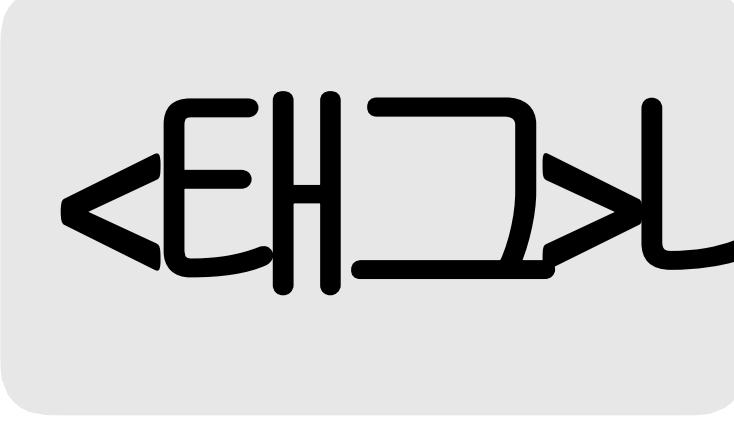


# HTML 기본 문법

**<h1>Hello HTML~</h1>**

<태그>내용</태그>

요소(Element)



<태그> 내용 </태그>

시작(열린) 태그(Tag)

<태그> 내용 </태그>

종료(닫힌) 태그

<태그> 내용 </태그>

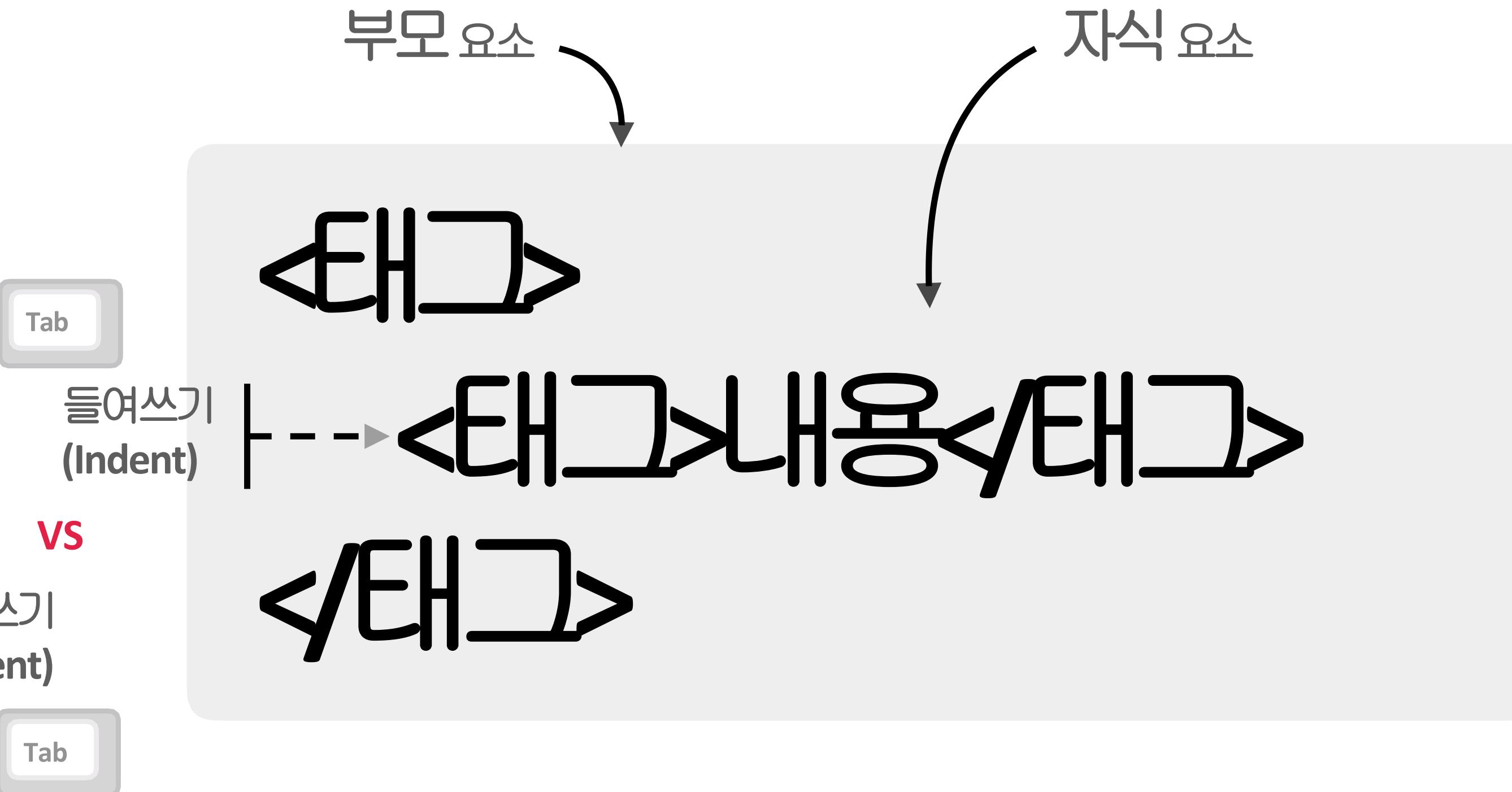
요소의 내용(Contents)

부모 요소

자식 요소

<태그><태그>내용</태그></태그>

The diagram illustrates the structure of nested HTML tags. It features a large, light-gray rounded rectangle containing the text "<태그><태그>내용</태그></태그>". Inside this rectangle, the first two '<태그>' tags are highlighted with a darker gray background. Two arrows point downwards from the text "부모 요소" (Parent Element) and "자식 요소" (Child Element) to the first and second '<태그>' tags respectively. The word "내용" (Content) is positioned between the two highlighted tags.



편리함!

HTML 1/2/3/4/5

<태그>

VS

<태그 />

안전함!

XHTML / HTML5

Attribute

Value

<태그 속성="값">내용</태그>

기능의 확장

<img />

이미지를 삽입하는 요소(태그)!  
어떤 이미지를 삽입해야 하지???  
이미지 이름은 뭐지??

이미지의 경로

```

```

이미지의 이름  
(대체 텍스트 / Alternate Text)



화면에 출력!

# **<input />**

사용자가 데이터를 입력하는 요소(태그)!  
어떤 데이터 타입을 입력 받을 거지?

hello world

화면에 출력!

데이터의 탑

text

사용자에게 일반 텍스트를 입력 받음!

`<input type="text" />`



화면에 출력!

데이터의 탑

checkbox

사용자에게 체크 여부를 입력 받음!

```
<input type="checkbox" />
```

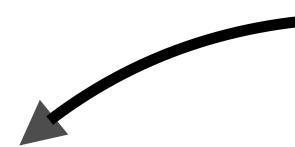
# 글자와 상자

요소가 화면에 출력되는 특성, 크게 2가지로 구분.

인라인(**Inline**) 요소 : 글자를 만들기 위한 요소들.

블록(**Block**) 요소 : 상자(레이아웃)를 만들기 위한 요소들.

```
<span>Hello</span>  
<span>World</span>
```



<span>/span>

대표적인 인라인 요소!  
본질적으로 아무것도 나타내지 않는,  
콘텐츠 영역을 설정하는 용도.

Hello World

요소가 수평으로 쌓임

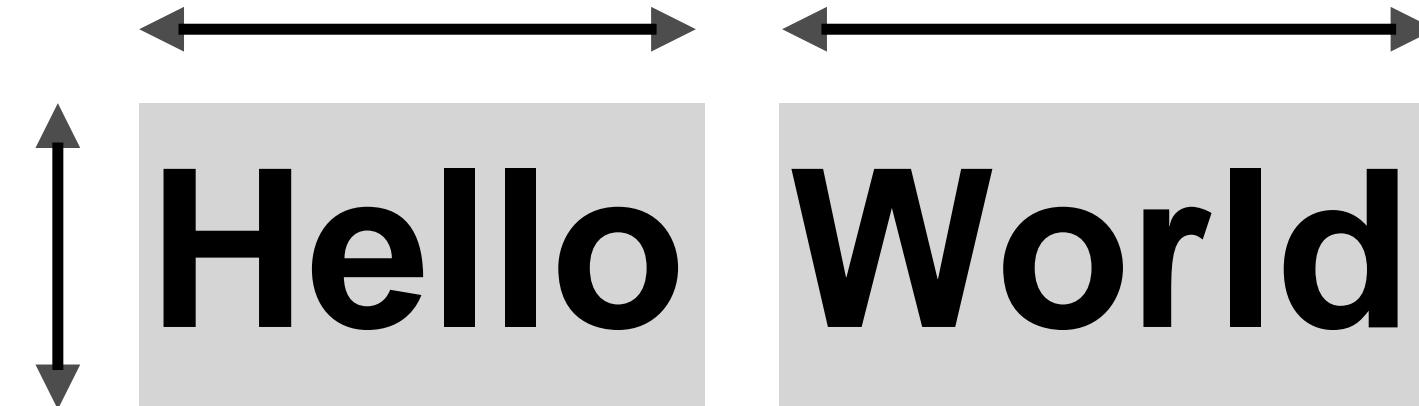


```
<span>Hello</span>
```

```
<span>World</span>
```

포함한 콘텐츠 크기만큼 자동으로 줄어듬!

포함한 콘텐츠 크기만큼  
자동으로 줄어듬!

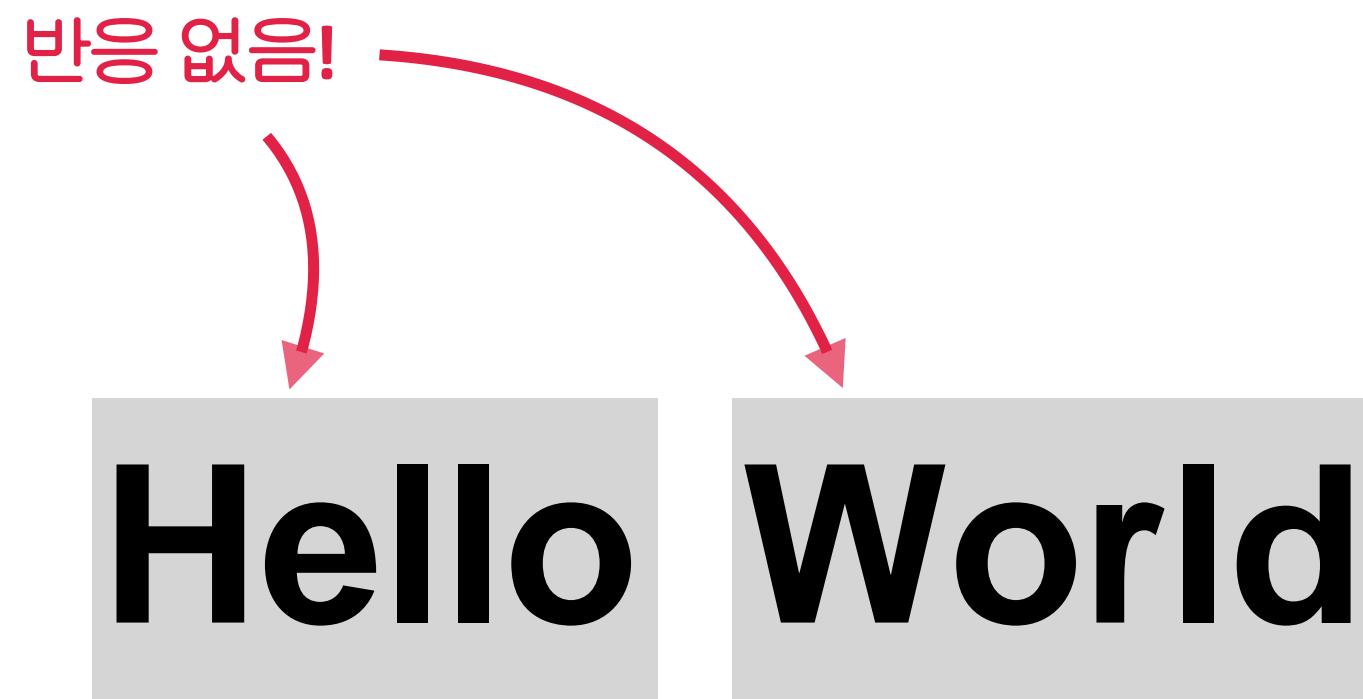


요소의 가로 너비를 지정하는 CSS 속성

```
<span style="width: 100px;">Hello</span>
```

```
<span style="height: 100px;">World</span>
```

요소의 세로 너비를 지정하는 CSS 속성

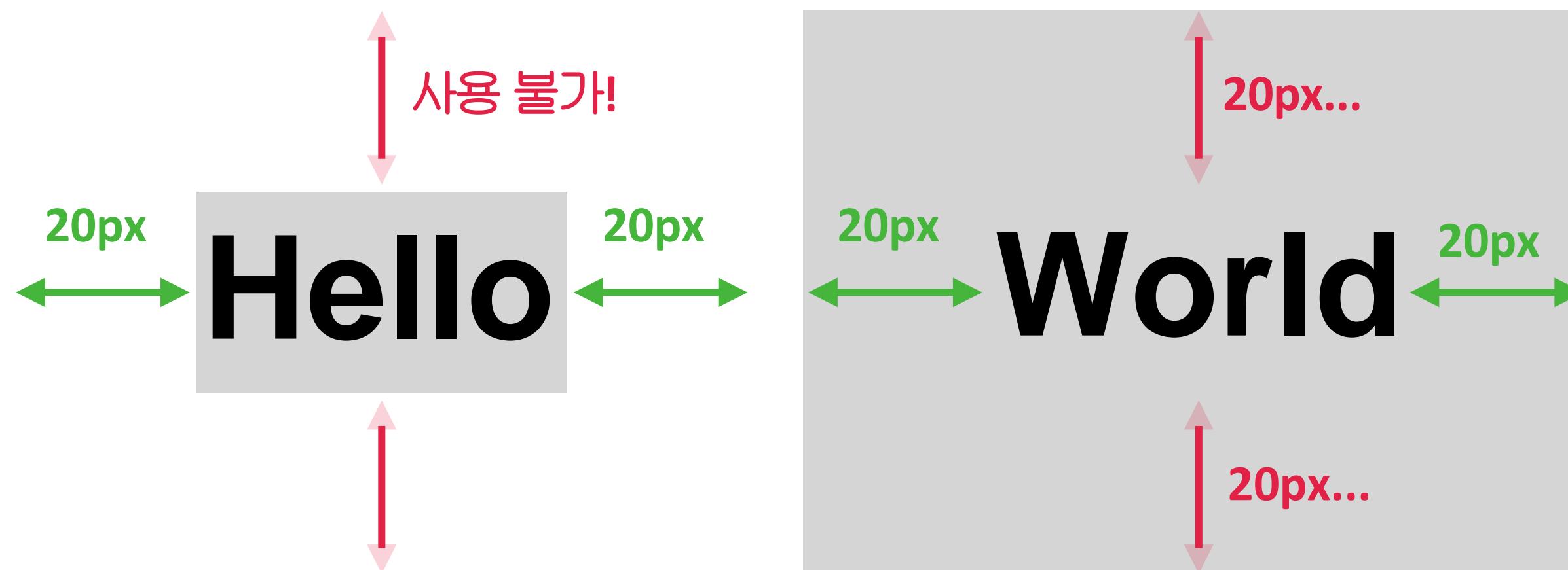


요소의 외부 여백을 지정하는 CSS 속성

```
<span style="margin: 20px 20px;">Hello</span>
```

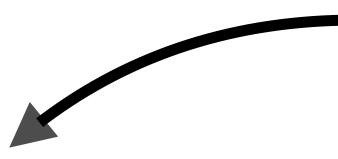
```
<span style="padding: 20px 20px;">World</span>
```

요소의 내부 여백을 지정하는 CSS 속성





```
<div>Hello</div>  
<div>World</div>
```



```
<div></div>
```

대표적인 블록 요소!  
본질적으로 아무것도 나타내지 않는,  
콘텐츠 영역을 설정하는 용도.

Hello  
World

요소가 수직으  
로 쌓임



```
<div>Hello</div>  
<div>World</div>
```



```
<div></div>
```

대표적인 블록 요소!  
본질적으로 아무것도 나타내지 않는,  
콘텐츠 영역을 설정하는 용도.

부모 요소의 크기만큼 자동으로 늘어남!

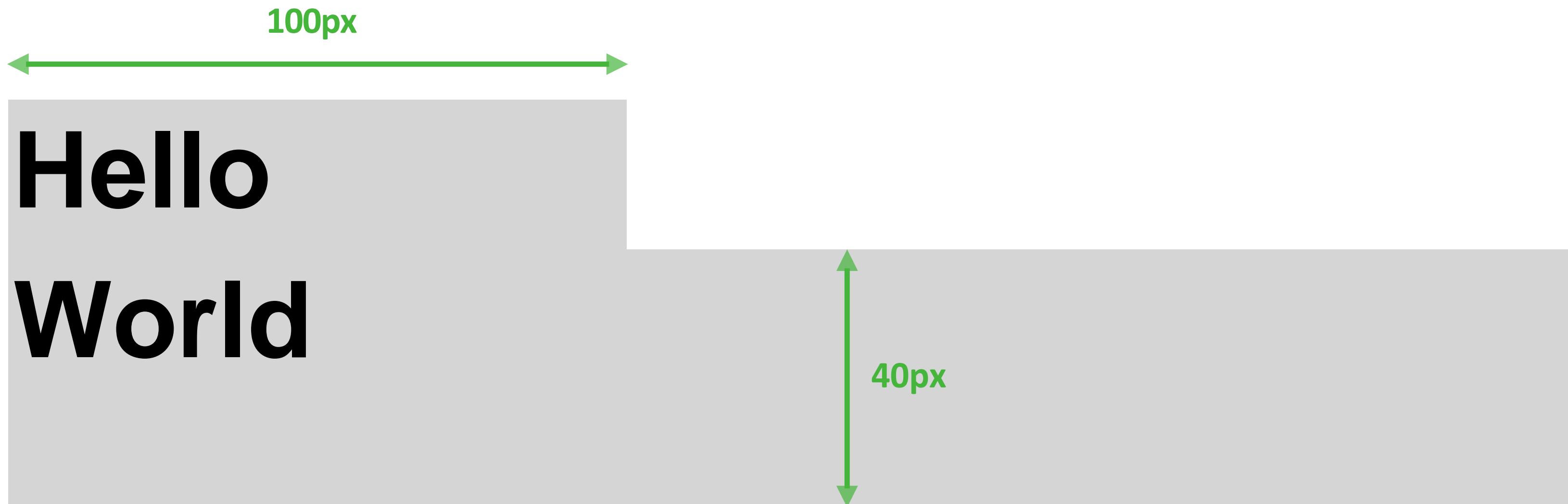


요소의 가로 너비를 지정하는 CSS 속성

```
<div style="width: 100px;">Hello</div>
```

```
<div style="height: 40px;">World</div>
```

요소의 세로 너비를 지정하는 CSS 속성

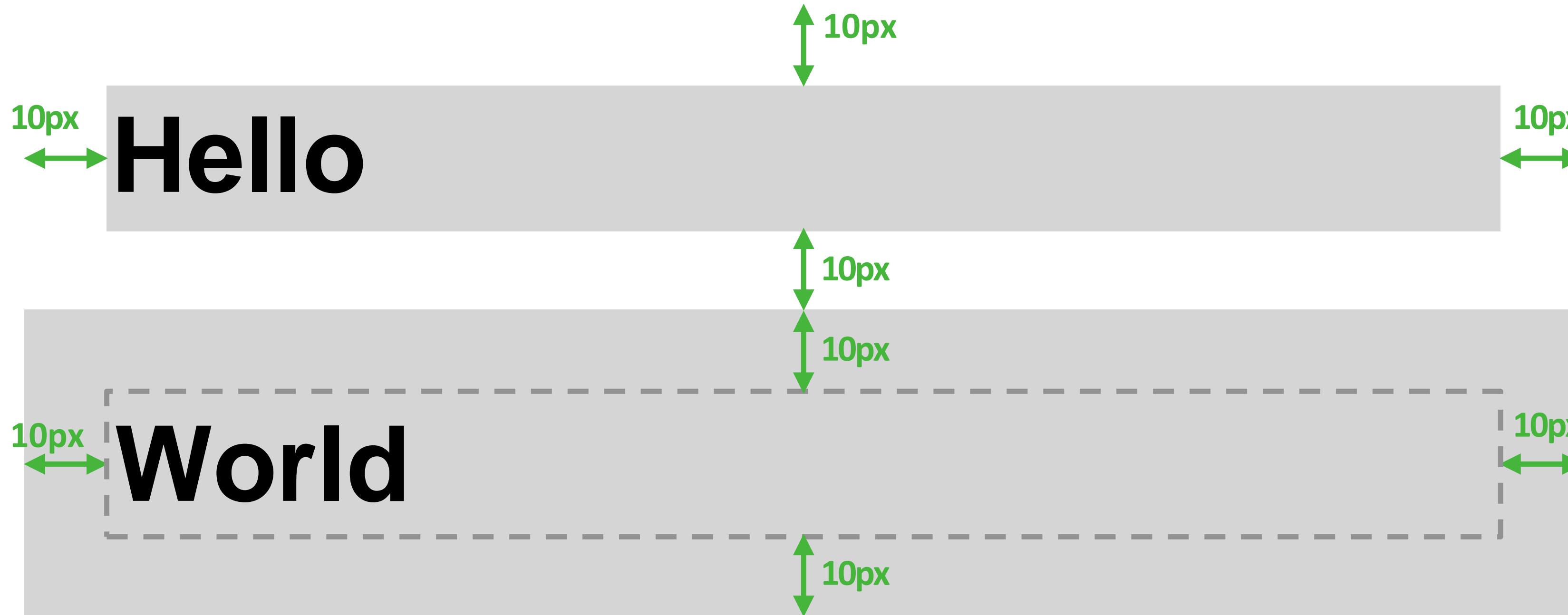


요소의 외부 여백을 지정하는 CSS 속성

```
<div style="margin: 10px;">Hello</div>
```

```
<div style="padding: 10px;">World</div>
```

요소의 내부 여백을 지정하는 CSS 속성



블록 요소

가능!

```
<div><div>/div></div>
```

```
<div><span>/span>/div>
```

가능!

인라인 요소

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```

문서의 전체 범위

HTML 문서가 어디에서 시작하고, 어디에서 끝나는지 알려주는 역할.

```
<!DOCTYPE html>  
<html>  
  <head>  
    </head>  
  <body>  
    </body>  
</html>
```

문서의 정보를 나타내는 범위

웹 브라우저가 해석해야 할  
웹 페이지의 제목, 설명, 사용할 파일 위치, 스타일(CSS) 같은,  
웹페이지의 보이지 않는 정보를 작성하는 범위.

```
<!DOCTYPE html>  
<html>  
  <head>  
    </head>  
    <b><body></body></b>  
  </html>
```

## 문서의 구조를 나타내는 범위

사용자 화면을 통해 보여지는  
로고, 헤더, 푸터, 내비게이션, 메뉴, 버튼, 이미지 같은,  
웹페이지의 보여지는 구조를 작성하는 범위.

```
<!DOCTYPE html>  
<html>  
  <head>  
    </head>  
    <body>  
      </body>  
    </html>
```

## 문서의 HTML 버전을 지정

HTML1

HTML2

HTML3

HTML4

XHTML

**HTML5 (표준)**

DOCTYPE(DTD, Document Type Definition)은 마크업 언어에서 문서 형식을 정의하며, 웹 브라우저가 어떤 HTML 버전의 해석 방식으로 페이지를 이해하면 되는지를 알려주는 용도.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Trans ...
```

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

문서의 HTML 버전을 지정

HTML1

HTML2

HTML3

HTML4

XHTML

HTML5 (표준)

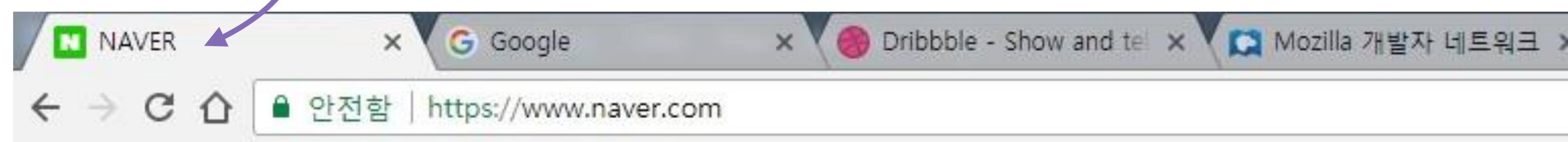
```
<!DOCTYPE html>
<html>
  <head>          문자인코딩 방식
    <meta charset="UTF-8" />
    <meta name="author" content="HEROPY" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  </head>          정보의 종류
  <body></body>
</html>
```

<meta />는 HTML 문서(웹페이지)의 제작자, 내용, 키워드 같은 여러 정보를 검색엔진이나 브라우저에게 제공.

```
<!DOCTYPE html>
<html>
| <head>
| | <title>Naver</title>
| </head>
| <body></body>
</html>
```

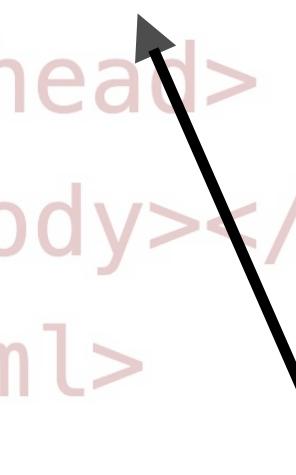
HTML 문서의 제목(title)을 정의.

웹 브라우저 탭에 표시됨!



```
<!DOCTYPE html>
<html>          필수 속성!
  <head>      가져올 문서와 관계
    <link rel="stylesheet" href="./main.css" />
    <link rel="icon" href="./favicon.png">
  </head>
  <body></body>
</html>
```

외부 문서를 가져와 연결할 때 사용.  
(대부분 CSS 파일)



```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        color: red;
      }
    </style>          작성된 CSS
  </head>
  <body></body>
</html>
```

스타일(css)를 HTML 문서 안에서 작성하는 경우에 사용.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="./main.js"></script>
    <script>
      console.log('Hello world!')
    </script>
  </head>
  <body></body>
</html>
```

1 자바스크립트(JS) 파일 가져오는 경우.

작성된 JS

2 자바스크립트(JS)를 HTML 문서 안에서 작성하는 경우.

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <div>
      <h1>오늘의 날씨</h1>
      <p>중부 집중호우, 남부는 열대야, 12호 태풍 북상 중...</p>
      
    </div>
  </body>
</html>
```

블록(상자) 요소

특별한 의미가 없는 구분을 위한 요소. (Division)

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <div>
      <h1>오늘의 날씨</h1>
      <p>중부 집중호우, 남부는 열대야, 12호 태풍 북상 중...</p>
      
    </div>
  </body>
</html>
```

블록(상자) 요소

제목을 의미하는 요소. (**Heading**)

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <div>
      <h1>오늘의 날씨</h1>
      <p>중부 집중호우, 남부는 열대야, 12호 태풍 북상 중..</p>
      
    </div>
  </body>
</html>
```

블록(상자) 요소

문장을 의미하는 요소. (Paragraph)

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <div>
      <h1>오늘의 날씨</h1>
      <p>중부 집중호우, 남부는 열대야, 12호 태풍 북상 중...</p>
      
    </div>
  </body>
</html>
```

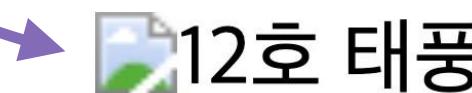
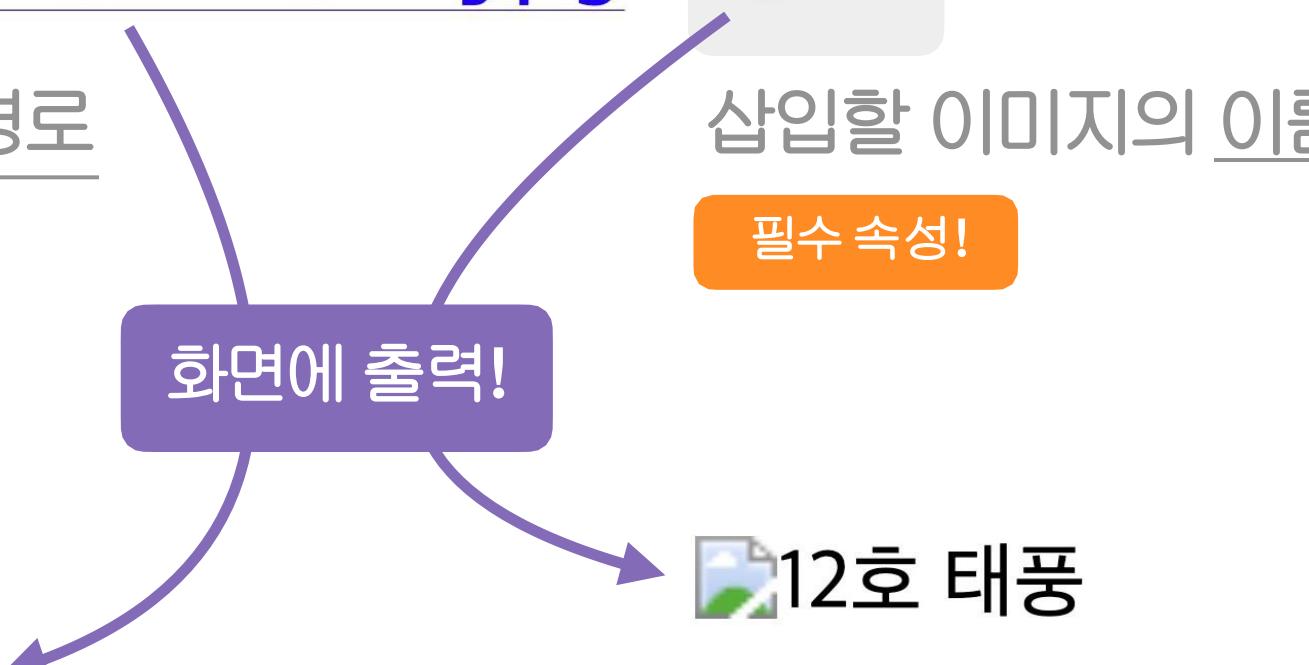
인라인(글자) 요소

이미지를 삽입하는 요소. (Image)

삽입할 이미지의 경로

필수 속성!

화면에 출력!



```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>제목1</h1>
    <h2>제목2</h2>
    <h3>제목3</h3>
    <h4>제목4</h4>
    <h5>제목5</h5>
    <h6>제목6</h6>
  </body>
</html>
```

블록(상자) 요소

제목을 의미하는 요소. (**Heading**) 숫자가  
작을수록 더 중요한 제목을 정의.

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <ul>
      <li>사과</li>
      <li>딸기</li>
      <li>수박</li>
      <li>오렌지</li>
    </ul>
  </body>
</html>
```

블록(상자) 요소

순서가 필요없는 목록의 집합을 의미. (Unordered List)

블록(상자) 요소

목록 내 각 항목 (List Item)

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
```

인라인(글자) 요소

다른/같은 페이지로 이동하는 하이퍼링크를 지정하는 요소.  
(Anchor)

```
    <a href="https://www.naver.com">NAVER</a>
```

```
    <a href="https://www.google.com" target="_blank">Google</a>
```

</body> 링크 URL

링크 URL의 표시(브라우저 탭) 위치

화면에 출력!

Naver Google

밑줄?? 파란색 글자??

```
<!DOCTYPE html>
<html>
|   <head></head>
|   <body>
|       <p>
|           |   <span>동해물</span>과 백두산이 마르고 닳도록
|       </p>
|   </body>
|</html>
```

인라인(글자) 요소

특별한 의미가 없는 구분을 위한 요소.

VS

<div></div>

<p>

<span>동해물</span>과 백두산이 마르고 닳도록

</p>

</body>

</html>

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      span { color: red; }
    </style>
  </head>
  <body>
    <p>
      <span>동해물</span>과 백두산이 마르고 닳도록
    </p>
  </body>
</html>
```

스타일(CSS) 추가!

```
span { color: red; }
```

```
</style>
```

스타일 적용

```
<span>동해물</span>과 백두산이 마르고 닳도록
```

동해물과 백두산이 마르고 닳도록

화면에 출력!

```
<!DOCTYPE html>
<html>
|   <head></head>
|   <body>
|       <p>
|           동해물과 백두산이<br/>마르고 닳도록
|       </p>
|   </body>
|</html>
```

인라인(글자) 요소

줄바꿈 요소. (Break)

화면에 출력!

줄바꿈 됨!

동해물과 백두산이  
마르고 닳도록

```
<!DOCTYPE html>
<html>
| <head></head>
| <body>
| | <input type="text" />
| </body>    입력받을 데이터의 타입
| </html>
```

인라인(글자) 요소      블록(상자) 요소      = `Inline-block`

사용자가 데이터를 입력하는 요소.

