

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.

스타일 상속

```
.animal {  
  color: ■ red;  
}
```

선택

```
<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>
```

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!





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

인라인 선언 - 1000점

21점 `.list li.item { color: red; }`

21점 `.list li:hover { color: red; }`

10점 `.box::before { content: "Good "; color: red; }`

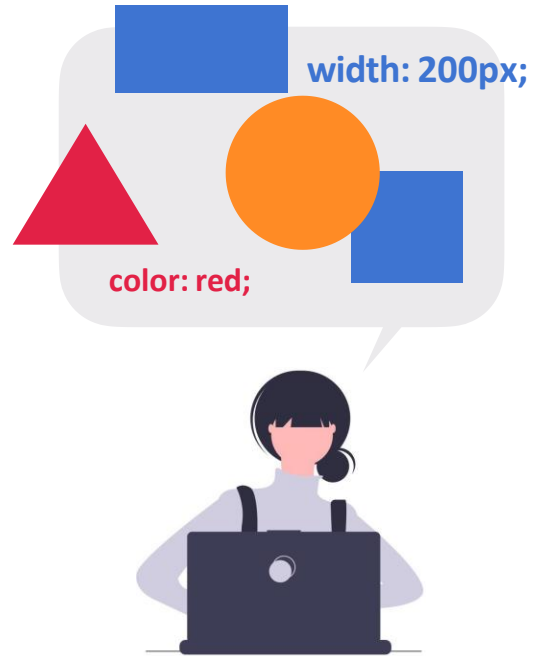
101점 `#submit span { color: red; }`

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
배치
플렉스(정렬)
전환
변환
띄움
애니메이션
그리드
다단
필터

박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

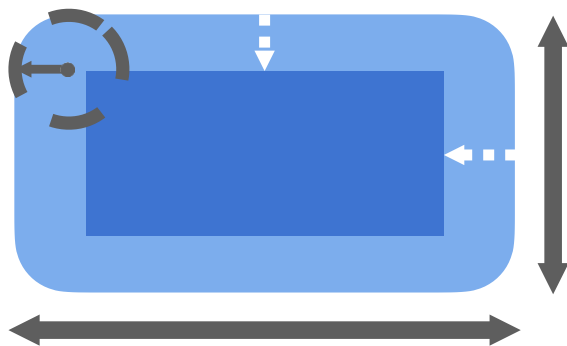
띄움

애니메이션

그리드

다단

필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

띄움

애니메이션

그리드

다단

필터

Hello world
Good morning~

박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

띄움

애니메이션

그리드

다단

필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

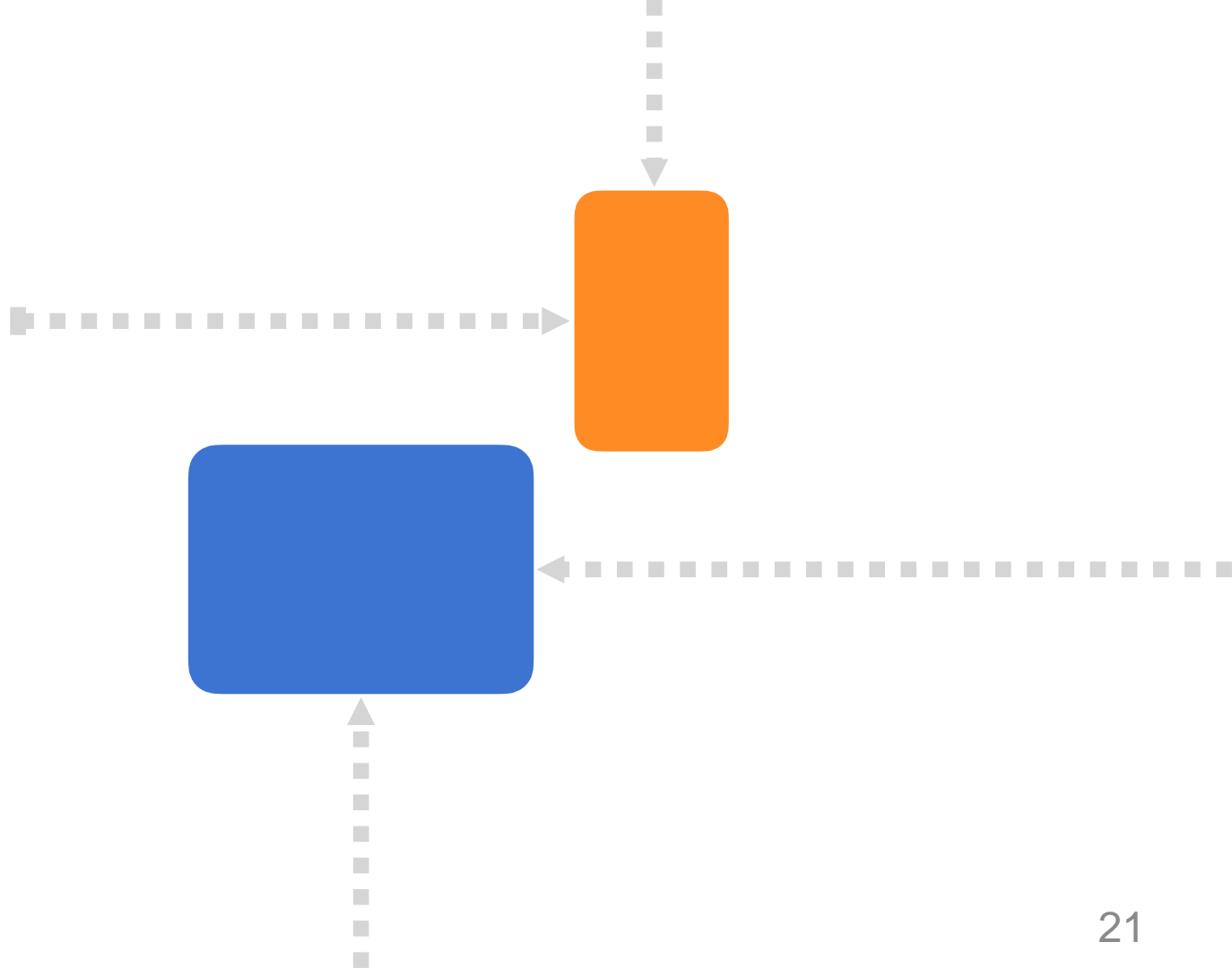
띄움

애니메이션

그리드

다단

필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

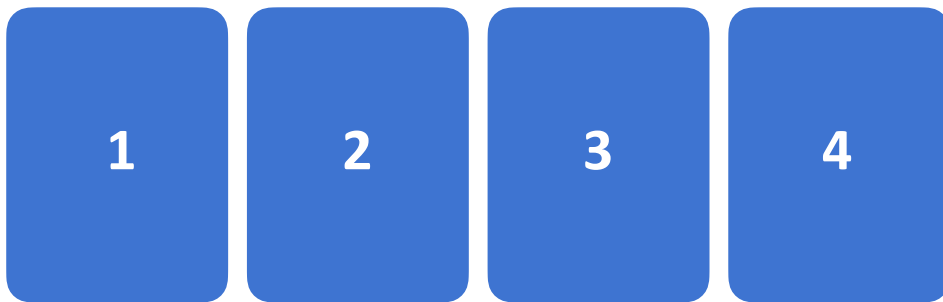
띄움

애니메이션

그리드

다단

필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

띄움

애니메이션

그리드

다단

필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

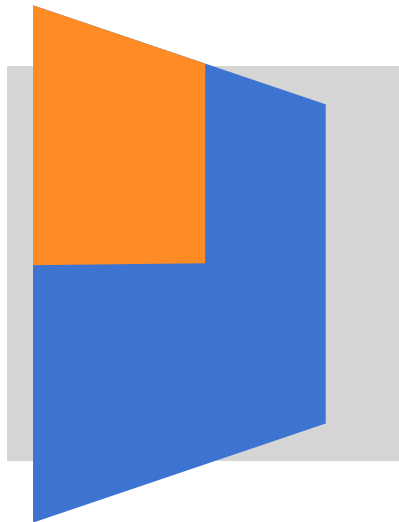
띄움

애니메이션

그리드

다단

필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

띄움

애니메이션

그리드

다단

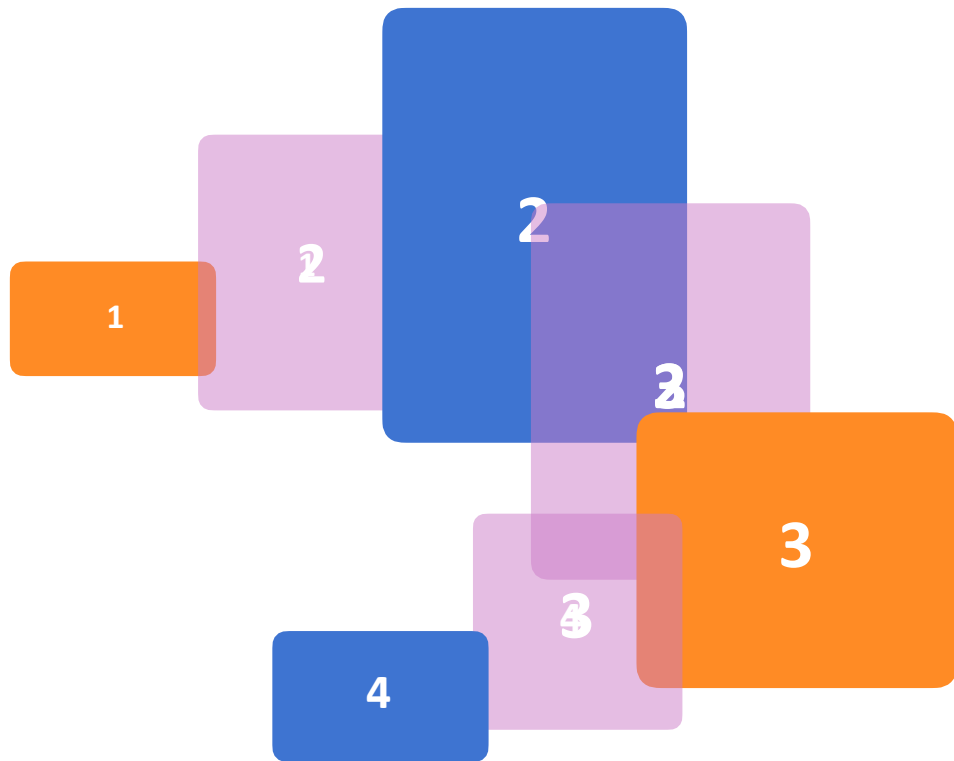
필터



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
배치
플렉스(정렬)
전환
변환
띄움
애니메이션
그리드
다단
필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

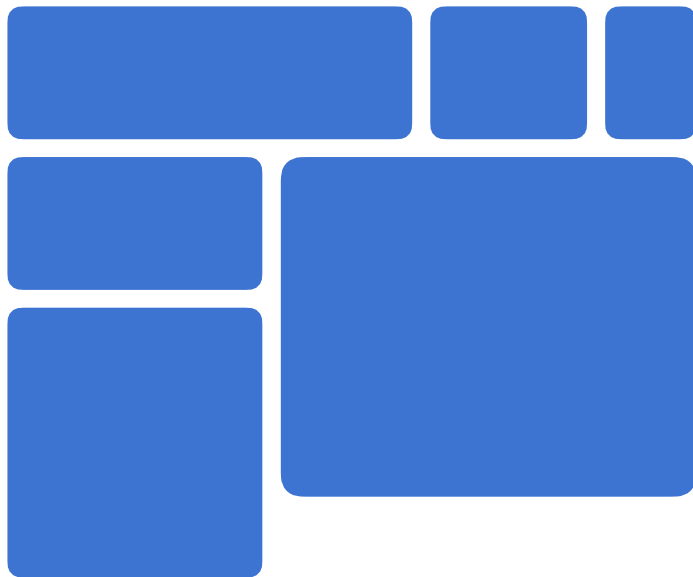
띄움

애니메이션

그리드

다단

필터



박스 모델

글꼴, 문자

배경

배치

플렉스(정렬)

전환

변환

띄움

애니메이션

그리드

다단

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

배치

플렉스(정렬)

전환

변환

띄움

애니메이션

그리드

다단

필터



박스 모델

요소의
가로/세
로 너비

width, height

기본값
(요소에 이미 들어있는 속성의 값)



auto

브라우저가 너비를 계산

단위

px, em, vw 등 단위로 지정

`Hello`
`World`

``

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



`<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!

Hello

World

포함한 콘텐츠 크기만큼
자동으로 줄어듬!

auto

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

px	픽셀
%	상대적 백분율
em	요소의 글꼴 크기
rem	루트 요소(html)의 글꼴 크기
vw	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 없음 브라우저

auto

가 여백을 계산

단위

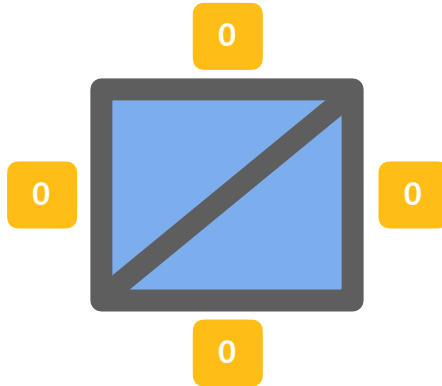
px, em, vw 등 단위로 지정

%

부모 요소의 가로 너비에 대한 비율로 지정

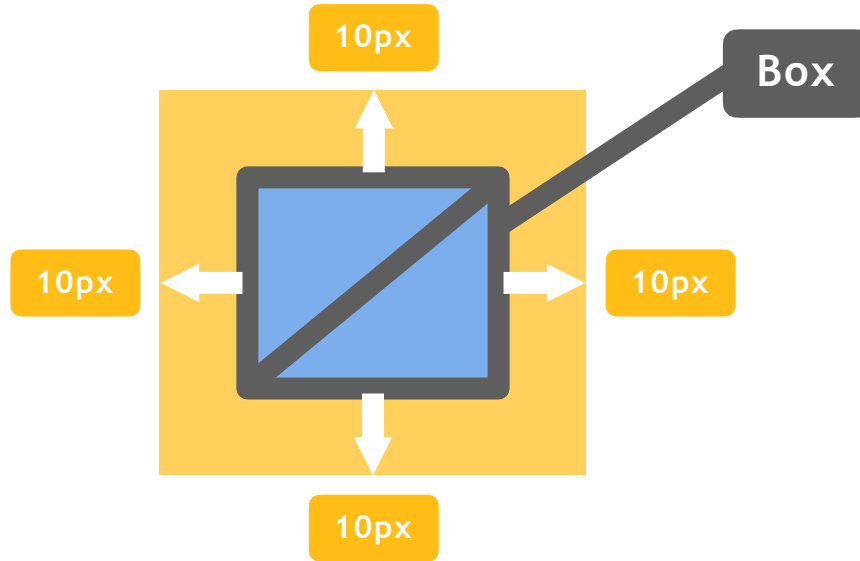
margin: 0;

top, right, bottom, left



margin: 10px;

top, right, bottom, left

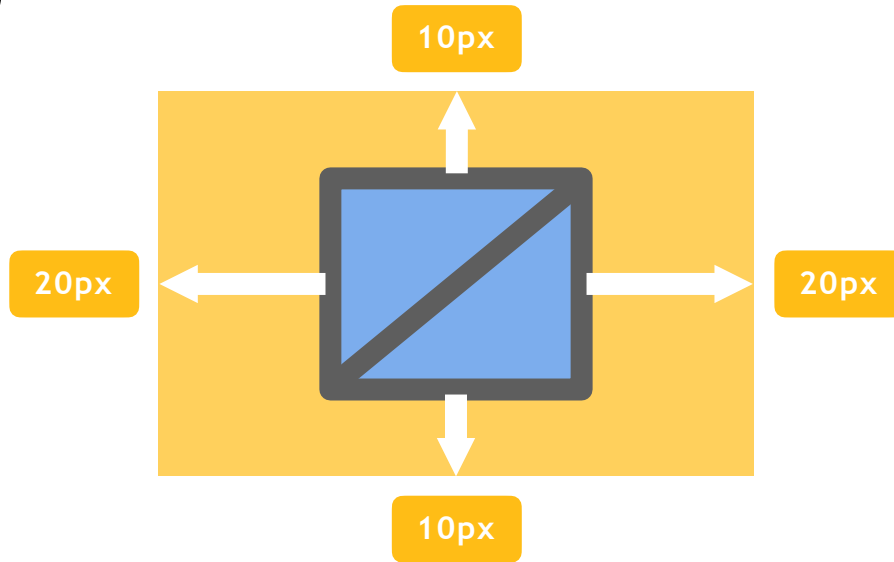


margin: 10px 20px;

top, bottom

left, right

상하.좌우.



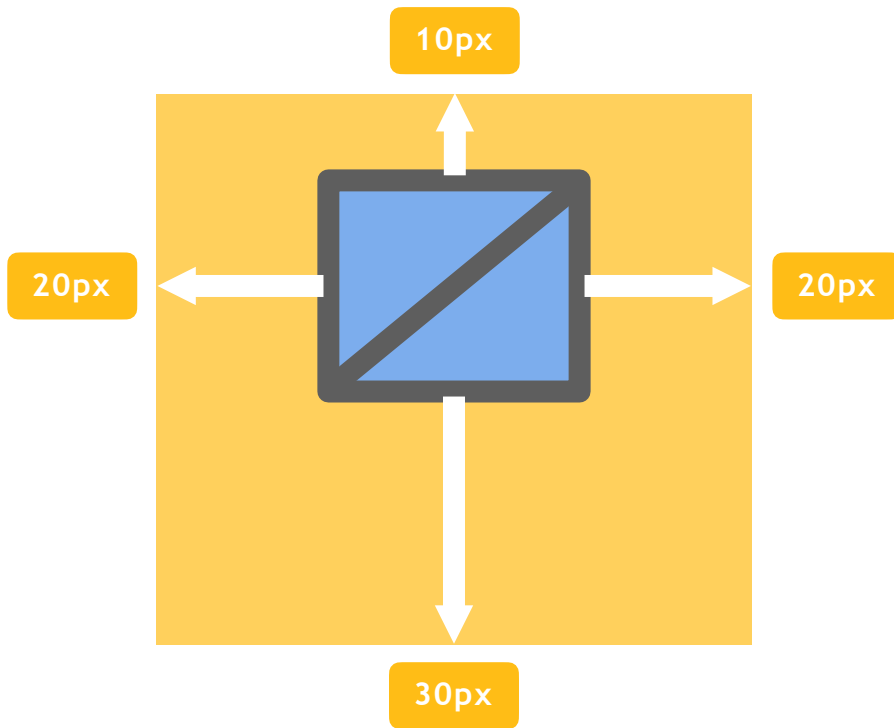
margin: 10px 20px 30px;

top

left, right

bottom

상.중.하



margin: 10px 20px 30px 40px;

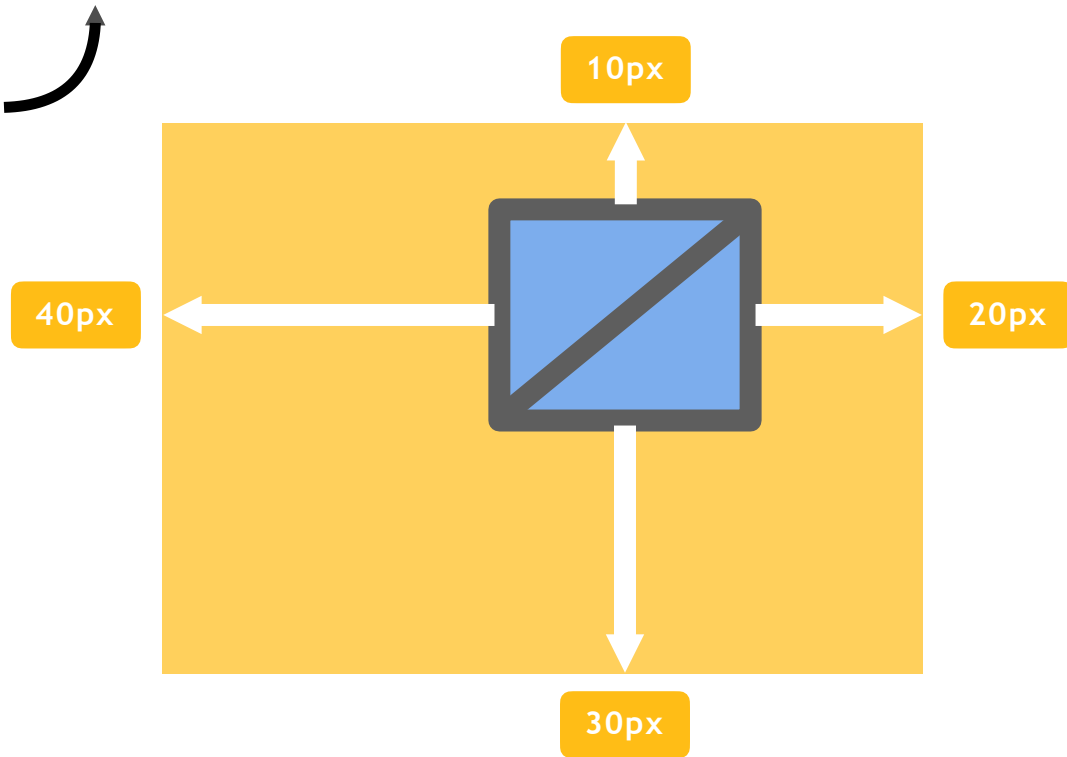
top

right

bottom

left

상.시계 방향



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

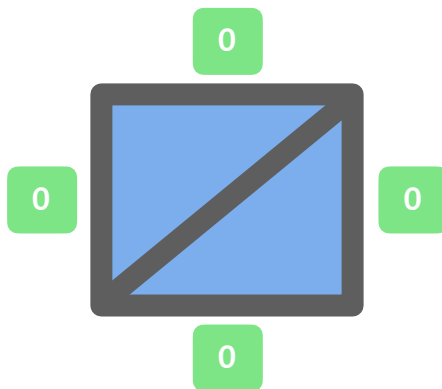
px, em, vw 등 단위로 지정

%

부모 요소의 가로 너비에 대한 비율로 지정

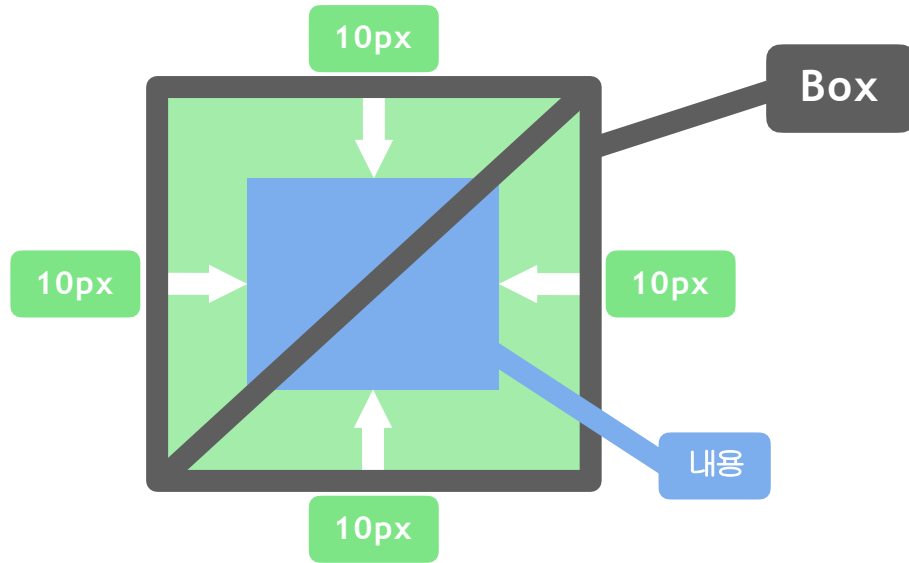
padding: 0;

top, right, bottom, left



padding: 10px;