```
<!DOCTYPE html>
<html>
| <head></head>
| <body>          입력받을 데이터의 타입
|   | <input type="text" />
| </body>
| </html>
```

화면에 출력!



```
<!DOCTYPE html>
<html>
| <head></head>
| <body>
| | <input type="text" value="HEROPY!" />
| </body>
| </html>
```

미리 입력된 값(데이터)

화면에 출력!



```
<!DOCTYPE html>
<html>
| <head></head>
| <body>
| | <input type="text" placeholder="이름을 입력하세요!" />
| </body>
| </html>
```

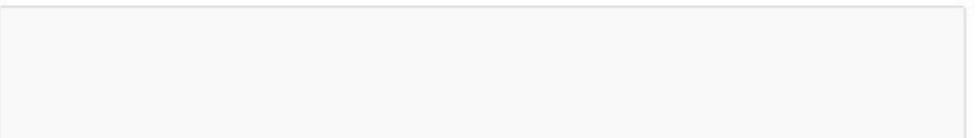
사용자가 입력할 값(데이터)의 힌트

화면에 출력!

이름을 입력하세요!

```
<!DOCTYPE html>
<html>
| <head></head>
| <body>          입력 요소 비활성화
|   | <input type="text" disabled />
|   </body>
| </html>
```

화면에 출력!



```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <label>
      <input type="checkbox" /> Apple
    </label>
    <label>
      <input type="checkbox" /> Banana
    </label>
  </body>
</html>
```

인라인(글자) 요소

라벨 가능 요소(input)의 제목.

사용자에게 체크 여부를 입력 받음!

<input type="checkbox" /> Apple

Apple  Banana

화면에 출력!

```
<!DOCTYPE html>
<html>
| <head></head>
| <body>
|   <label>
|     <input type="checkbox" /> Apple
|   </label>
|   <label>
|     <input type="checkbox" checked /> Banana
|   </label>
| </body>
| </html>
```



```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
```

**<label>**

사용자에게 체크 여부를  
그룹에서 1개만 입력 받음!

  | <input type="radio" name="fruits" />

  | </label>

  | <label>

  | | <input type="radio" name="fruits" /> Banana

  | </label>

  | </body>

</html>

1개만 선택 가능 (택1)

Apple  Banana

'fruits'란 이름의 그룹

화면에 출력!

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <table>
      <tr>
        <td>A</td><td>B</td>
      </tr>
      <tr>
        <td>C</td><td>D</td>
      </tr>
    </table>
  </body>
</html>
```

테이블 요소  
= table

표 요소, 행(Row)과 열(Column)의 집합.

화면에 출력!

A	B
C	D

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <table>
      <tr>
        <td>A</td><td>B</td>
      </tr>
      <tr>
        <td>C</td><td>D</td>
      </tr>
    </table>
  </body>
</html>
```

테이블 요소  
= table-row

행(Row)을 지정하는 요소. (Table Row)

화면에 출력!

A	B
C	D

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <table>
      <tr>
        <td>A</td><td>B</td>
      </tr>
      <tr>
        <td>C</td><td>D</td>
      </tr>
    </table>
  </body>
</html>
```

테이블 요소

= table-cell

열(Column)을 지정하는 요소. (Table Data)

화면에 출력!

A	B
C	D

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <table>
      <tr>
        <td>A</td><td>B</td>
      </tr>
      <tr>
        <td>C</td><td>D</td>
      </tr>
    </table>
  </body>
</html>
```



```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <!--애국가-->
    <div>
      
      <h1>애국가</h1>
      <p>동해물과 백두산이 마르고 닳도록</p>
    </div>
  </body>
</html>
```

Ctrl ⌘ 혹은 Cmd /

<!-- Comment -->

수정사항이나 설명 등을 작성(주석).

브라우저는 이 태그를 해석하지 않기 때문에 화면에 내용이 표시되지 않음.

# **CSS 기본 문법**

**선택자 { 속성: 값; }**

스타일(CSS)을 적용할 대상 (Selector)

선택자 { 속성: 값; }

스타일(CSS)의 종류 (Property)

선택자 { 속성: 값; }

스타일(CSS)의 값 (Value)

**선택자 { 속성: 값; }**

속성은 값이다

선택자 { 속성: 값; 속성: 값; }

스타일 범위의 시작

스타일 범위의 끝

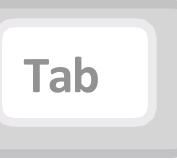
A diagram illustrating a CSS rule. The rule is displayed in a light gray rounded rectangle: `div { color: red; }`. A curved arrow labeled "글자색" (text color) points to the word "color". Another curved arrow points to the word "red", which is enclosed in a small red square labeled "빨강" (red).

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

```
div { color: red; margin: 20px; }
```

요소 외부 여백

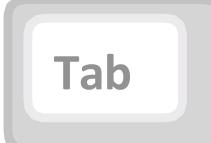
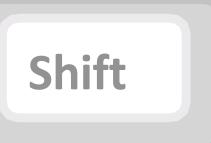
20픽셀



들여쓰기  
(Indent)

vs

내어쓰기  
(Outdent)



```
div {  
    color: red;  
    margin: 20px;  
}
```

```
/* 설명 작성  
 */ div {  
    color: red;  
    margin: 20px;  
}
```

주석 시작

`/* 설명 작성`

`*/ div {`

`color: red;`

`margin: 20px;`

`}`

주석 끝

브라우저는 이 범위를 해석하지 않음

`Ctrl`

혹은

`Cmd`

`/`

`/`

# CSS 선언 방식

내장 방식

링크 방식

인라인 방식

@import 방식

```
<style>
  div {
    color: red;
    margin: 20px;
  }
</style>
```

내장 방식

<style>/</style>의 내용(Contents)으로 스타일을 작성하는 방식

인라인 방식

요소의 style 속성에 직접 스타일을 작성하는 방식(  
선택자 없음)

```
<div style="color: red; margin: 20px;"></div>
```

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

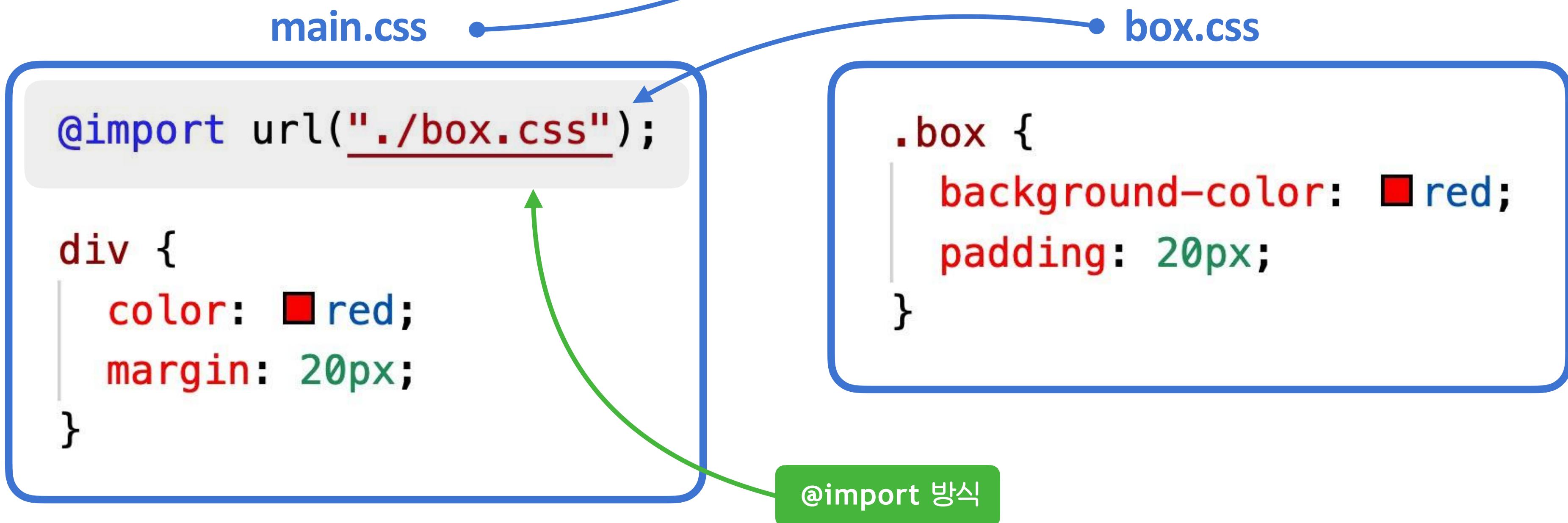
링크 방식

<link>로 외부 CSS 문서를  
가져와서 연결하는 방식

main.css

```
div {  
    color: red;  
    margin: 20px;  
}
```

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



CSS의 `@import` 규칙으로 CSS 문서 안에서 또 다른 CSS 문서를 가져와 연결하는 방식

# CSS 선택자

기본

복합

가상 클래스

가상 요소

속성

\*

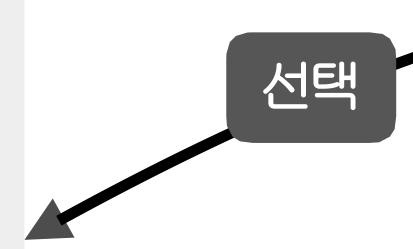
기본

전체 선택자 (Universal Selector)

모든 요소를 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li>오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span>오렌지</span>
</div>
```

선택



```
* {
  color: red;
```

# ABC

기본

태그 선택자 (Type Selector)

태그 이름이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li>오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span>오렌지</span>
</div>
```

선택

```
li {
  color: red;
}
```

기본

클래스 선택자 (Class Selector)

.ABC

HTML class 속성의 값이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
.orange {
  color: red;
}
```

기본

아이디 선택자 (ID Selector)

# #ABC

HTML id 속성의 값이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li id="orange" class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
#orange {
  color: red;
}
```

# ABCXYZ

복합

일치 선택자 (Basic Combinator)

선택자 ABC와 XYZ를 동시에 만족하는 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
span.orange {
  color: red;
```

# ABC > XYZ

복합

자식 선택자 (Child Combinator)

선택자 ABC의 자식 요소 XYZ 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
ul > .orange {
  color: red;
```

# ABC XYZ

복합

하위(후손) 선택자 (Descendant Combinator)

선택자 ABC의 하위 요소 XYZ 선택!  
띄어쓰기가 선택자의 기호!

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
<span class="orange">오렌지</span>
```



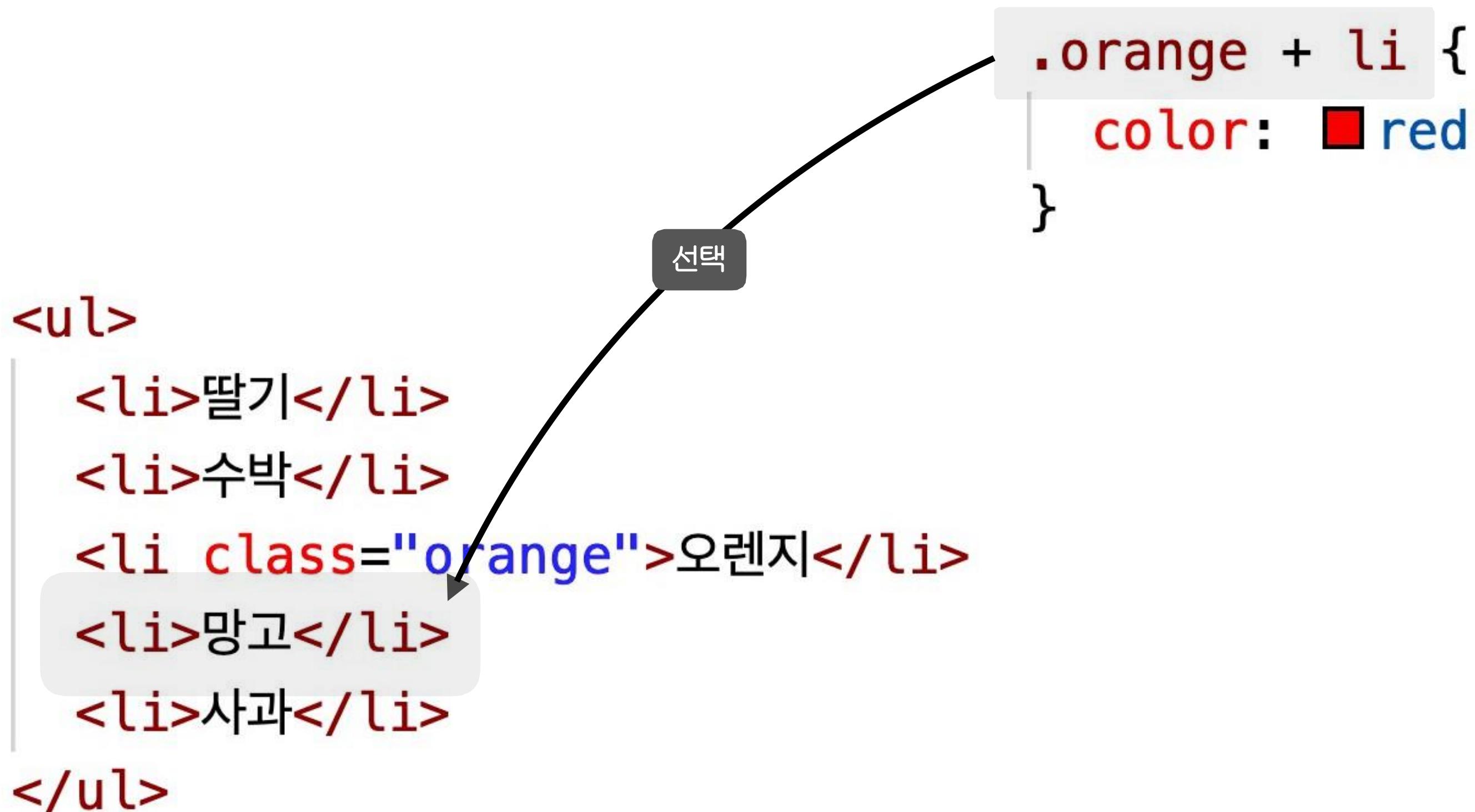
```
div .orange {
  color: red;
}
```

# ABC + XYZ

복합

인접 형제 선택자 (Adjacent Sibling Combinator)

선택자 ABC의 다음 형제 요소 XYZ 하나를 선택.



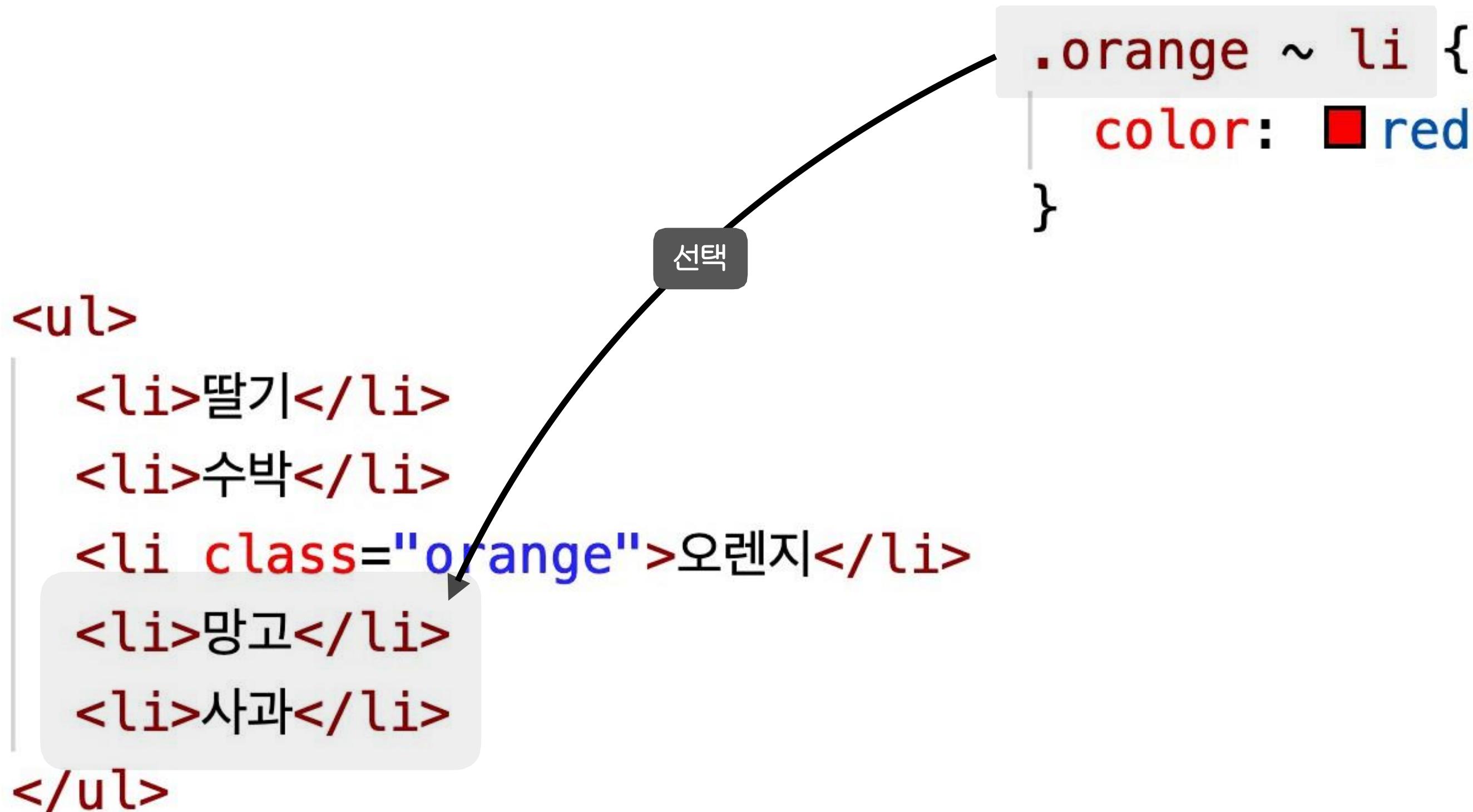
# ABC ~ XYZ

복합

일반 형제 선택자 (General Sibling Combinator)



선택자 ABC의 다음 형제 요소 XYZ 모두를 선택.



# ABC:hover

가상 클래스 선택자 (Pseudo-Classes)

HOVER

선택자 ABC 요소에 마우스 커서가 올라가 있는 동안 선택.



a:hover {  
color: red;  
}

```
<a href="https://www.naver.com">NAVER</a>
```

# ABC:active

가상 클래스 선택자 (Pseudo-Classes)

ACTIVE

선택자 ABC 요소에 마우스를 클릭하고 있는 동안 선택.



```
a:active {  
    color: red;  
}
```

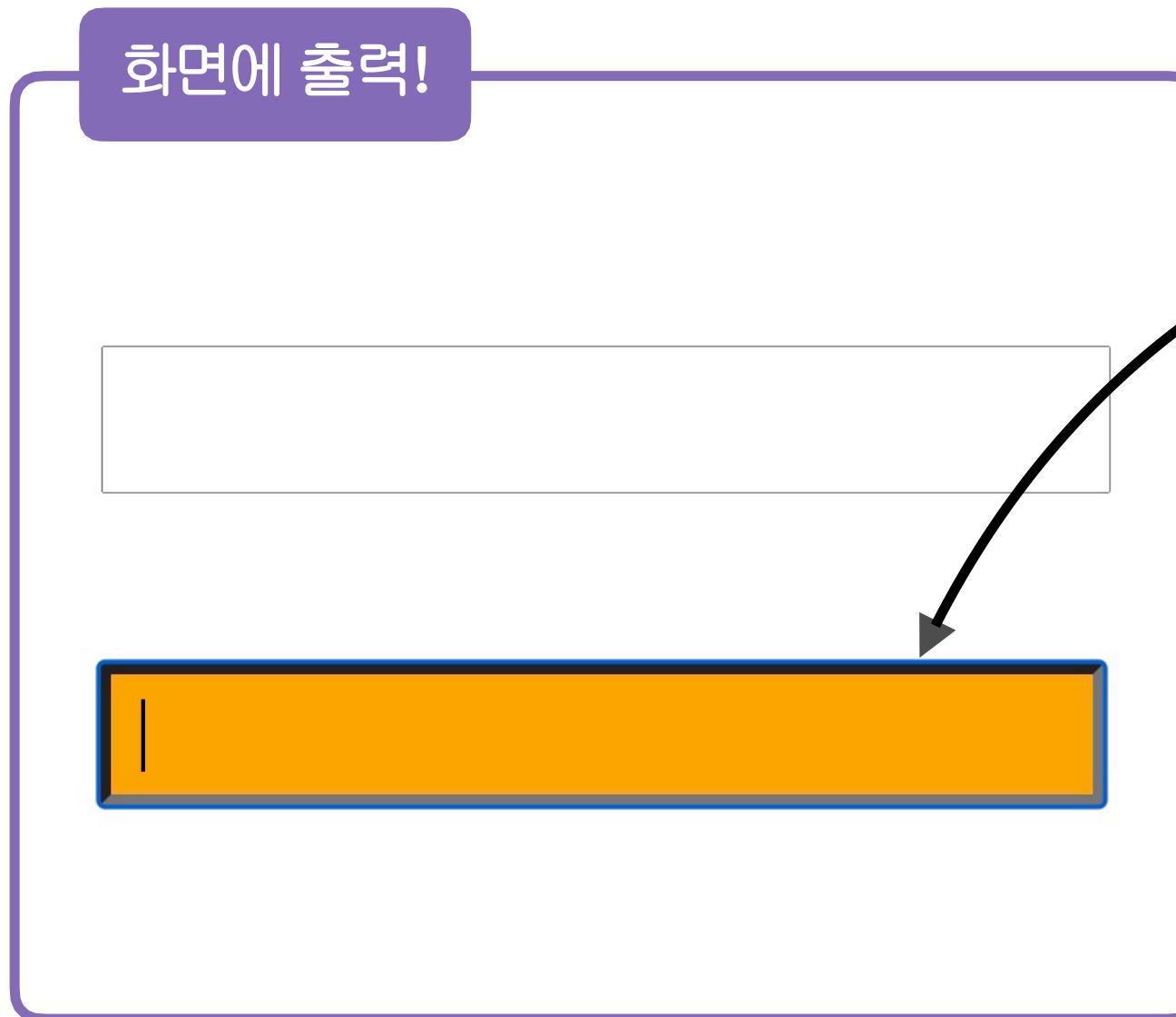
```
<a href="https://www.naver.com">NAVER</a>
```

# ABC:focus

가상 클래스 선택자 (Pseudo-Classes)

FOCUS

선택자 ABC 요소가 포커스되면 선택.



input:focus {  
background-color: orange;  
}

# ABC:first-child

가상 클래스 선택자 (Pseudo-Classes)

FIRST CHILD

선택자 ABC가 형제 요소 중 첫째라면 선택.

```
<div class="fruits">  
  <span>딸기</span>  
  <span>수박</span>  
  <div>오렌지</div>  
  <p>망고</p>  
  <h3>사과</h3>  
</div>
```

선택

```
.fruits span:first-child {  
  color: red;  
}
```

# ABC:first-child

가상 클래스 선택자 (Pseudo-Classes)

FIRST CHILD

선택자 ABC가 형제 요소 중 첫째라면 선택.

?

```
.fruits div:first-child {  
    color: red;  
}
```

```
<div class="fruits">  
    <span>딸기</span>  
    <span>수박</span>  
    <div>오렌지</div>  
    <p>망고</p>  
    <h3>사과</h3>  
</div>
```

# ABC:last-child

가상 클래스 선택자 (Pseudo-Classes)

LAST CHILD

선택자 ABC가 형제 요소 중 막내라면 선택.

선택

```
.fruits h3:last-child {  
    color: red;  
}
```

```
<div class="fruits">  
    <span>딸기</span>  
    <span>수박</span>  
    <div>오렌지</div>  
    <p>망고</p>  
    <h3>사과</h3>  
</div>
```

# ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes)

NTH CHILD

선택자 ABC가 형제 요소 중 (n)째라면 선택.

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

선택

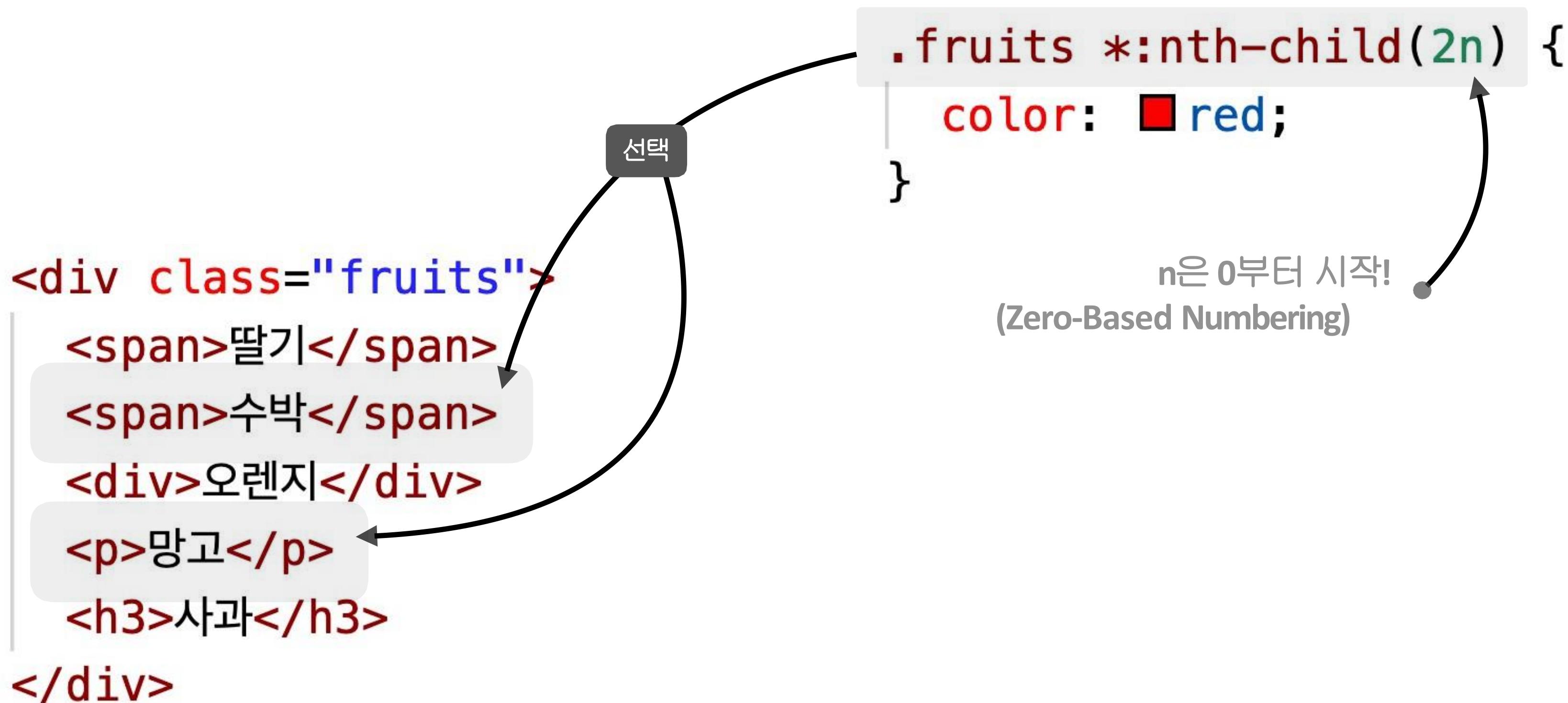
```
.fruits *:nth-child(2) {
  color: red;
```

# ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes)

NTH CHILD

선택자 ABC가 형제 요소 중 (n)째라면 선택.

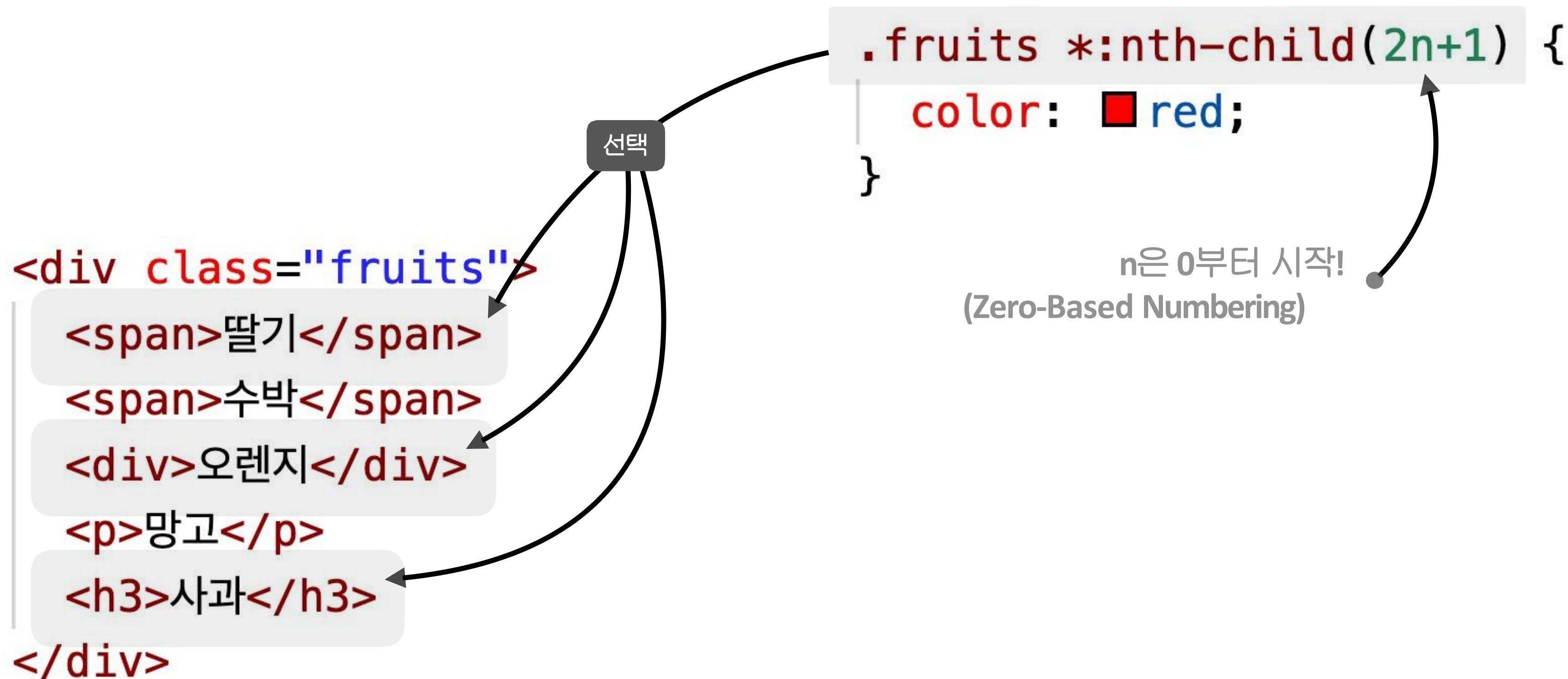


# ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes)

NTH CHILD

선택자 ABC가 형제 요소 중 (n)째라면 선택.



# ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes)

NTH CHILD



선택자 ABC가 형제 요소 중 (n)째라면 선택.

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

선택

```
.fruits *:nth-child(n+2) {
  color: red;
```

n은 0부터 시작!  
(Zero-Based Numbering)

# ABC:not(XYZ)

부정 선택자 (Negation)

NOT

선택자 XYZ가 아닌 ABC 요소 선택.

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

선택

```
.fruits *:not(span) {
  color: red;
}
```

# ABC::before

인라인(글자) 요소

화면에 출력!

앞! Content!

```
<div class="box">
```

Content!

```
</div>
```

가상 요소 선택자 (Pseudo-Elements)

BEFORE

선택자 ABC 요소의 내부 앞에 내용(Content)을 삽입.

```
.box::before {  
    content: "앞!";  
}
```

::before

# ABC::after

인라인(글자) 요소

화면에 출력!

Content! 뒤!

```
<div class="box">
```

Content!

```
</div>
```

가상 요소 선택자 (Pseudo-Elements)

AFTER

선택자 ABC 요소의 내부 뒤에 내용(Content)을 삽입.

```
.box::after {  
    content: "뒤!";  
}
```

::after

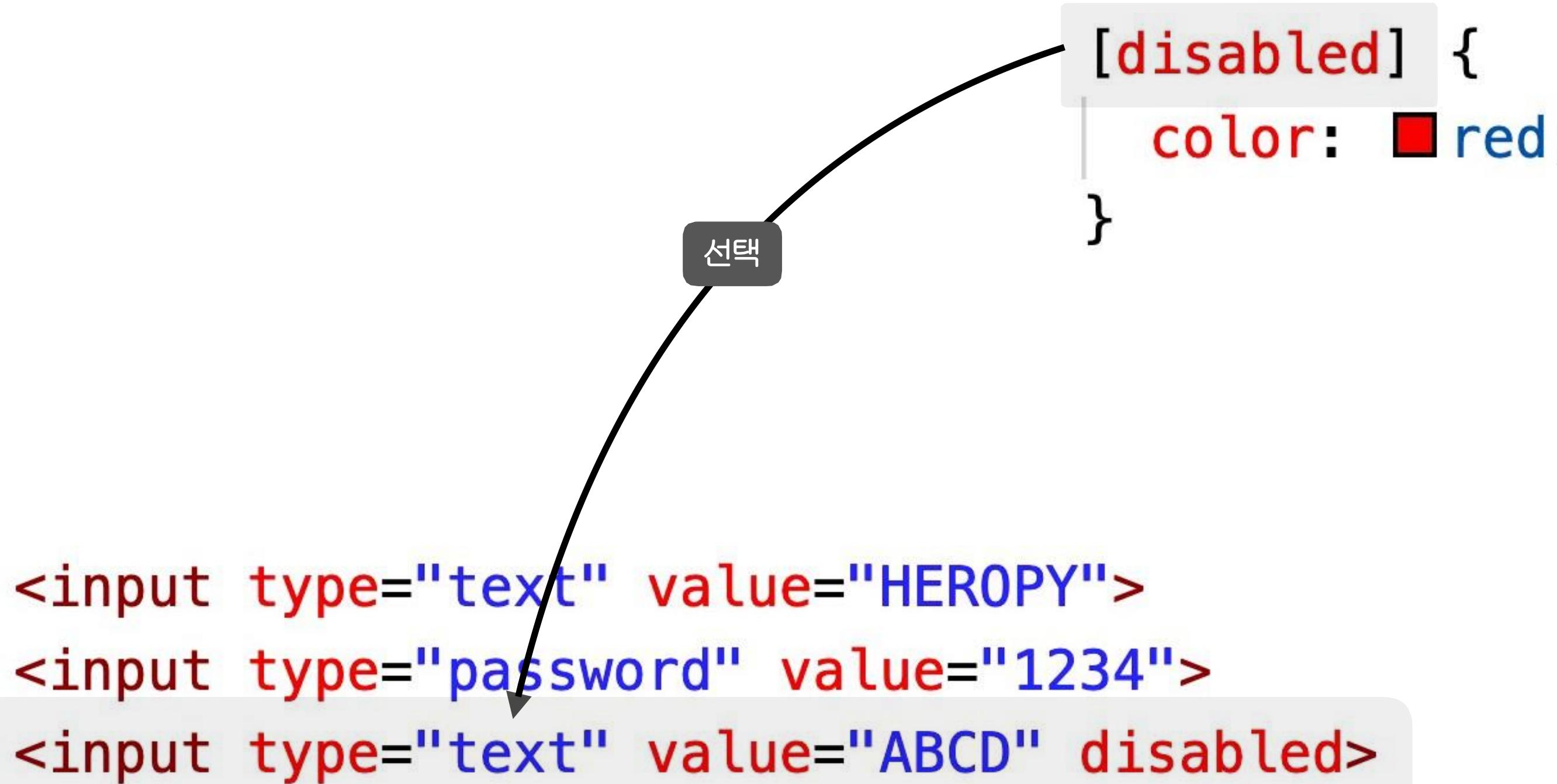


# [ABC]

속성 선택자(Attribute)

ATTR

속성 ABC을 포함한 요소 선택



# [ABC]

속성 선택자(Attribute)

ATTR

속성 ABC을 포함한 요소 선택

선택

```
[type] {  
  color: red;  
}
```

```
<input type="text" value="HER0PY">  
<input type="password" value="1234">  
<input type="text" value="ABCD" disabled>
```

# [ABC="XYZ"]

속성 선택자(Attribute)

ATTR=VALUE

속성 ABC을 포함하고 값이 XYZ인 요소 선택.

선택

```
[type="password"] {  
    color: red;  
}
```

```
<input type="text" value="HER0PY">  
<input type="password" value="1234">  
<input type="text" value="ABCD" disabled>
```

**스타일 상속**

```
.animal {  
    color: red;  
}  
  
<div class="ecosystem">생태계  
<div class="animal">동물  
    <div class="tiger">호랑이</div>  
    <div class="lion">사자</div>  
    <div class="elephant">코끼리</div>  
</div>  
<div class="plant">식물</div>  
</div>
```

선택

화면에 출력!

생태계  
동물  
호랑이  
사자  
코끼리  
식물

?

# 상속되는 CSS 속성들..

모두 글자/문자 관련 속성들!

(모든 글자/문자 속성은 아님 주의!)

**font-style** : 글자 기울기

**font-weight** : 글자 두께

**font-size** : 글자 크기|line-

**height** : 줄 높이|font-

**family** : 폰트(서체)|color

: 글자 색상

**text-align** : 정렬

...

**선택자 우선순위**

우선순위란, 같은 요소가 여러 선언의 대상이 된 경우,  
어떤 선언의 CSS 속성을 우선 적용할지 결정하는 방법

1, 점수가 높은 선언이 우선함!

2, 점수가 같으면, 가장 마지막에 해석된 선언이 우선함!

선택

```
<div  
    id="color_yellow"  
    class="color_green"  
    style="color: orange;">  
    Hello world!  
</div>
```

```
div { color: red !important; }  
#color_yellow { color: yellow; }  
.color_green { color: green; }  
div { color: blue; }  
* { color: darkblue; }  
body { color: violet; }
```

과연 글자색은??

!important - 999999999점

```
{ color: red !important; }  
{ color: yellow; }  
{ color: green; }  
{ color: blue; }  
{ color: darkblue; }  
{ color: violet; }
```

div

ID 선택자 - 100점

#color\_yellow

Class 선택자 - 10점

.color\_green

태그 선택자 - 1점

div

전체 선택자 - 0점

\*

body

상속 - X

```
<div  
    id="color_yellow"  
    class="color_green"     인라인 선언 - 1000점  
    style="color: orange;">  
    Hello world!  
</div>
```

21점

```
.list li.item { color: ■ red; }
```

21점

```
.list li:hover { color: ■ red; }
```

11점

```
.box::before { content: "Good "; color: ■ red; }
```

101점

```
#submit span { color: ■ red; }
```

22점

```
header .menu li:nth-child(2) { color: ■ red; }
```

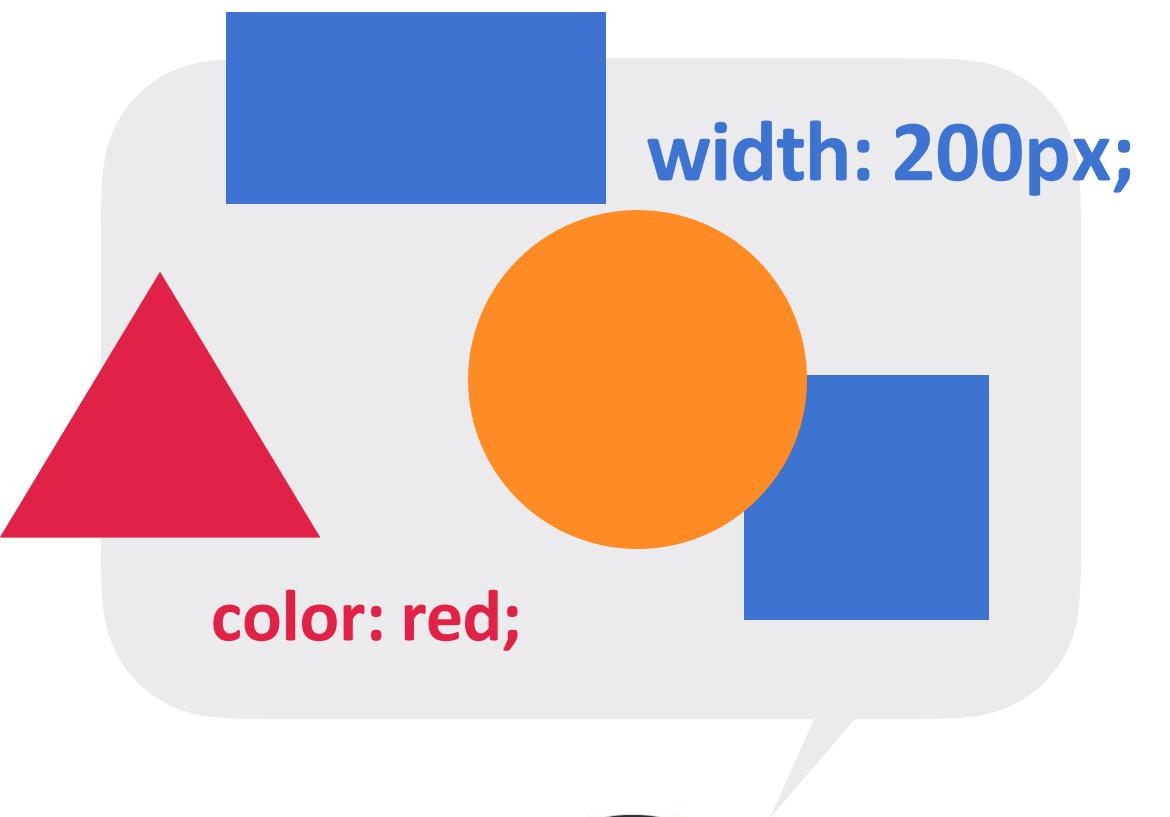
1점

```
h1 { color: ■ red; }
```

10점

```
:not(.box) { color: ■ red; }
```

# 속성(Properties)



박스 모델

글꼴, 문자

배경

배치플렉스(정렬) 전환

변환

띄움

애니메이션

그리드 다

단

필터

# 박스 모델

글꼴, 문자

배경

배치플렉스(

정렬) 전환

변환

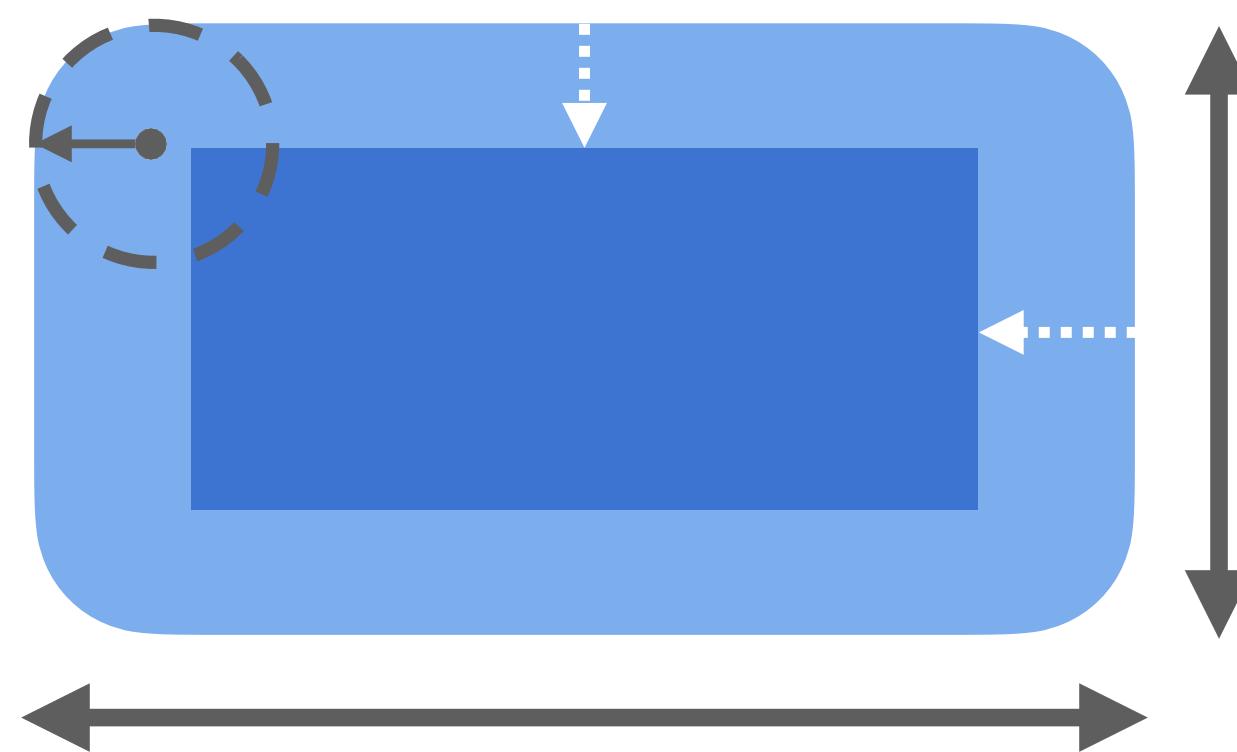
띄움

애니메이션

그리드 다

단

필터



박스 모델

글꼴, 문자

배경

배치플렉스(

정렬)전환

변환

띄움

애니메이션

그리드 다

단

필터

Hello world  
Good morning~

박스 모델

글꼴, 문자

**배경**

배치플렉스(

정렬)전환

변환

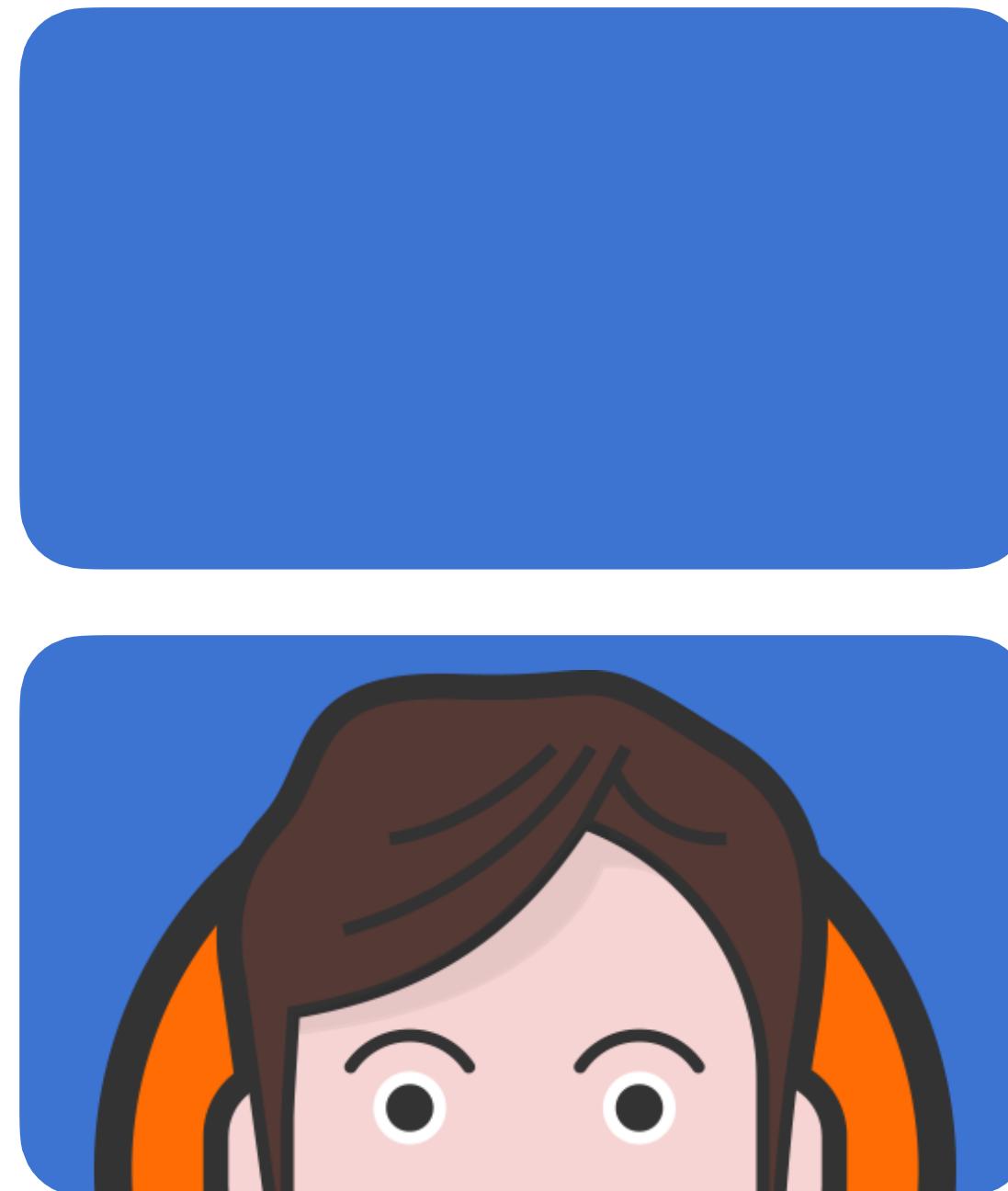
띄움

애니메이션

그리드 다

단

필터



박스 모델

글꼴, 문자

배경

**배치**플렉스(

정렬)전환

변환

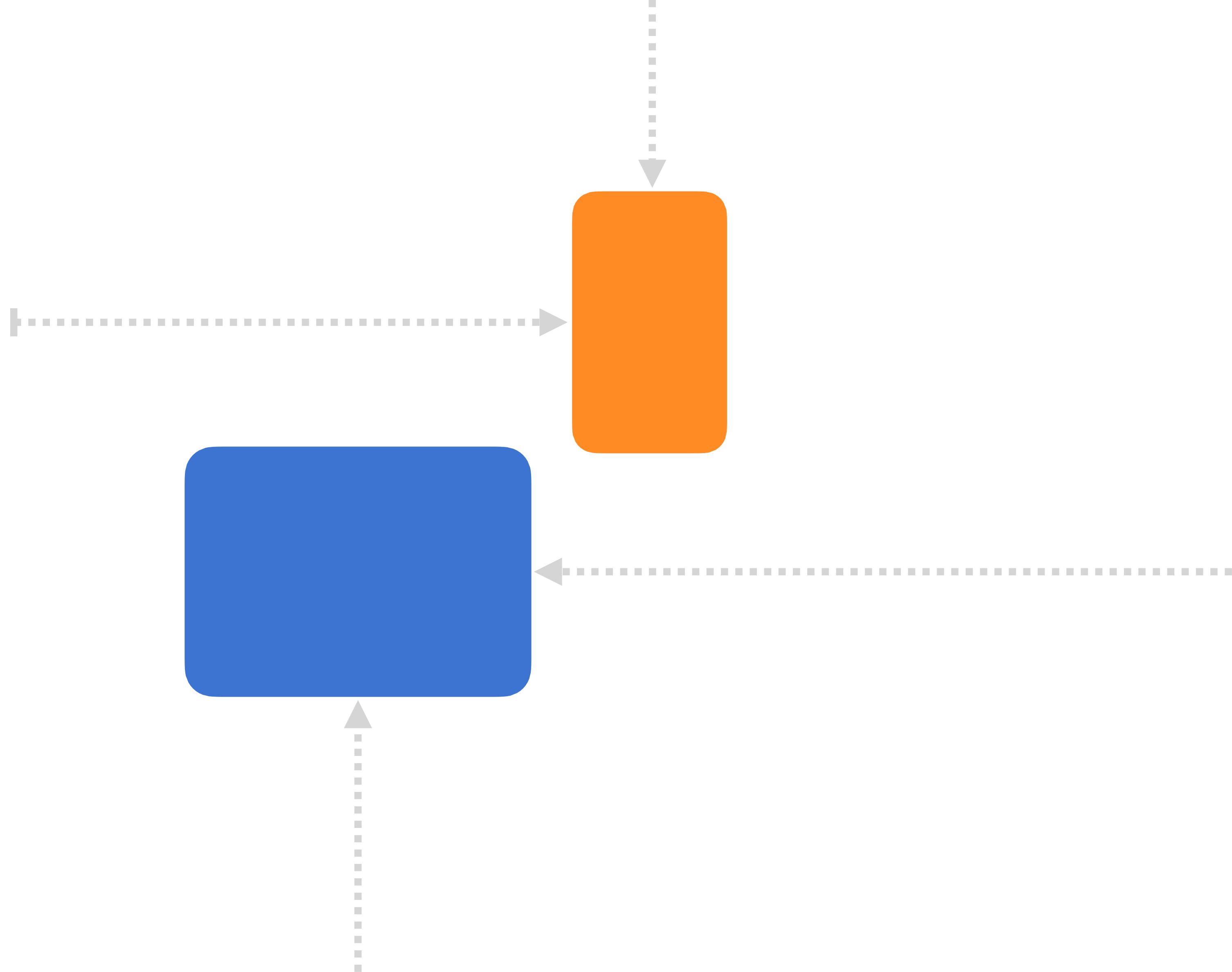
띄움

애니메이션

그리드 다

단

필터



박스 모델

글꼴, 문자

배경

배치 플렉스(

정렬) 전환

변환

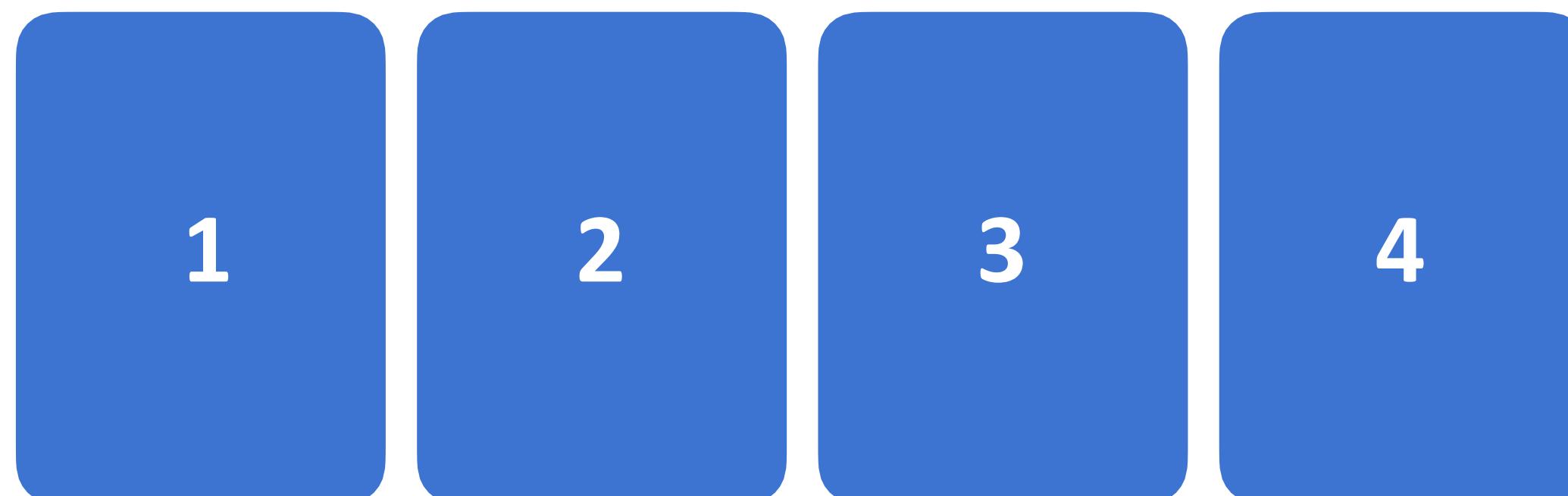
띄움

애니메이션

그리드 다

단

필터



박스 모델

글꼴, 문자

배경

배치플렉스(

정렬) **전환**

변환

띄움

애니메이션

그리드 다

단

필터



박스 모델

글꼴, 문자

배경

배치플렉스(

정렬) 전환

변환

띄움

애니메이션

그리드 다

단

필터



박스 모델

글꼴, 문자

배경

배치플렉스(정렬)전환

변환

**띄움**

애니메이션

그리드 디자인

단

필터



**Lorem  
Ipsum** is  
simply dummy  
text of the  
printing and

Ipsum has been the industry's  
standard dummy text ever  
since the 1500s, when an

박스 모델

글꼴, 문자

배경

배치플렉스(

정렬) 전환

변환

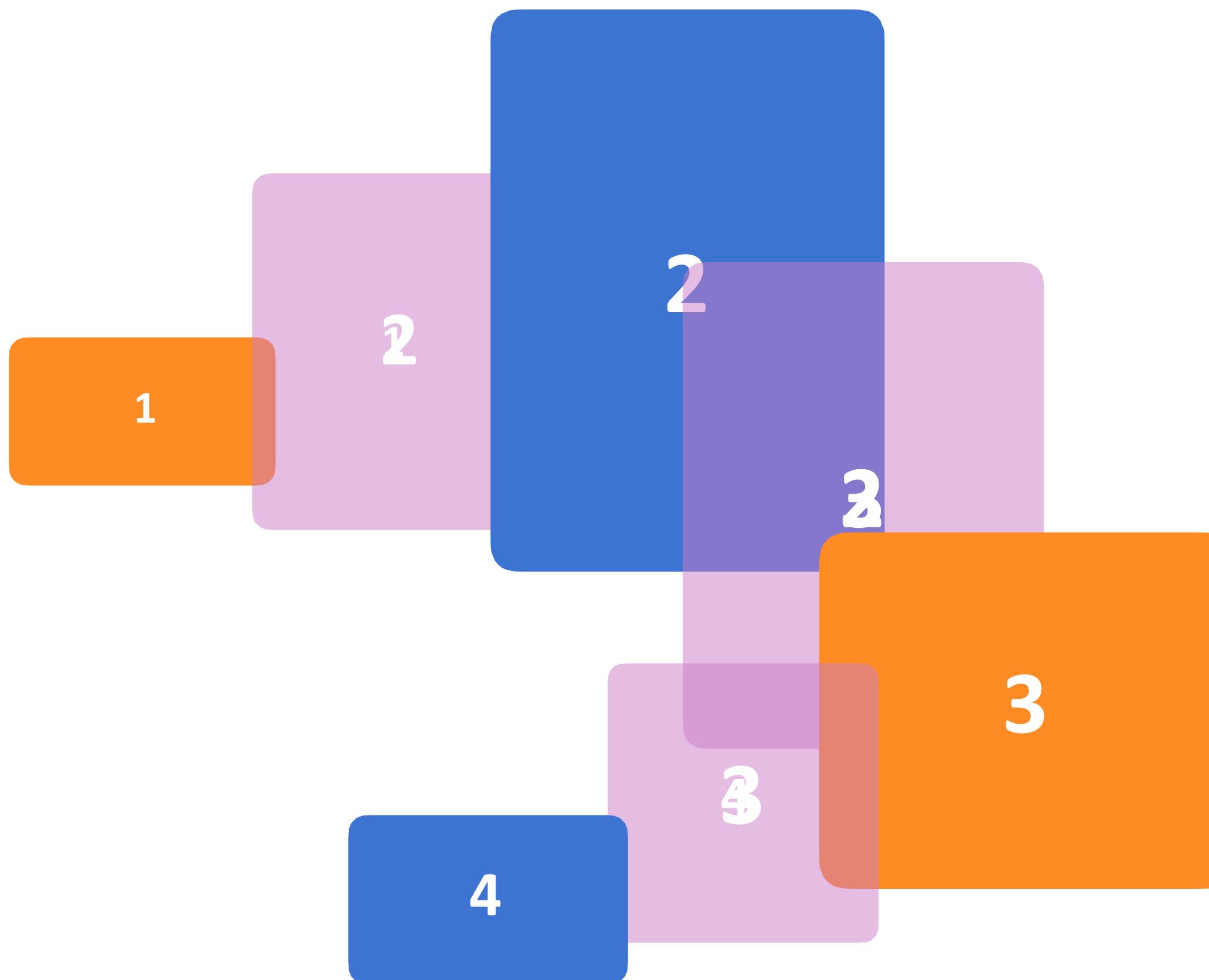
띄움

**애니메이션**

그리드 다

단

필터



박스 모델

글꼴, 문자

배경

배치플렉스(

정렬) 전환

변환

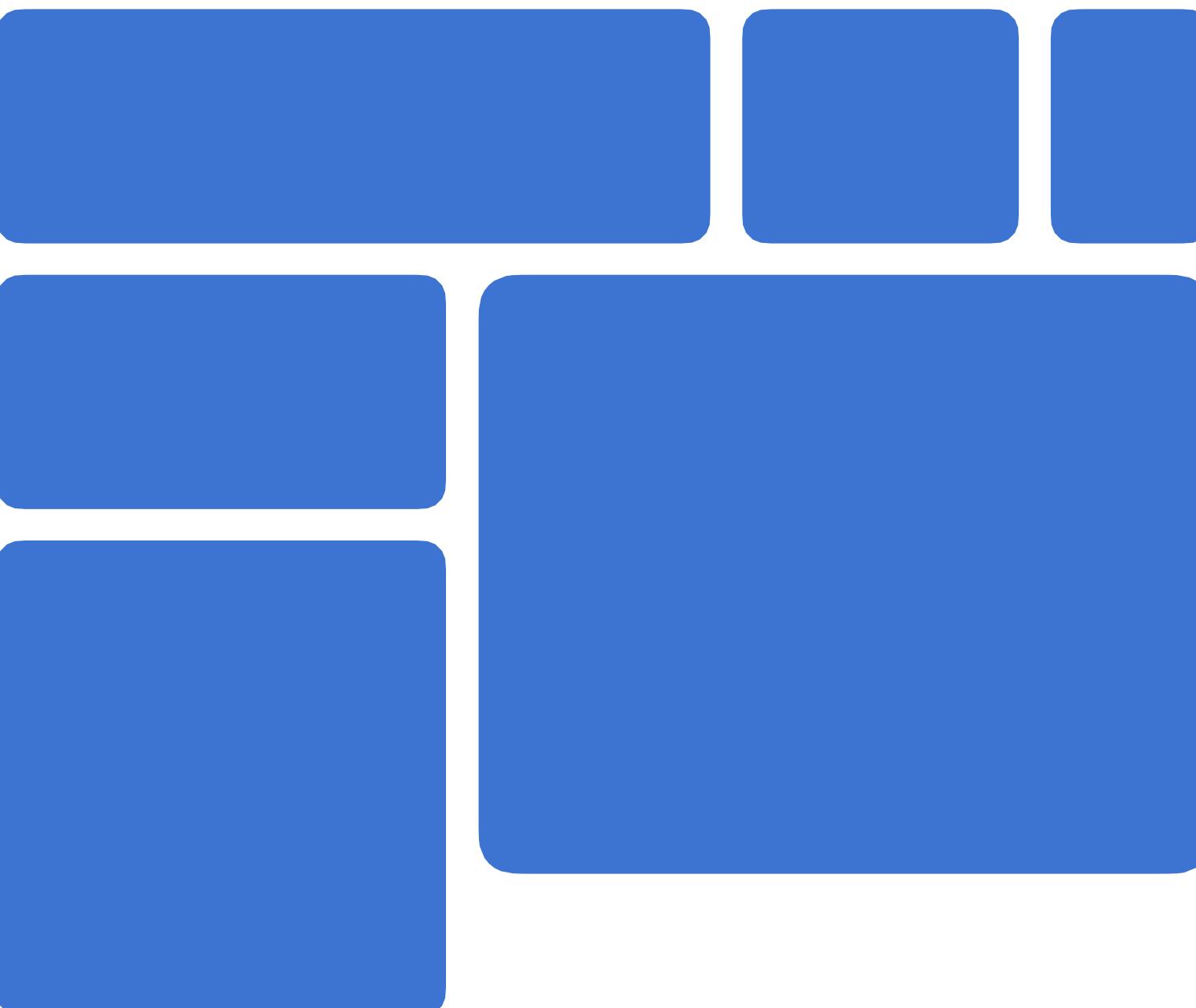
띄움

애니메이션

**그리드** 다

단

필터



박스 모델

글꼴, 문자

배경

배치플렉스(정렬)전환

변환

띄움

애니메이션

그리드 디

단

필터

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been

the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type

박스 모델

글꼴, 문자

배경

배치플렉스(

정렬) 전환

변환

띄움

애니메이션

그리드 다

단

필터



박스 모델

요소의 가로/세로 너비

# width, height

기본값

(요소에 이미 들어있는 속성의 값)

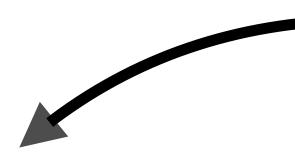
auto

브라우저가 너비를 계산

단위

px, em, vw 등 단위로 지정

```
<span>Hello</span>  
<span>World</span>
```



<span>/span>

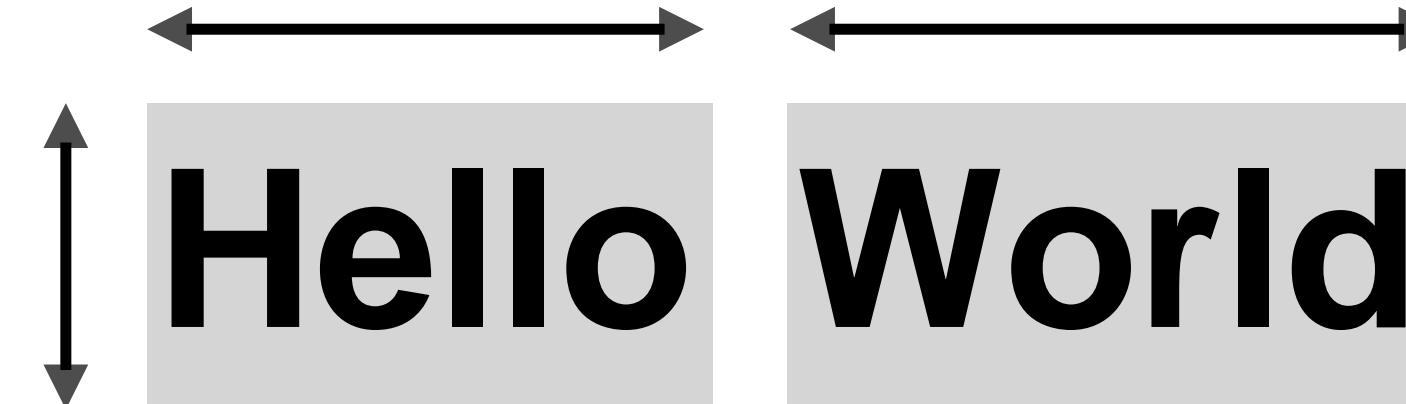
대표적인 인라인 요소!  
본질적으로 아무것도 나타내지 않는,  
콘텐츠 영역을 설정하는 용도.

auto

포함한 콘텐츠 크기만큼 자동으로 줄어듬!