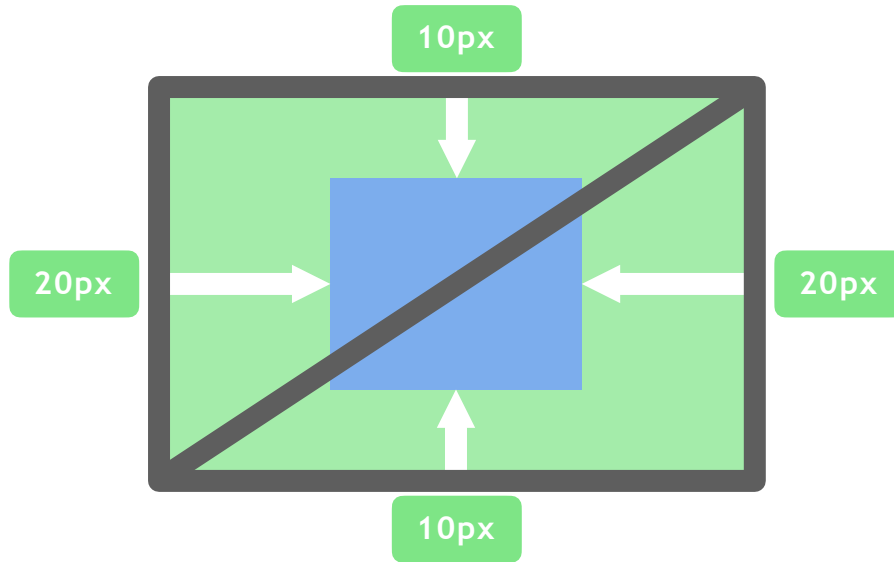
top, right, bottom, left



padding: 10px 20px;

top, bottom

left, right

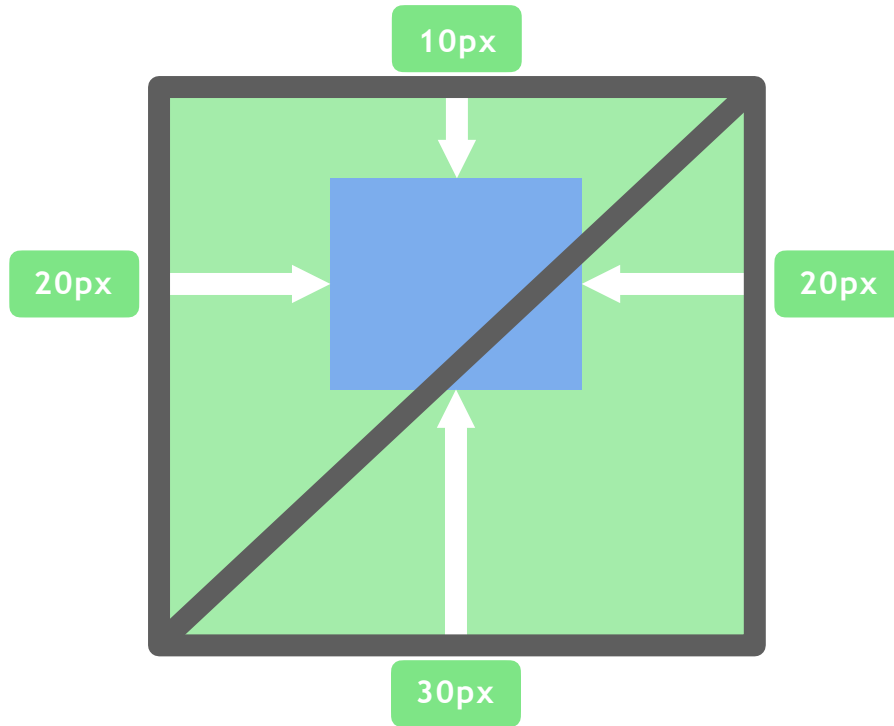


padding: 10px 20px 30px;

top

left, right

bottom



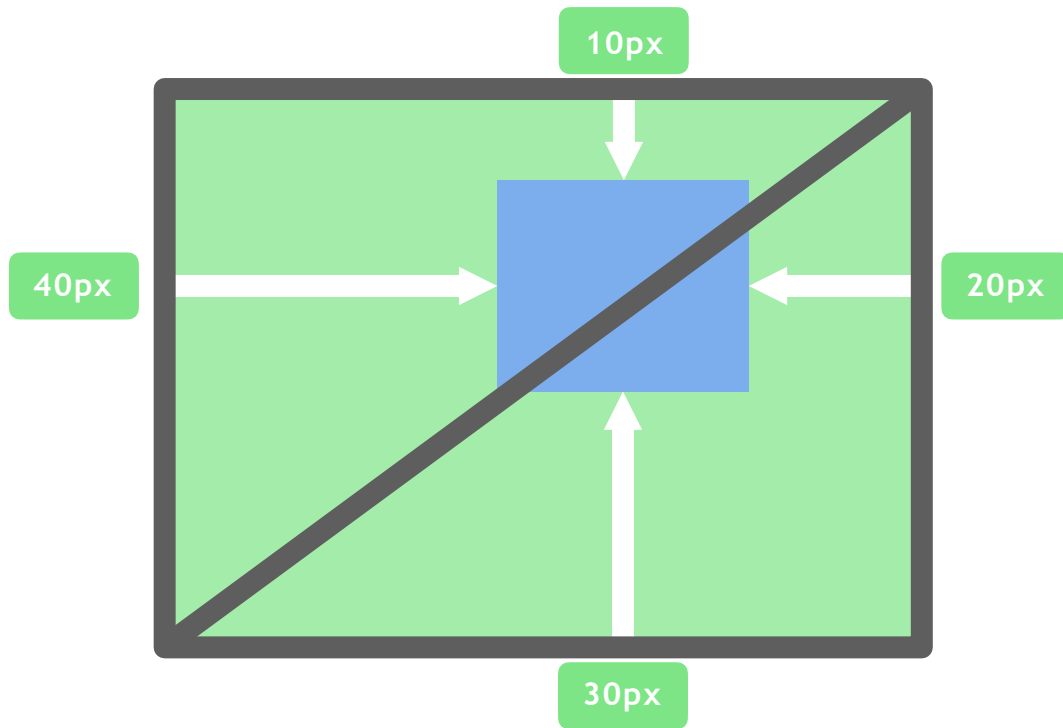
padding: 10px 20px 30px 40px;

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

BOX MODEL



보더

요소의 크기가 커져요!

요소의 테두리 선을 지정하는 단축 속성

border: 선-두께 선-종류 선-색상;

border-width

border-style

border-color

border: medium none ■ black;

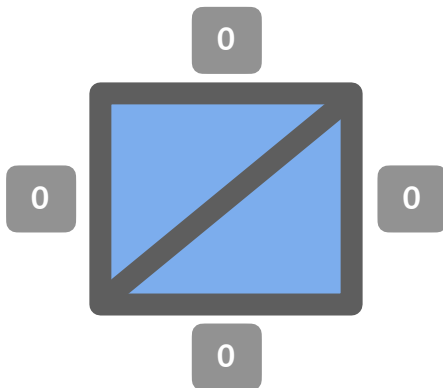
border-width

border-style

border-color

기본값
(요소에 이미 들어있는 속성의 값)

선의 종류가 없어서(**none**)
출력되지 않아요!

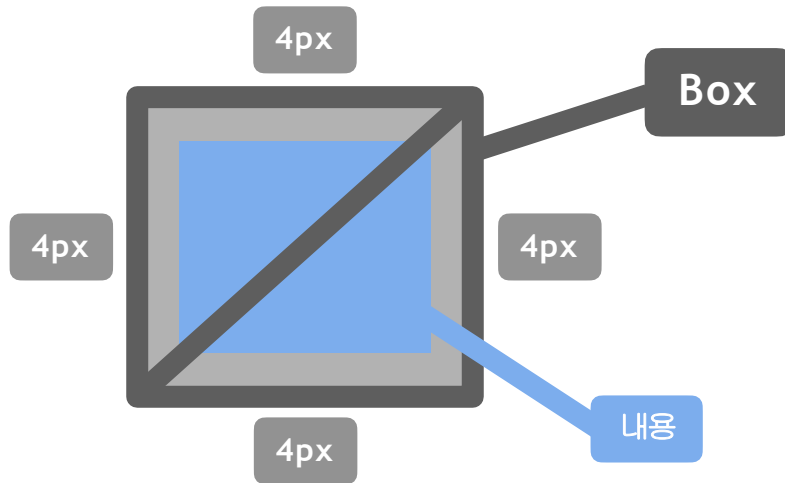


border: 4px solid ■ black;

border-width

border-style

border-color

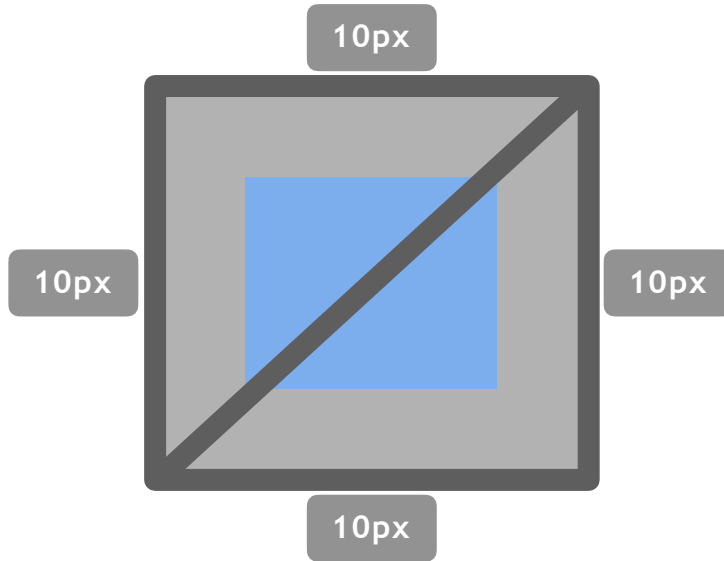


border: 10px solid ■ black;

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 ;

border-style

none	선 없음
solid	실선 (일반 선)
dotted	점선
dashed	파선
double	두 줄 선
groove	홈이 파여있는 모양
ridge	숫은 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 이름16	red, tomato, royalblue
색상코드RGB	진수 색상(Hexadecimal Colors) 빛의	#000, #FFFFFF
RGBA	삼원색	rgb(255, 255, 255)
	빛의 삼원색 + 투명도	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 모
서리를 둥
글게 깎음

border-radius

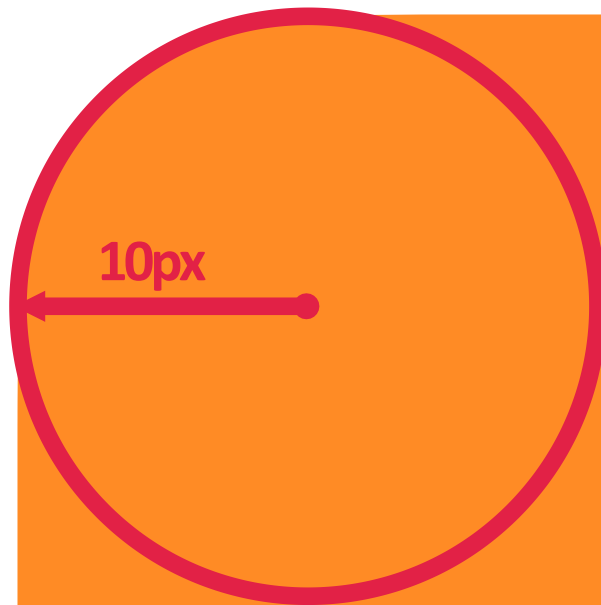
0

둥글게 없음

단위

px, em, vw 등 단위로 지정

`border-radius: 10px;`



`border-radius: 0;`

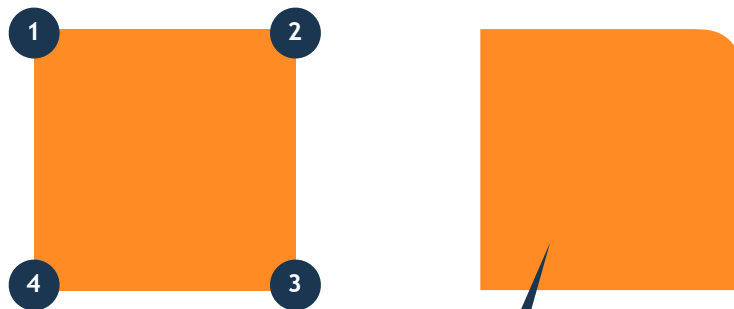


`border-radius: 10px;`



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





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

요소의 크기 계산 기준을 지정

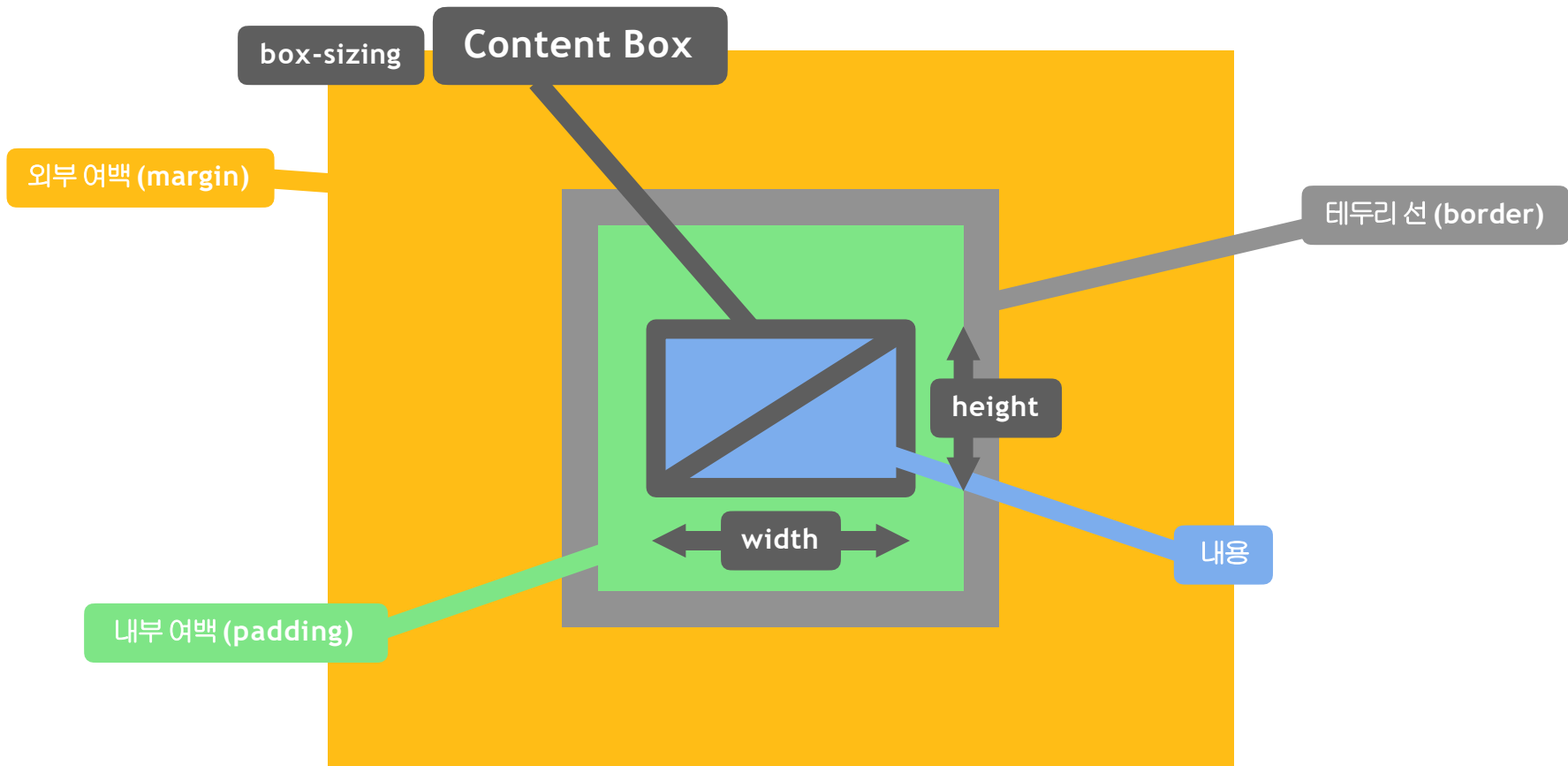
box-sizing

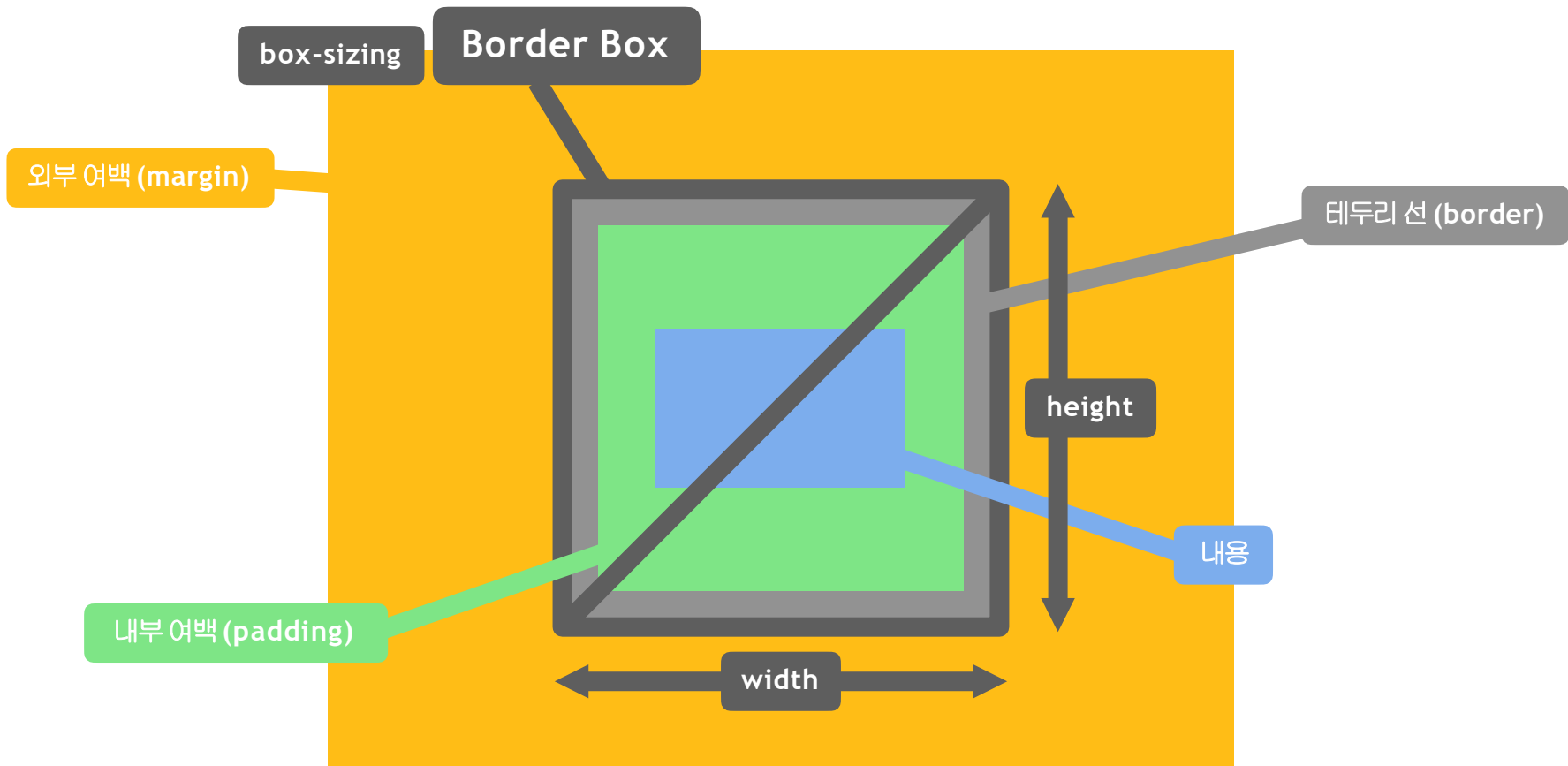
content-box

요소의 내용(content)으로 크기 계산

border-box

요소의 내용 + padding + border로 크기 계산





```
box-sizing: content-box;
width: 100%;
```

```
box-sizing: content-box;
width: 100%;
border: solid #586DCD 10px;
padding: 5px;
```

```
box-sizing: border-box;
width: 100%;
border: solid #586DCD 10px;
padding: 5px;
```

Parent container

Child container

```
box-sizing: content-box;
width: 100%;
```

```
box-sizing: content-box;
width: 100%;
border: solid #586DCD 10px;
padding: 5px;
```

```
box-sizing: border-box;
width: 100%;
border: solid #586DCD 10px;
padding: 5px;
```

Parent container

Child container

```
box-sizing: content-box;
width: 100%;
```

```
box-sizing: content-box;
width: 100%;
border: solid #586DCD 10px;
padding: 5px;
```

```
box-sizing: border-box;
width: 100%;
border: solid #586DCD 10px;
padding: 5px;
```

Parent container

Child container

요소의 크기 이상으로 내용이 넘쳤을 때, 보여짐을 제어하는 단축 속성

overflow

visible

넘친 내용을 그대로 보여줌

hidden

넘친 내용을 잘라냄

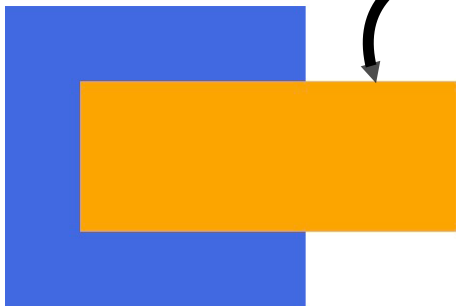
scroll

넘친 내용을 잘라냄, 스크롤바 생성

auto

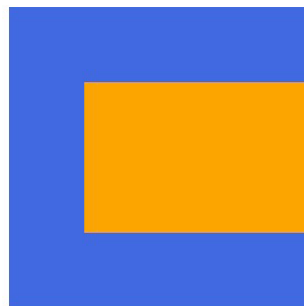
넘친 내용이 있는 경우에만 잘라내고 스크롤바 생성

visible



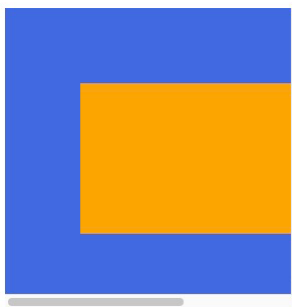
내용 넘침

hidden



잘라냄

scroll



y축

스크롤바
(x축, y축)

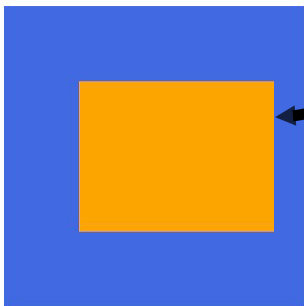
x축



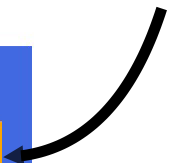
auto



visible



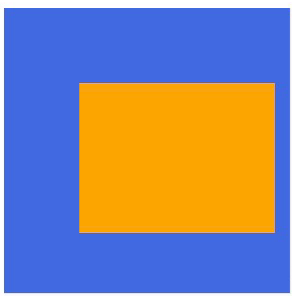
내용 넘치지 않음!



hidden



scroll



y축

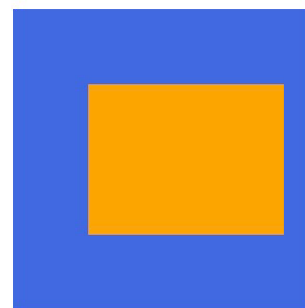


스크롤바
(x축, y축)

x축



auto



overflow: visible;

overflow: hidden;

overflow: clip;

overflow: scroll;

overflow: auto;

Michaelmas term lately over, and the Lord Chancellor sitting in Lincoln's Inn Hall. Implacable November weather. As much mud in the streets as if the waters...

overflow: visible;

overflow: hidden;

overflow: clip;

overflow: scroll;

overflow: auto;

Michaelmas term lately over, and the Lord Chancellor sitting in Lincoln's Inn Hall. Implacable November weather. As much mud in the streets as if the waters...

overflow: visible;

overflow: hidden;

overflow: clip;

overflow: scroll;

overflow: auto;

Michaelmas term lately over, and the Lord Chancellor sitting in Lincoln's Inn Hall. Implacable November weather. As much mud in the streets as if the waters...

overflow: visible;

overflow: hidden;

overflow: clip;

overflow: scroll;

overflow: auto;

Michaelmas term lately over, and the Lord Chancellor sitting in Lincoln's Inn Hall. Implacable November weather. As much mud in the streets as if the waters...