포함한 콘텐츠 크기만큼  
자동으로 줄어듬!

auto



```
<div>Hello</div>  
<div>World</div>
```

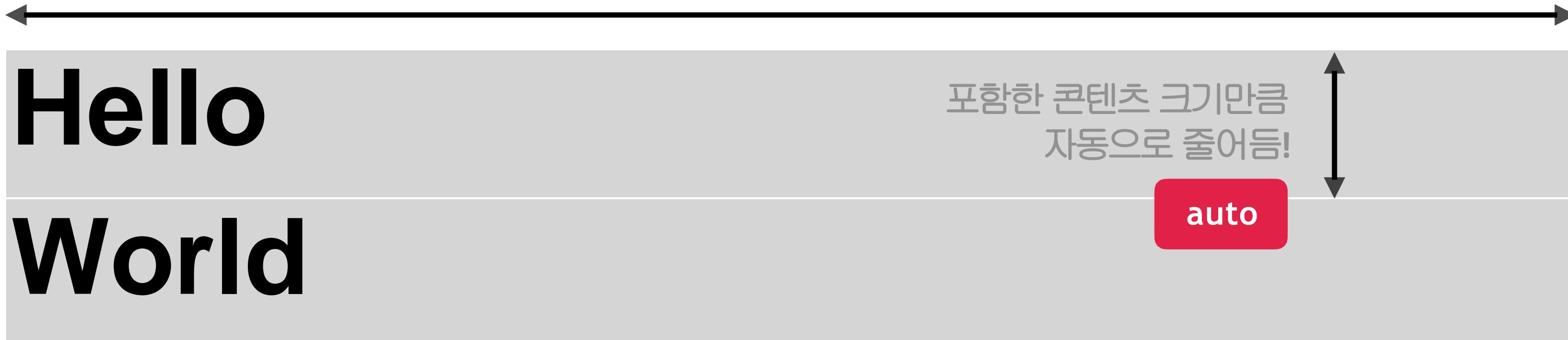


```
<div></div>
```

대표적인 블록 요소!  
본질적으로 아무것도 나타내지 않는,  
콘텐츠 영역을 설정하는 용도.

auto

부모 요소의 크기만큼 자동으로 늘어남!



요소가 커질 수 있는 최대 가로/세로 너비

# max-width, max-height

none

최대 너비 제한 없음

auto

브라우저가 너비를 계산

단위

px, em, vw 등 단위로 지정

요소가 작아질 수 있는 최소 가로/세로 너비

# min-width, min-height

0

최소 너비 제한 없음

auto

브라우저가 너비를 계산

단위

px, em, vw 등 단위로 지정

표현 단위

# 단위

<b>px</b>	픽셀
<b>%</b>	상대적 백분율
<b>em</b>	요소의 글꼴 크기
<b>rem</b>	루트 요소(html)의 글꼴 크기
<b>vw</b>	뷰포트 가로 너비의 백분율 뷰포트 세로 너비의 백분율

## 요소의 외부 여백(공간)을 지정하는 단축 속성

가로(세로) 너비가 있는 요소의  
가운데 정렬에 활용해요!

# margin

음수를 사용할 수 있어요!

0

외부 여백 없음브라우저

auto

가 여백을 계산

단위

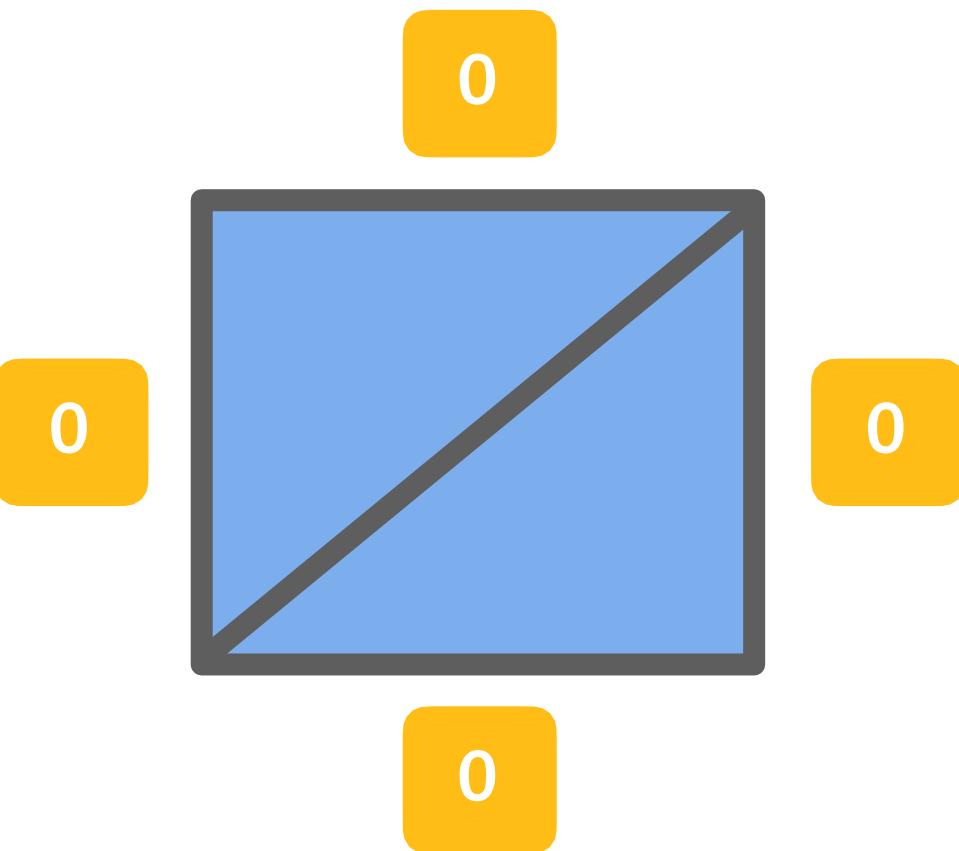
px, em, vw 등 단위로 지정

%

부모 요소의 가로 너비에 대한 비율로 지정

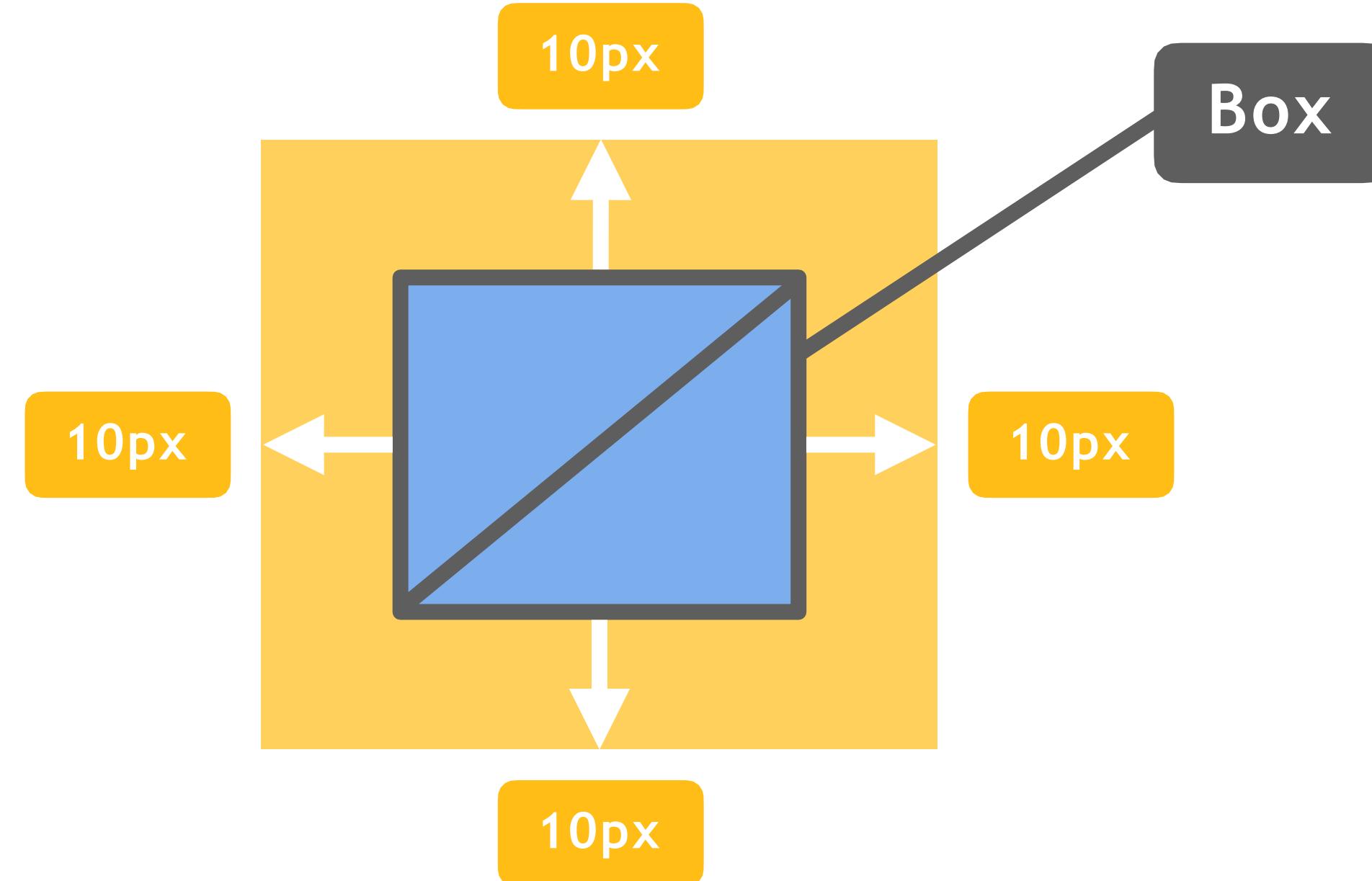
**margin: 0;**

top, right, bottom, left



**margin: 10px;**

top, right, bottom, left

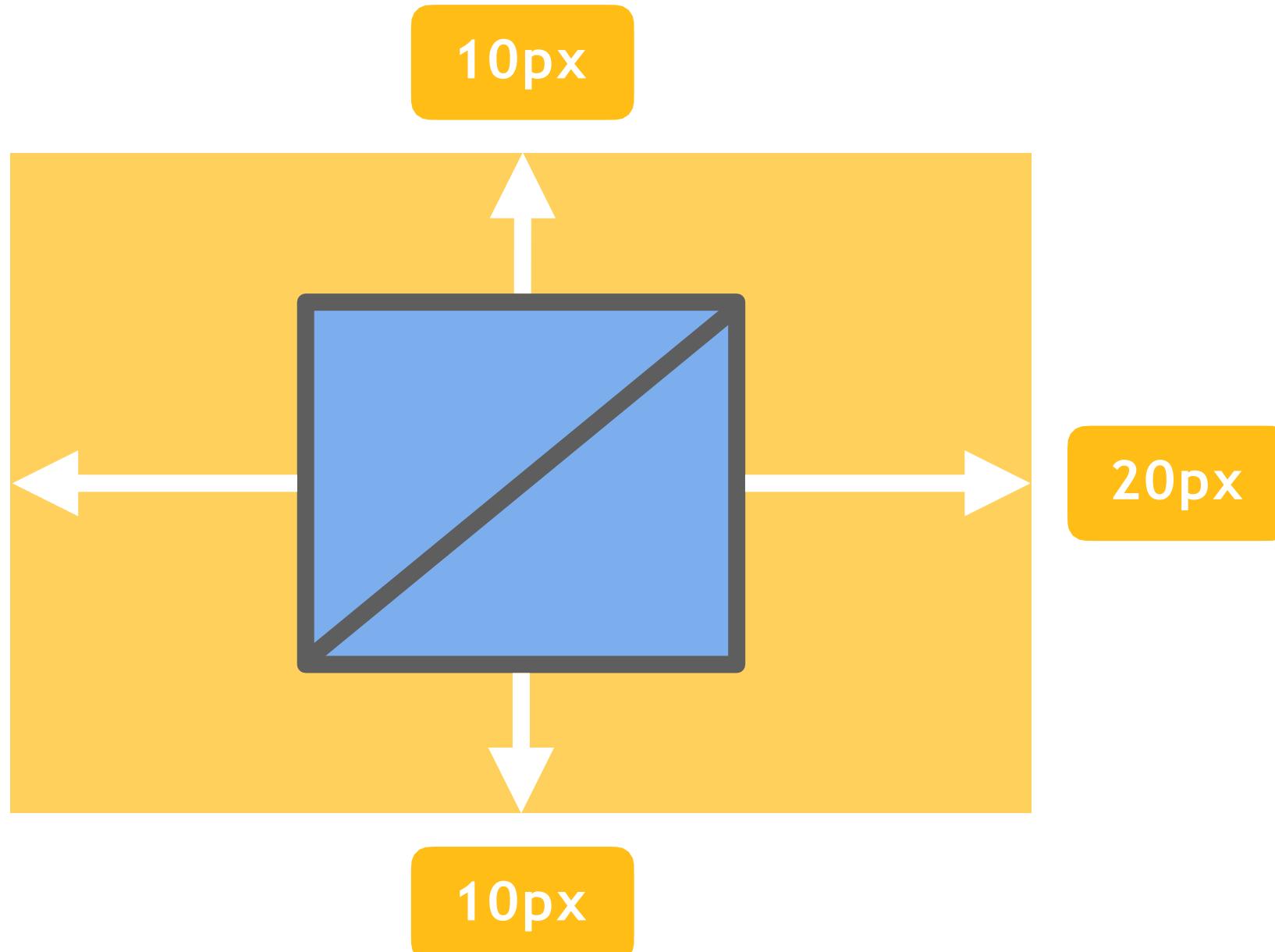


**margin: 10px 20px;**

top, bottom

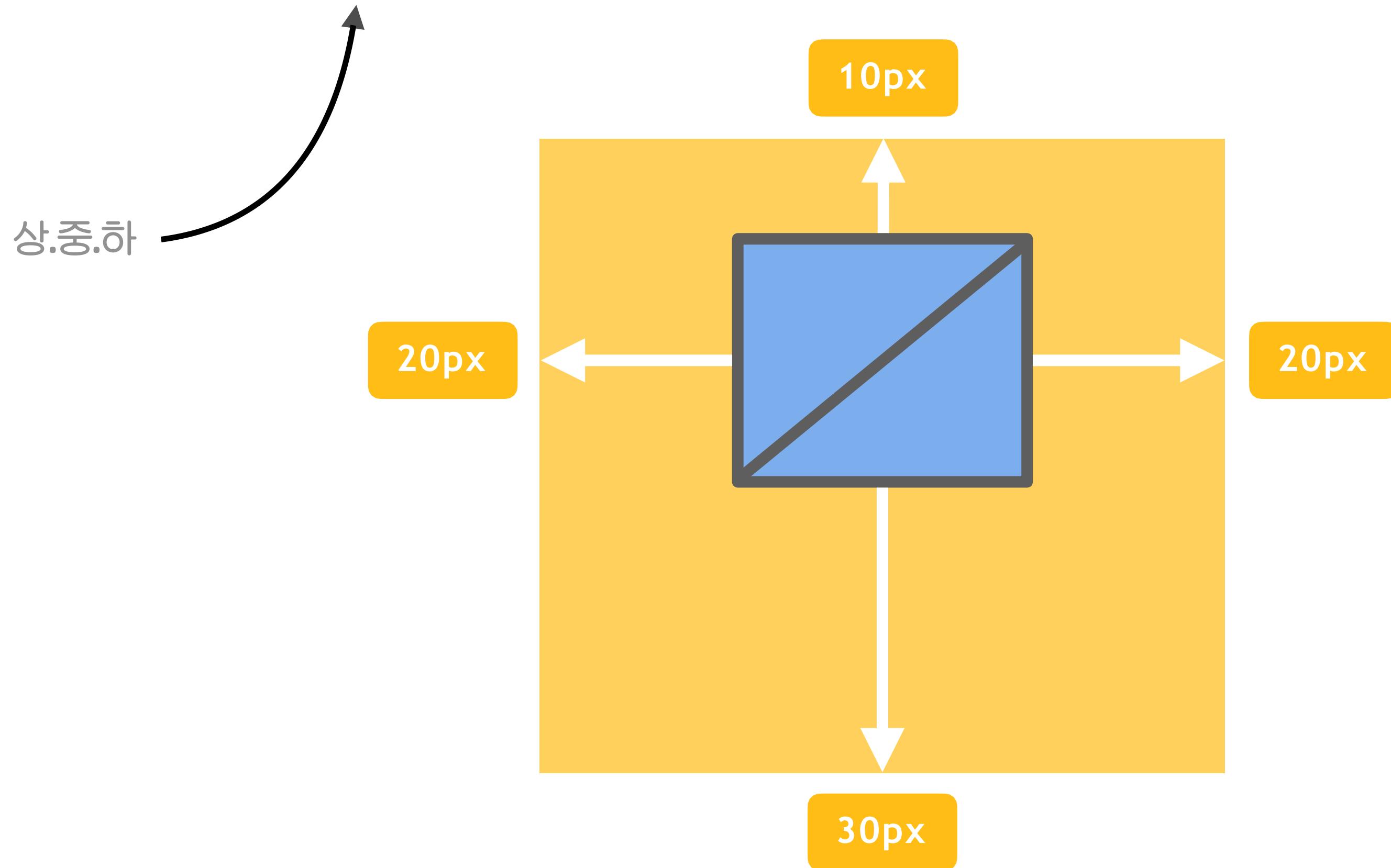
left, right

상하좌우.



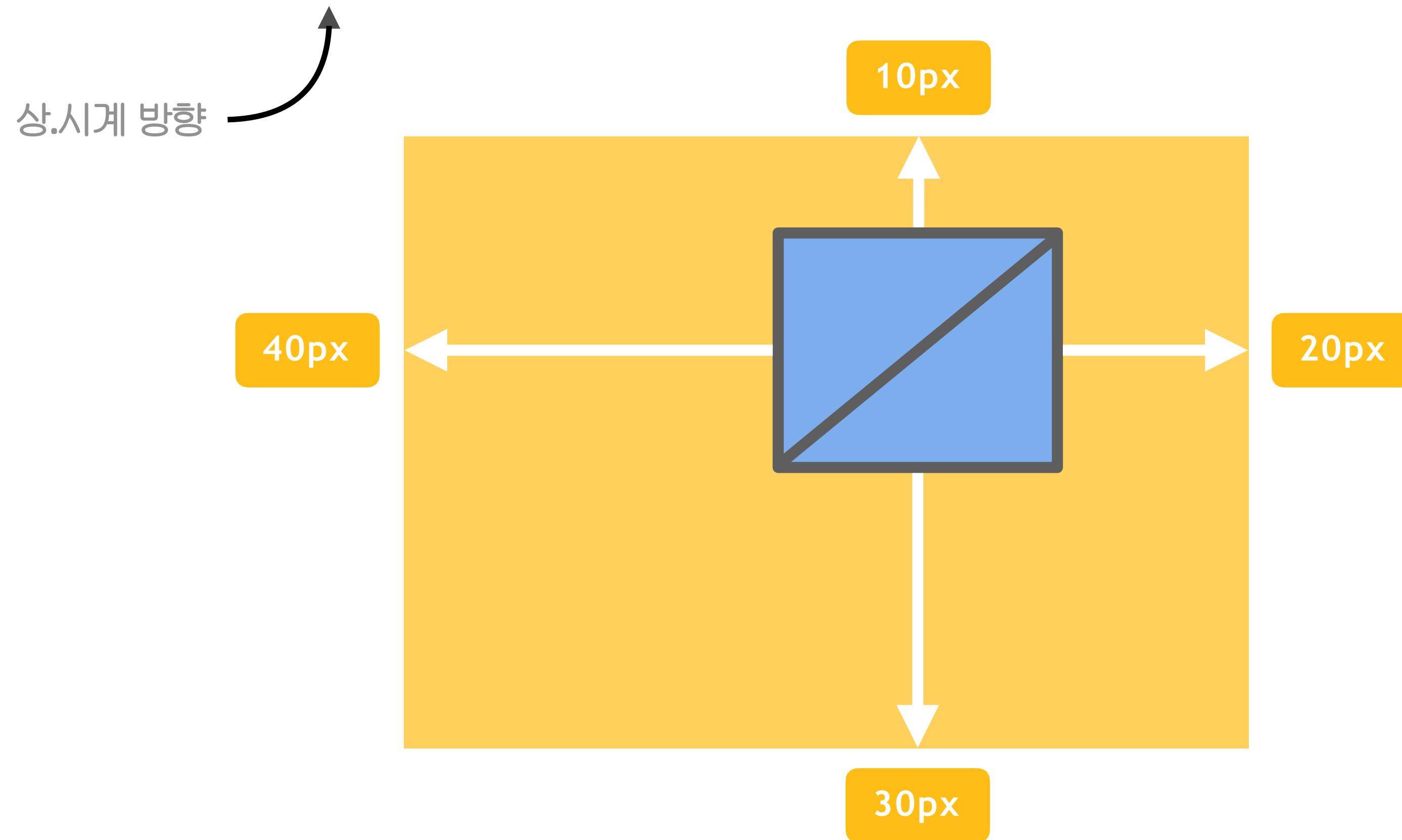
**margin: 10px 20px 30px;**

top    left, right    bottom



**margin: 10px 20px 30px 40px;**

top right bottom left



**margin:** top, right, bottom, left ;

**margin:** top, bottom left, right ;

**margin:** top left, right bottom ;

**margin:** top right bottom left ;

요소의 외부 여백(공간)을 지정하는 기타 개별 속성들

# margin-방향

**margin-top**

**margin-bottom**

**margin-left**

**margin-right**

요소의 내부 여백(공간)을 지정하는 단축 속성

# padding



요소의 크기가 커져요!

0

내부 여백 없음

단위

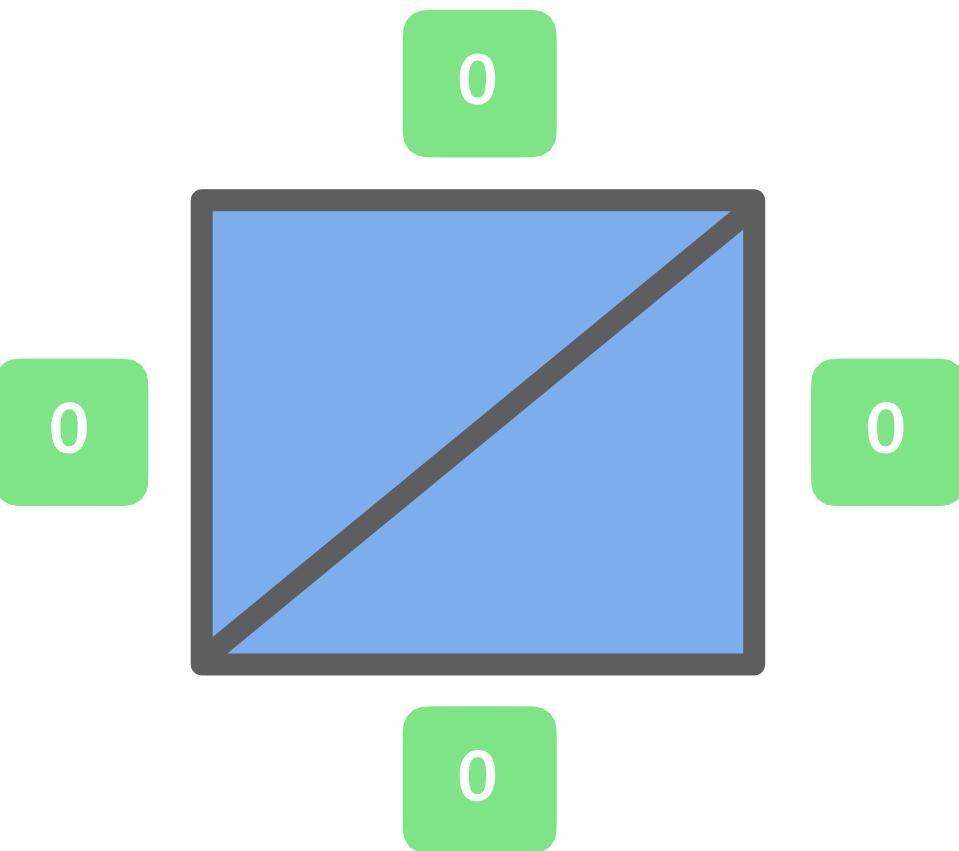
px, em, vw 등 단위로 지정

%

부모 요소의 가로 너비에 대한 비율로 지정

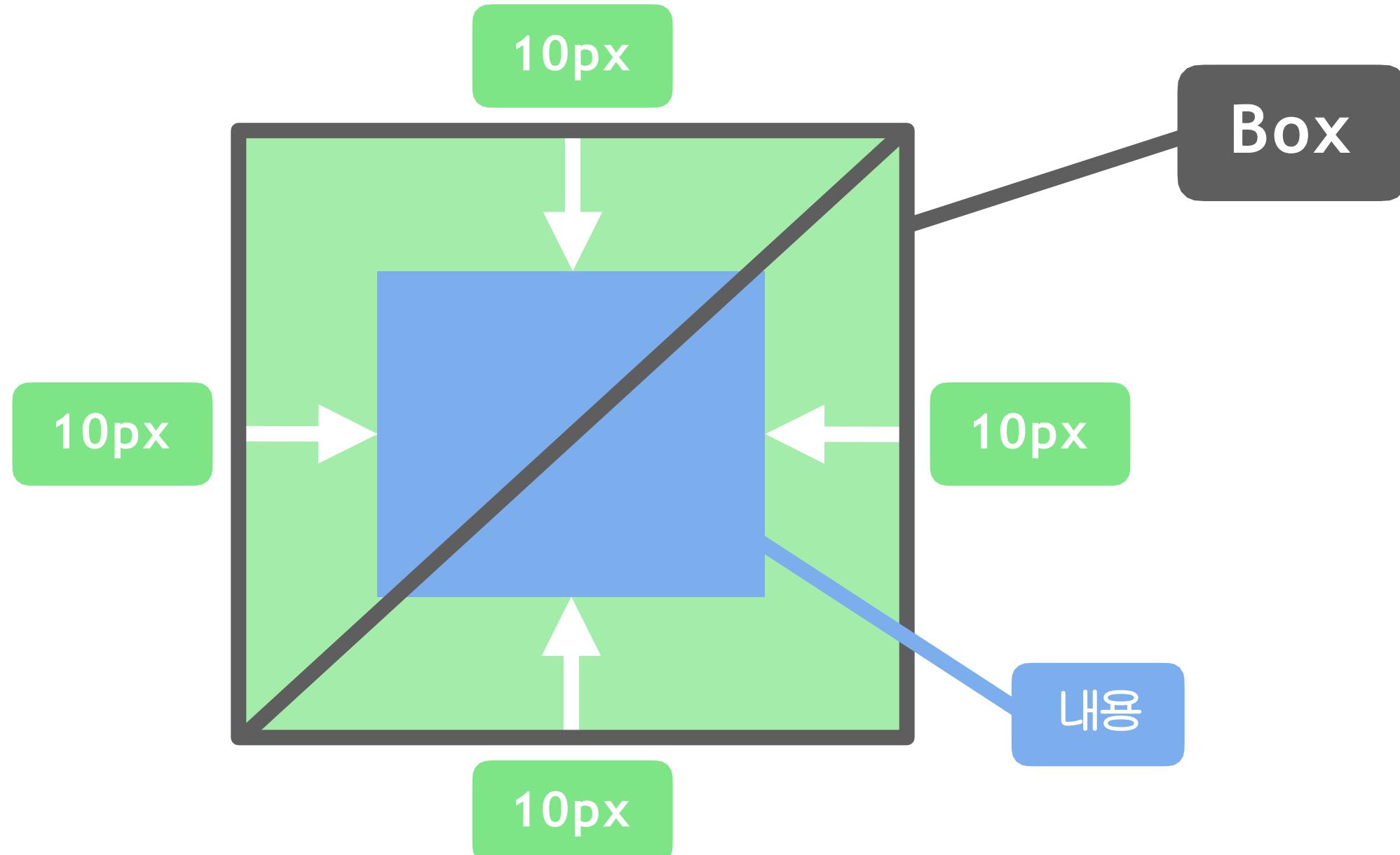
**padding: 0;**

top, right, bottom, left



**padding: 10px;**

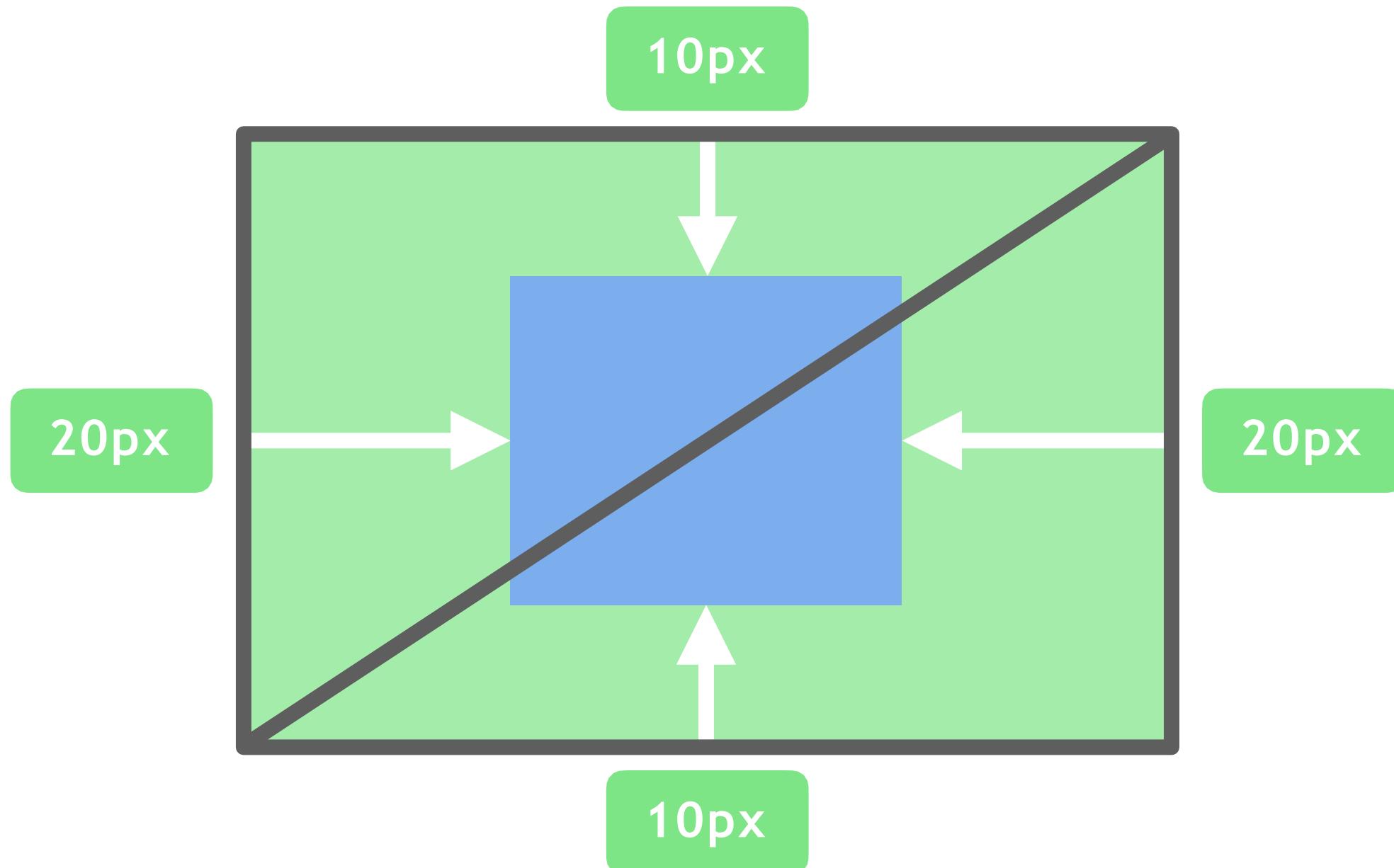
top, right, bottom, left



**padding: 10px 20px;**

top, bottom

left, right

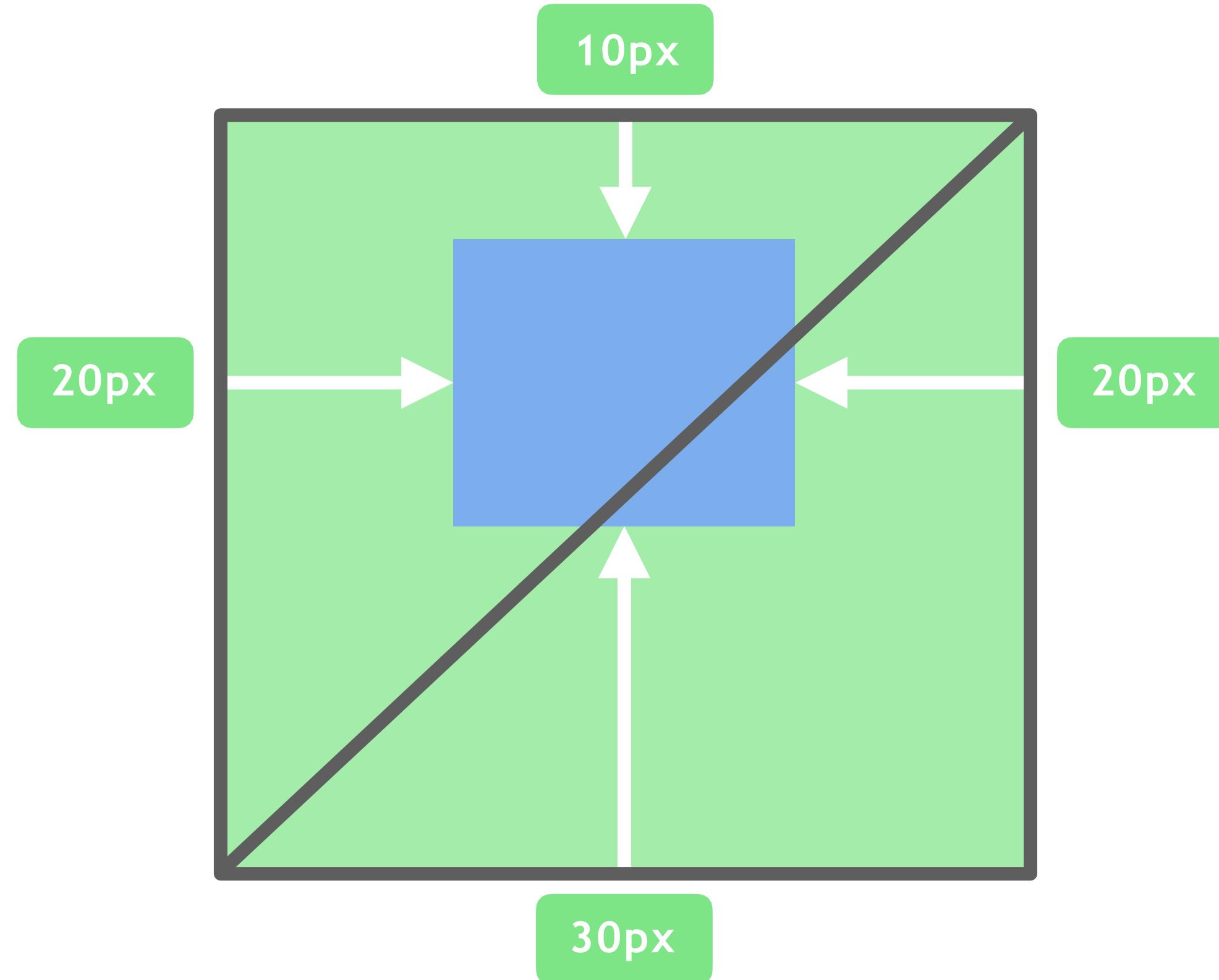


**padding:** 10px 20px 30px;

top

left, right

bottom



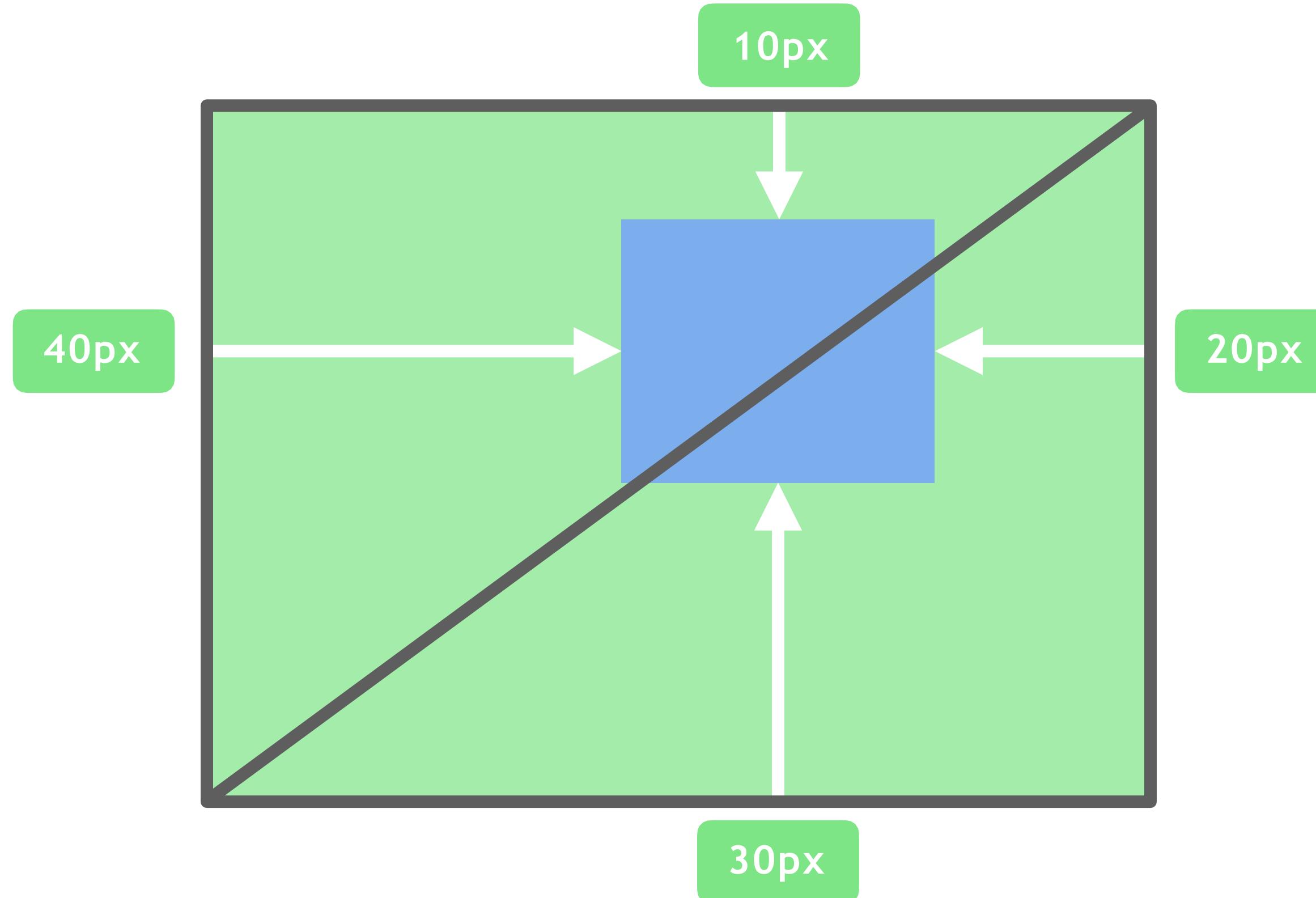
**padding:** 10px 20px 30px 40px;

top

right

bottom

left



**padding:** top, right, bottom, left ;

**padding:** top, bottom left, right ;

**padding:** top left, right bottom ;

**padding:** top right bottom left ;

요소의 내부 여백(공간)을 지정하는 기타 개별 속성들

# padding-방향

**padding-top**

**padding-bottom**

**padding-left**

**padding-right**

보더

요소의 크기가 커져요!

**border: 선-두께**

border-width

요소의 테두리 선을 지정하는 단축 속성

**선-종류**

border-style

border-color

**선-색상;**

**border:** medium none ■ black;

border-width

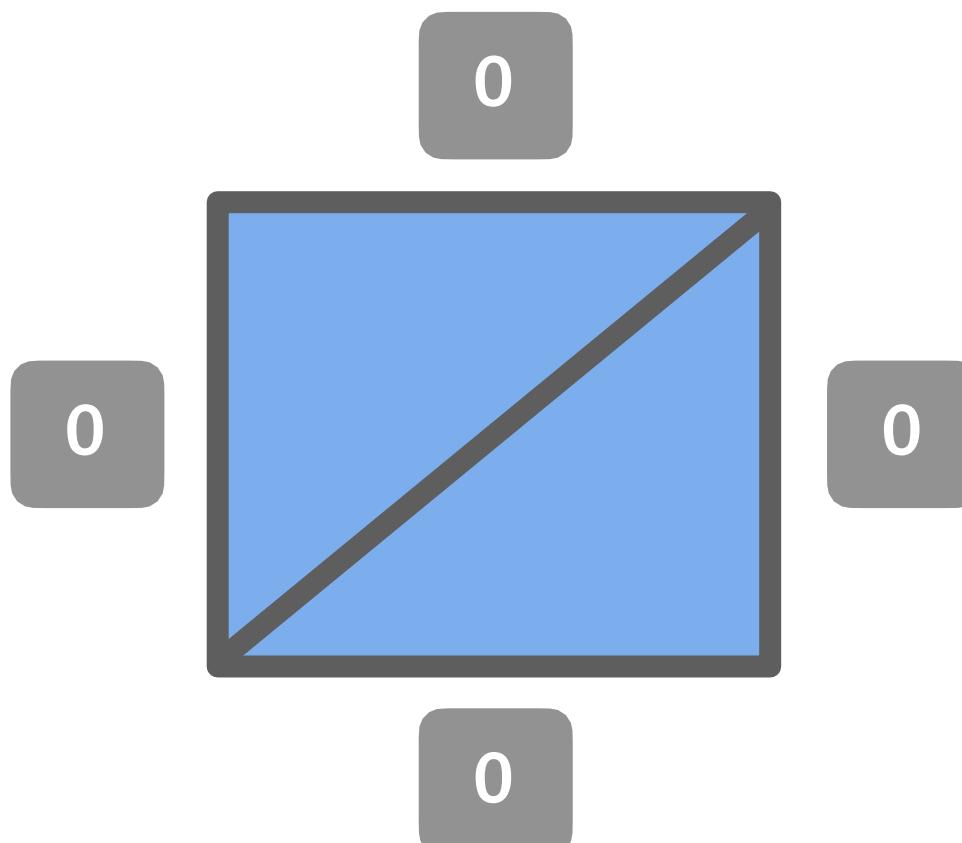
border-style

border-color

기본값

(요소에 이미 들어있는 속성의 값)

선의 종류가 없어서(none)  
출력되지 않아요!

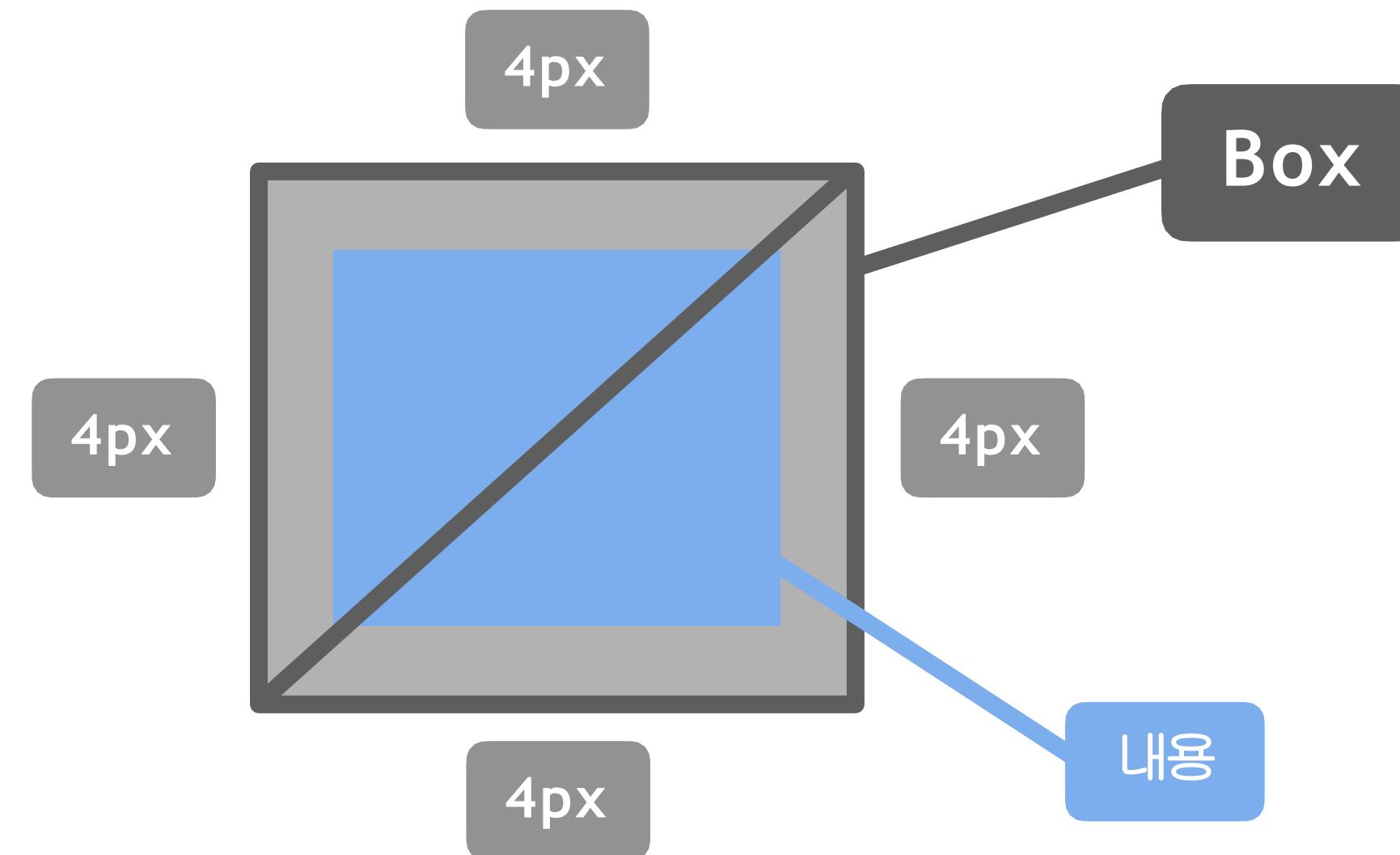


**border: 4px solid black;**

border-width

border-style

border-color

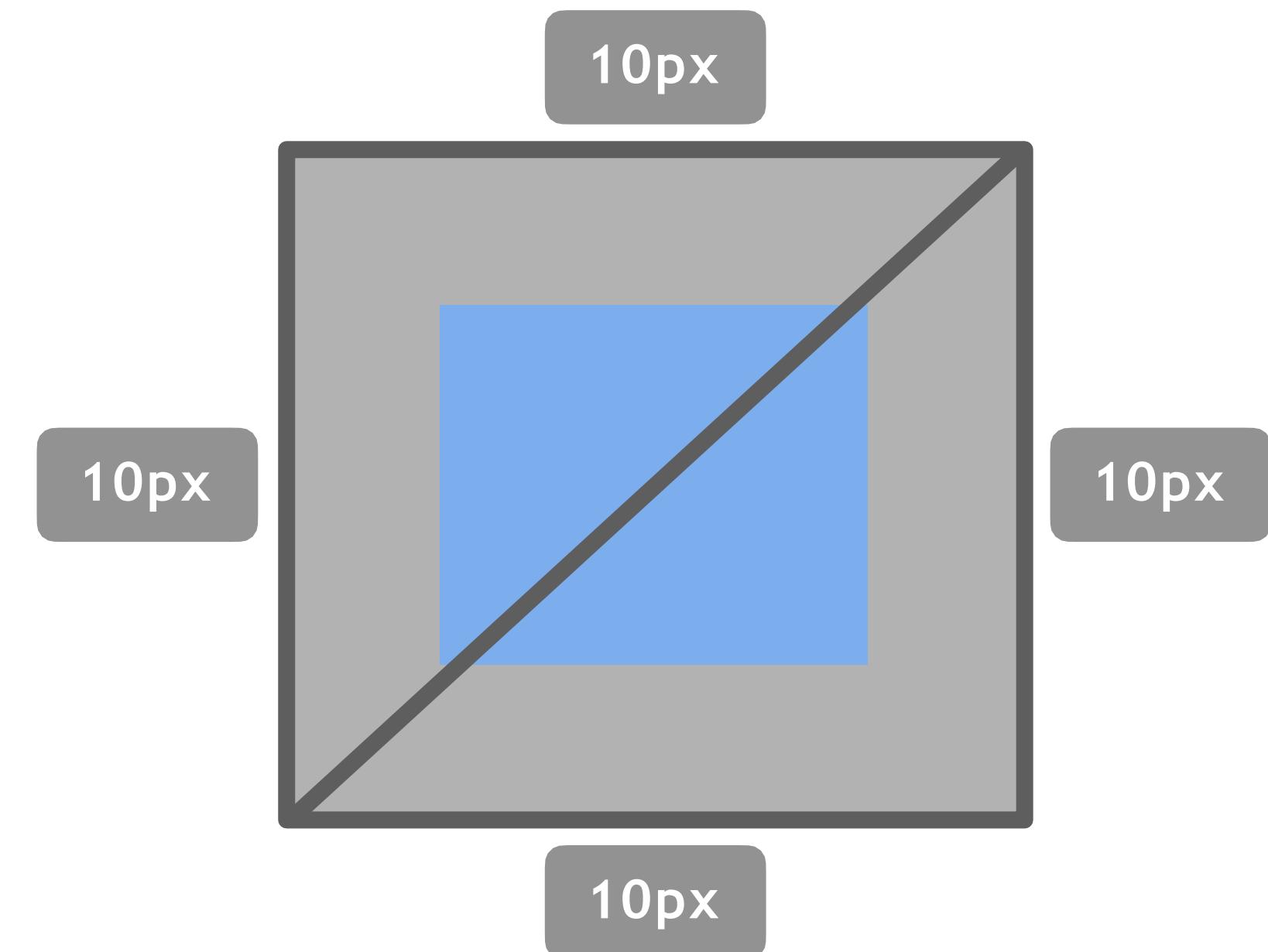


**border: 10px solid black;**

border-width

border-style

border-color



요소 테두리 선의 두께

# border-width

medium

중간 두께

thin

얇은 두께

thick

두꺼운 두께

단위

px, em, % 등 단위로 지정

**border-width:**

top, right, bottom, left ;

**border-width:**

top, bottom      left, right ;

**border-width:**

top      left, right      bottom ;

**border-width:**

top      right      bottom      left ;

## 요소 테두리 선의 종류

# border-style

**none** 선 없음

**solid** 실선 (일반 선)

**dotted** 점선

**dashed** 파선

**double** 두 줄 선

**groove** 홈이 파여있는 모양

**ridge** 솟은 모양 (groove의 반대)

**inset** 요소 전체가 들어간 모양

**outset** 요소 전체가 나온 모양

**border-style:** top, right, bottom, left ;

**border-style:** top, bottom left, right ;

**border-style:** top left, right bottom ;

**border-style:** top right bottom left ;

요소 테두리 선의 색상을 지정하는 단축 속성

# border-color

black 검정색

색상 선의 색상

transparent 투명

**border-color:** top, right, bottom, left ;

**border-color:** top, bottom left, right ;

**border-color:** top left, right bottom ;

**border-color:** top right bottom left ;

색을 사용하는 모든 속성에 적용 가능한 색상 표현

# 색상 표현

색상 이름Hex	브라우저에서 제공하는 색상 이름16진수 색상(Hexadecimal Colors) 빛의 삼원색	red, tomato, royalblue #000, #FFFFFF rgb(255, 255, 255)
색상코드RGB		#000, #FFFFFF
RGBA		rgba(0, 0, 0, 0.5)
HSL	색상, 채도, 명도	hsl(120, 100%, 50%)
HSLA	색상, 채도, 명도 + 투명도	hsla(120, 100%, 50%, 0.3)

요소의 테두리 선을 지정하는 기타 속성들

**border-방향**

**border-방향-속성**

**border-top:** 두께 종류 색상;

**border-top-width:** 두께;

**border-top-style:** 종류;

**border-top-color:** 색상;

**border-bottom:** 두께 종류 색상;

**border-bottom-width:** 두께;

**border-bottom-style:** 종류;

**border-bottom-color:** 색상;

**border-left:** 두께 종류 색상;

**border-left-width:** 두께;

**border-left-style:** 종류;

**border-left-color:** 색상;

**border-right:** 두께 종류 색상;

**border-right-width:** 두께;

**border-right-style:** 종류;

**border-right-color:** 색상;

요소의 모서리를 둥글게 깎음

# border-radius

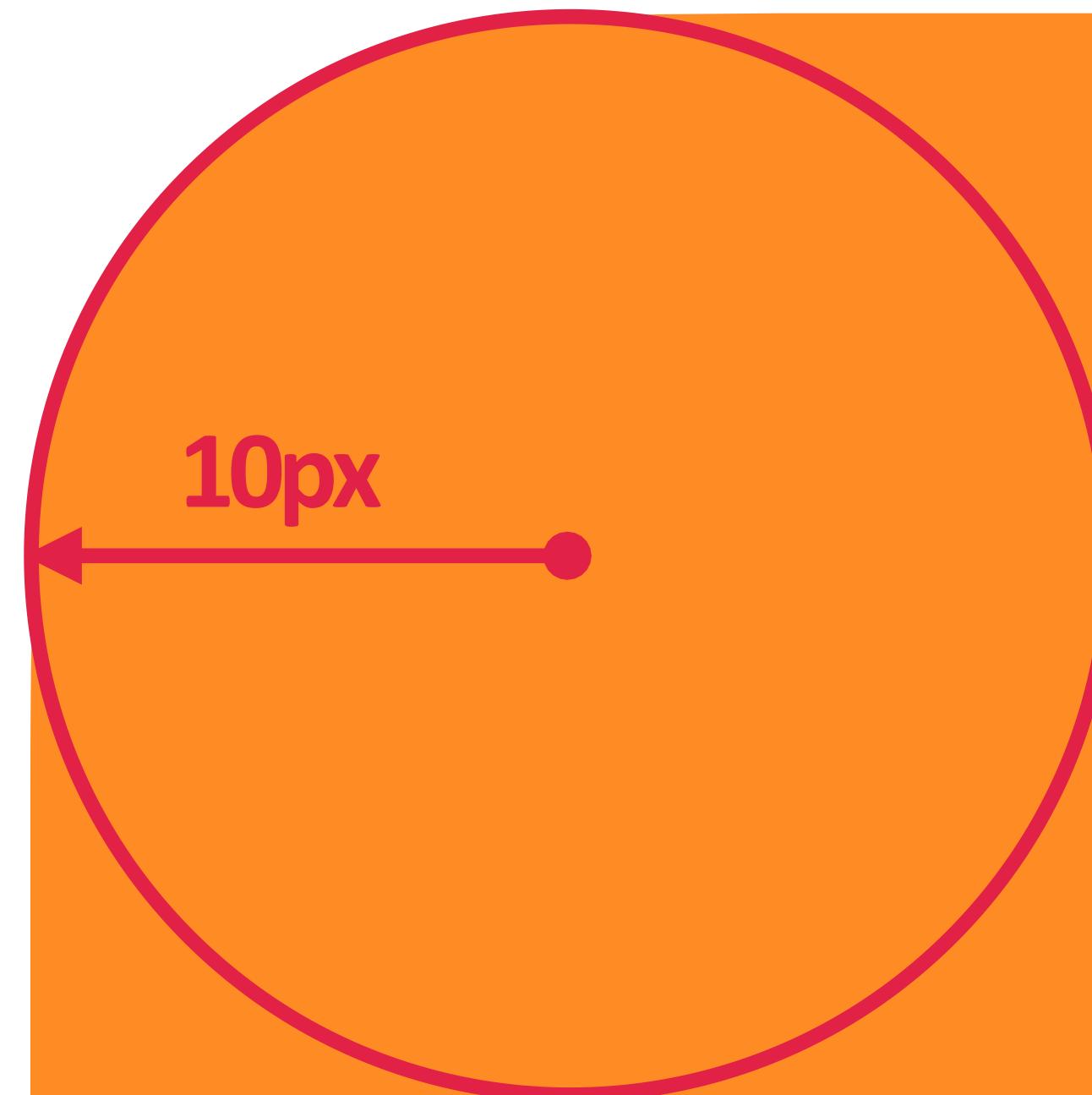
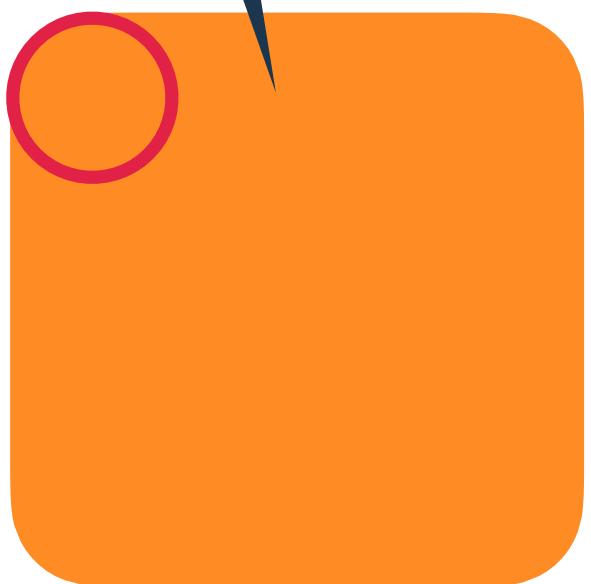
0