요소의 크기 이상으로 내용이 넘쳤을 때, 보여짐을 제어하는 개별 속성들

overflow-x
overflow-y

요소의 화면 출력(보여짐) 특성

whether an element is treated as a [block or inline box](#) and the layout used for its children, such as [flow layout](#), [grid](#) or [flex](#)

display

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..

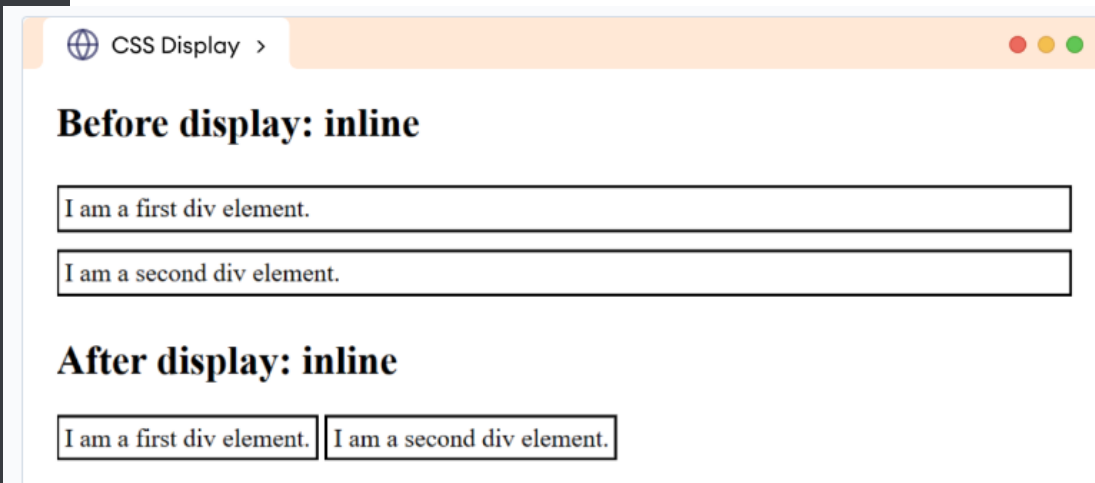
따로 지정해서 사용하는 값



```
div {
  border: 2px solid black;
  margin-bottom: 12px;
  padding: 4px;
}

div.after {
  display: inline;
}

<body>
  <h2>Before display: inline</h2>
  <div>I am a first div element.</div>
  <div>I am a second div element.</div>
  <!-- Adding display: inline in divs -->
  <h2>After display: inline</h2>
  <div class="after">I am a first div element.</div>
  <div class="after">I am a second div element.</div>
</body>
```



```

span {
  width: 60px;
  height: 30px;
  padding: 10px;
  margin: 20px;
  border: 2px solid black;
  background-color: greenyellow;
}

span.after {
  display: inline-block;
}

<body>
  <h2>Before display: inline-block</h2>
  <p>
    In this paragraph, the <span>span</span> element is
    color in green-yellow color. The width and height don't
    work, the padding value doesn't push surrounding
    content away, and the top/bottom margins don't work.
  </p>
  <!-- Adding display: inline-block the divs -->
  <h2>After display: inline-block</h2>
  <p>
    In this paragraph, the <span class="after">span</span>
    element is color in green-yellow color. The width and
    height work, the padding value push surrounding
    content away, and the top/bottom margins work.
  </p>
</body>

```

Before display: inline-block

In this paragraph, the span element is color in green-yellow color. The width and height don't work, the padding value doesn't push surrounding content away, and the top/bottom margins don't work.

After display: inline-block

In this paragraph, the span element is color in green-yellow color. The width and height work, the padding value push surrounding content away, and the top/bottom margins work.

요소 투명도

opacity

1

불투명

0~1

0부터 1 사이의 소수점 숫자

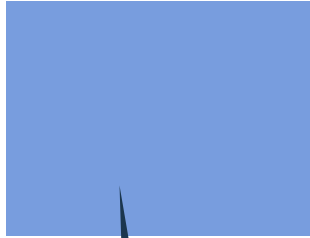
opacity: 0.07;



opacity: 0.4;



opacity: 0.7;



opacity: 1;



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,
7단계의 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

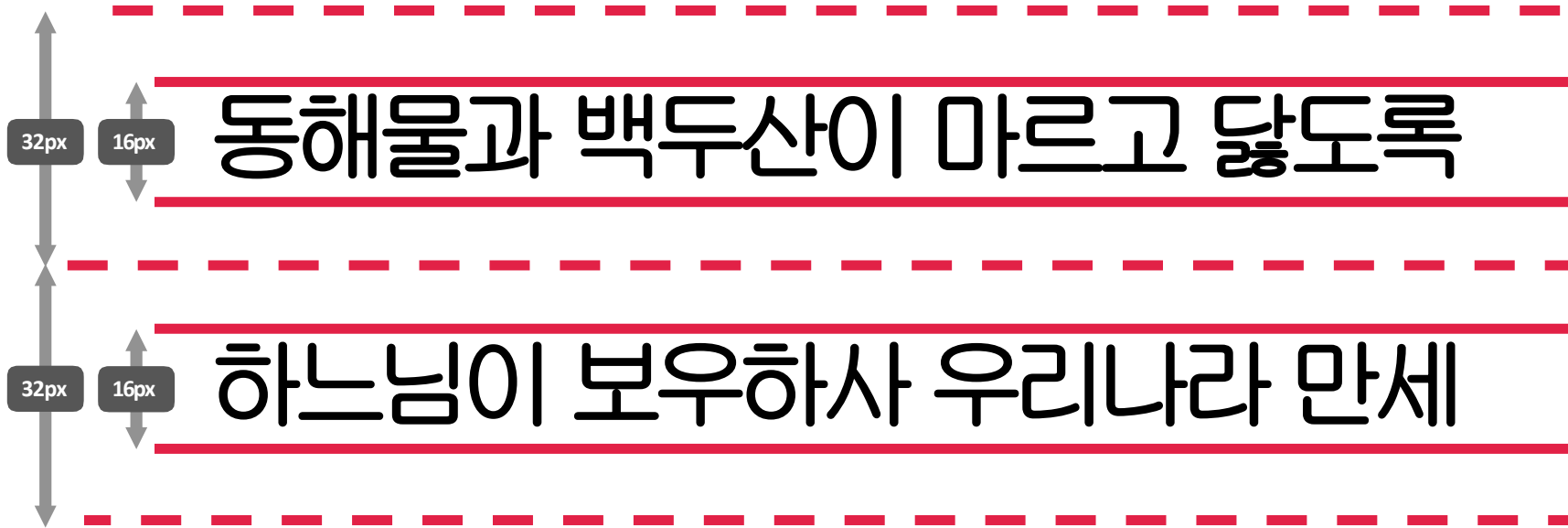
%

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

2배 차이

필수로 작성!

글꼴(서체) 지정

font-family: 글꼴1, "글꼴2", ... 글꼴계열;

띄어쓰기 등 특수문자가 포함된
글꼴 이름은 큰 따옴표로 묶어야 합니다~

Hello World!

serif

바탕체 계열

Hello World!

sans-serif

고딕체 계열

He l l o W o r l d !

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

none

장식 없음

동해물과 백두산이 마르고 닳도록

underline

밑줄

동해물과 백두산이 마르고 닳도록

overline

윗줄

~~동해물과 백두산이 마르고 닳도록~~

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)

문자 첫 줄의 들여쓰기

text-indent

음수를 사용할 수 있어요!
반대는 내어쓰기(outdent)입니다.

0

들여쓰기 없음

단위

px, em, rem 등 단위로 지정

%

요소의 가로 너비에 대한 비율

배경

요소의 배경 색상

background-color

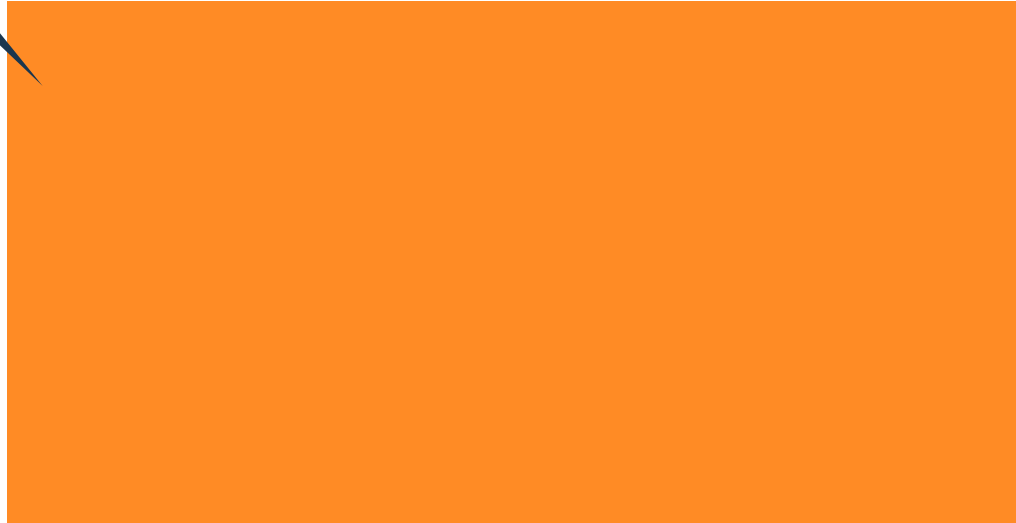
transparent

투명함

색상

지정 가능한 색상

`background-color: orange;`



요소의 배경 이미지 삽입

background-image

none

이미지 없음

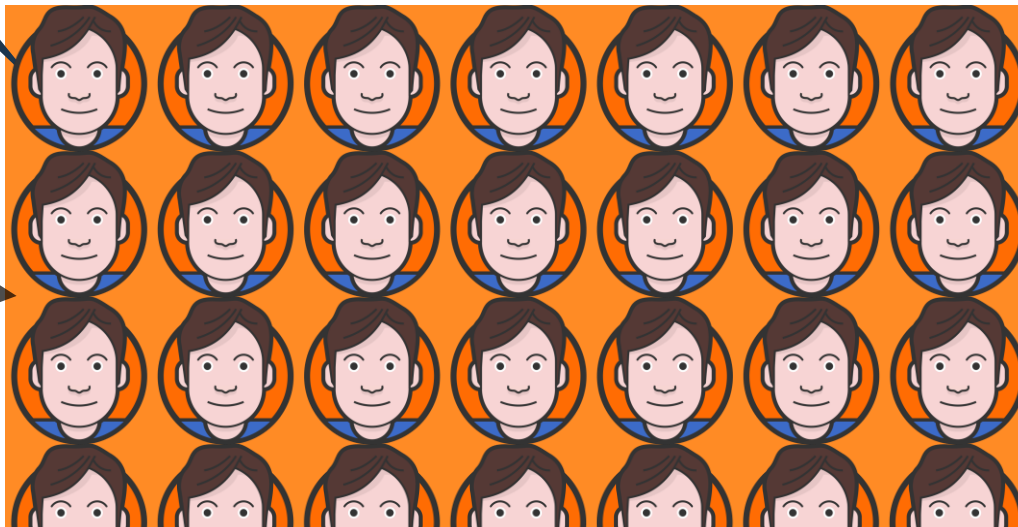
url("경로")

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;
```

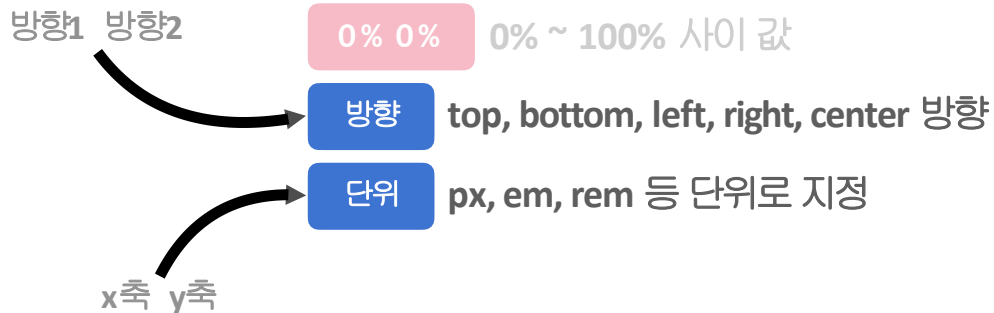


```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;
```



요소의 배경 이미지 위치

background-position

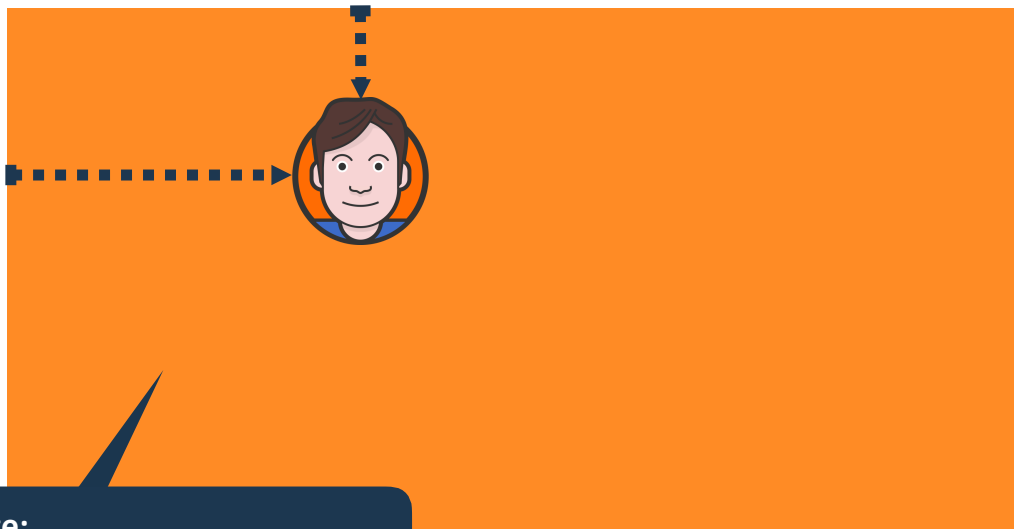




```
background-color: orange;  
background-image: url("./images/heropy.png");  
background-repeat: no-repeat;  
background-position: top right;
```




```
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 맞춤

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

120

```
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!

요소의 위치 지정 기준

top
bottom
left
right
z-index

position

static

기준 없음

relative

요소 자신을 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

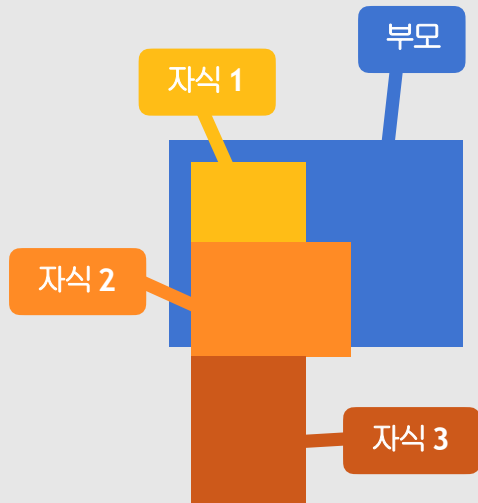
top, bottom, left, right

auto

브라우저가 계산

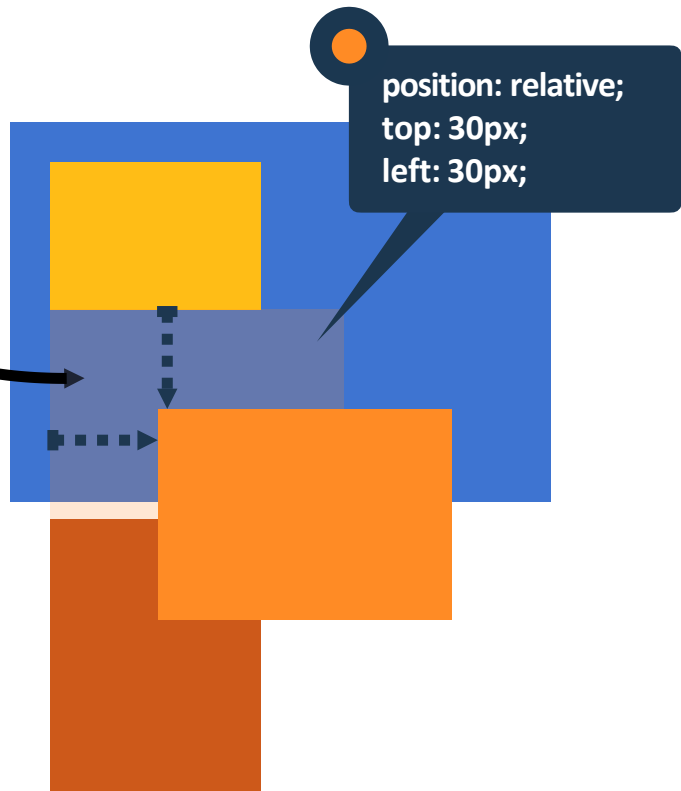
단위

px, em, rem 등 단위로 지정



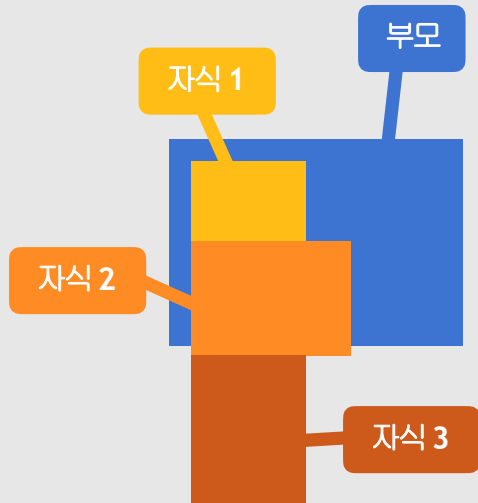
HTML 구조

배치 전 자리는
비어 있어요!

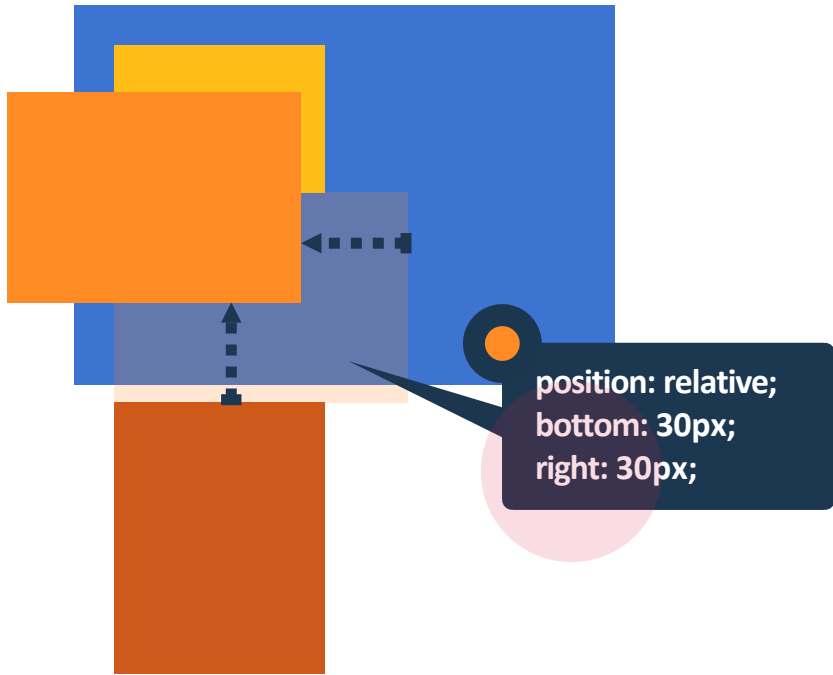


relative

요소 자신을 기준으로 배치!

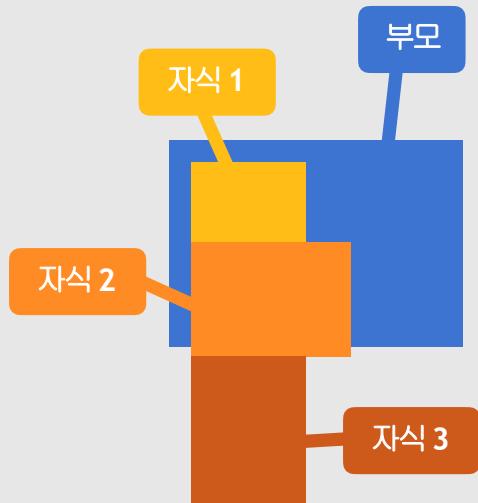


HTML 구조



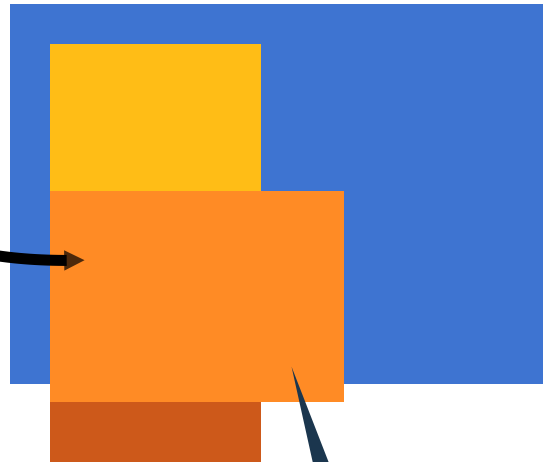
relative

요소 자신을 기준으로 배치!



HTML 구조

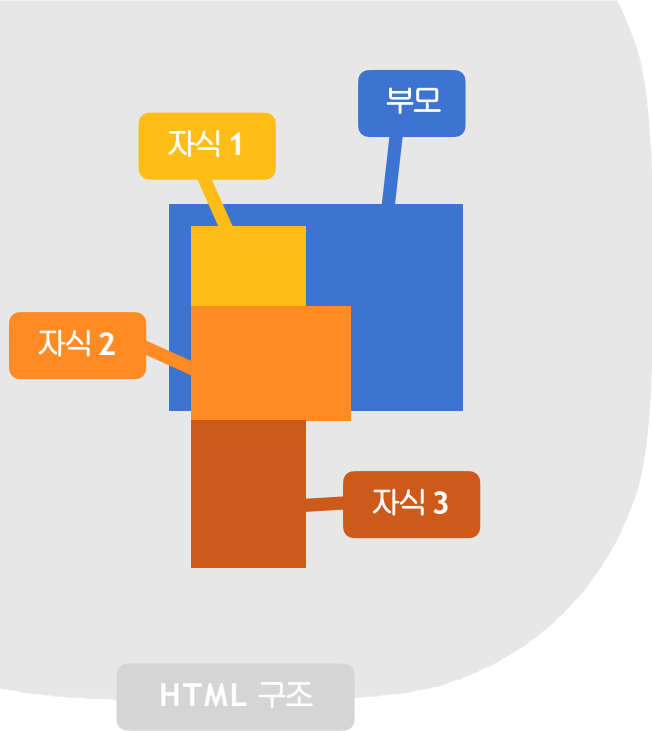
붕~ 뜨면서요
소가 겹쳐요



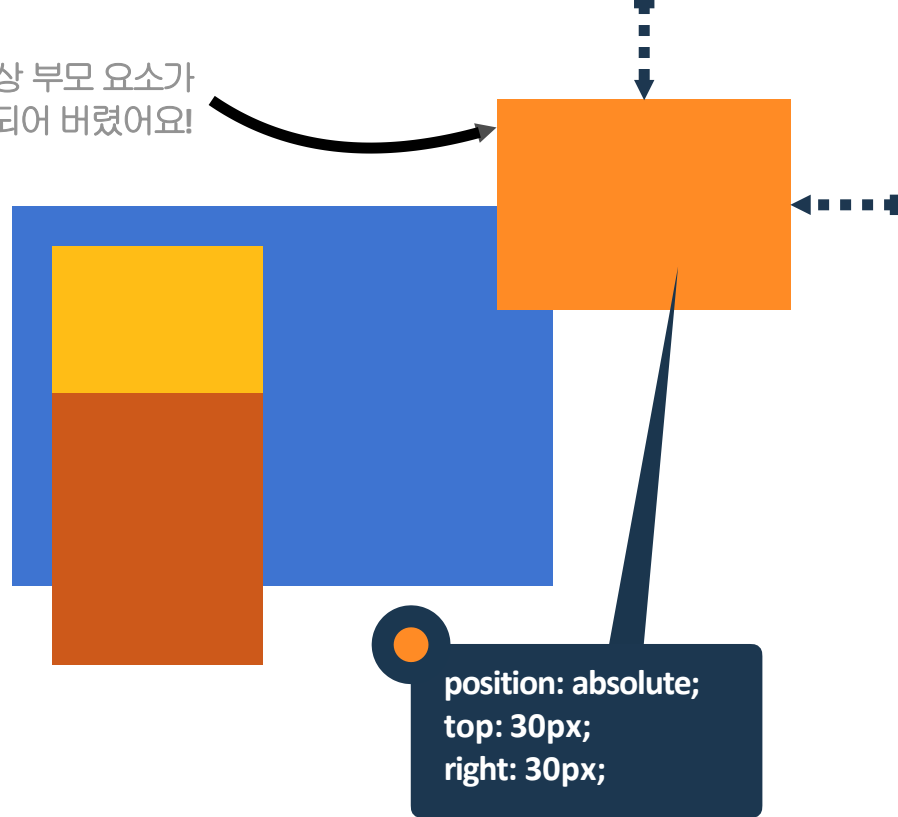
`position: absolute;`

absolute

위치 상 부모 요소를 기준으로 배치!

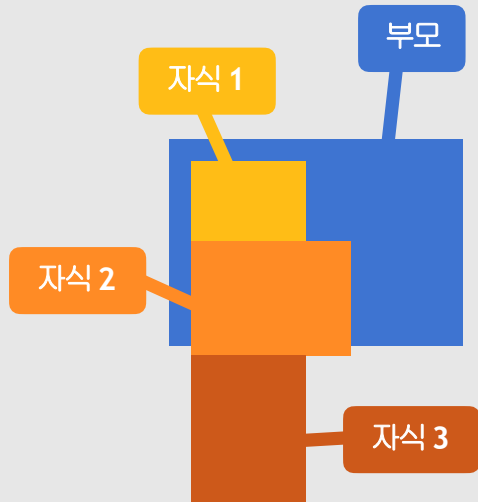


위치 상 부모 요소가
뷰포트가 되어 버렸어요!



absolute

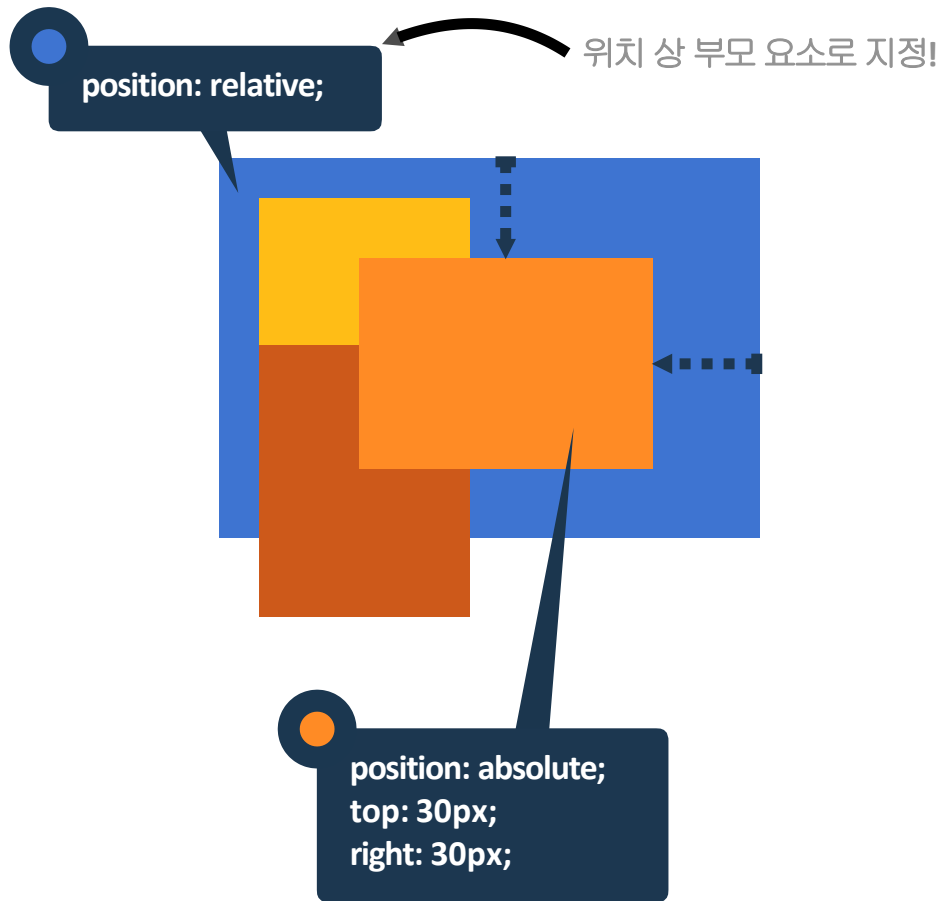
위치 상 부모 요소를 기준으로 배치!

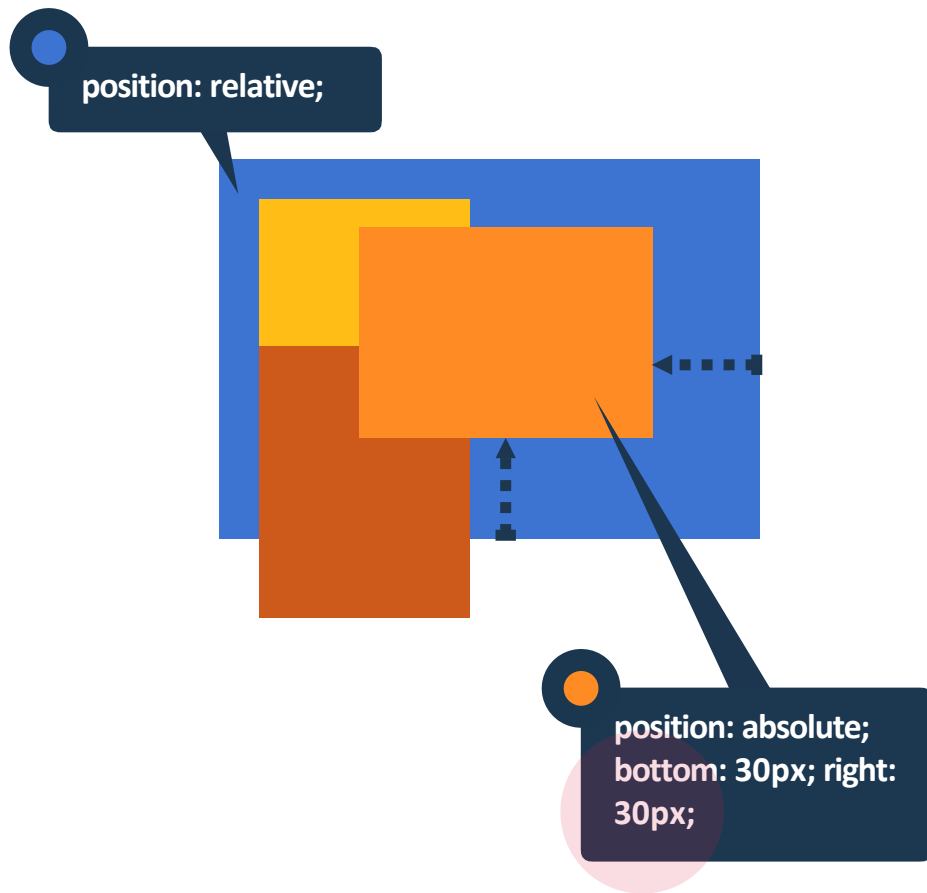
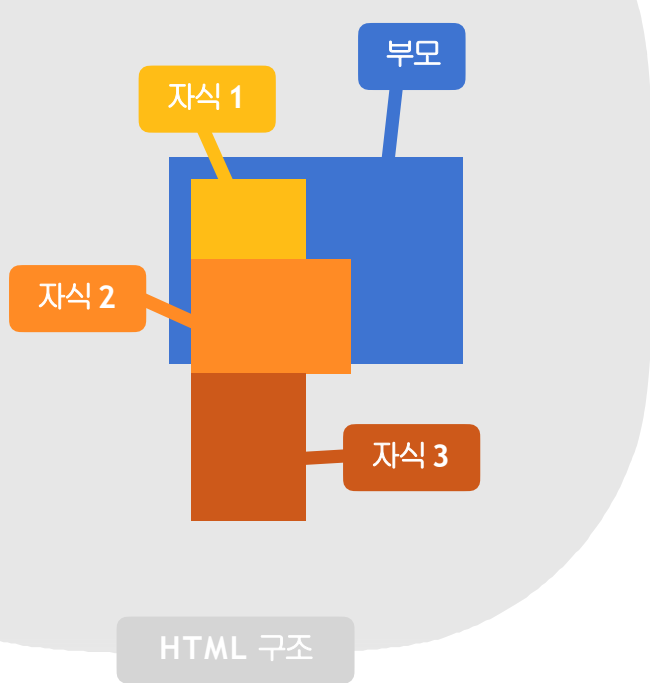


HTML 구조

absolute

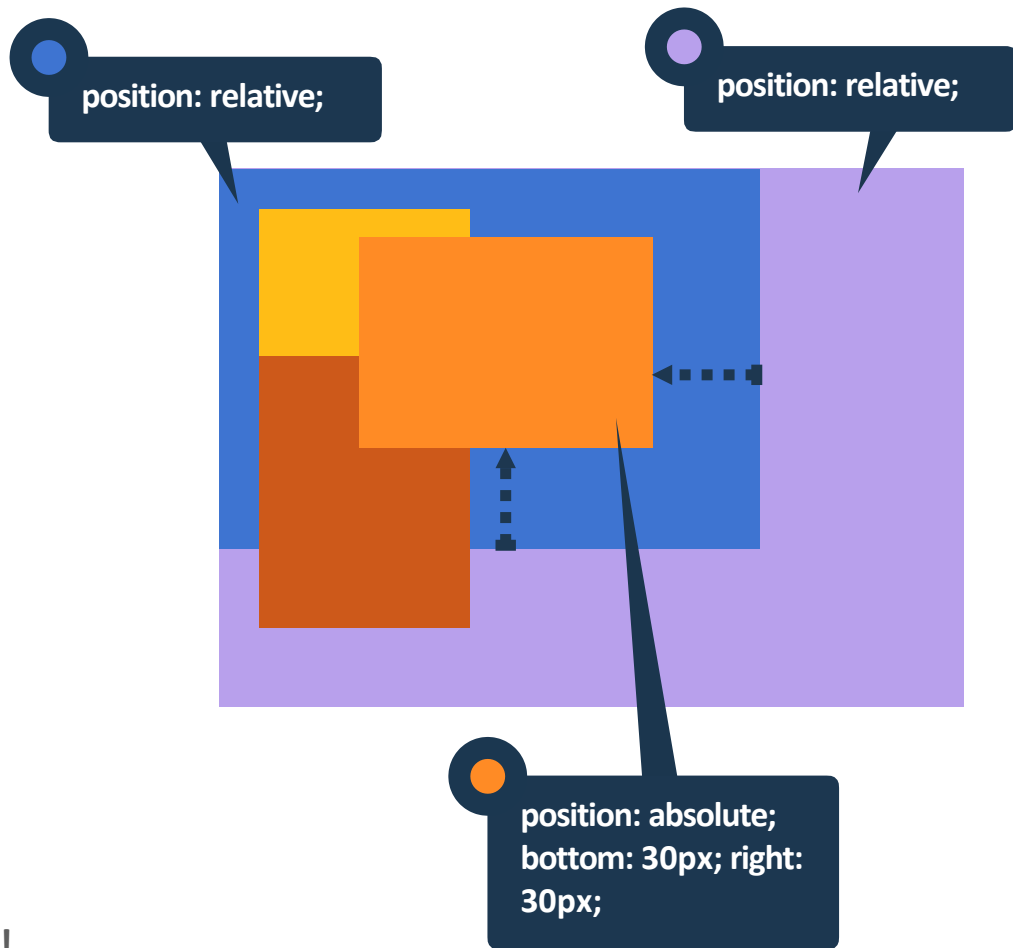
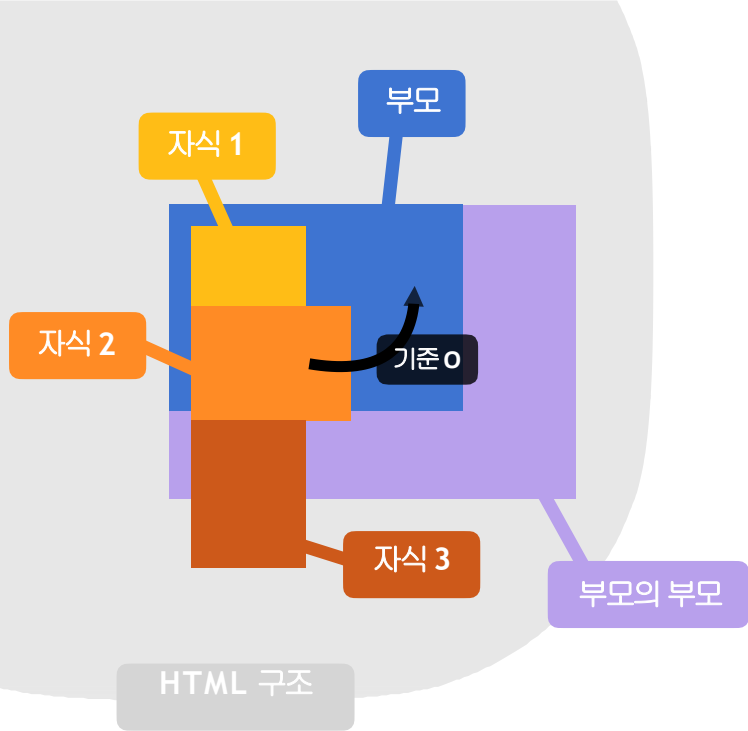
위치 상 부모 요소를 기준으로 배치!





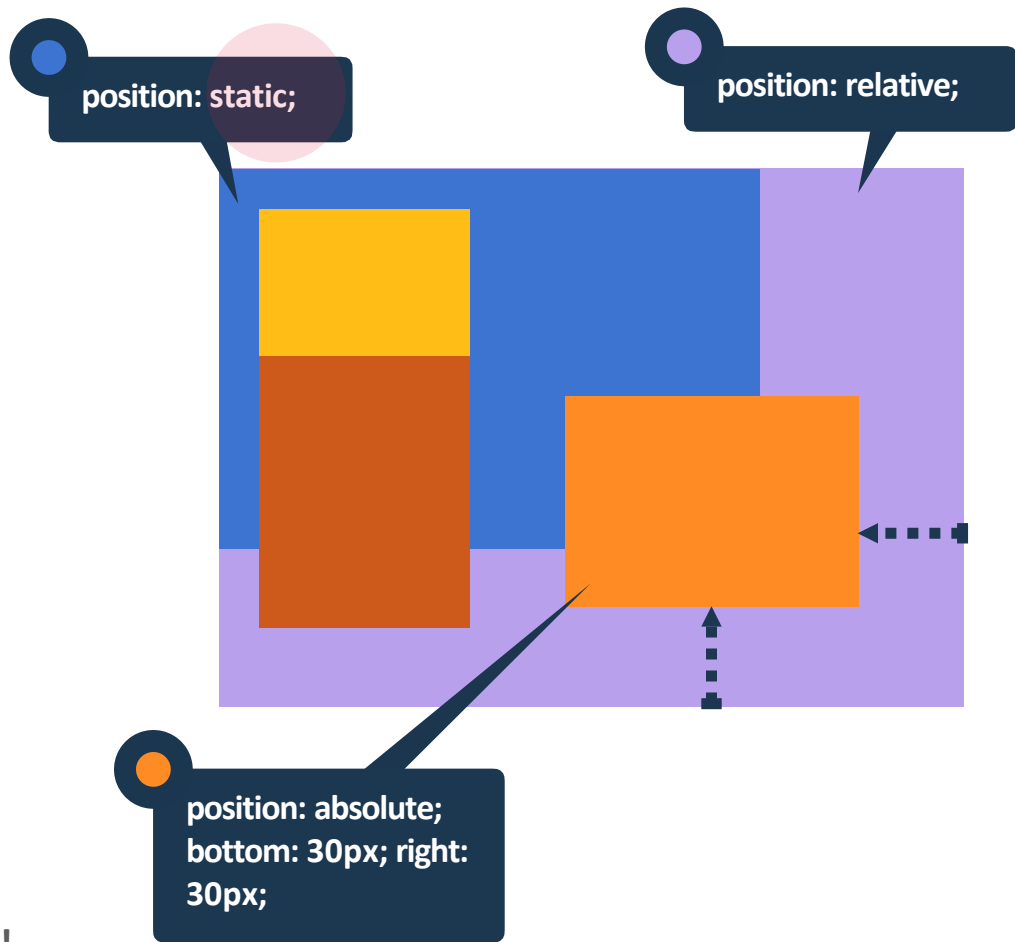
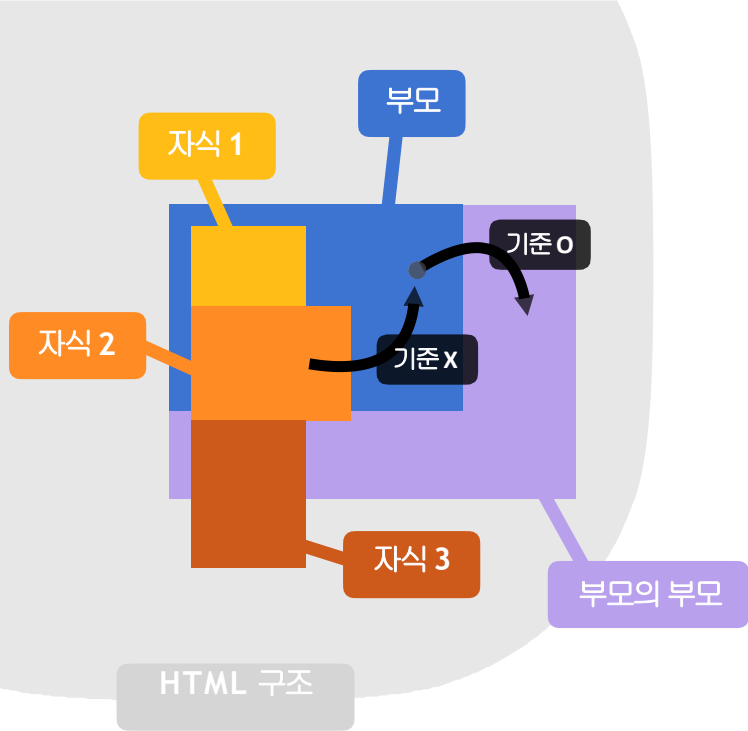
absolute

위치 상 부모 요소를 기준으로 배치!



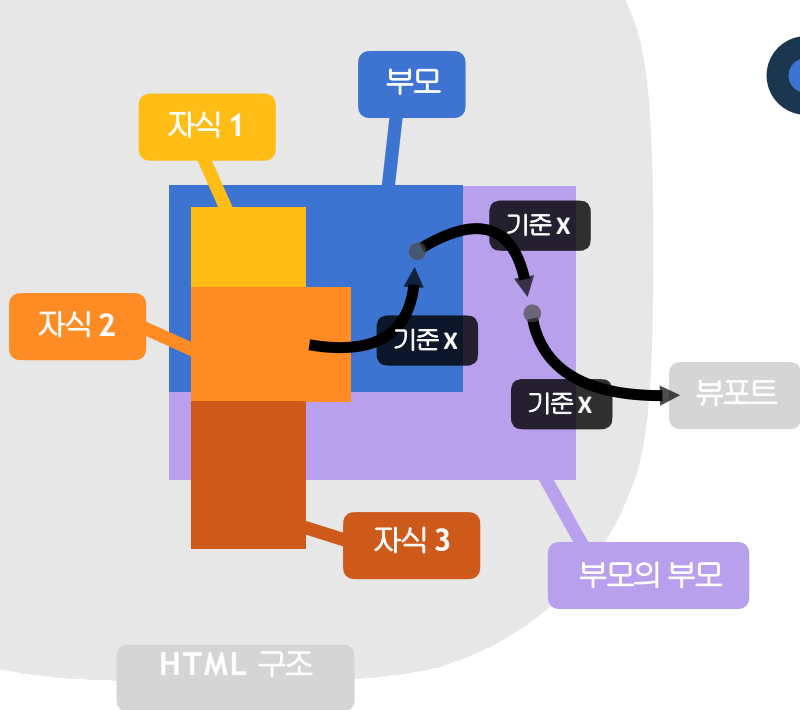
absolute

위치 상 부모 요소를 기준으로 배치!



absolute

위치 상 부모 요소를 기준으로 배치!



position: static;

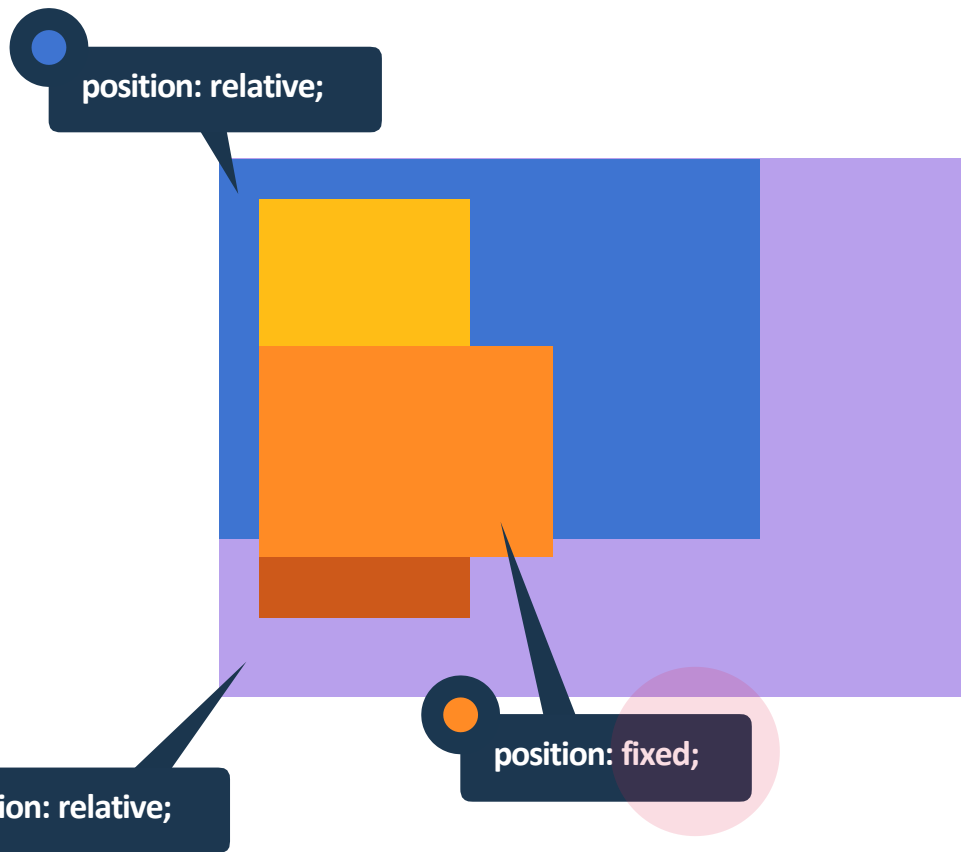
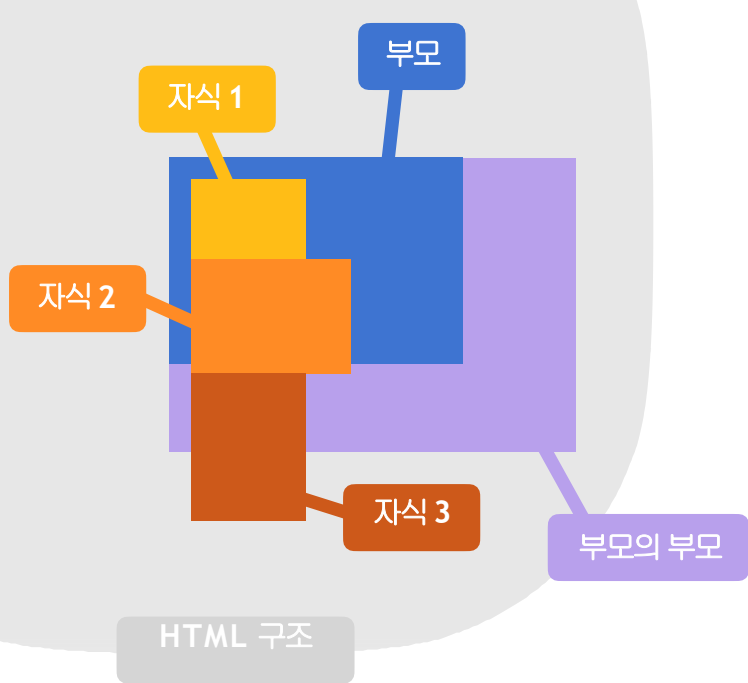
position: static;



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

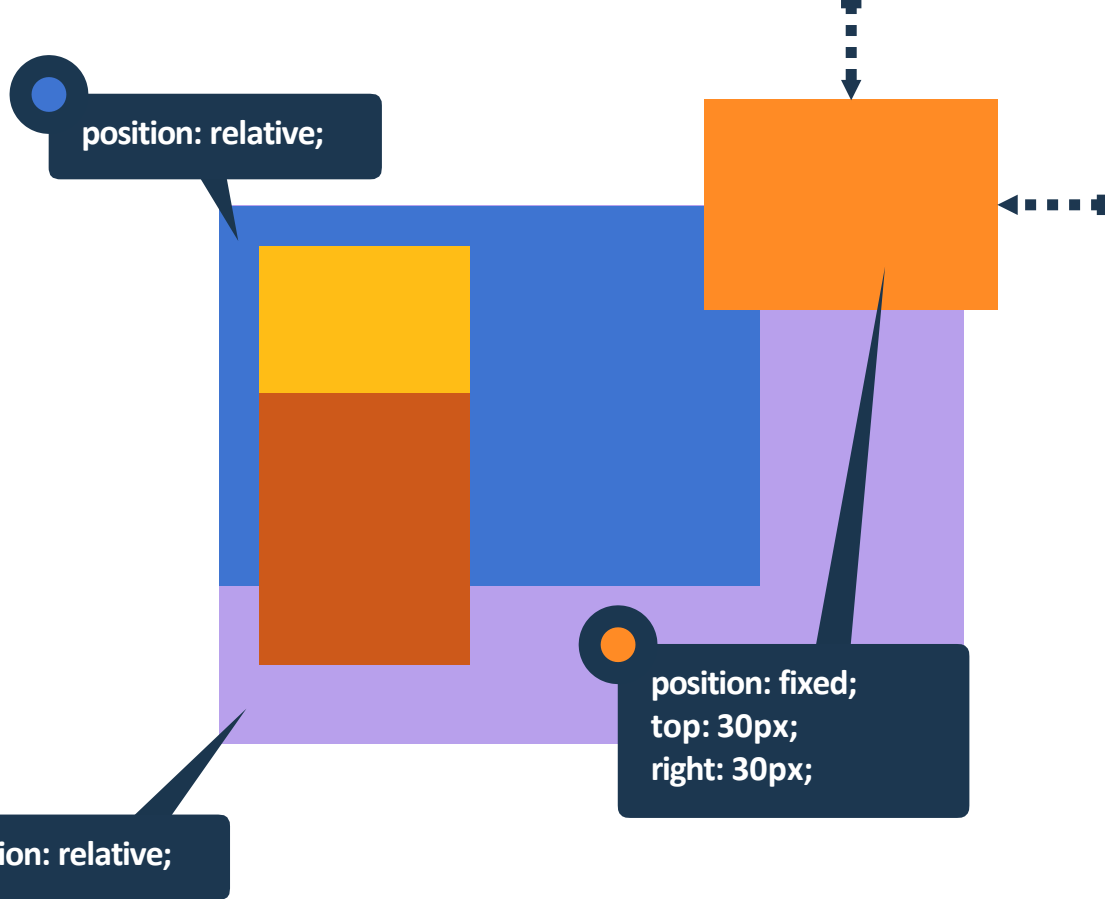
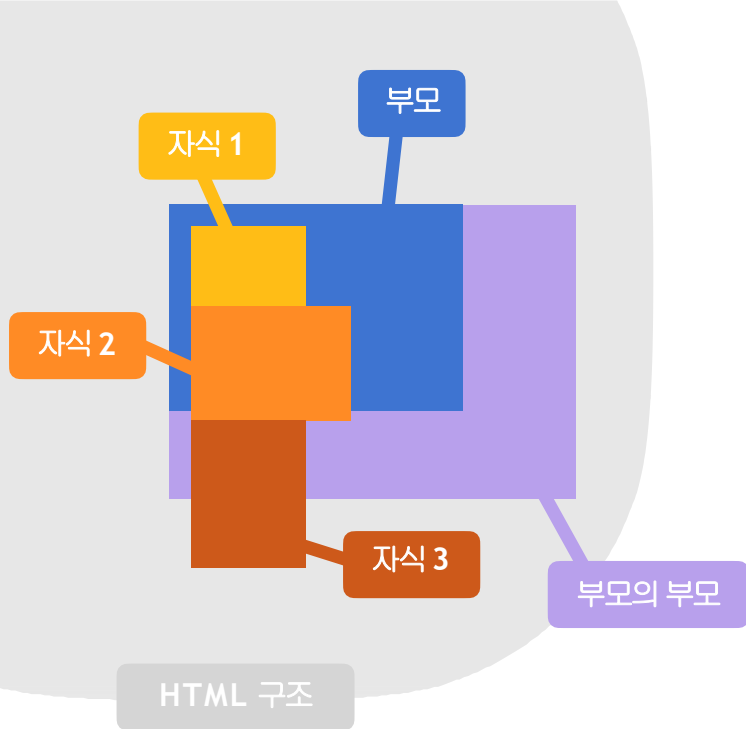


absolute 위치 상 부모 요소를 기준으로 배치!



fixed

뷰포트(브라우저)를 기준으로 배치!



fixed

뷰포트(브라우저)를 기준으로 배치!

CSS Demo: position

RESET

position: static;

position: relative;

top: 40px; left: 40px;

position: absolute;

top: 40px; left: 40px;

position: sticky;

top: 20px;

In this demo you can control the position property for the yellow box.

To see the effect of sticky positioning, select the position: sticky option and scroll this container.

The element will scroll along with its container, until it is at the top of the container (or reaches the offset specified in top), and will then stop scrolling, so it stays visible.

CSS Demo: position

RESET

position: static;

position: relative;

top: 40px; left: 40px;

position: absolute;

top: 40px; left: 40px;

position: sticky;

top: 20px;

In this demo you can control the position property for the yellow box.

To see the effect of sticky positioning, select the position: sticky option and scroll this container.

The element will scroll along with its container, until it is at the top of the container (or reaches the offset specified in top), and will then stop scrolling, so it stays visible.