둥글게 없음

단위

px, em, vw 등 단위로 지정

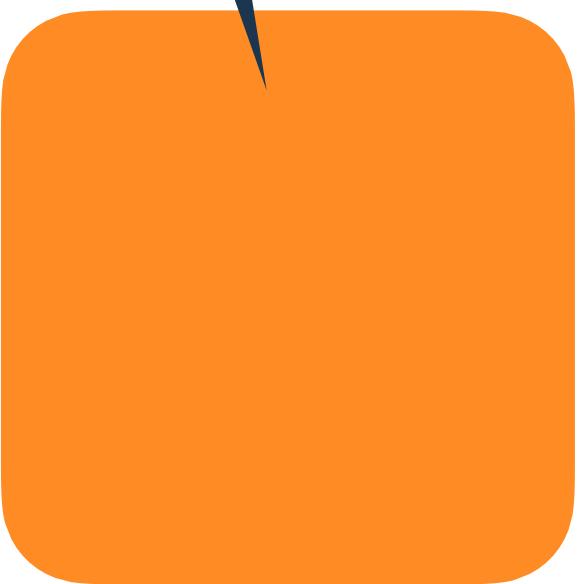
`border-radius: 10px;`



`border-radius: 0;`

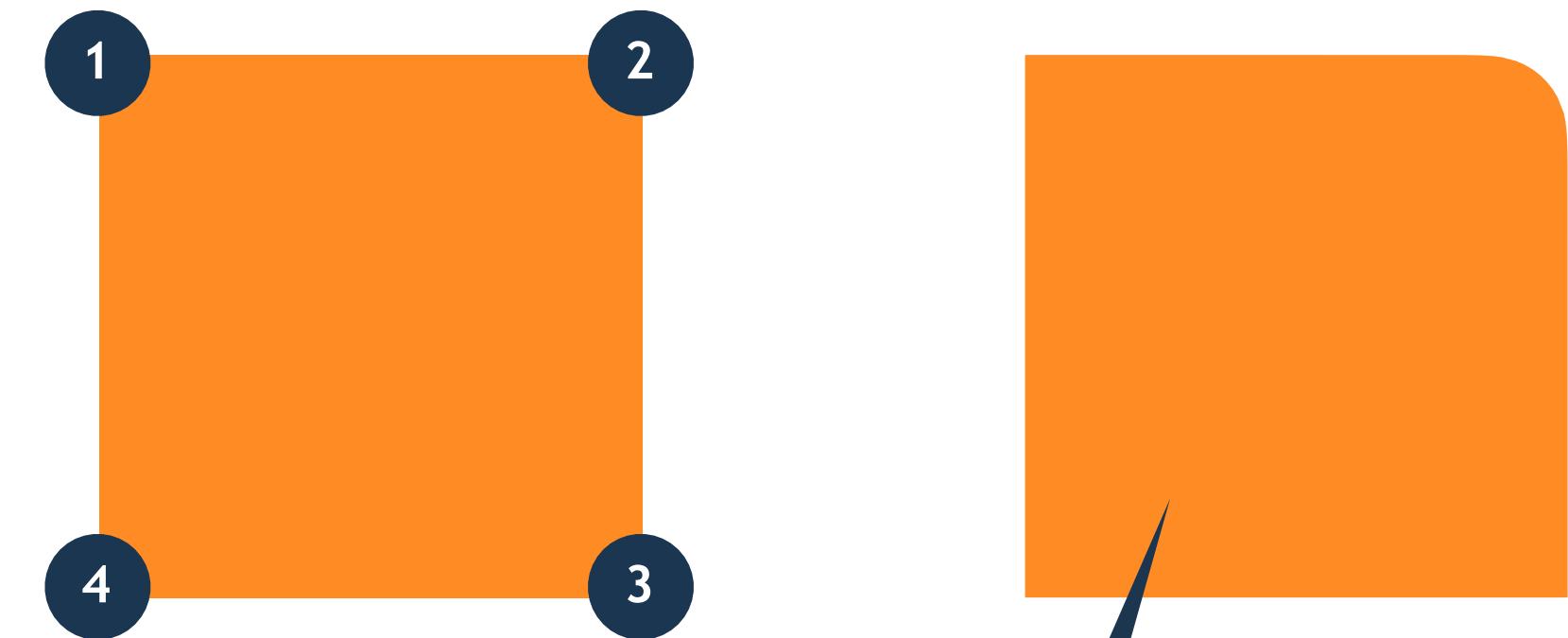


`border-radius: 10px;`



`border-radius: 0 10px 0 0;`





`border-radius: 0 10px 0 0;`

요소의 크기 계산 기준을 지정

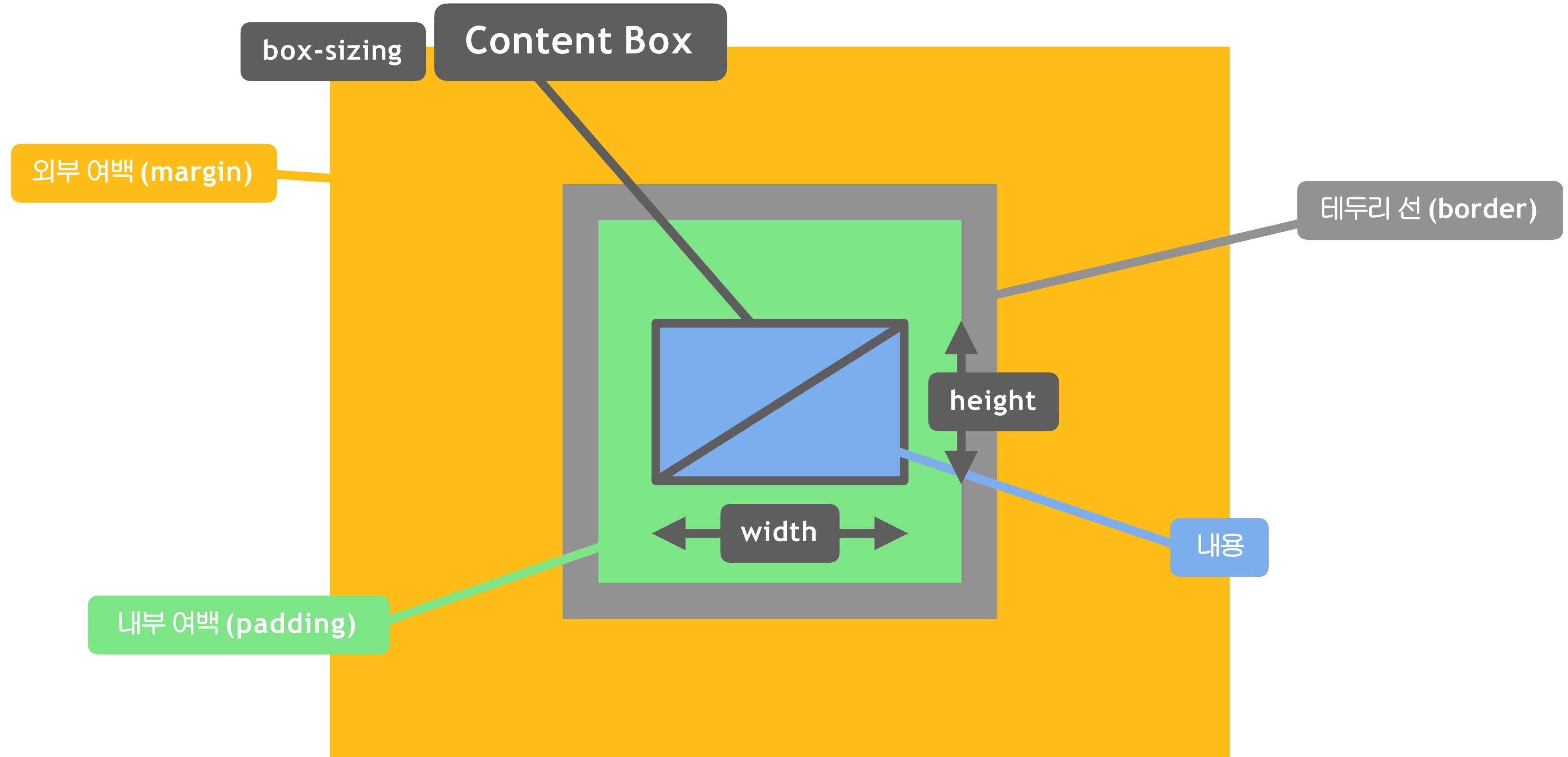
# box-sizing

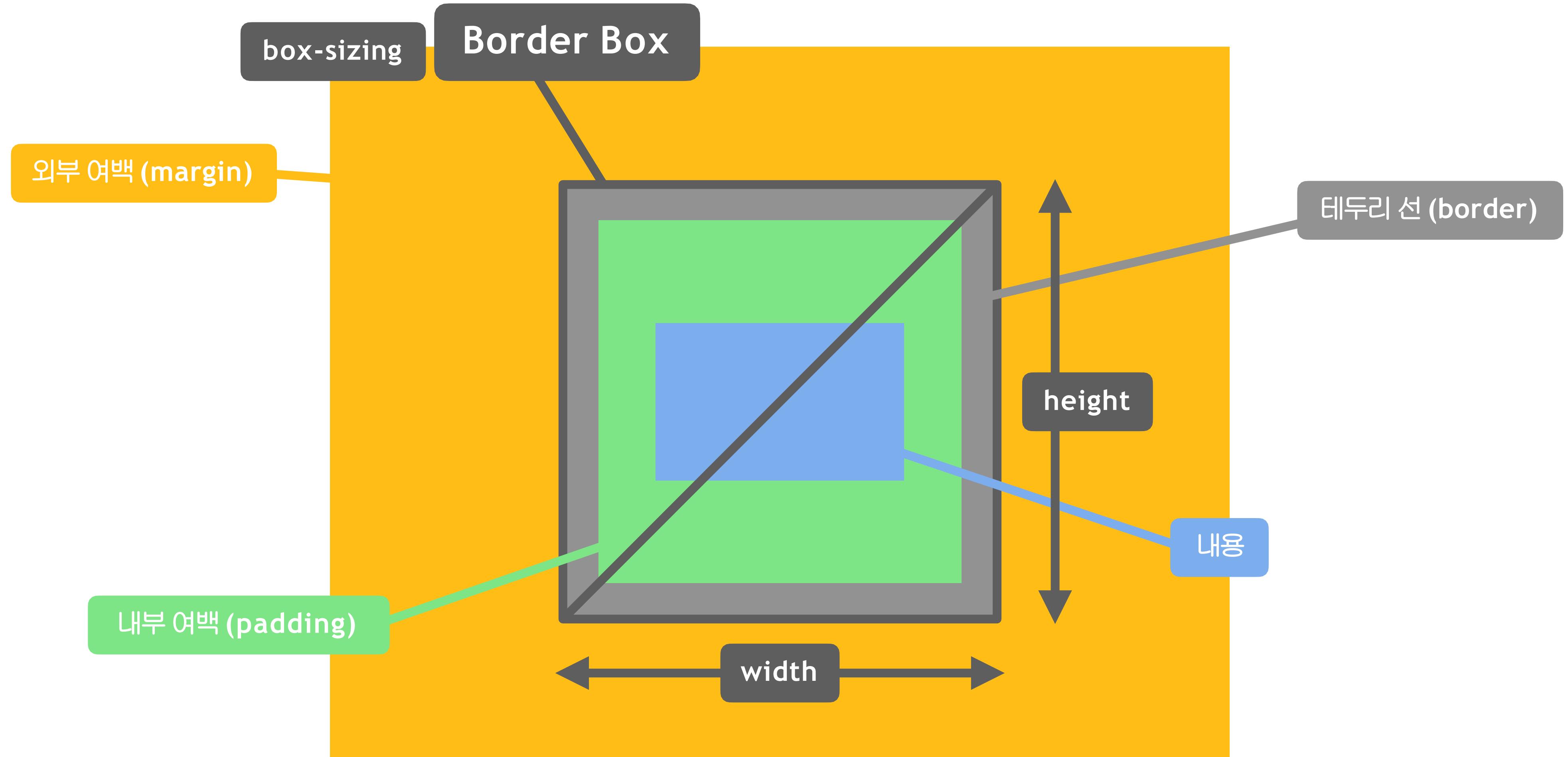
content-box

요소의 내용(content)으로 크기 계산

border-box

요소의 내용 + padding + border로 크기 계산





요소의 크기 이상으로 내용이 넘쳤을 때, 보여짐을 제어하는 단축 속성

# overflow

**visible**

넘친 내용을 그대로 보여줌

**hidden**

넘친 내용을 잘라냄

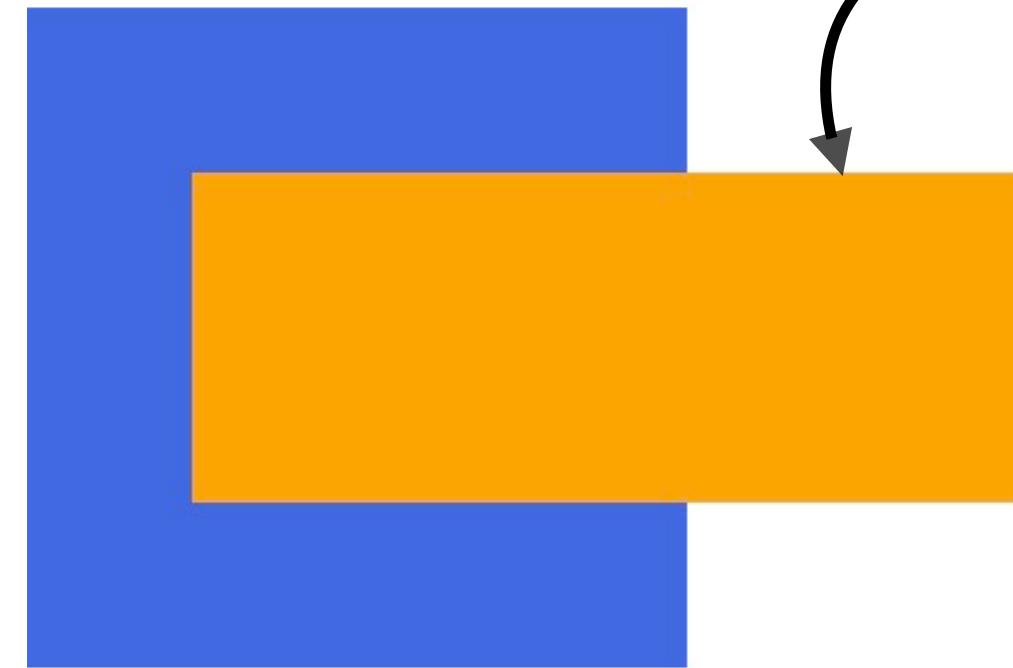
**scroll**

넘친 내용을 잘라냄, 스크롤바 생성

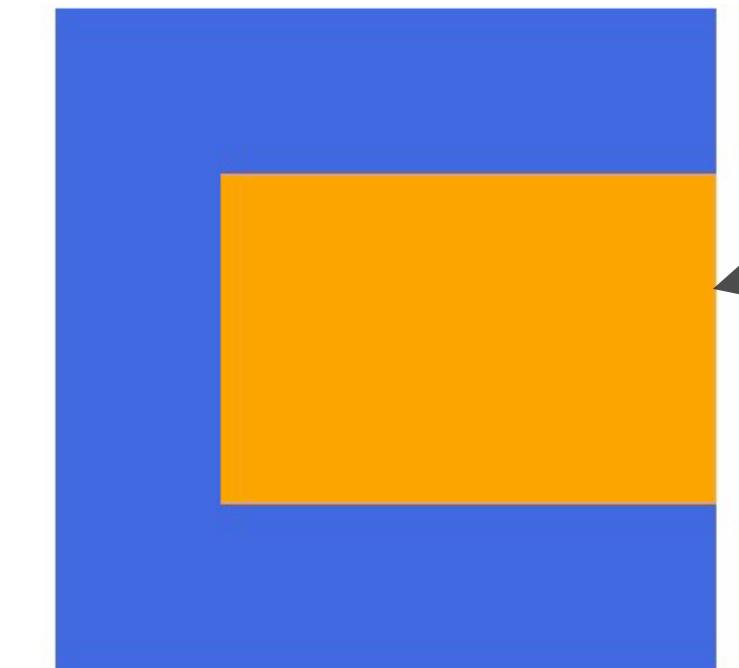
**auto**

넘친 내용이 있는 경우에만 잘라내고 스크롤바 생성

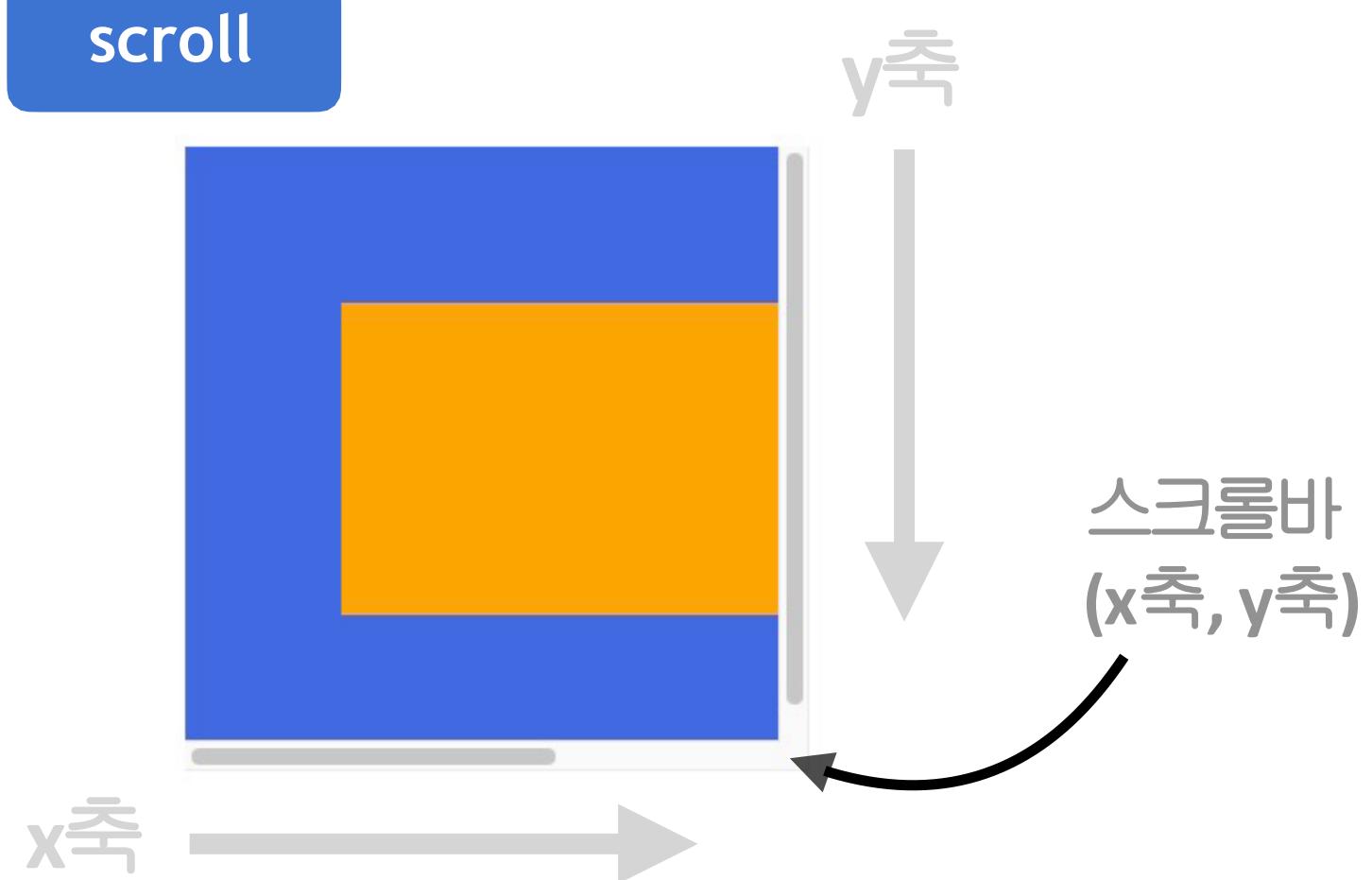
**visible**



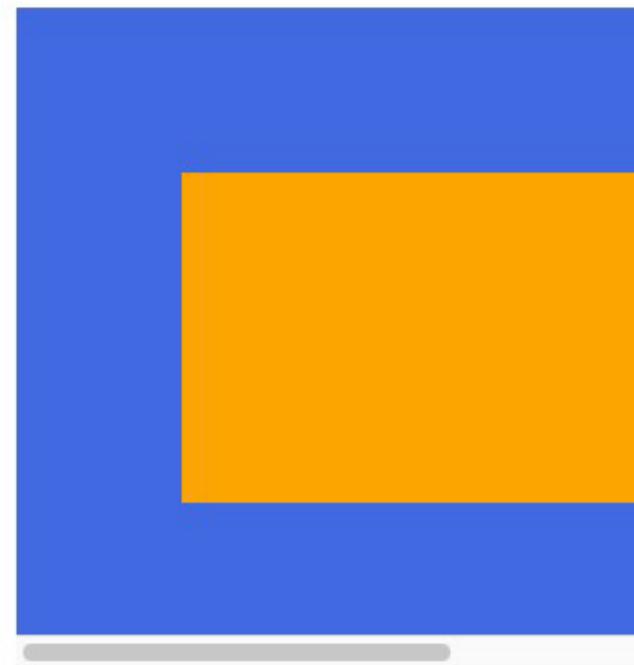
**hidden**



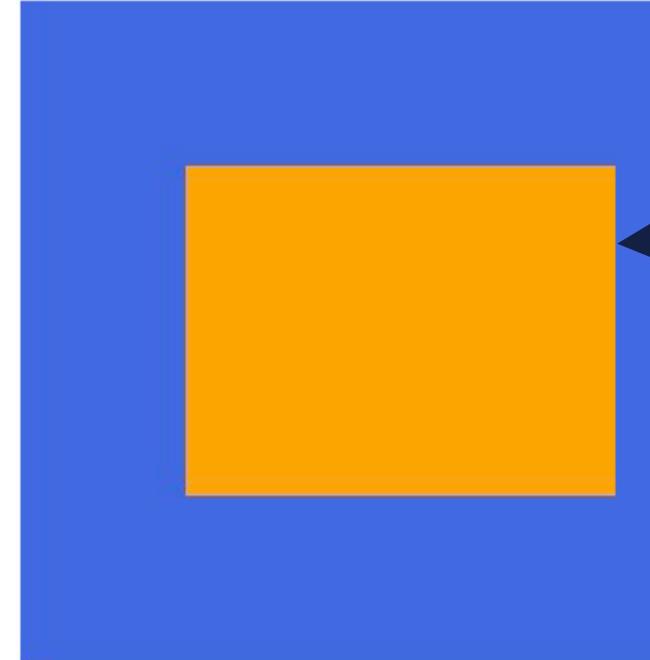
**scroll**



**auto**

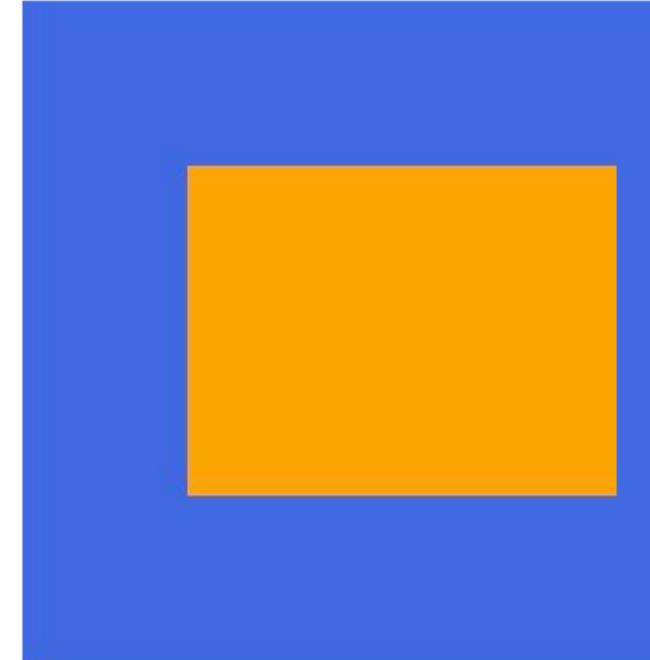


**visible**

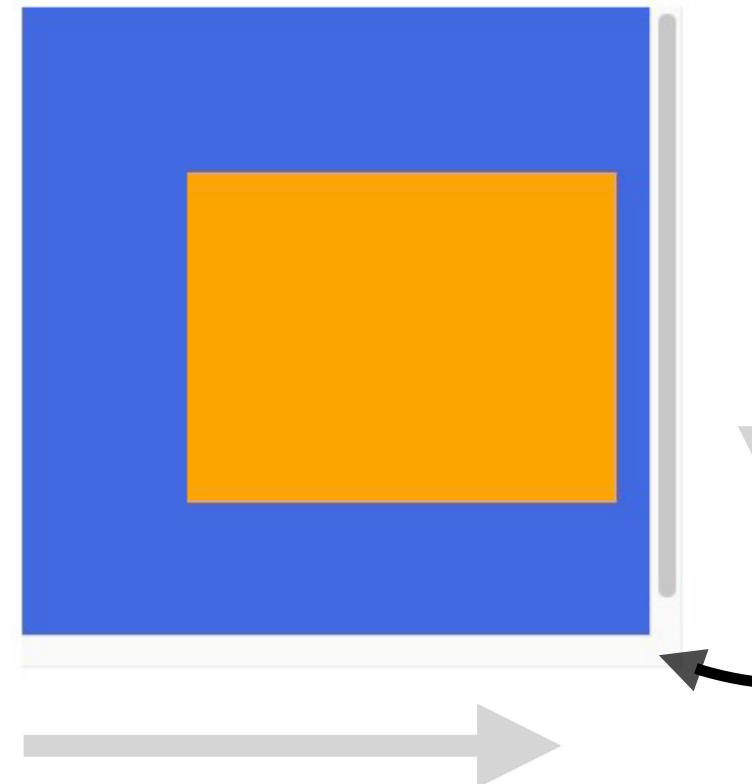


내용 넘치지 않음!

**hidden**

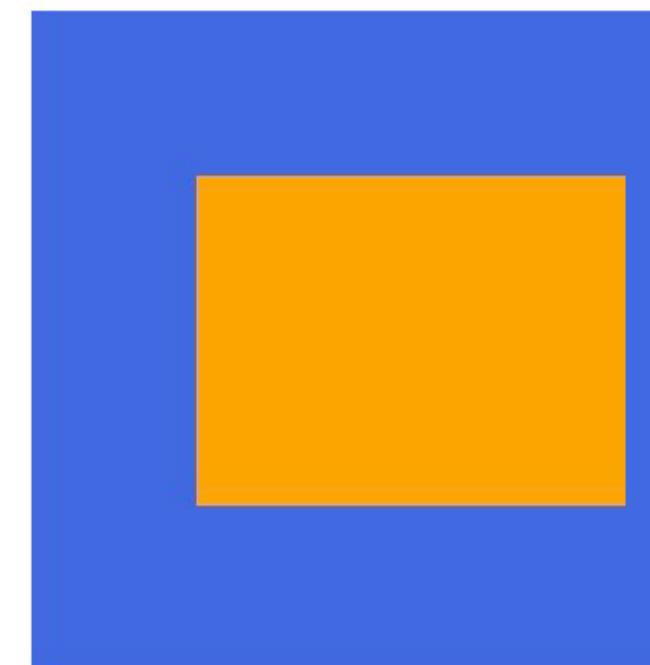


**scroll**



스크롤바  
(x축, y축)

**auto**



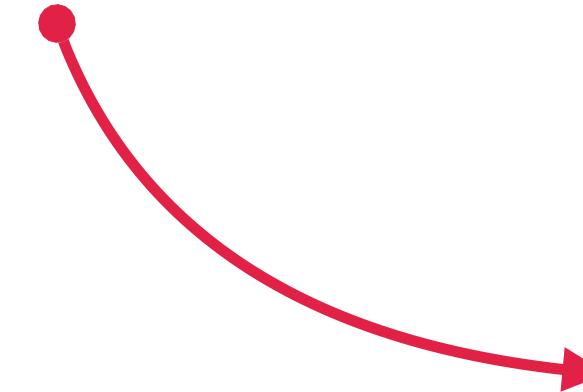
요소의 크기 이상으로 내용이 넘쳤을 때, 보여짐을 제어하는 개별 속성들

**overflow-x**

**overflow-y**

## 요소의 화면 출력(보여짐) 특성

각 요소에 이미 지정되어 있는 값



**block**

상자(레이아웃) 요소

**inline**

글자 요소

**inline-block**

글자 + 상자 요소

**flex**

플렉스 박스 (1차원 레이아웃)

**grid**

그리드 (2차원 레이아웃)

**none**

보여짐 특성 없음, 화면에서 사라짐

**기타**

table, table-row, table-cell 등..

따로 지정해서 사용하는 값



# display

## 요소 투명도

# opacity

1

불투명

0~1

0부터 1사이의 소수점 숫자

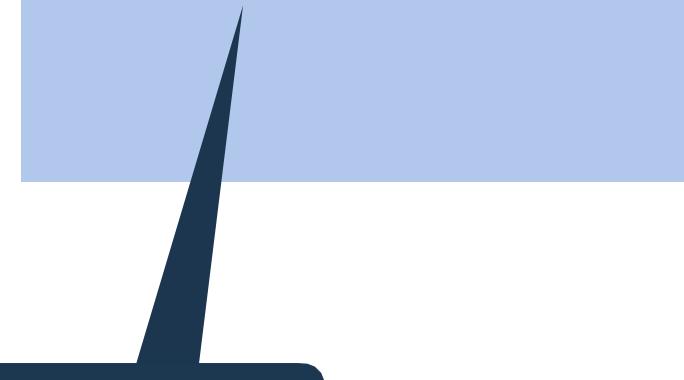
**opacity: 0.07;**



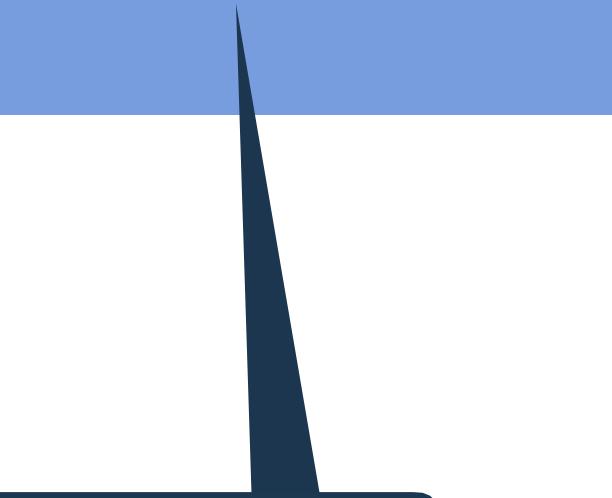
**opacity: 1;**



**opacity: 0.4;**



**opacity: 0.7;**



글꼴

## 글자의 기울기

# font-style

normal

기울기 없음

italic

이텔릭체

oblique

기울어진 글자

## 글자의 두께(가중치)

# font-weight

normal, 400

기본 두께

bold, 700

두껍게

bolder

상위(부모) 요소보다 더 두껍게

lighter

상위(부모) 요소보다 더 얇기

100 ~ 900

100단위의 숫자 9개,  
normal과 bold 이외 두께

## 글자의 크기

# font-size

16px

기본 크기

단위

px, em, rem 등 단위로 지정

%

부모 요소의 폰트 크기에 대한 비율

smaller

상위(부모) 요소보다 작은 크기

larger

상위(부모) 요소보다 큰 크기

xx-small ~ xx-large

가장 작은 크기 ~ 가장 큰 크기까지,  
단계의 크기를 지정

한 줄의 높이, 행간과 유사

# line-height

normal 브라우저의 기본 정의를 사용

숫자 요소의 글꼴 크기의 배수로 지정

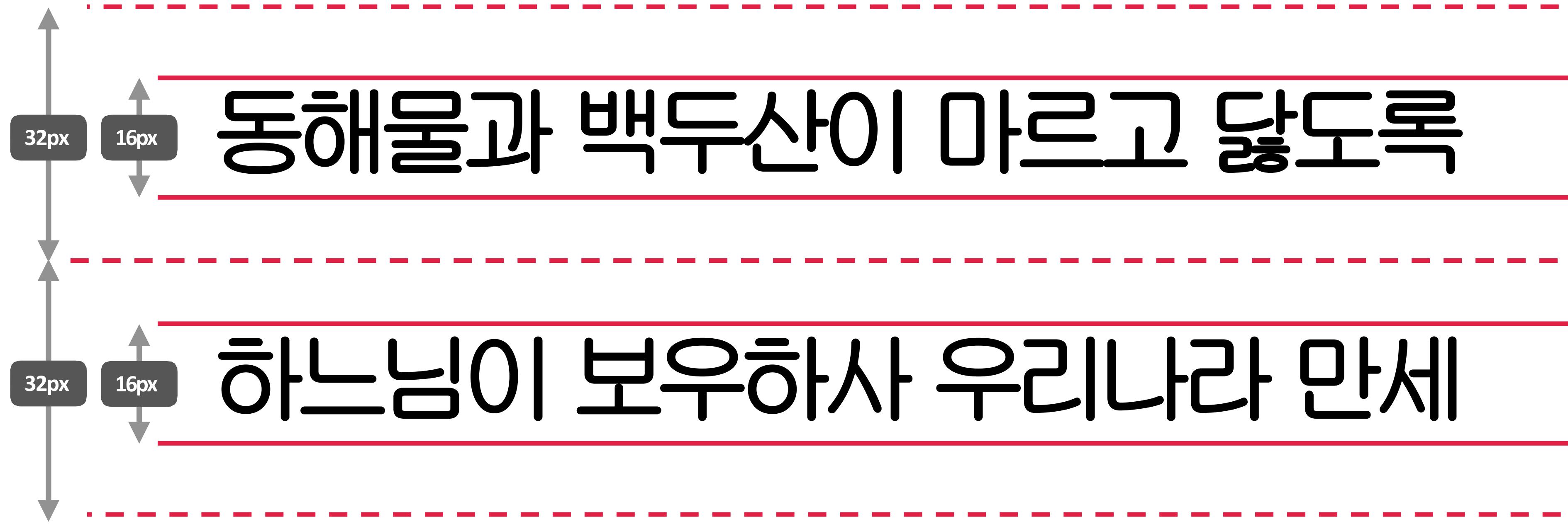
단위 px, em, rem 등의 단위로 지정

% 요소의 글꼴 크기의 비율로 지정

동해물과 백두산이 마르고 닳도록

하느님이 보우하사 우리나라 만세

```
font-size: 16px;  
line-height: 32px;  
/* line-height: 2; */  
/* line-height: 200%; */
```



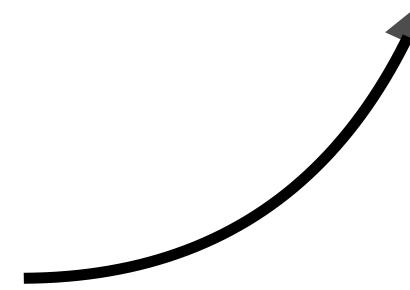
```
font-size: 16px;           2배 차이
line-height: 32px;
/* line-height: 2; */
/* line-height: 200%; */
```

**font-family: 글꼴1, "글꼴2", ... 글꼴계열;**

글꼴(서체) 지정

필수로 작성!

띄어쓰기 등 특수문자가 포함된  
글꼴 이름은 큰 따옴표로 묶어야 합니다~



글꼴계열;

Hello World!

serif

바탕체 계열

Hello World!

sans-serif

고딕체 계열

Hello Wor ld!

monospace

고정너비(가로폭이 동등) 글꼴 계열

*Hello World!*

cursive

필기체 계열

**EMPIRE**

fantasy

장식 글꼴 계열

문자

글자의 색상

color

rgb(0,0,0)

검정색

색상

기타 지정 가능한 색상

## 문자의 정렬 방식

# text-align

-  **left** 왼쪽 정렬
-  **right** 오른쪽 정렬
-  **center** 가운데 정렬
-  **justify** 양쪽 정렬

## 문자의 장식(선)

# text-decoration

화면에 출력!

동해물과 백두산이 마르고 닳도록

동해물과 백두산이 마르고 닳도록

동해물과 백두산이 마르고 닳도록

~~동해물과 백두산이 마르고 닳도록~~

none      장식 없음

underline      밑줄

overline      윗줄

line-through      중앙 선

동해물과 백두산이 마르고 닳도록 하느  
님이 보우하사 우리나라 만세 무궁화 삼천리 화  
려 강산 대한 사람 대한으로 길이 보전하세

들여쓰기(50px)

음수를 사용할 수 있어요!  
반대는 내어쓰기(outdent)입니다.