CSS Demo: position

RESET

position: static;

position: relative;

top: 40px; left: 40px;

position: absolute;

top: 40px; left: 40px;

position: sticky;

top: 20px;

In this demo you can control the position property for the yellow box.

To see the effect of sticky positioning, select the position: sticky option and scroll this container.

The element will scroll along with its container, until it is at the top of the container (or reaches the offset specified in top), and will then stop scrolling, so it stays visible.

136

요소 쌓임 순서(Stack order)

어떤 요소가 사용자와 더 가깝게 있는지(위에 쌓이는지) 결정

- 1 요소에 **position** 속성의 값이 있는 경우 위에 쌓임.(기본값 **static** 제외)
- 2 1번 조건이 같은 경우, **z-index** 속성의 숫자 값이 높을 수록 위에 쌓임.
- 3 1번과 2번 조건까지 같은 경우, **HTML**의 다음 구조일 수록 위에 쌓임.

요소의 쌓임 정도를 지정

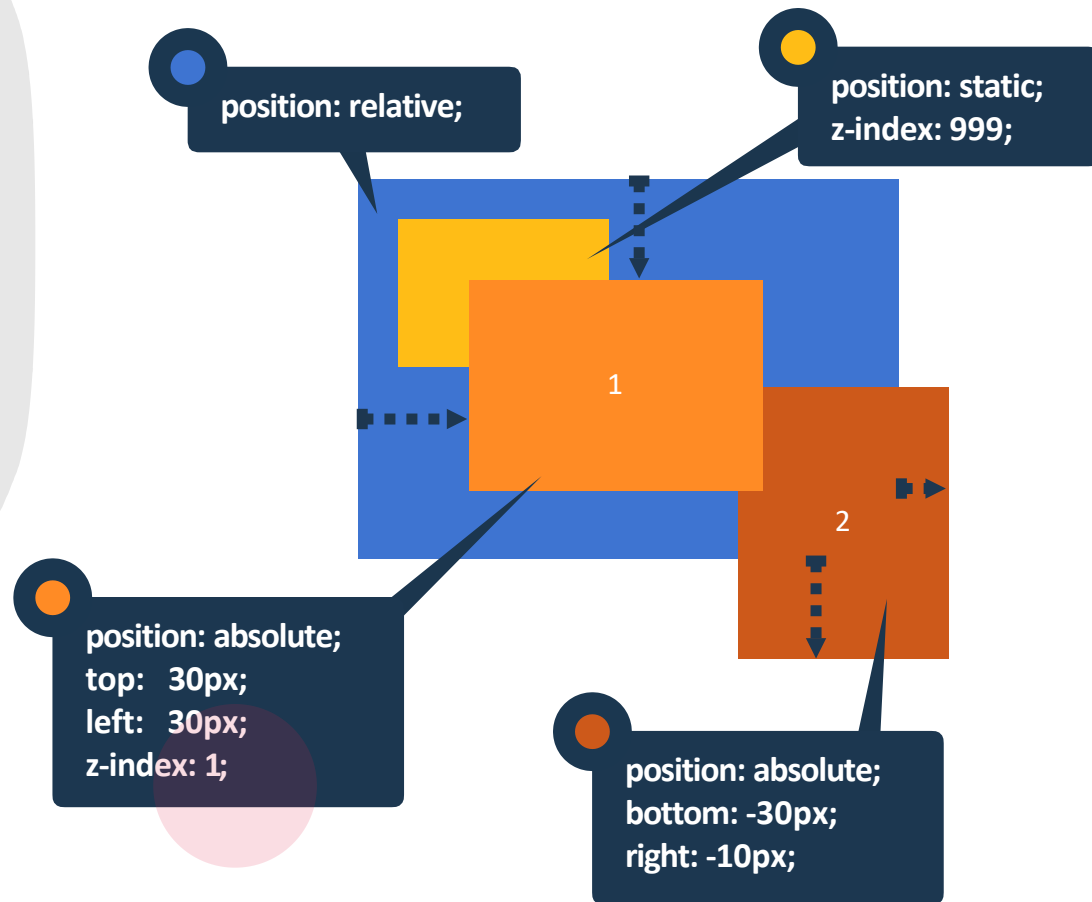
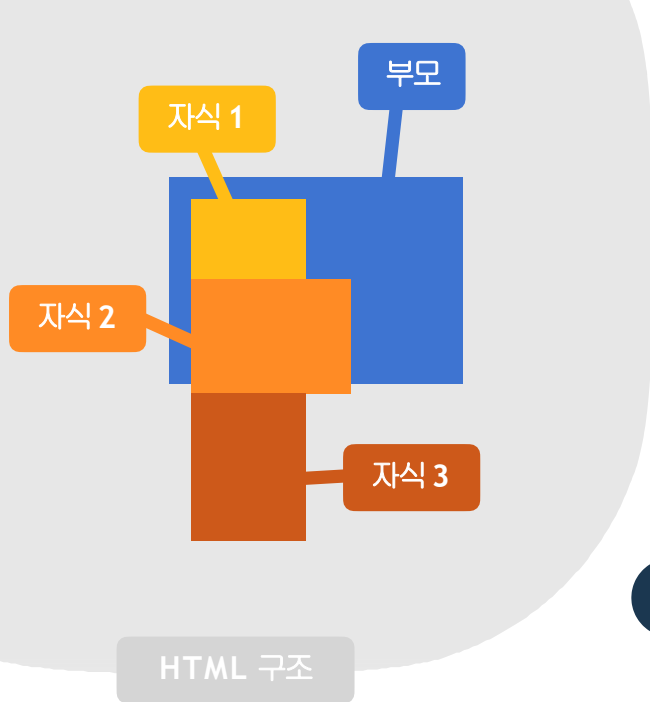
z-index

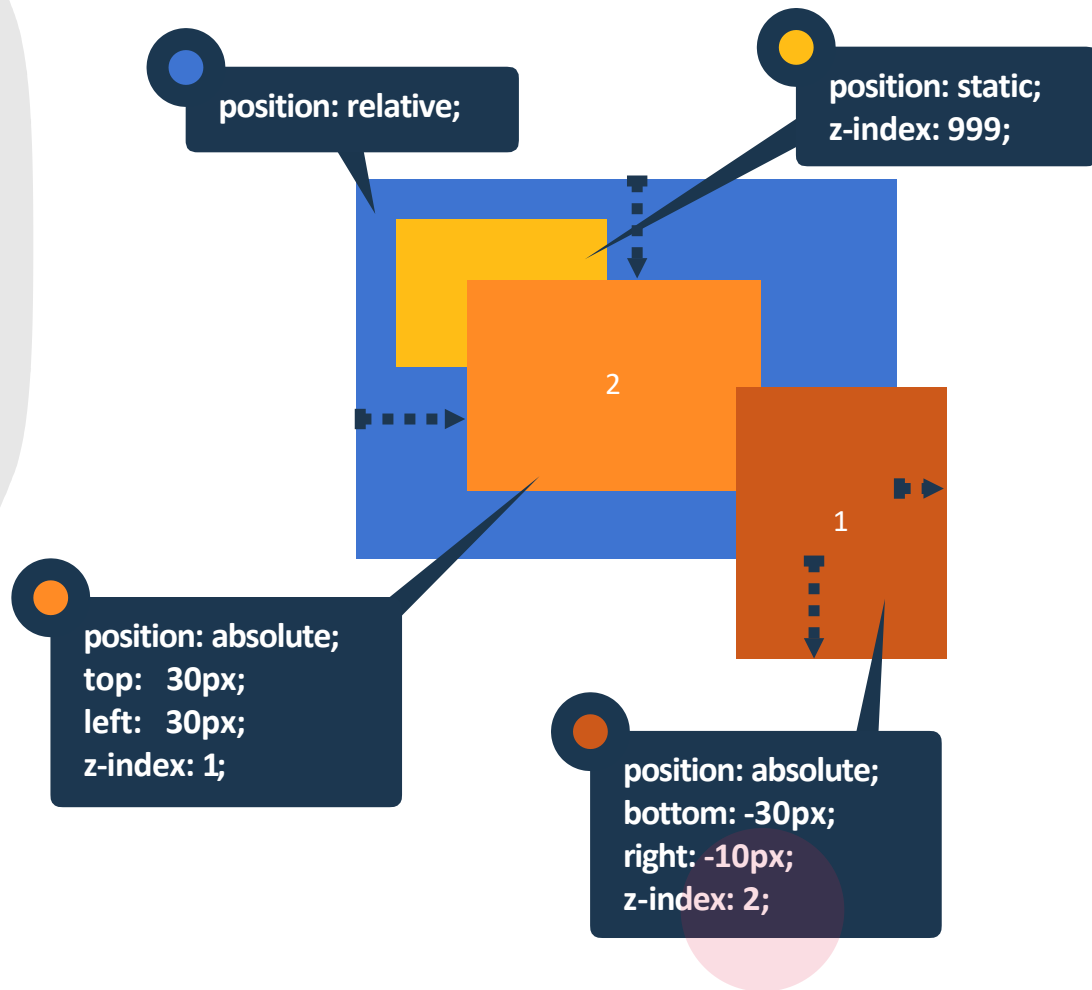
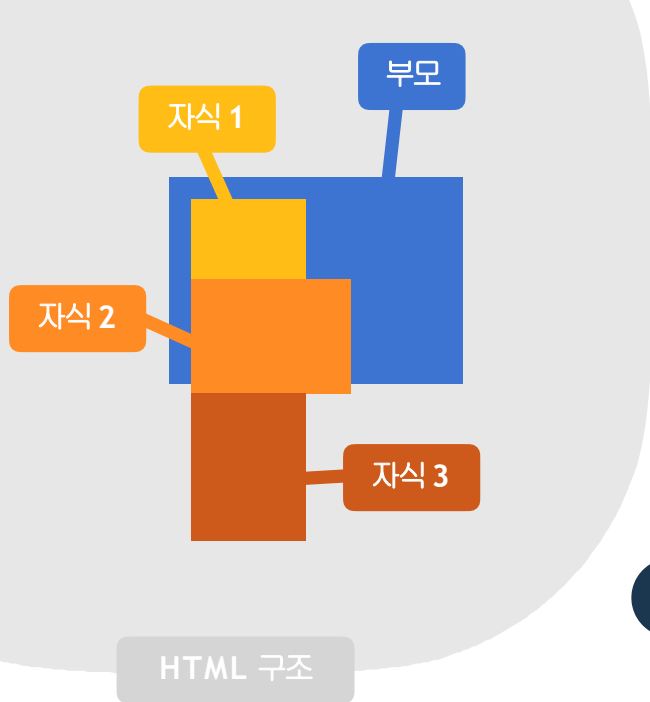
auto

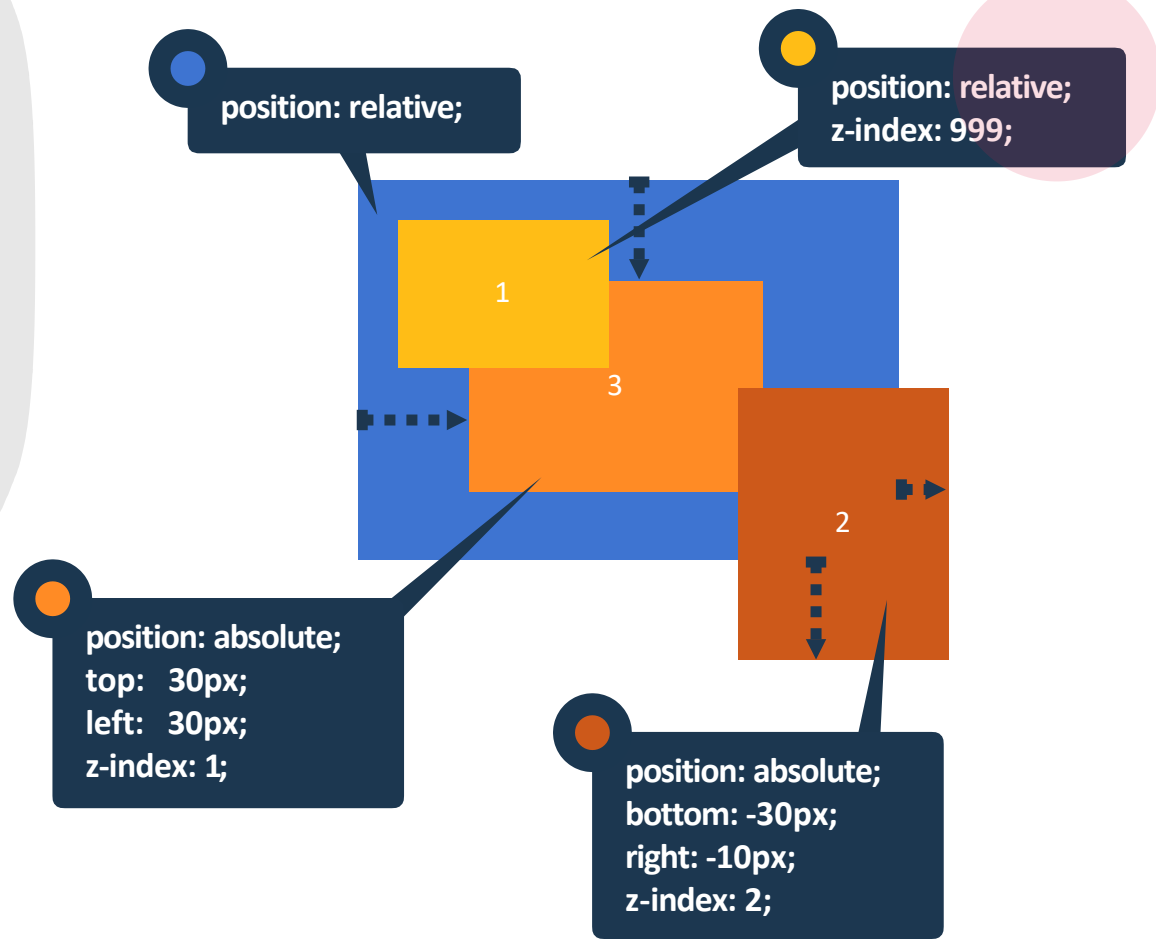
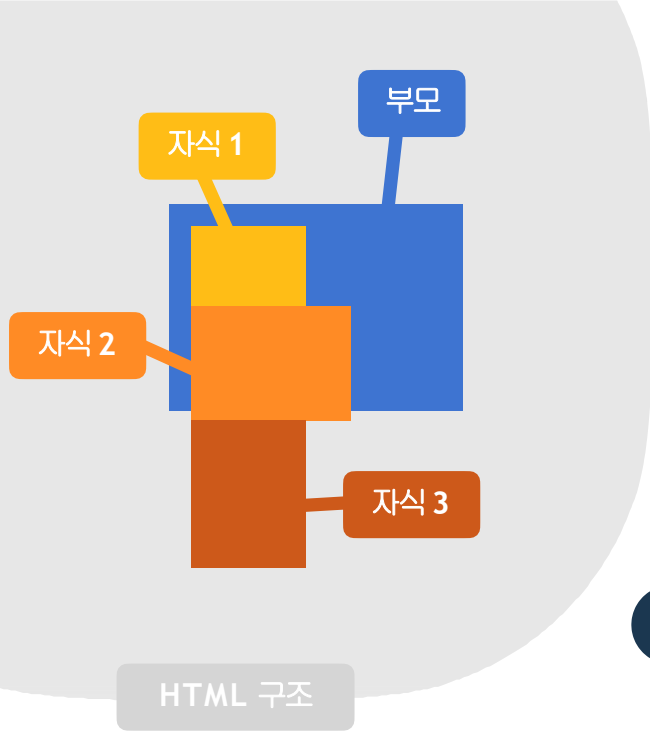
부모 요소와 동일한 쌓임 정도

숫자

숫자가 높을 수록 위에 쌓임

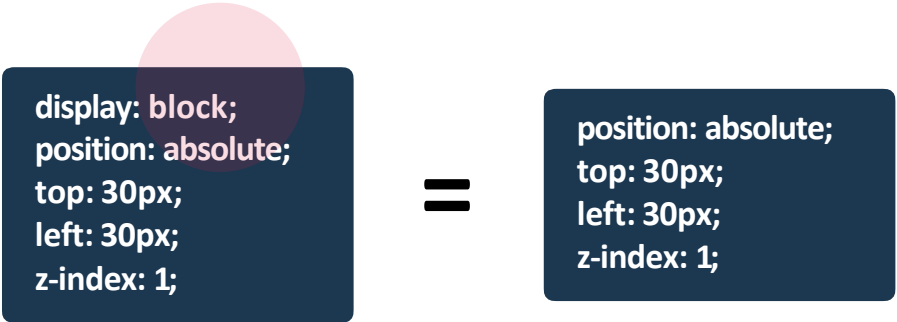






요소의 display가 변경됨

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



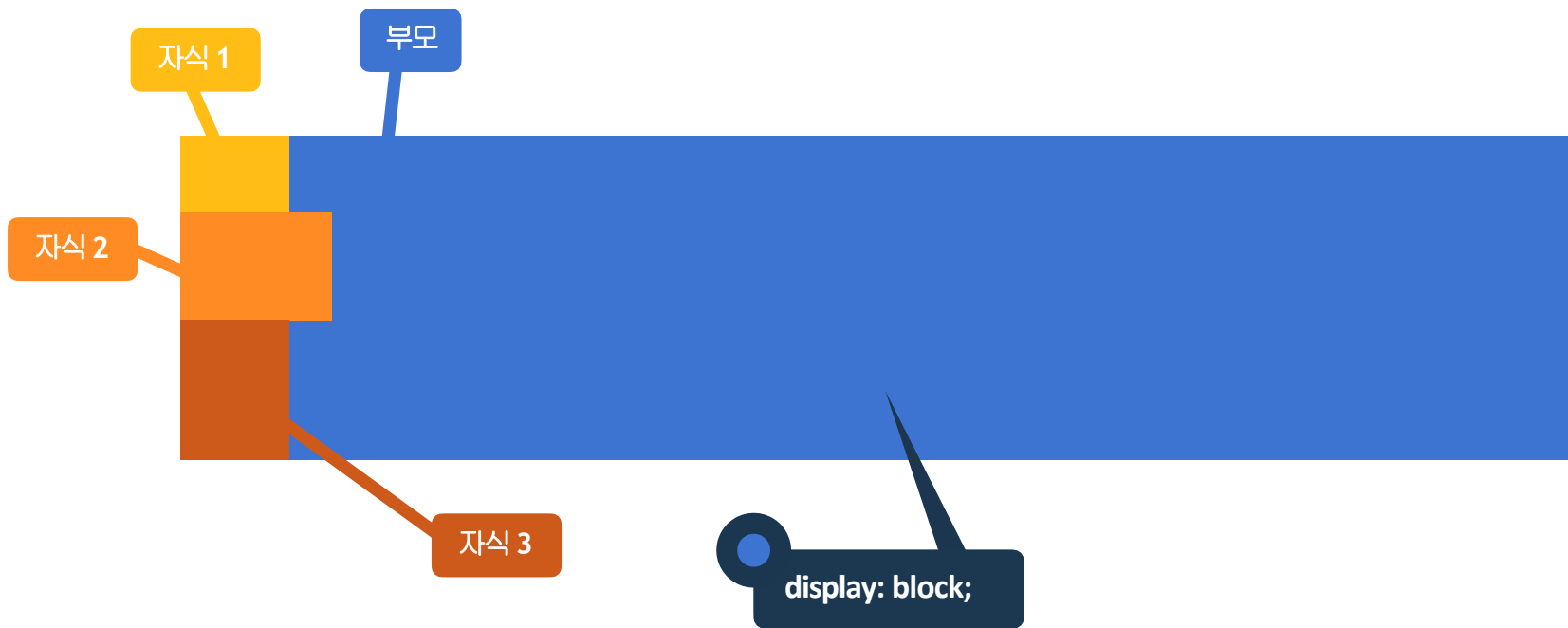
A diagram illustrating the equivalence between two CSS property sets. On the left, a dark blue rounded rectangle contains the text: `display: block;`, `position: absolute;`, `top: 30px;`, `left: 30px;`, and `z-index: 1;`. A light pink circle is partially visible behind the top of this rectangle. In the center, a black equals sign (=) is positioned. To the right, another dark blue rounded rectangle contains the text: `position: absolute;`, `top: 30px;`, `left: 30px;`, and `z-index: 1;`.

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

=

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

플렉스(정렬)



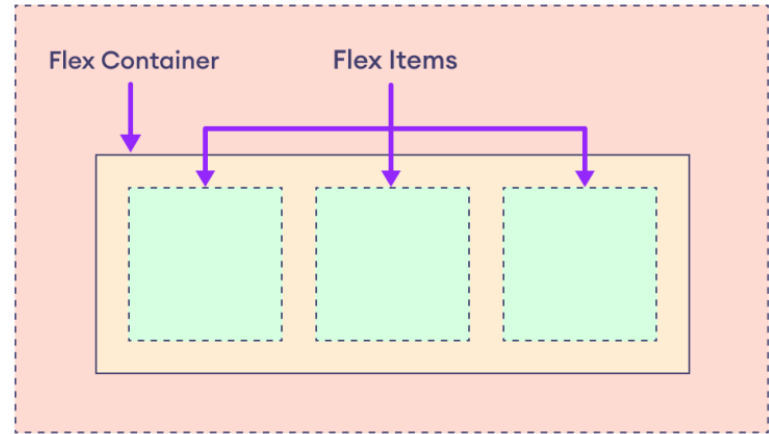
Flex Container Flex Item

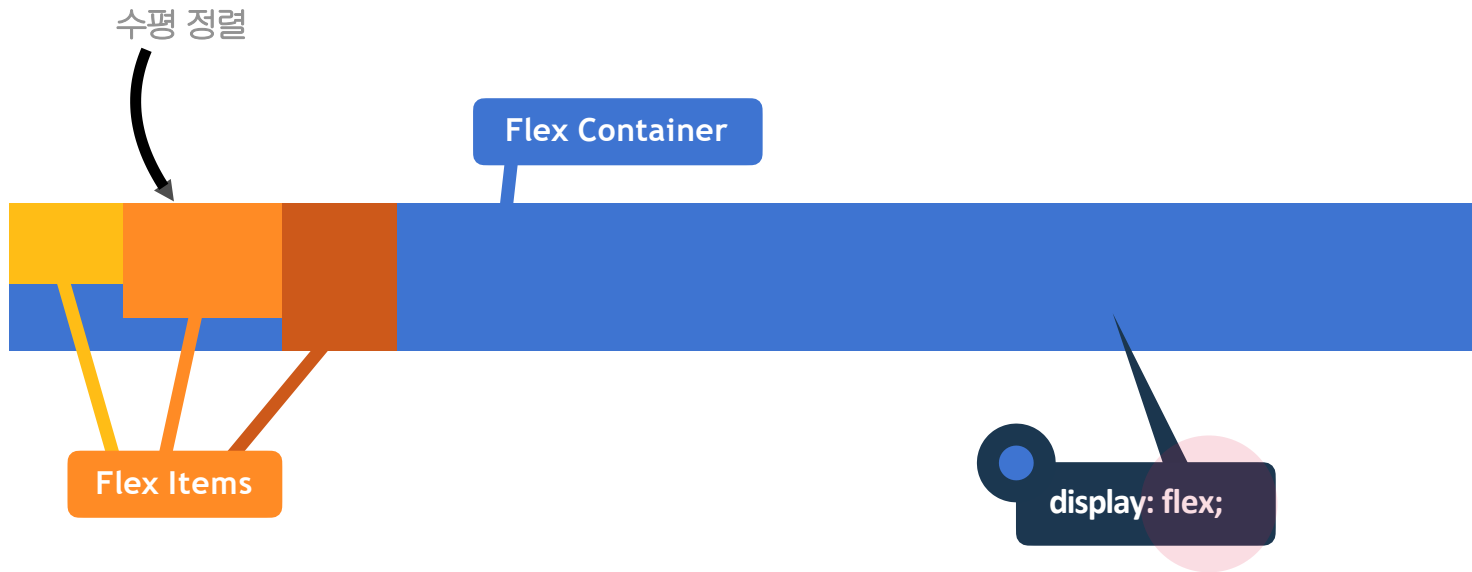
In order to create a flex layout, we first need to set up a flex container. Any element can be set as a flex container using

display: flex declaration

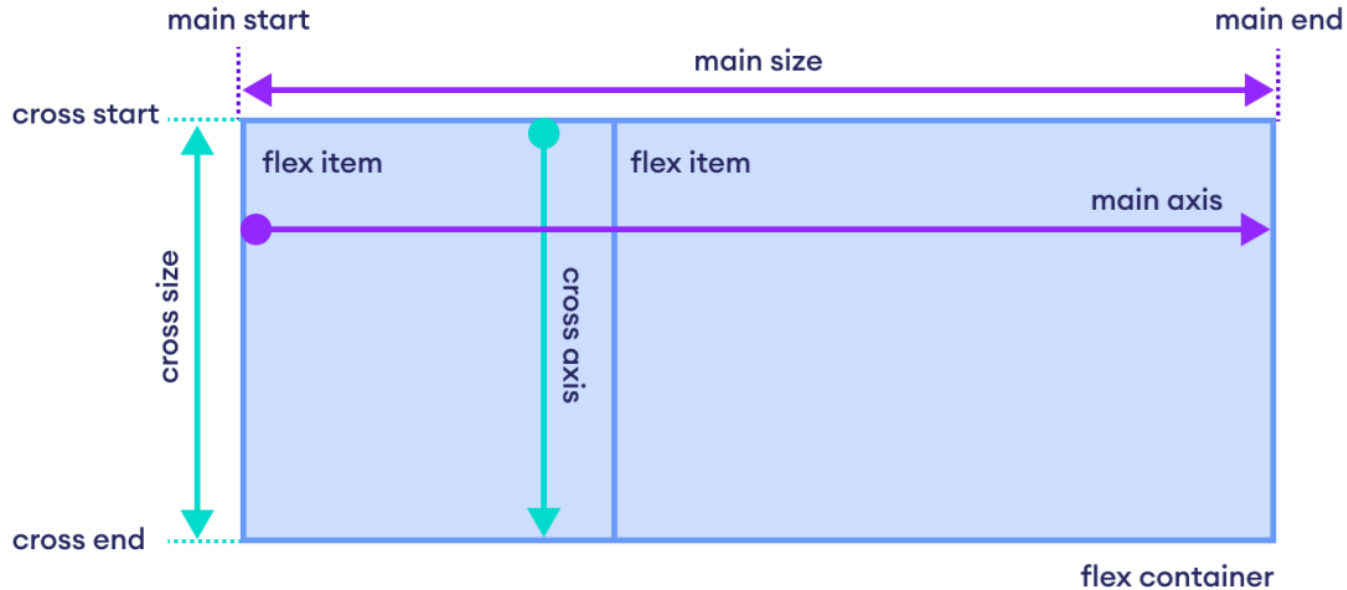
The elements within a flex container are referred to as flex items.

The available space within the container can be distributed efficiently among the flex items.





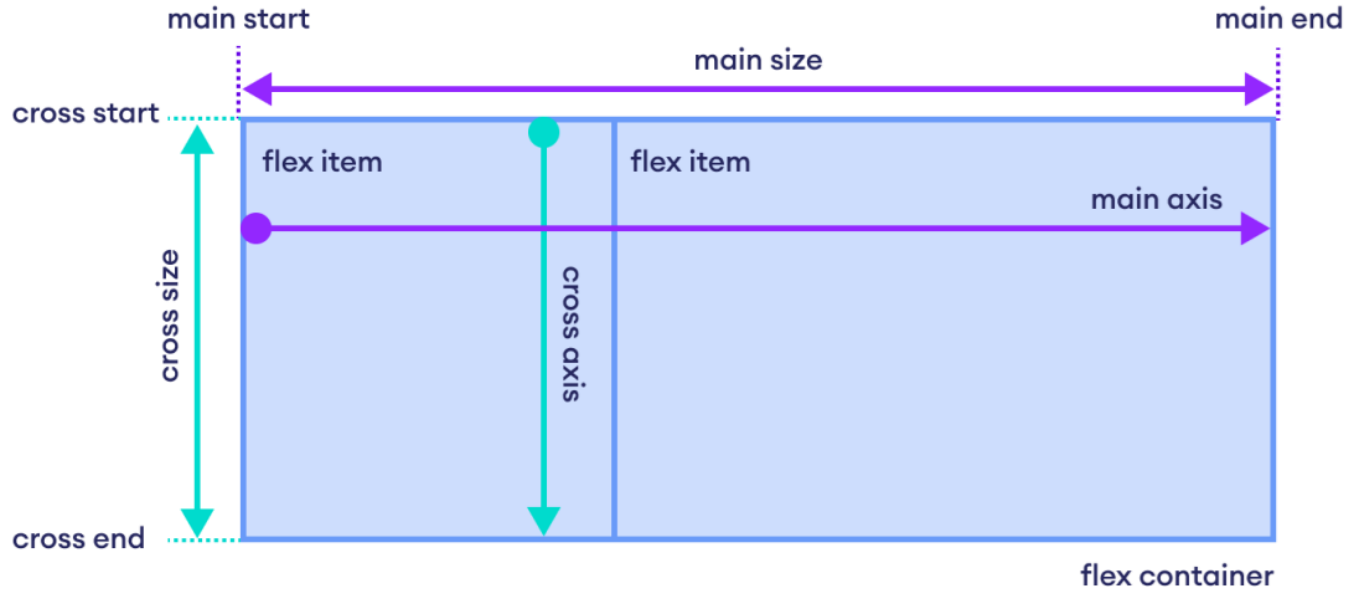
`flex-direction: row;`



main axis: The main axis refers to the primary axis along which the flex items are arranged within the flex container. It can be in either horizontal or vertical direction.

The main axis depends upon the value of the flex-direction property.

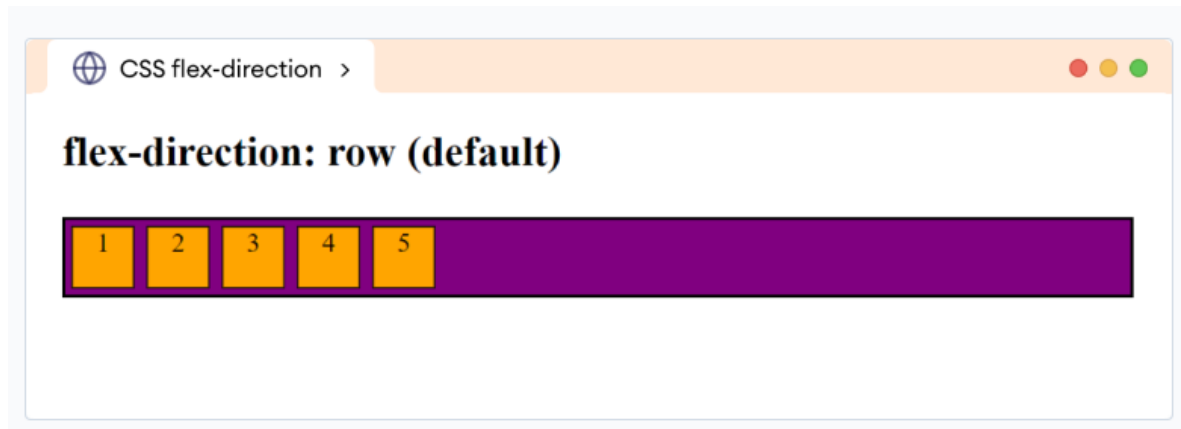
`flex-direction: row;`



main start and main end, main size

cross axis, cross start and end, cross size

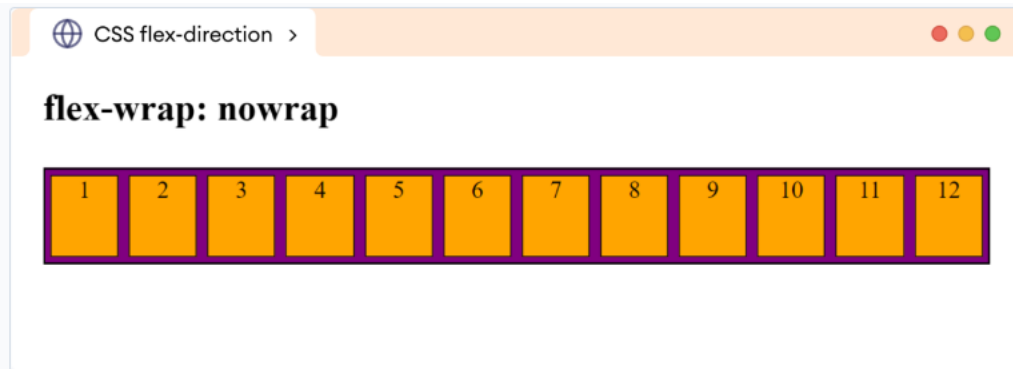
```
div.container {  
  display: flex;  
  flex-direction: row;  
  
  border: 2px solid black;  
  background-color: purple;  
}  
  
div.box {  
  width: 40px;  
  height: 40px;  
  text-align: center;  
  border: 1px solid black;  
  background-color: orange;  
  margin: 4px;  
}  
  
<body>  
  <h2>flex-direction: row (default)</h2>  
  <div class="container">  
    <div class="box">1</div>  
    <div class="box">2</div>  
    <div class="box">3</div>  
    <div class="box">4</div>  
    <div class="box">5</div>  
  </div>  
</body>
```



```
div.container {  
  display: flex;  
  flex-direction: row;  
  
  /* prevents wrapping of flex items; default value */  
  flex-wrap: nowrap;  
  
  border: 2px solid black;  
  background-color: purple;  
}  
  
div.box {  
  width: 80px;  
  height: 60px;  
  text-align: center;  
  border: 1px solid black;  
  background-color: orange;  
  margin: 4px;  
}
```

```
<body>  
  <h2>flex-wrap: nowrap</h2>  
  <div class="container">  
    <div class="box">1</div>  
    <div class="box">2</div>  
    <div class="box">3</div>  
    <div class="box">4</div>  
    <div class="box">5</div>  
    <div class="box">6</div>  
    <div class="box">7</div>  
    <div class="box">8</div>  
    <div class="box">9</div>  
    <div class="box">10</div>  
    <div class="box">11</div>  
    <div class="box">12</div>  
  </div>>
```

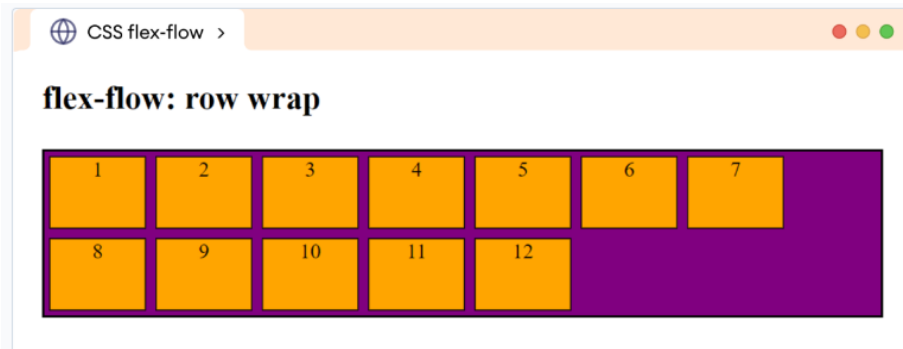
Here, all the flex items are squeezed in the horizontal direction, and their original width is ignored



```
div.container {  
  display: flex;  
  
  flex-flow: row wrap;  
  
  border: 2px solid black;  
  background-color: purple;  
}  
  
div.box {  
  width: 80px;  
  height: 60px;  
  text-align: center;  
  border: 1px solid black;  
  background-color: orange;  
  margin: 4px;  
}
```

```
<body>  
  <h2>flex-flow: row wrap</h2>  
  <div class="container">  
    <div class="box">1</div>  
    <div class="box">2</div>  
    <div class="box">3</div>  
    <div class="box">4</div>  
    <div class="box">5</div>  
    <div class="box">6</div>  
    <div class="box">7</div>  
    <div class="box">8</div>  
    <div class="box">9</div>  
    <div class="box">10</div>  
    <div class="box">11</div>  
    <div class="box">12</div>  
  </div>  
</body>
```

Here, all the flex items are squeezed in the horizontal direction, and their original width is ignored



```

div.container {
  display: flex;

  /* default value */
  justify-content: flex-start;

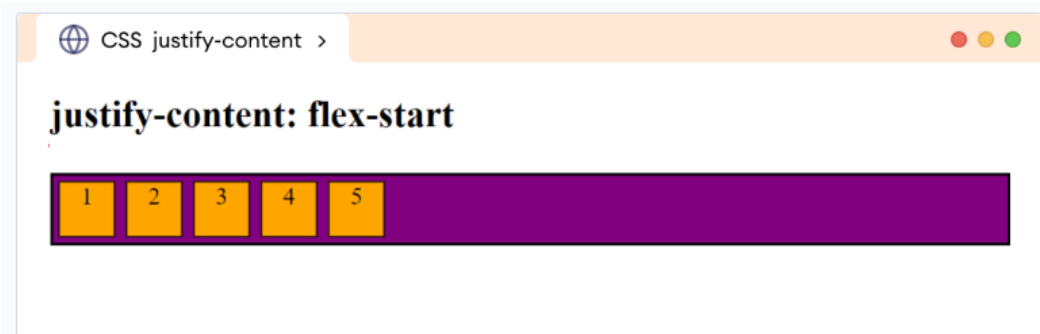
  border: 2px solid black;
  background-color: purple;
}

div.box {
  width: 40px;
  height: 40px;
  text-align: center;
  border: 1px solid black;
  background-color: orange;
  margin: 4px;
}