문자 첫 줄의 들여쓰기

# text-indent

0

들여쓰기 없음

단위

px, em, rem 등 단위로 지정

%

요소의 가로 너비에 대한 비율

배경

요소의 배경 색상

# background-color

transparent

투명함

색상

지정 가능한 색상

`background-color: orange;`



요소의 배경 이미지 삽입

# background-image

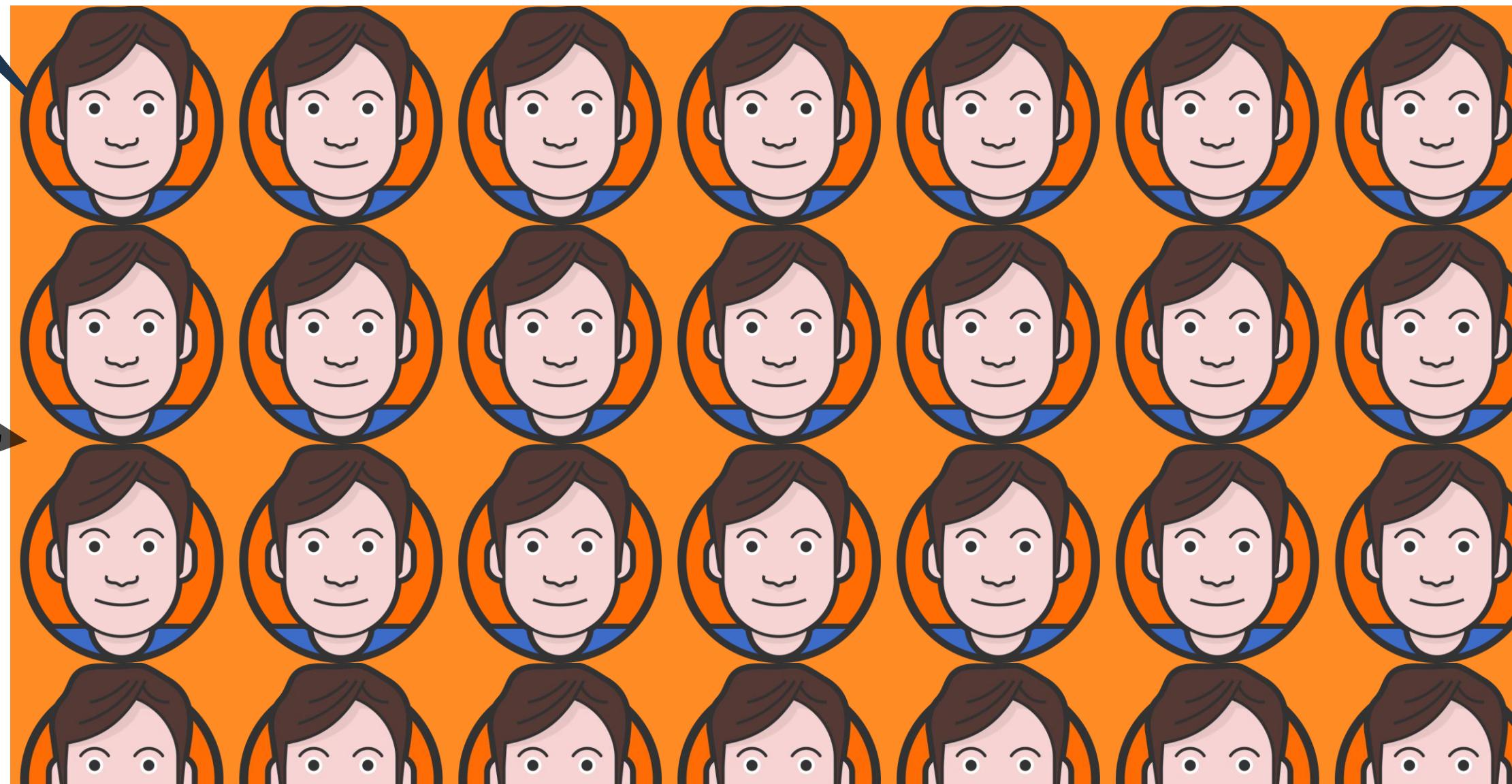
`none` 이미지 없음

`url("경로")` 이미지 경로

```
background-color: orange;  
background-image: url("./images/heropy.png");
```



배경 색상은  
이미지 뒤에 나와요!



## 요소의 배경 이미지 반복

# background-repeat

**repeat** 이미지를 수직, 수평 반복

**repeat-x** 이미지를 수평 반복

**repeat-y** 이미지를 수직 반복

**no-repeat** 반복 없음

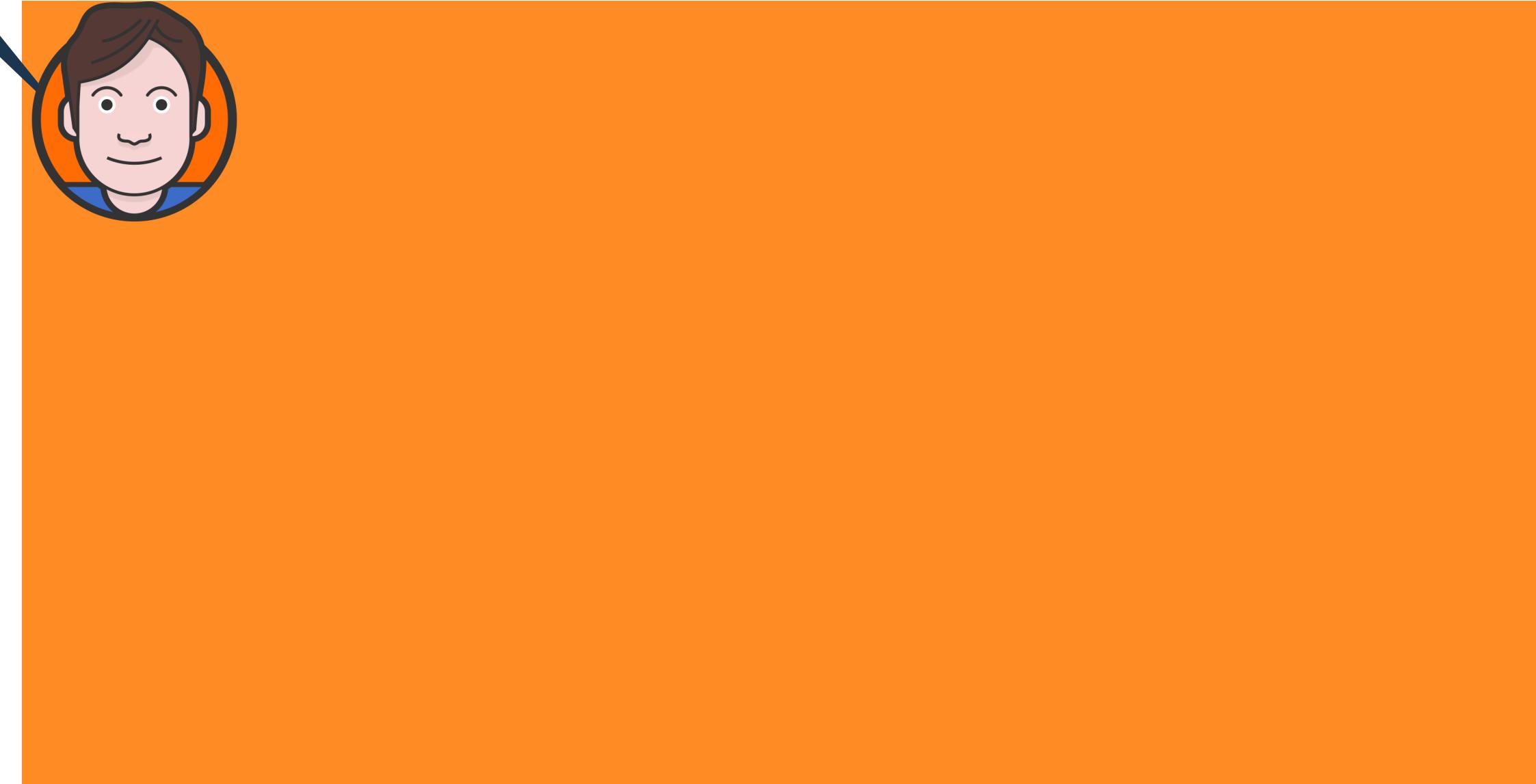
```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: repeat-x;
```



```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: repeat-y;
```



```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;
```



## 요소의 배경 이미지 위치

# background-position

방향1 방향2

0% 0%

0% ~ 100% 사이 값

방향

top, bottom, left, right, center 방향

단위

px, em, rem 등 단위로 지정

x축 y축



```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-position: top right;
```



```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-position: center;
```



```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-position: 100px 30px;
```

요소의 배경 이미지 크기

# background-size

auto

이미지의 실제 크기

단위

px, em, rem 등 단위로 지정

cover

비율을 유지, 요소의 더 넓은 너비에 맞춤비

contain

율을 유지, 요소의 더 짧은 너비에 맞춤

```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-size: cover;
```



```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-size: contain;
```



요소의 배경 이미지 스크롤 특성

# background-attachment

scroll

이미지가 요소를 따라서 같이 스크롤

fixed

이미지가 뷰포트에 고정, 스크롤 X

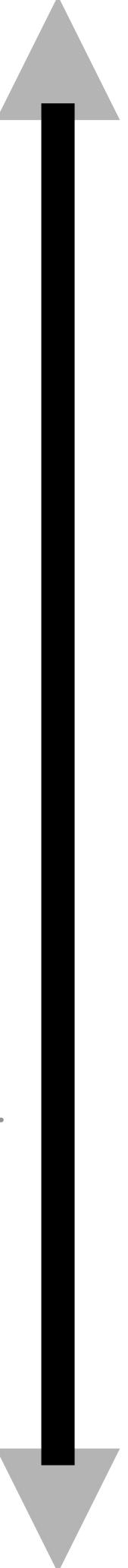
local

요소 내 스크롤 시 이미지가 같이 스크롤

```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-size: cover;  
background-attachment: scroll;
```



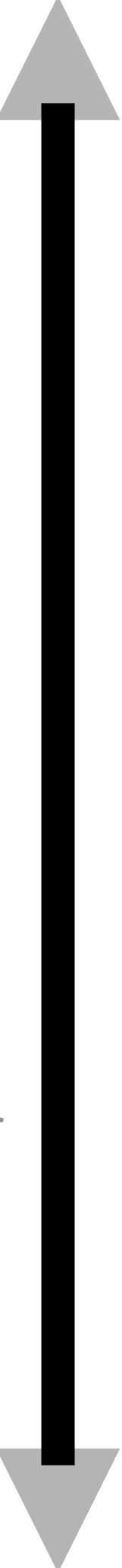
화면 스크롤



```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-size: cover;  
background-attachment: fixed;
```



화면 스크롤



배치

position과 같이 사용하는 CSS 속성들!  
모두 음수를 사용할 수 있어요!

**top**  
**bottom**  
**left**  
**right**  
**z-index**

요소의 위치 지정 기준

# position

**static** 기준 없음

**relative** 요소자신을 기준

**absolute** 위치 상 부모 요소를 기준

**fixed** 뷰포트(브라우저)를 기준

**sticky** 스크롤 영역 기준

위치 상 부모 요소를  
꼭 확인해야 해요!

요소의 각 방향별 거리 지정

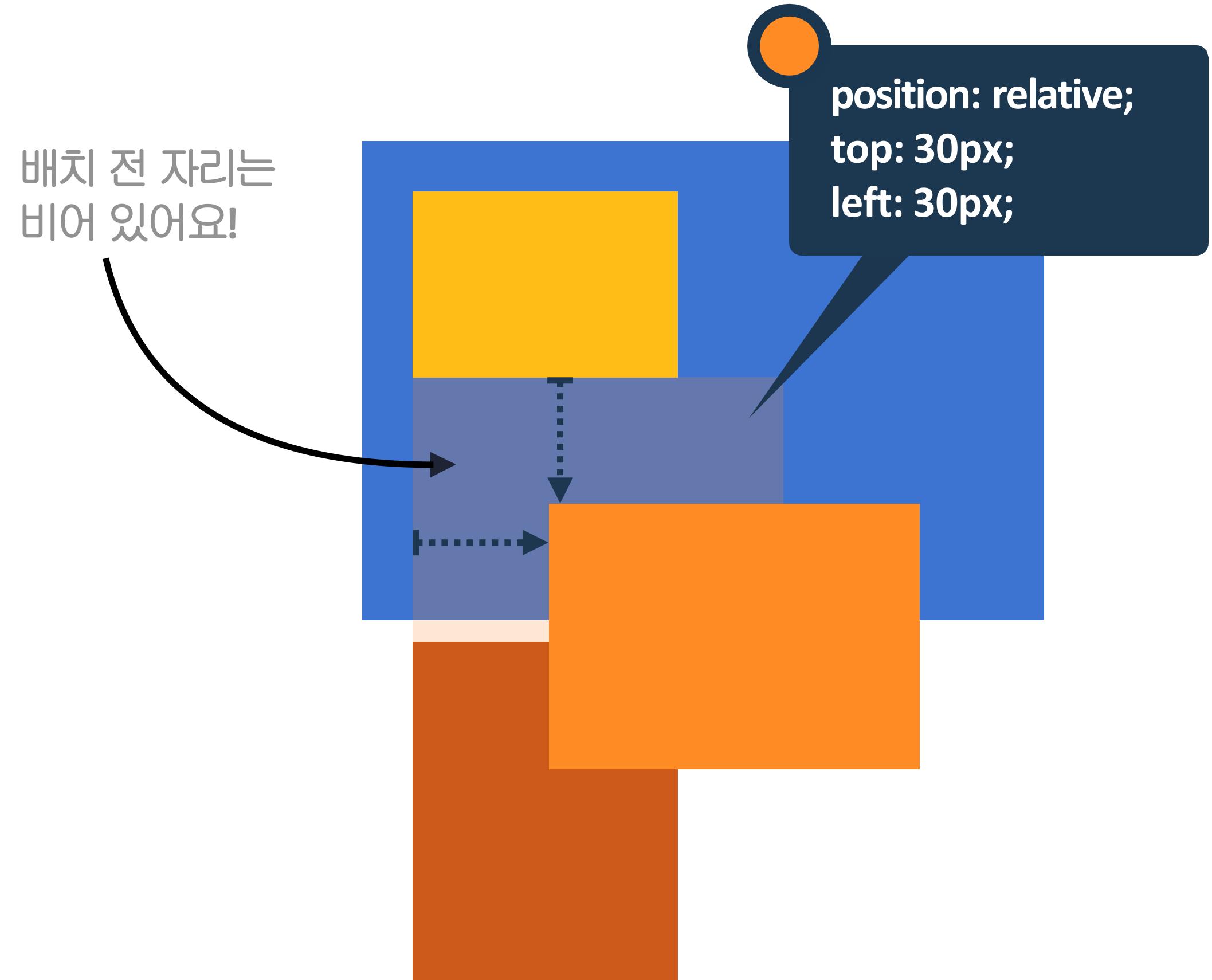
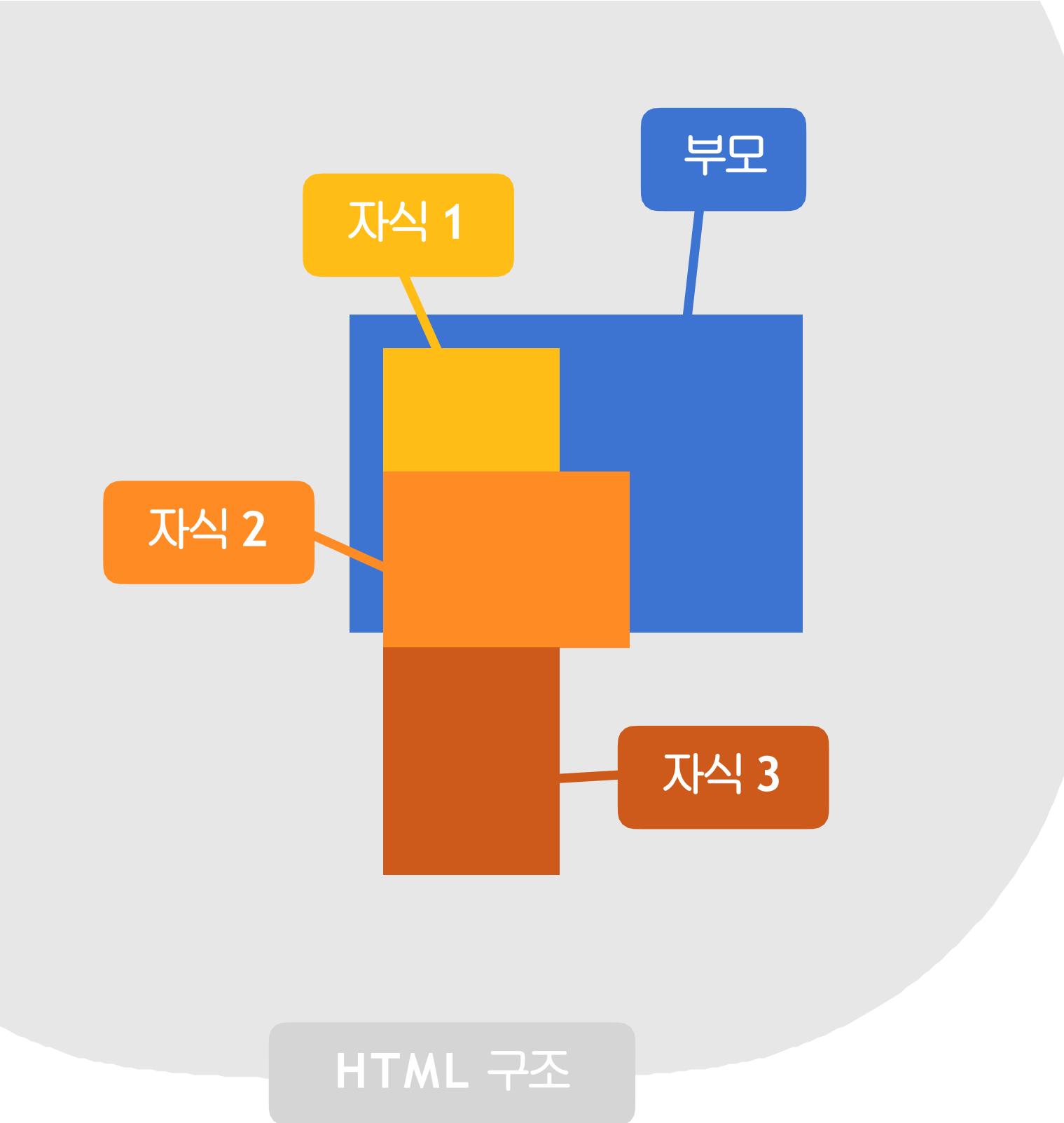
# top, bottom, left, right

auto

브라우저가 계산

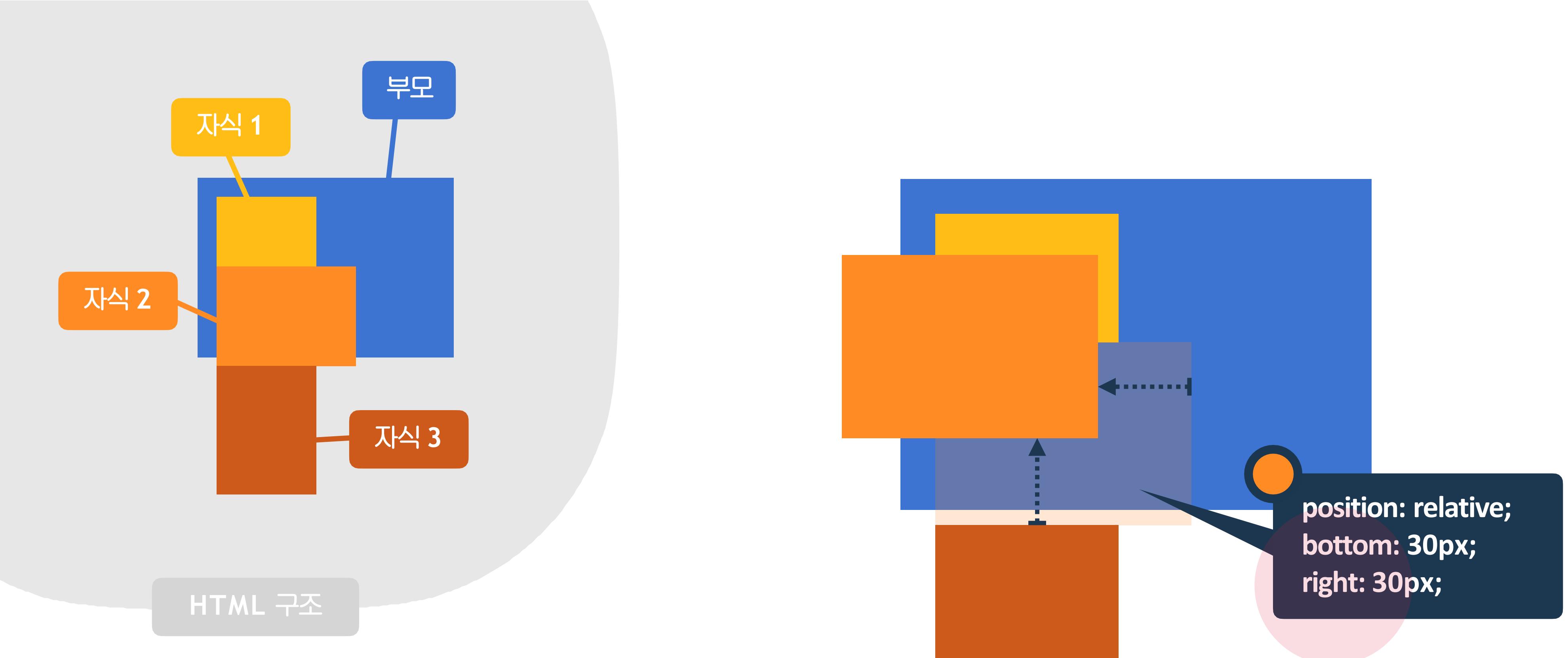
단위

px, em, rem 등 단위로 지정



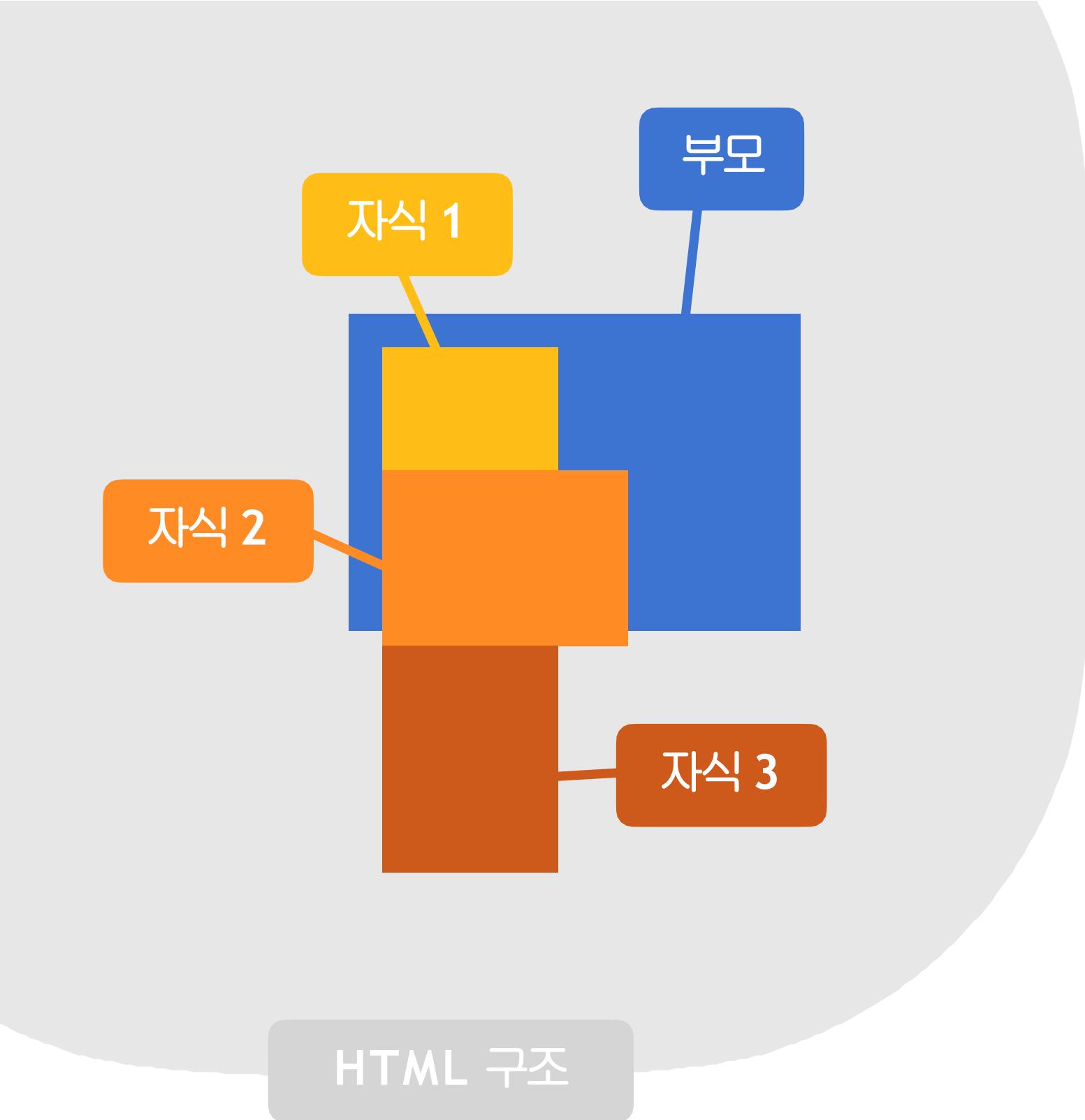
relative

요소 자신을 기준으로 배치!

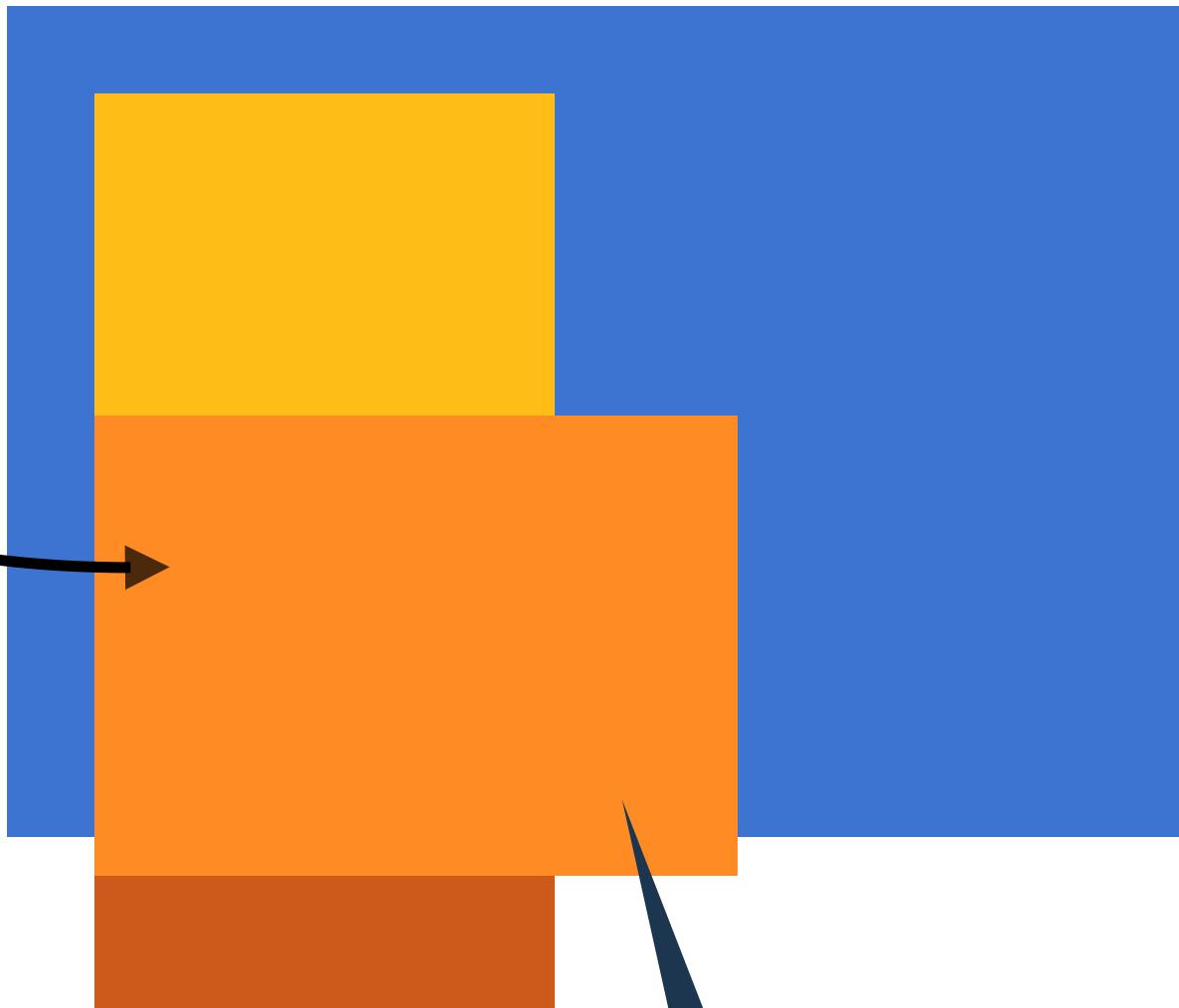


**relative**

요소 자신을 기준으로 배치!