<body>
  <h2>justify-content; flex-start</h2>
  <div class="container">
    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
    <div class="box">4</div>
    <div class="box">5</div>
  </div>
</body>

```

- The justify-content property is used to distribute the available space between the flex items along the main axis.
- Property are flex-start, center, flex-end, space-between, space-around, and space-evenly.



justify-content: center



justify-content: space-between



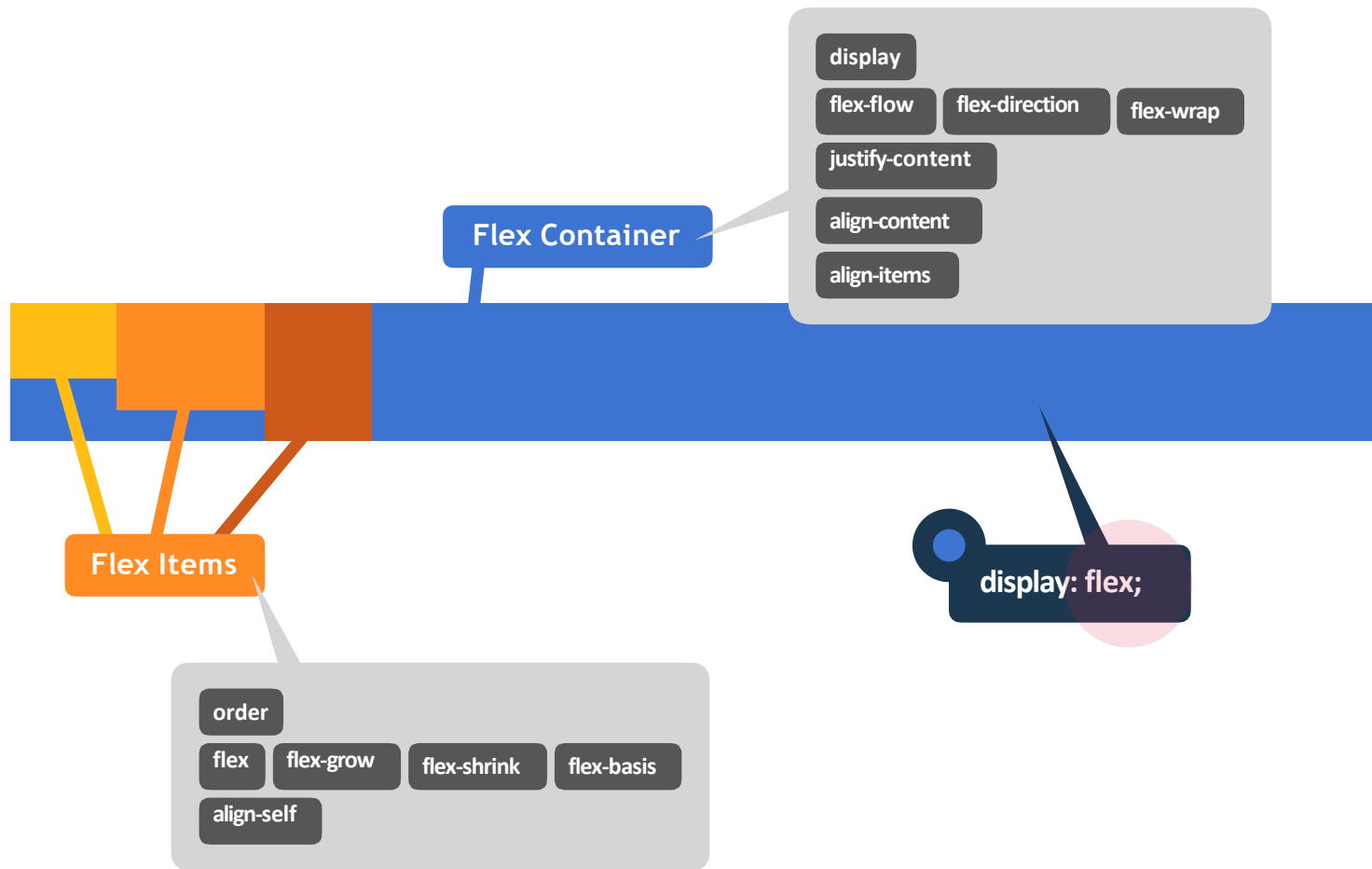
justify-content: space-around



justify-content: space-evenly



- Value aligns the first item to the starting edge (main-start) and the last item to the ending edge (main-end) within a flex container. The remaining space is evenly divided between the flex items
- Evenly divides the space between items. However, there is half of that space before the first and after the last item
- Evenly divides the space between the flex items



Flex Container의 화면 출력(보여짐) 특성

display

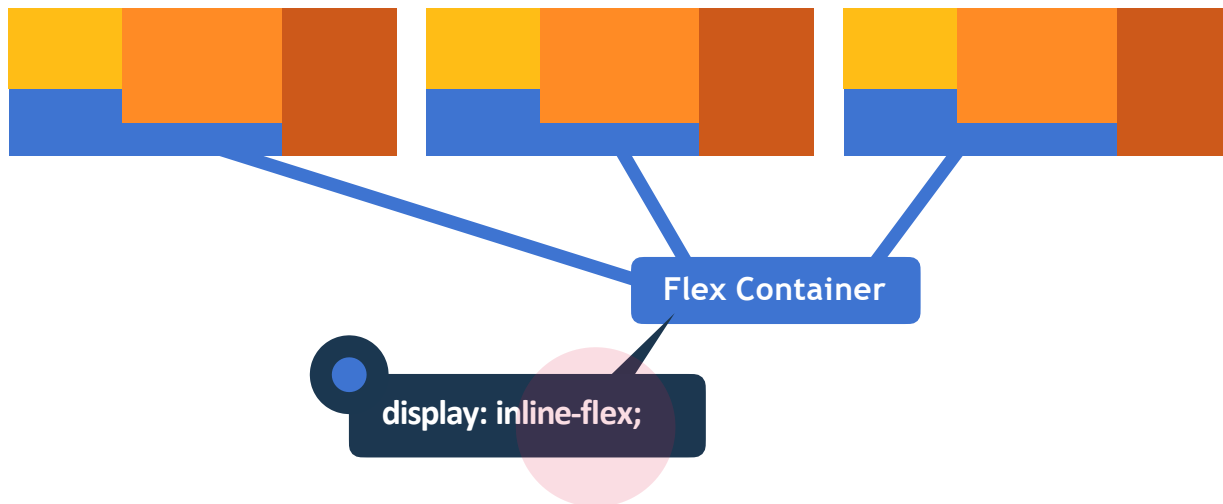
flex

블록 요소와 같이 Flex Container 정의

inline-flex

인라인 요소와 같이 Flex Container 정의





주 축을 설정

flex-direction

row

행 축 (좌 ⇒ 우)

row-reverse

행 축 (우 ⇒ 좌)

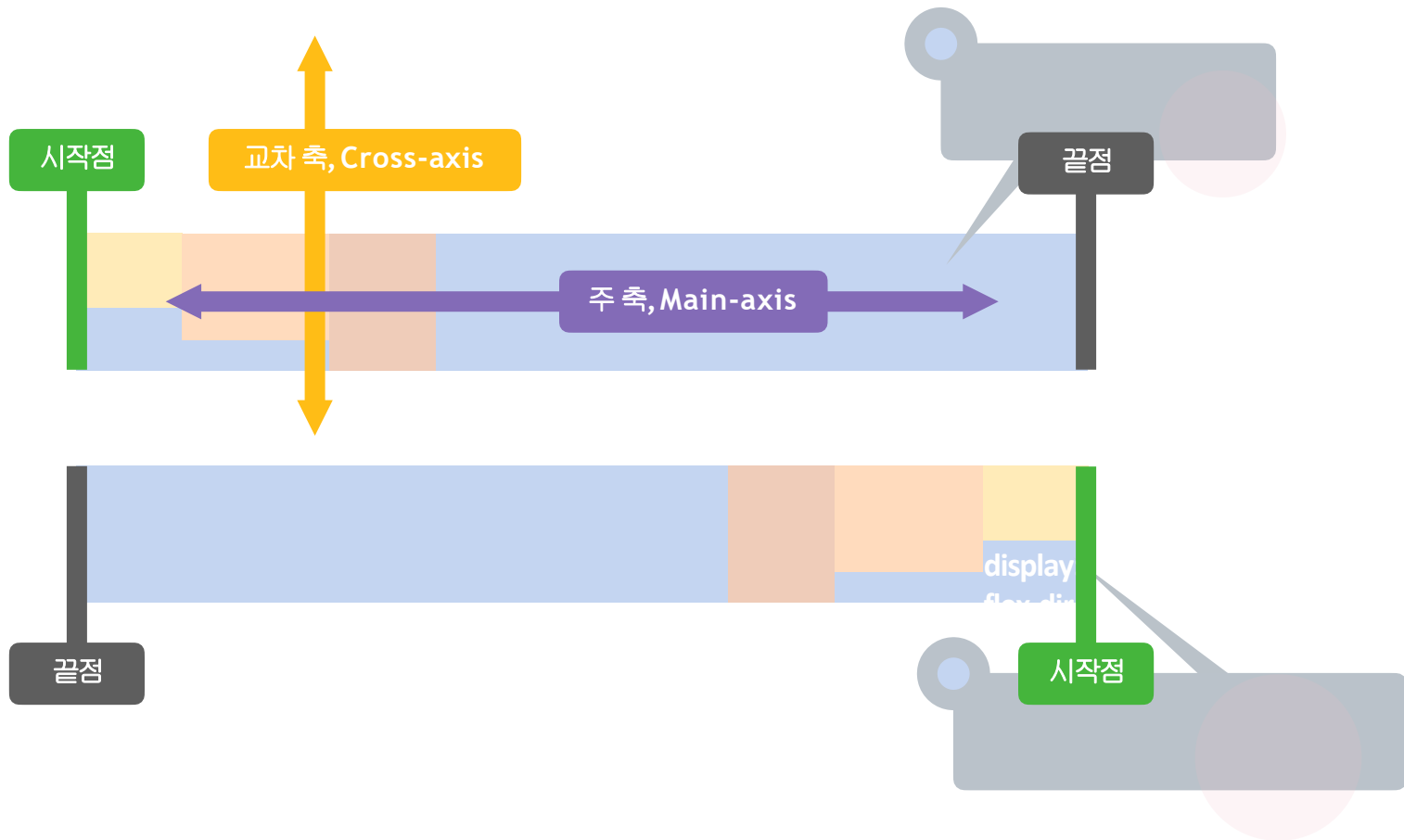
column

열 축 (위 ⇒ 아래)

column-reverse

열 축 (아래 ⇒ 위)







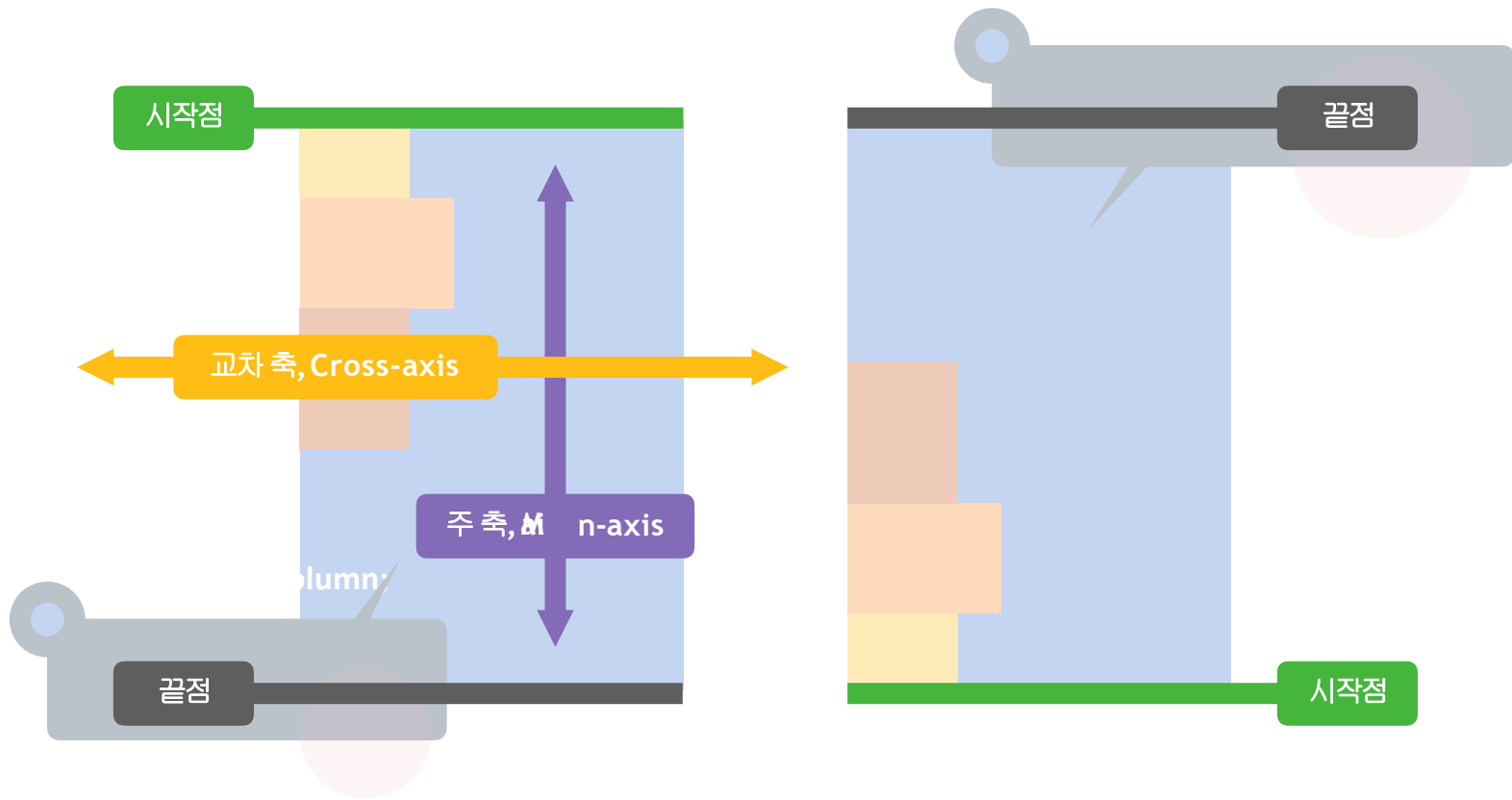
A large blue square represents a flex container. On the left side, three smaller squares are stacked vertically. From top to bottom, they are yellow, orange, and brown. A dark blue callout box with a pointer to the container contains the text: `display: flex;` and `flex-direction: column;`. A light pink circle is partially visible behind the callout box.

`display: flex;`
`flex-direction: column;`



A large blue square represents a flex container. On the left side, three smaller squares are stacked vertically. From top to bottom, they are brown, orange, and yellow. A dark blue callout box with a pointer to the container contains the text: `display: flex;` and `flex-direction: column-reverse;`. A light pink circle is partially visible behind the callout box.

`display: flex;`
`flex-direction: column-reverse;`



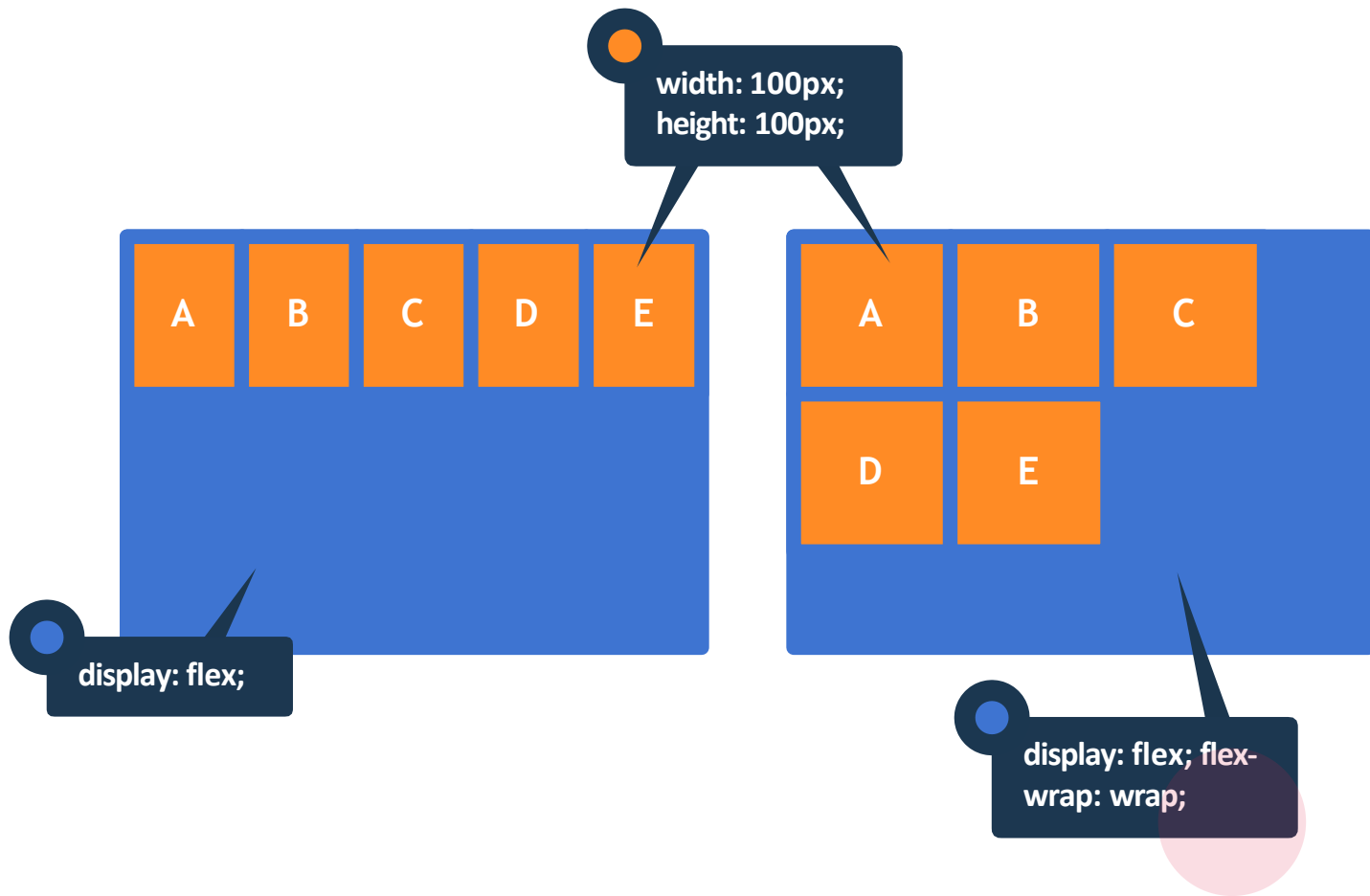
Flex Items 묶음(줄 바꿈) 여부

flex-wrap

nowrap 묶음(줄 바꿈) 없음

wrap 여러 줄로 묶음

wrap-reverse wrap의 반대 방향으로 묶음



주 축의 정렬 방법

justify-content

flex-start

Flex Items를 시작점으로 정렬

flex-end

Flex Items를 끝점으로 정렬

center

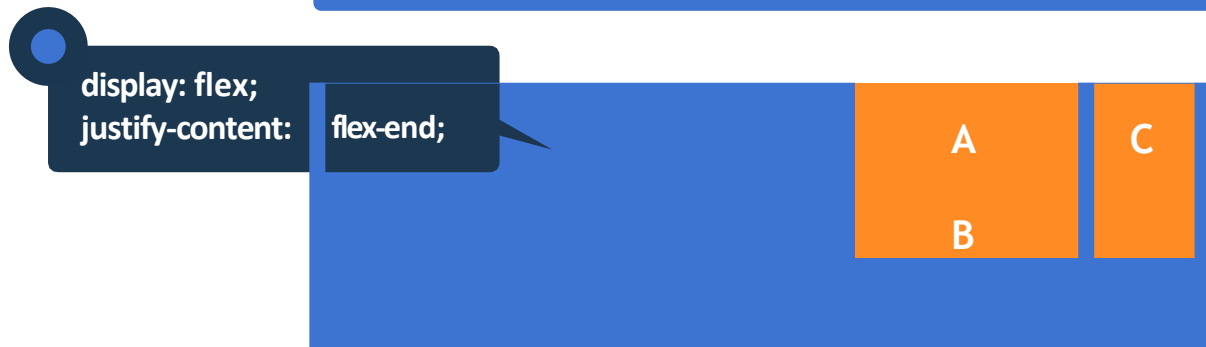
Flex Items를 가운데 정렬

space-between

각 Flex Item 사이를 균등하게 정렬

space-around

각 Flex Item의 외부 여백을 균등하게 정렬



교차 축의 여러 줄 정렬 방법

align-content

stretch

Flex Items를 시작점으로 정렬

flex-start

Flex Items를 시작점으로 정렬

flex-end

Flex Items를 끝점으로 정렬

center

Flex Items를 가운데 정렬

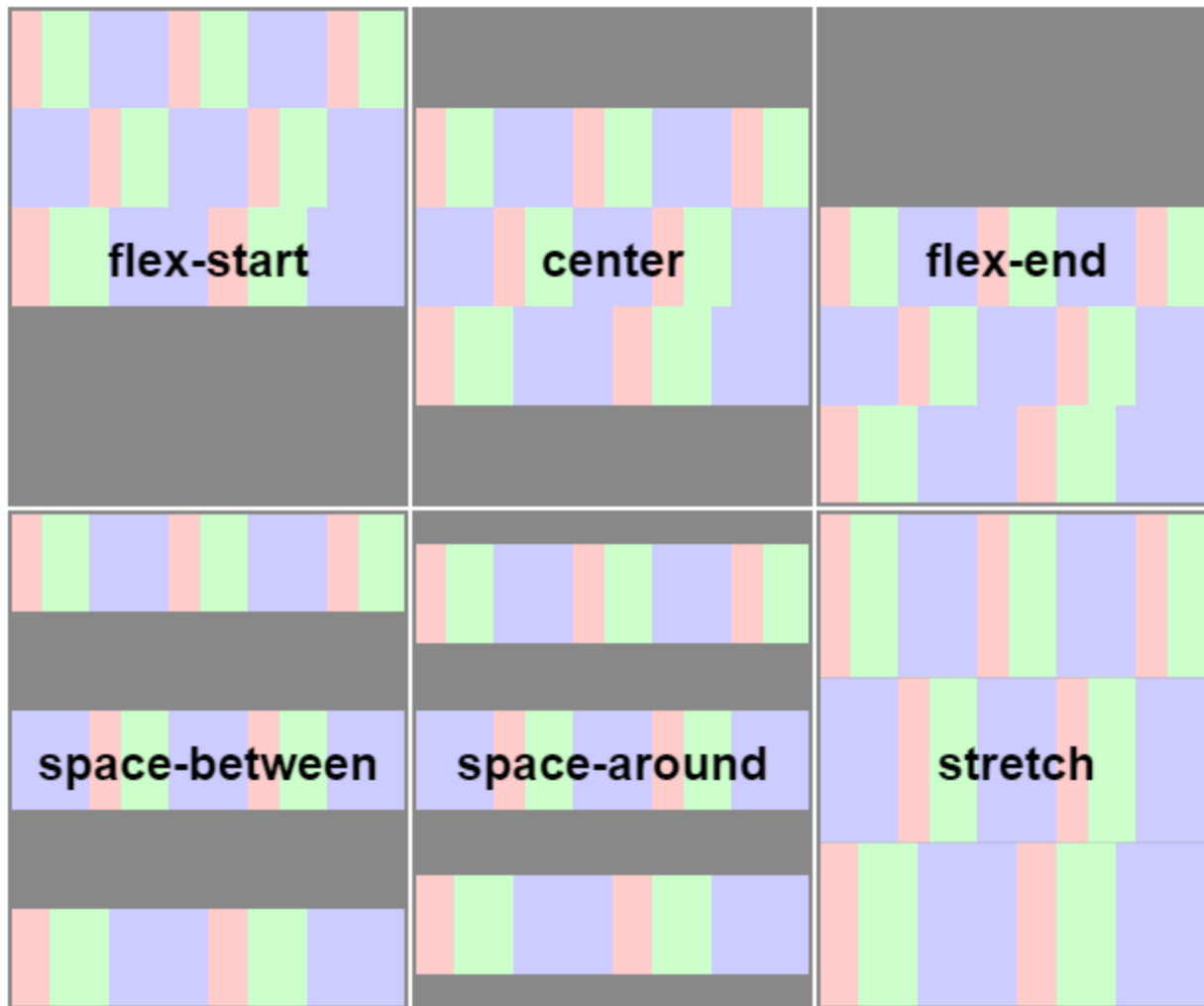
space-between

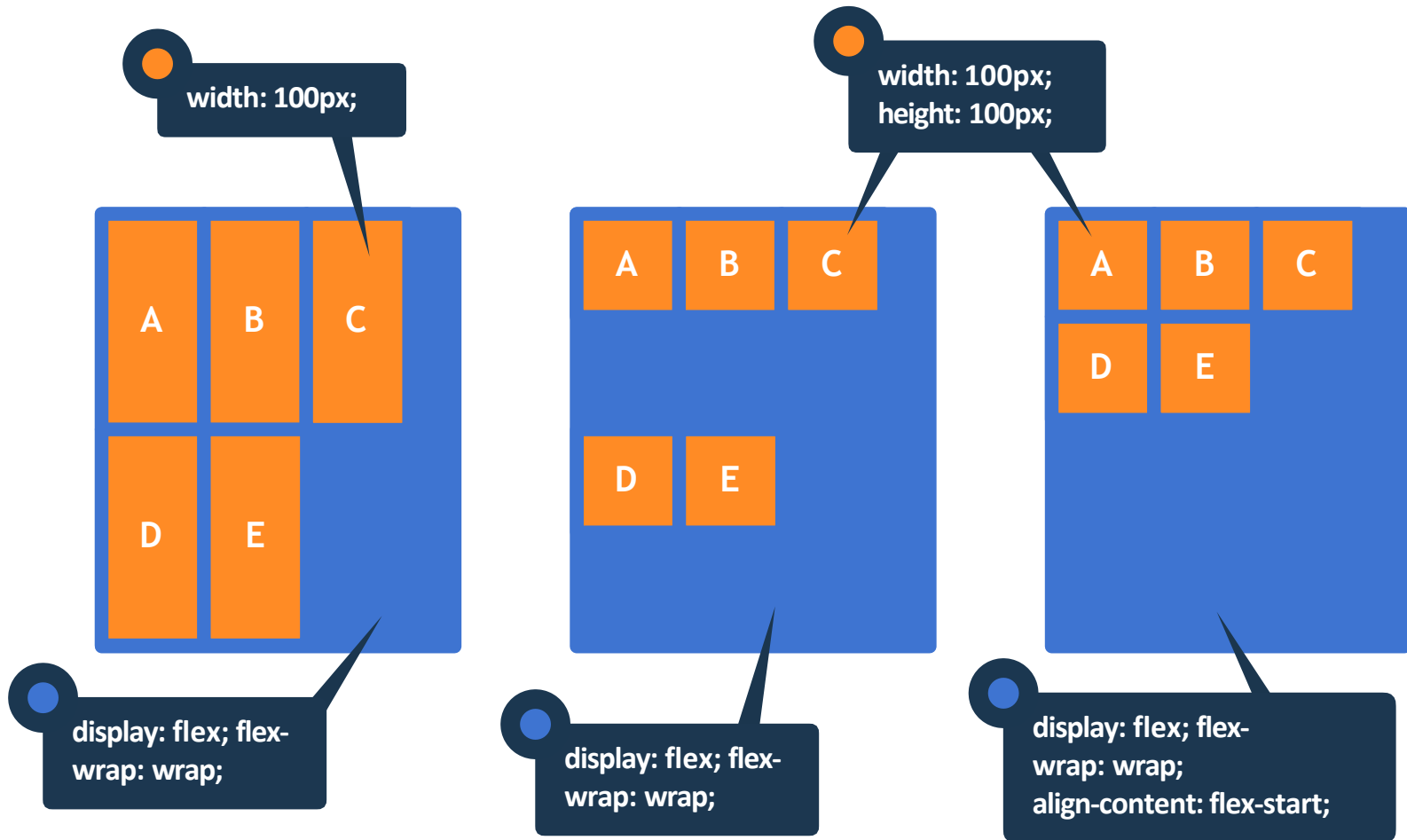
각 Flex Item 사이를 균등하게 정렬

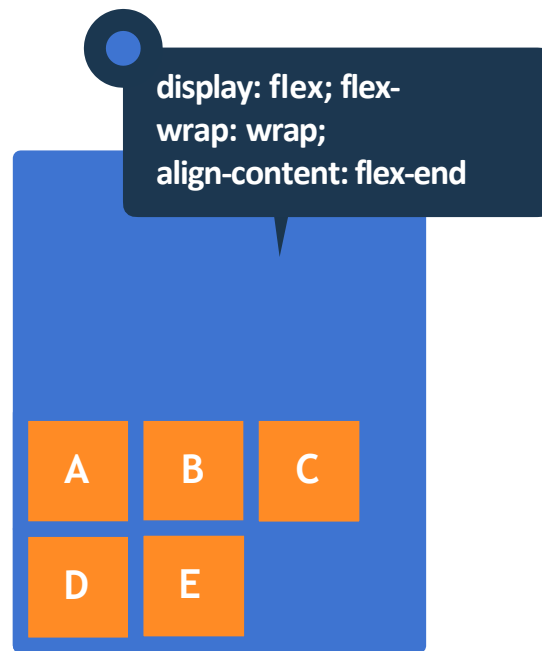
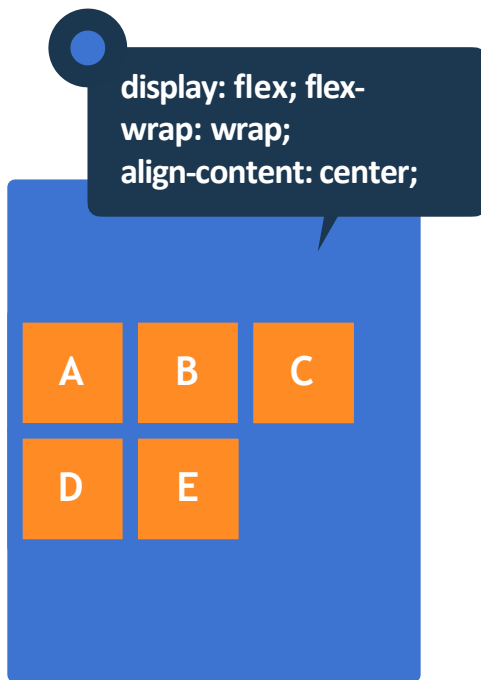
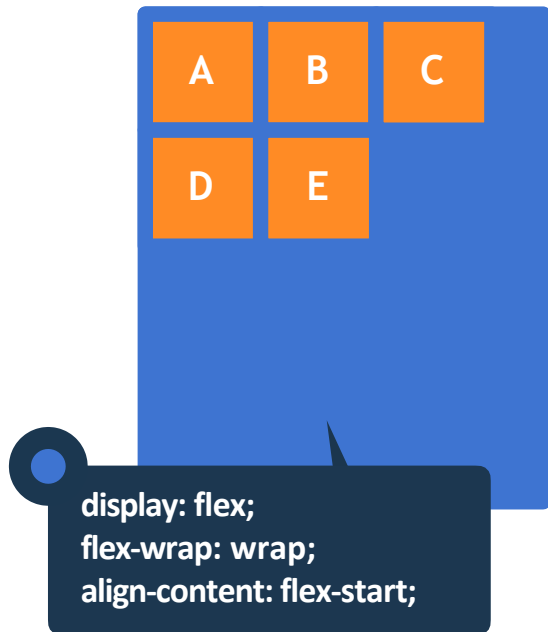
space-around

각 Flex Item의 외부 여백을 균등하게 정렬

The align-content property controls the alignment of **flex lines** when there is extra space in the **cross-axis**





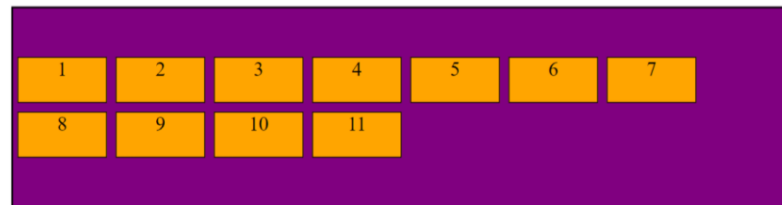