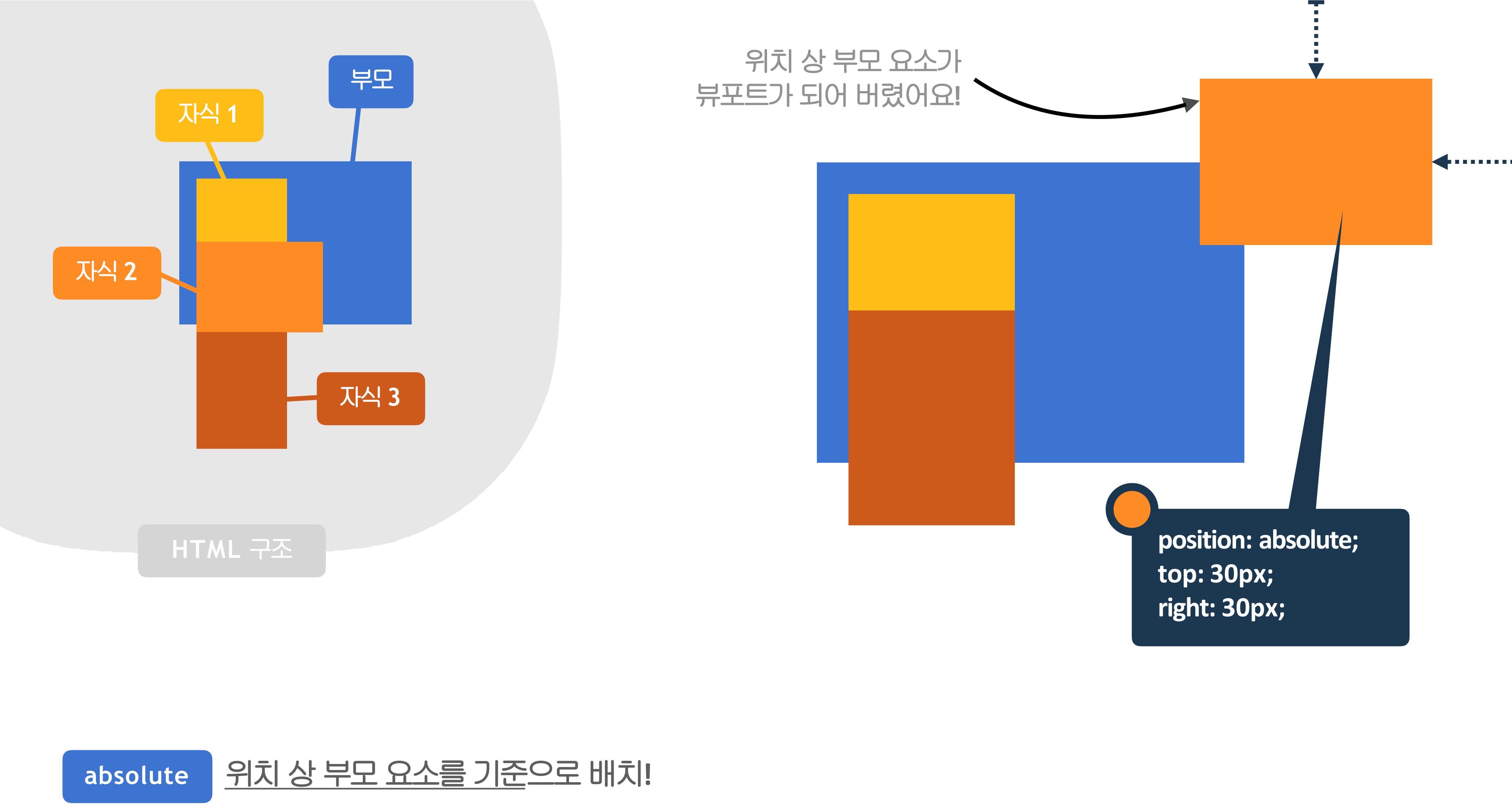


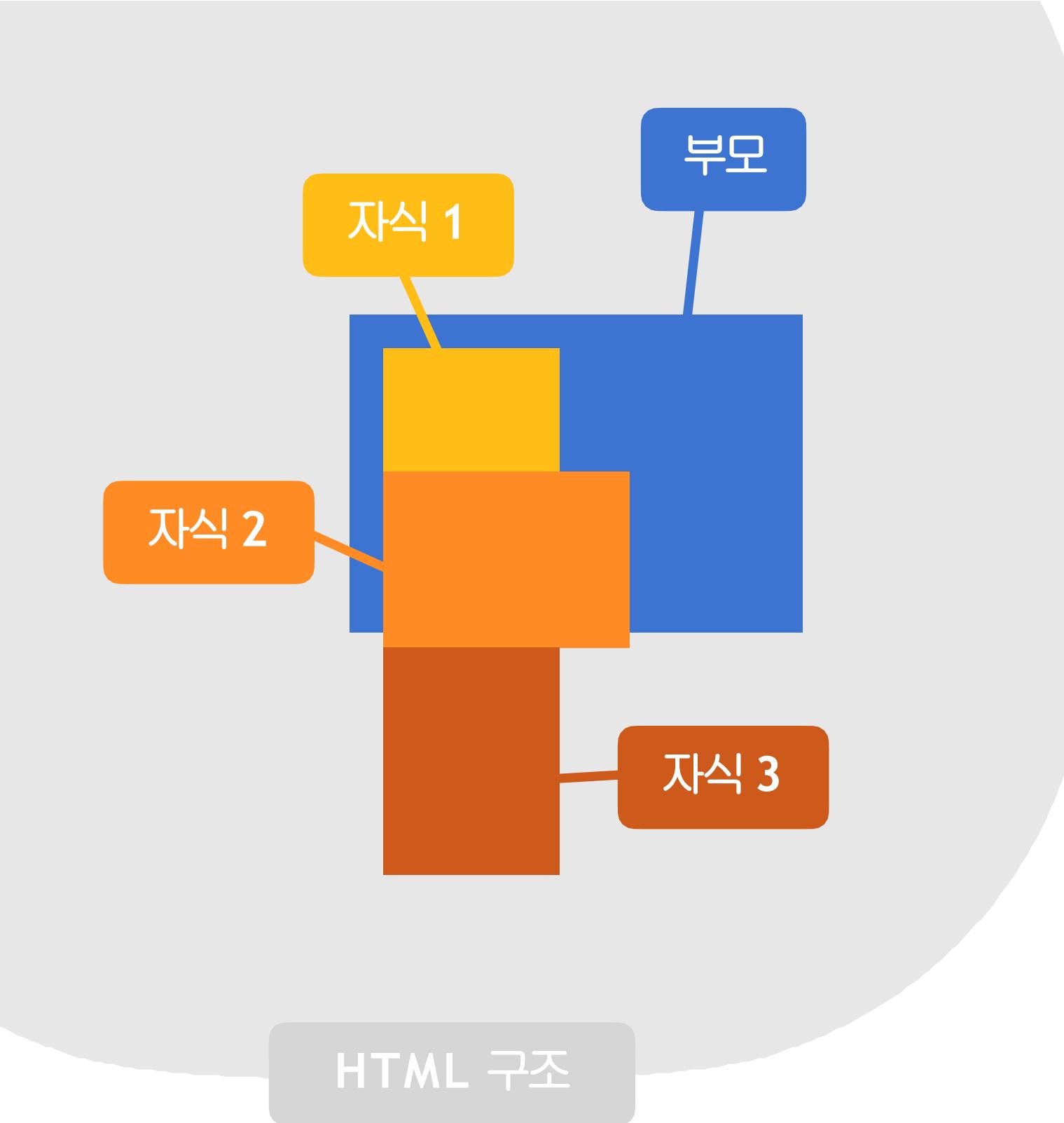
붕~ 뜨면서요  
소가 겹쳐요



absolute

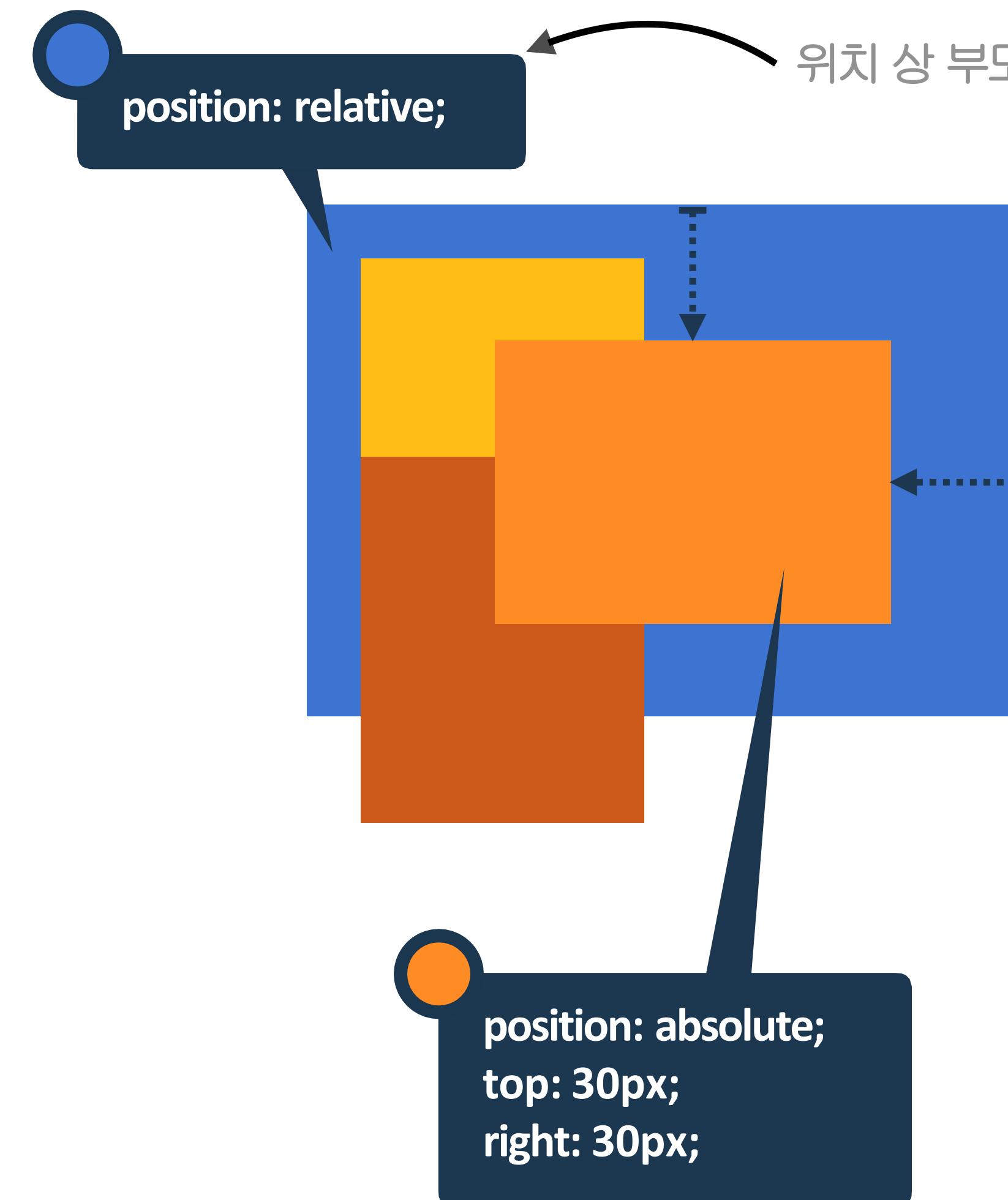
위치 상 부모 요소를 기준으로 배치!

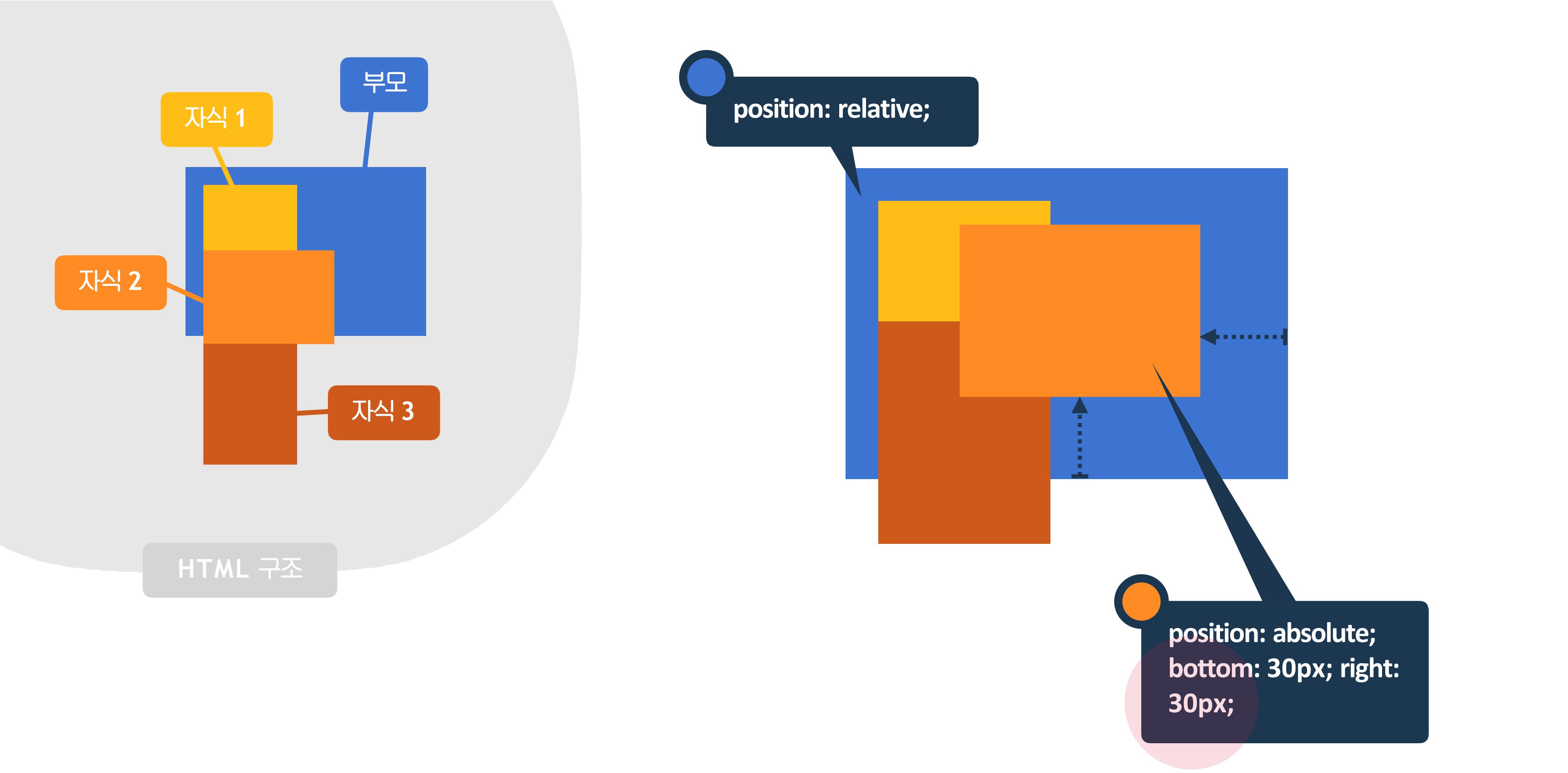




absolute

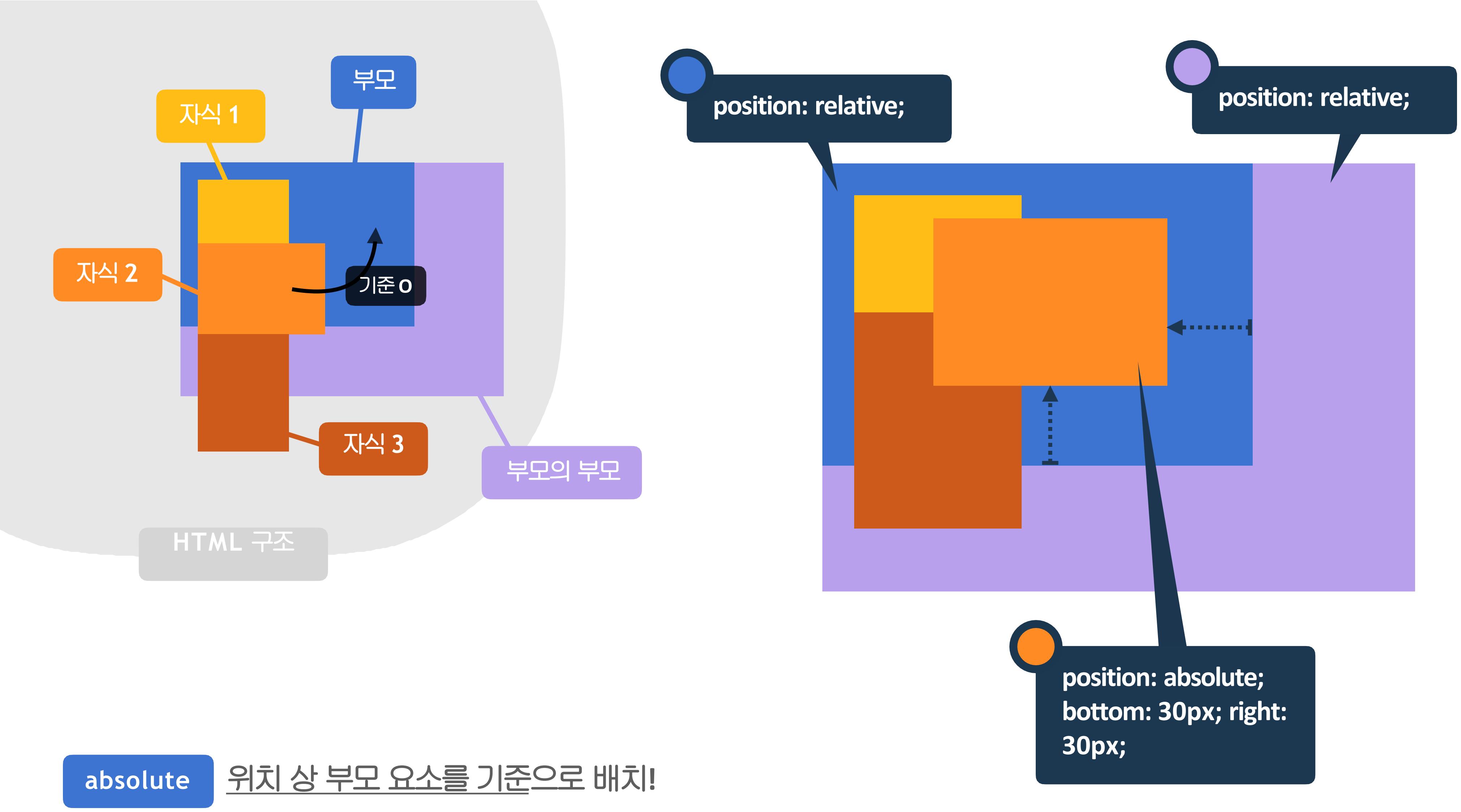
위치 상 부모 요소를 기준으로 배치!

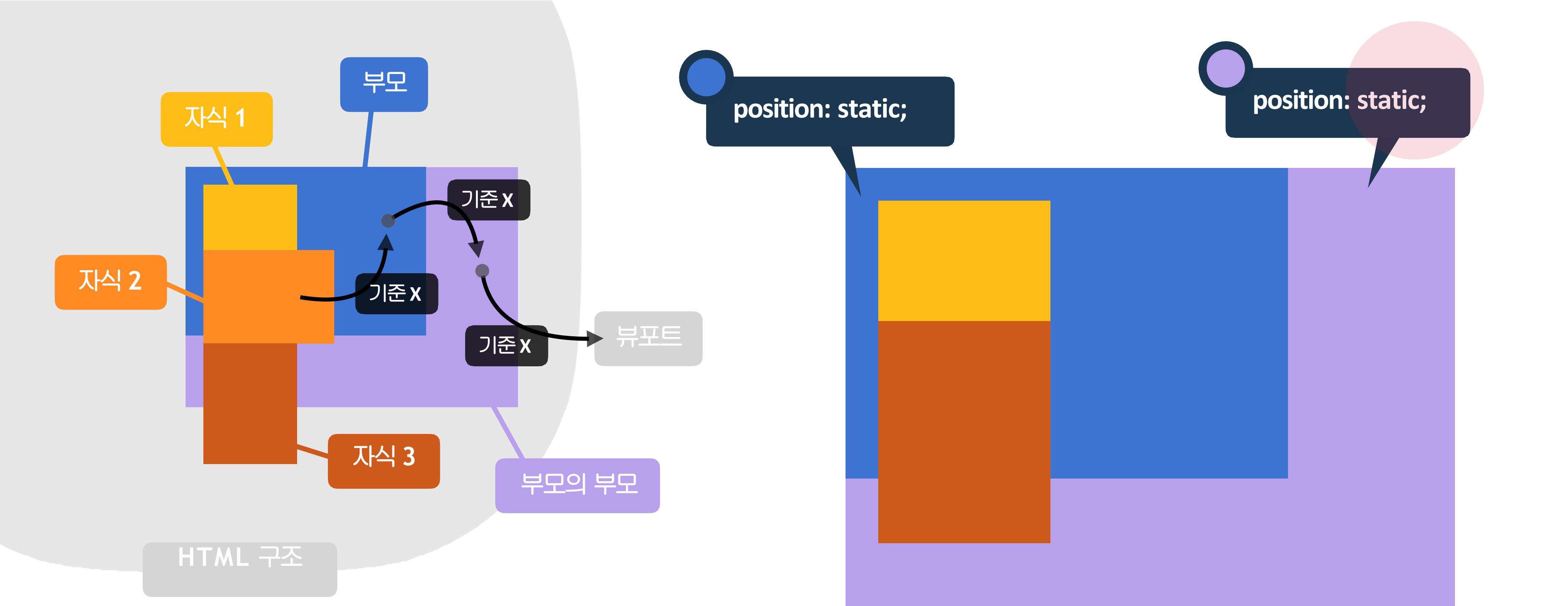




absolute

위치 상 부모 요소를 기준으로 배치!

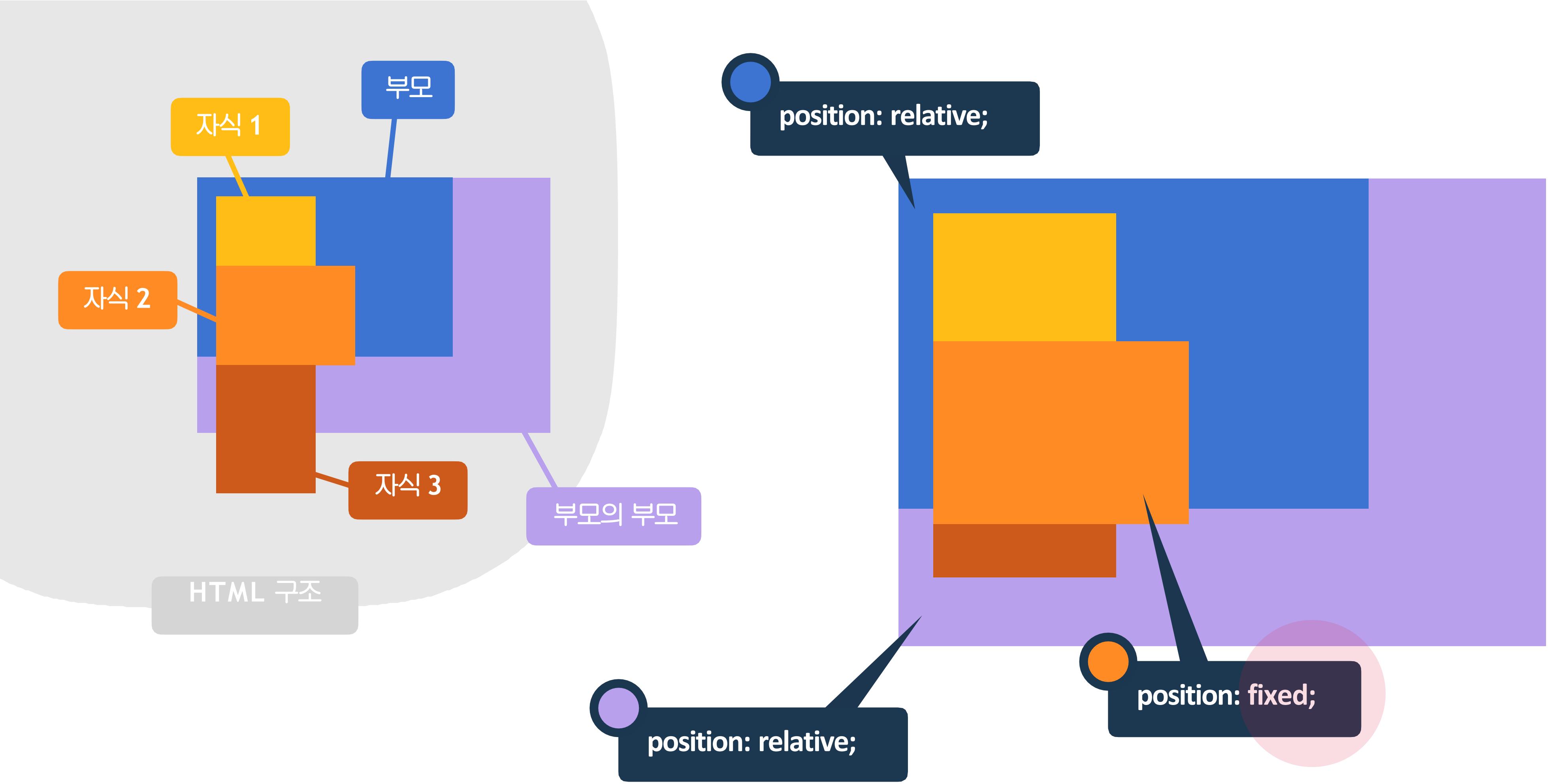




**absolute**

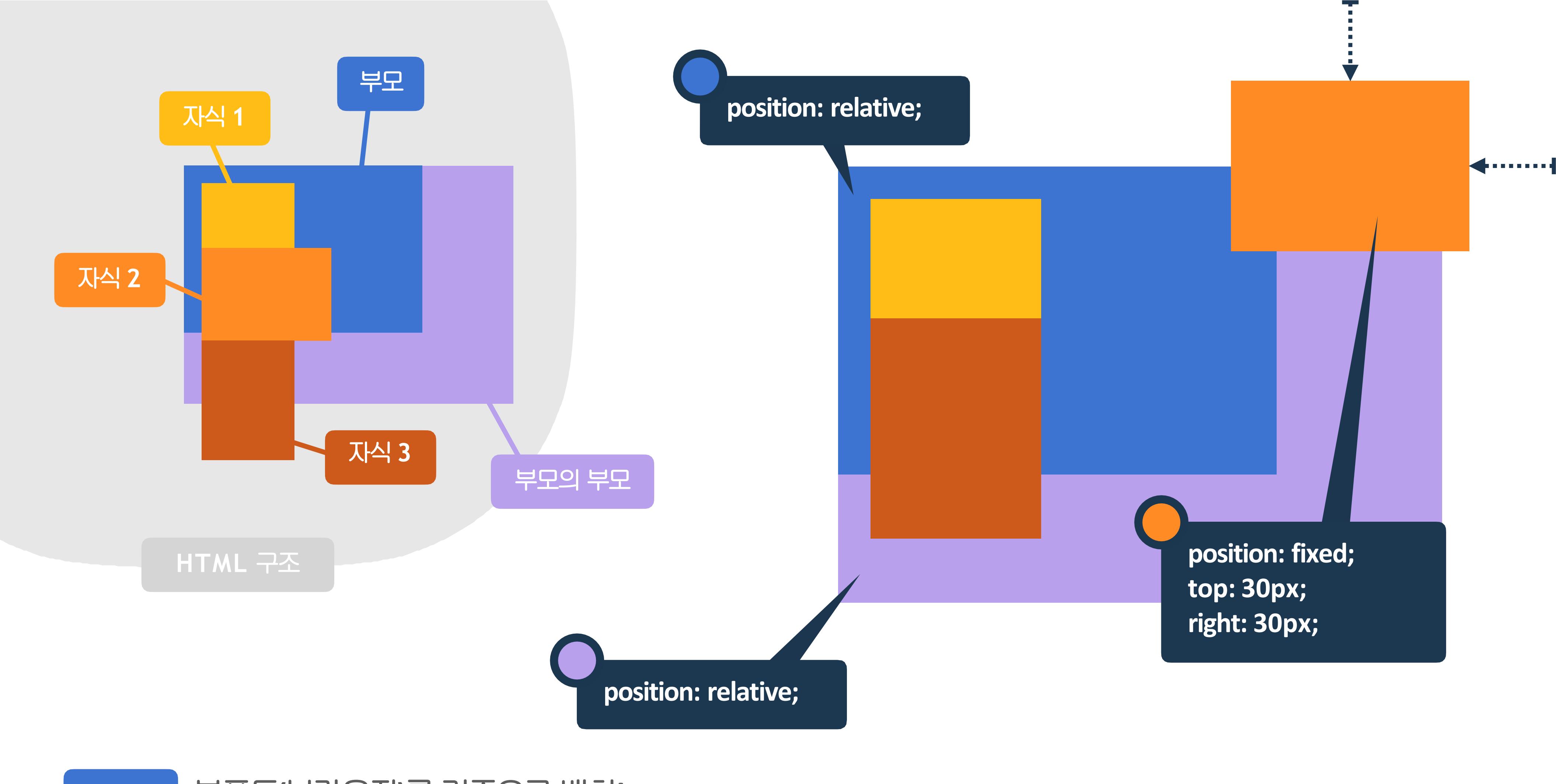
위치 상 부모 요소를 기준으로 배치!

`position: absolute;  
bottom: 30px; right:  
30px;`



fixed

뷰포트(브라우저)를 기준으로 배치!

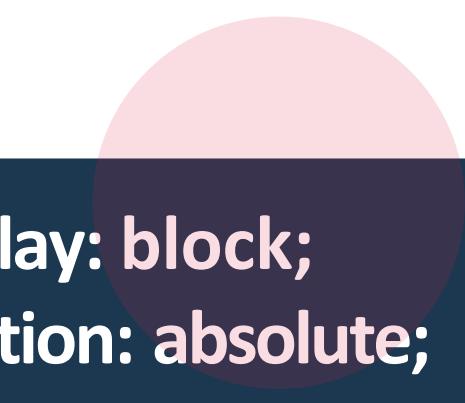


fixed

뷰포트(브라우저)를 기준으로 배치!

# 요소의 display가 변경됨

position 속성의 값으로 absolute, fixed가 지정된 요소는,  
display 속성이 block으로 변경됨.



```
display: block;  
position: absolute;  
top: 30px;  
left: 30px;  
z-index: 1;
```

=

```
position: absolute;  
top: 30px;  
left: 30px;  
z-index: 1;
```

# JavaScript 실행 학습

**vs Code에서 main.js 연결 테스트!**

# 표기법

dash-case(kebab-case)

snake\_case

camelCase

ParcelCase

HTML

CSS

# dash-case(kebab-case)

the-quick-brown-fox-jumps-over-the-lazy-dog

HTML

CSS

snake\_case

the\_quick\_brown\_fox\_jumps\_over\_the\_lazy\_dog

JS

camelCase

theQuickBrownFoxJumpsOverTheLazyDog

JS

# PascalCase

TheQuickBrownFoxJumpsOverTheLazyDog

# Zero-based Numbering

o 기반 번호 매기기!

특수한 경우를 제외하고 0부터 숫자를 시작합니다.



# 주석

Comments



# 데이터 종류(자료형)

String

Number

Boolean

Undefined

Null Object

Array

```
let a = 123;  
const n = obj.name;  
document.querySelector('.data-abc')  
{ name: 'Heropy', age: 85 }  
  
function mount(params) {  
  return this;  
}
```

















# 변수

데이터를 저장하고 참조(사용)하는 데이터의 이름

`var, let, const`







# 예약어

특별한 의미를 가지고 있어, 변수나 함수 이름 등으로 사용할 수 없는 단어

Reserved Word



**break, case, catch, continue, default, delete, do, else, false, finally, for, function, if, in, instanceof, new, null, return, switch, this, throw, true, try, typeof, var, void, while, with, abstract, boolean, byte, char, class, const, debugger, double, enum, export, extends, final, float, goto, implements, import, int, interface, long, native, package, private, protected, public, short, static, super, synchronized, throws, transient, volatile, as, is, namespace, use, arguments, Array, Boolean, Date, decodeURI, decodeURIComponent, encodeURI, Error, escape, eval, EvalError, Function, Infinity, isFinite, isNaN, Math, NaN, Number, Object, parseFloat, parseInt, RangeError, ReferenceError, RegExp, String, SyntaxError, TypeError, undefined, unescape, URIError ...**

# 함수

특정 동작(기능)을 수행하는 일부 코드의 집합(부분)

function











# 조건문

조건의 결과(truthy, falsy)에 따라 다른 코드를 실행하는 구문

if, else





# **DOM API**

**Document Object Model, Application Programming Interface**











```
// HTML 요소(Element) 모두 검색/찾기  
const boxEls = document.querySelectorAll('.box');  
console.log(boxEls);
```

// 찾은 요소들 반복해서 함수 실행!

// 익명 함수를 인수로 추가!

```
boxEls.forEach(function () { }) ;
```

// 첫 번째 매개변수(boxEl): 반복중인 요소.

// 두 번째 매개변수(index): 반복중인 번호

```
boxEls.forEach(function (boxEl, index) { }) ;
```

// 출력!

```
boxEls.forEach(function (boxEl, index) {  
    boxEl.classList.add(`order-${index + 1}`);  
    console.log(index, boxEl);  
} );
```



# 메소드 체이닝

Method Chaining



```
const a = 'Hello~';  
// split: 문자를 인수 기준으로 쪼개서 배열로 반환.  
// reverse: 배열을 뒤집기.  
// join: 배열을 인수 기준으로 문자로 병합해 반환.  
  
const b = a.split(' ').reverse().join(''); // 메소드 체이닝  
...  
  
console.log(a); // Hello~  
console.log(b); // ~olleH
```

Q.

The quick brown fox

위 문장을 camelCase(낙타 표기법)로 작성하시오!

A.

theQuickBrownFox

Q.

```
let fruits = ['Apple', 'Banana', 'Cherry'];
```

위 데이터를 활용해 'Banana'를 콘솔 출력하시오!

A.

```
console.log(fruits[1]);
```

Q.

불린 데이터(Boolean)에서  
거짓을 의미하는 데이터는?

A.

false

Q.

'값이 의도적으로 비어있음'을 의미하는 데이터는?

A.

null

Q.

{ }

위 데이터의 종류는?

A.

Object(객체 데이터)

Q.

```
let obj = { abc: 123 };
console.log(obj.xyz);
```

위 코드를 통해 콘솔 출력될 값(데이터)은?

A.

undefined

Q.

값(데이터)을 재할당할 수 없는  
변수 선언 키워드는?

A.

const

Q.

함수에서 값(데이터)을 반환하기 위해  
사용하는 키워드는?

A.

return

Q.

sum(2, 4);

위 함수 호출에서 2, 4를 무엇이라 하는가?

A.

인수(Arguments)

Q.

```
function sum(a, b) {  
    return a +b;  
}
```

위 함수 선언의 a, b와 같이, 함수 호출에서 전달받은 **인수**를 함수 내부로 **전달하기** 위한 **변수**를 무엇이라 하는가?

A.

매개변수(Parameters)

Q.

이름이 없는 함수를 무엇이라 하는가?

A.

## 익명 함수(Anonymous Function)

Q.

hello 이름의 함수 표현을 작성하고 호출하시오!

A.

```
const hello = function () { };  
hello();
```

Q.

```
const user = {  
  getName: function () {}  
}
```

위 코드의 `getName`과 같이,  
함수가 할당된 객체 데이터의  
속성(Property)을 무엇이라 하는가?

A.

메소드(Method)

Q.

조건이 참(true)인 조건문을 작성하시오!

A.

```
if (true) { }
```

Q.

가져온 JS 파일을 HTML 문서 분석 이후에 실행하도록  
지시하는 HTML 속성(Attribute)은?

A.

defer

Q.

```
<div class="box">Box!!</div>
```

위 HTML 요소의 내용(Content)을 콘솔 출력하시오!

A.

```
const boxEl = document.querySelector('.box');  
console.log(boxEl.textContent);
```

Q.

값(데이터)을 재할당할 목적의  
변수 선언 키워드는?

A.

let

Q.

const boxEl = document.querySelector('.box'); 위 코드의 boxEl 요소에 클릭(click) 이벤트를 추가해, 클릭 시 'Hello~'를 콘솔 출력하시오!

A.

```
boxEl.addEventListener('click', function () {  
    console.log('Hello~');  
});
```

Q.

<div>1</div>

<div>2</div>

위 2개의 DIV 요소에 JS로  
class="hello"를 추가하시오!

A.

```
const divEls = document.querySelectorAll('div');

divEls.forEach(function (divEl) {

    divEl.classList.add('hello');

});
```

Q.

'HEROPY'.split(' ').reverse( ).join(' '); 위와  
같이, 메소드를 이어 작성하는 방법을  
무엇이라 하는가?

A.

메소드 체이닝(Method Chaining)

Q.

`const boxEl = document.querySelector('.box');` 위  
코드의 `boxEl` 요소에 HTML 클래스 속성의 값으로  
'active'가 포함되어 있으면,  
'포함됨!'을 콘솔 출력하시오!

A.

```
if (boxEl.classList.contains('active')) {  
    console.log('포함됨!');  
}
```