```
div.container {
  height: 180px;
  display: flex;
  /* setting a flex-wrap value is important */
  flex-wrap: wrap;
  align-content: *****;
  border: 2px solid black;
  background-color: purple;
}
```

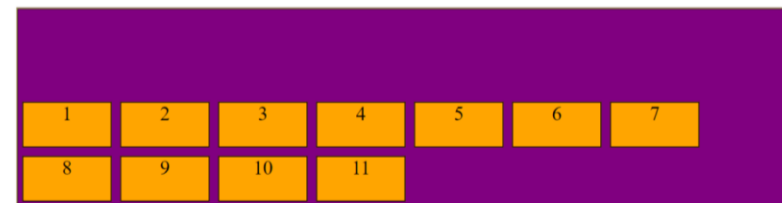
```
div.box {
  width: 80px;
  height: 40px;
  text-align: center;
  border: 1px solid black;
  background-color: orange;
  margin: 4px;
}
```

```
<body>
  <h2>align-content: ***** </h2>
  <div class="container">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
    <div class="box box4">4</div>
    <div class="box box5">5</div>
    <div class="box box6">6</div>
    <div class="box box7">7</div>
    <div class="box box8">8</div>
    <div class="box box9">9</div>
    <div class="box box10">10</div>
    <div class="box box11">11</div>
  </div>
</body>>
```

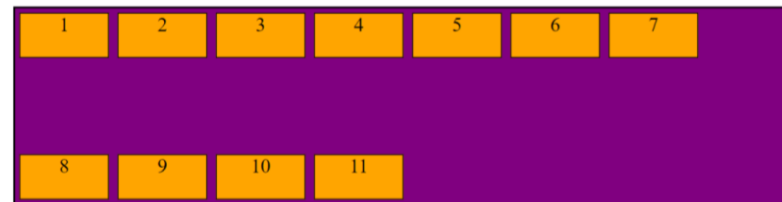
align-content: center



align-content: flex-end



align-content: space-between

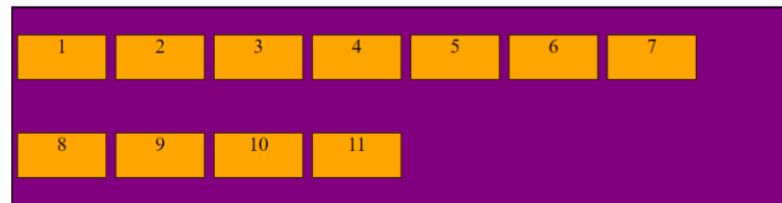


```
div.container {
  height: 180px;
  display: flex;
  /* setting a flex-wrap value is important */
  flex-wrap: wrap;
  align-content: *****;
  border: 2px solid black;
  background-color: purple;
}
```

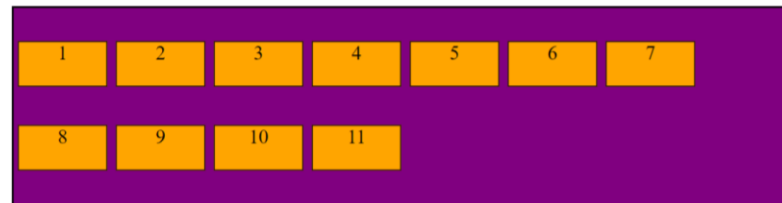
```
div.box {
  width: 80px;
  height: 40px;
  text-align: center;
  border: 1px solid black;
  background-color: orange;
  margin: 4px;
}
```

```
<body>
  <h2>align-content: ***** </h2>
  <div class="container">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
    <div class="box box4">4</div>
    <div class="box box5">5</div>
    <div class="box box6">6</div>
    <div class="box box7">7</div>
    <div class="box box8">8</div>
    <div class="box box9">9</div>
    <div class="box box10">10</div>
    <div class="box box11">11</div>
  </div>
</body>>
```

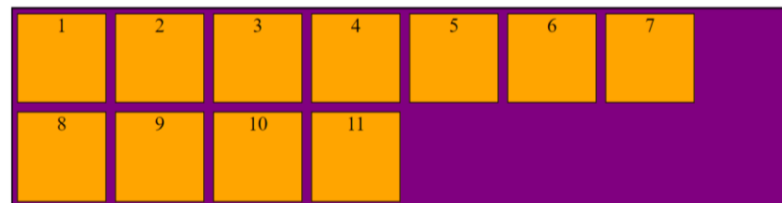
align-content: space-around



align-content: space-evenly



align-content: stretch



교차 축의 한 줄 정렬 방법

align-items

stretch

Flex Items를 교차 축으로 늘림

flex-start

Flex Items를 각 줄의 시작점으로 정렬

flex-end

Flex Items를 각 줄의 끝점으로 정렬

center

Flex Items를 각 줄의 가운데 정렬

baseline

Flex Items를 각 줄의 문자 기준선에 정렬

The align-items property distributes the available space between the **flex items** along the **cross** axis

The space is distributed **vertically** for **flex-direction: row** and horizontally for flex-direction: column

The possible values for the align-items property are

- flex-start (default value),
- center,
- flex-end,
- baseline, and
- stretch.

```
div.container {
  height: 160px;
  display: flex;

  /* stretches the flex items */
  align-items: *****;

  border: 2px solid black;
  background-color: purple;
}
```

```
div.box {
  width: 40px;

  text-align: center;
  border: 1px solid black;
  background-color: orange;
  margin: 4px;
}
```

```
<body>
  <h2>align-items: ***** </h2>
  <div class="container">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
    <div class="box box4">4</div>
    <div class="box box5">5</div>
  </div>
</body>
```

```
div.container {
  height: 160px;
  display: flex;
  align-items: baseline;
  border: 2px solid black;
  background-color: purple;
}

div.box {
  width: 40px;
  height: 40px;
  text-align: center;
  border: 1px solid black;
  background-color: orange;
  margin: 4px;
}

div.box3 {
  width: 90px;
  height: 90px;
  line-height: 70px;
  background-color: skyblue;
}

div.box4 {
  width: 100px;
  height: 100px;
  line-height: 140px;
  background-color: greenyellow;
}
```

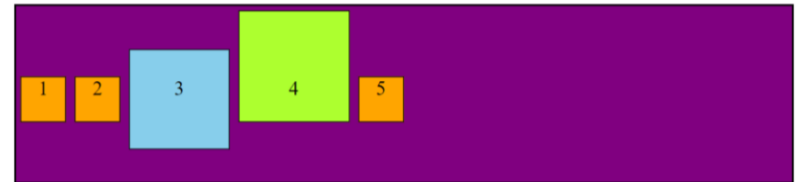
align-items: center



align-items: flex-end



align-items: baseline



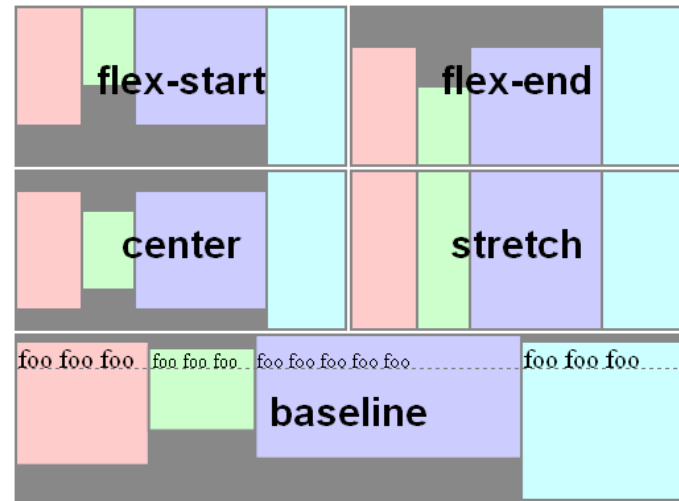
align-items: stretch



The align-items property distributes the available space between the flex items along the cross axis

Controls the alignment of all items on the cross axis

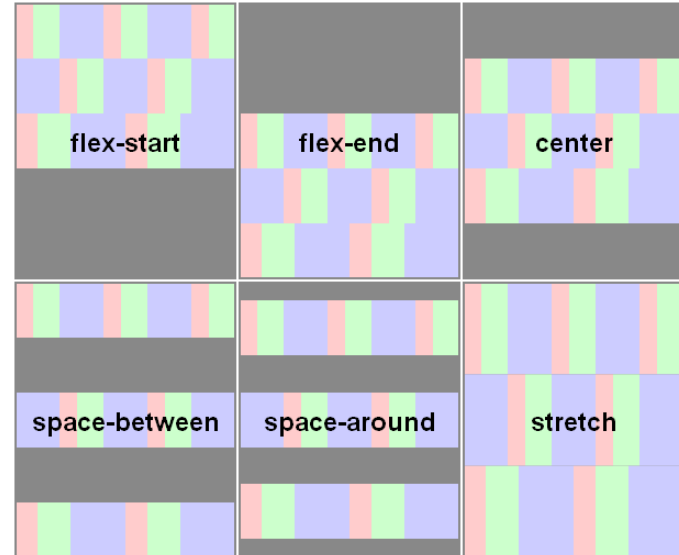
Single line



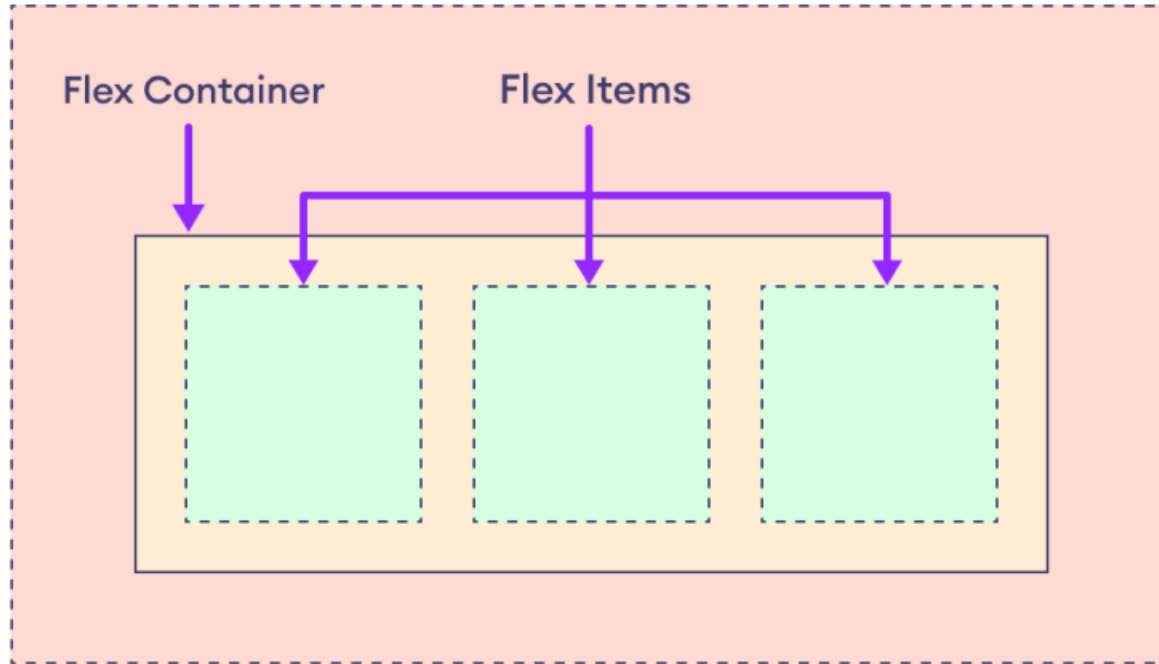
The align-content property controls the alignment of **flex lines** when there is extra space in the **cross-axis**

Controls the space between flex lines on the cross axis.

two and more lines and with wrap



Flex Item



Flex Item의 순서

order

0

순서 없음

숫자

숫자가 작을 수록 먼저

The flex items are arranged in the order they appear in the HTML document by default.

We can use the order property to specify the order of flex items in the flex container.

```
div.container {
  display: flex;
  flex-direction: row;
  background-color: purple;
}
div.box {
  width: 80px;
  height: 40px;
  text-align: center;
  background-color: orange;
  margin: 8px;
}
div.box1 {
  order: 3;
}
div.box2 {
  order: 1;
}
div.box3 {
  order: 2;
}
```

```
<body>
  <h2>Order Property</h2>
  <div class="container">
    <div class="box box1">1 <br />order: 3;</div>
    <div class="box box2">2 <br />order: 1;</div>
    <div class="box box3">3 <br />order: 2;</div>
  </div>
</body>
```

🌐 CSS order >

Order Property

2 order: 1;	3 order: 2;	1 order: 3;
----------------	----------------	----------------

Flex Item의 증가 너비 비율

flex-grow

0

증가 비율 없음

숫자

증가 비율

The flex-grow property determines how much flex items expand relative to the other items in the flex container. It takes a unitless number representing a **proportion of the remaining space** in the container.

```

/* styles for both containers */
div.container {
  display: flex;
  flex-direction: row;
  background-color: purple;
}

div.box {
  width: 80px;
  height: 40px;
  text-align: center;
  background-color: orange;
  margin: 8px;
}

/* styles for second container */
div.container2 div.box {
  flex-grow: 1;
}

```

```

<body>
  <h2>flex-grow: 0 (default value)</h2>

  <!-- container without flex-grow -->
  <div class="container container1">
    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
    <div class="box">4</div>
    <div class="box">5</div>
  </div>

  <h2>flex-grow: 1</h2>

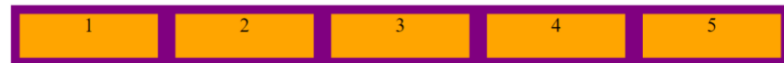
  <!-- container with flex-grow -->
  <div class="container container2">
    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
    <div class="box">4</div>
    <div class="box">5</div>
  </div>
</body>

```

flex-grow: 0 (default value)



flex-grow: 1



“Flex-grow: 1” causes the flex items to grow proportionally, taking the available space within the second flex container.

```
div.container {
  display: flex;
  flex-direction: row;
  background-color: purple;
}
div.box {
  width: 40px;
  height: 40px;
  text-align: center;
  background-color: orange;
  margin: 8px;
}
div.box2 {
  flex-grow: 2;
}
div.box4 {
  flex-grow: 1;
}
```

```
<body>
  <div class="container">
    <div class="box box1">1</div>
    <div class="box box2">2 <br />flex-grow: 2;</div>
    <div class="box box3">3</div>
    <div class="box box4">4 <br />flex-grow: 1;</div>
    <div class="box box5">5</div>
  </div>
</body>
```



The flex-grow: 2 makes a flex item grow twice as much as the flex-grow: 1 item when **there's extra space** in the flex container.

Flex Item의 감소 너비 비율

flex-shrink

1

Flex Container 너비에 따라 감소 비율 적용감

숫자

소 비율

The flex-shrink property determines how much flex items shrink relative to the other items when there is not enough space in the flex container.

The flex-shrink value represents a proportion factor for shrinking the flex item. The flex items shrink faster with the high value.

```
<body>
  <h2>flex-shrink: 1 (default value) for all flex items</h2>
  <div class="container container1">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
    <div class="box box4">4</div>
    <div class="box box5">5</div>
    <div class="box box6">6</div>
  </div>
```

```
<h2>flex-shrink: 2 on the third item</h2>
```

```
<div class="container container2">
  <div class="box box1">1</div>
  <div class="box box2">2</div>
  <div class="box box3">3</div>
  <div class="box box4">4</div>
  <div class="box box5">5</div>
  <div class="box box6">6</div>
</div>
</body>
```

```
div.container {
  display: flex;
  flex-direction: row;
  background-color: purple;
}
div.box {
  width: 200px;
  height: 40px;
  text-align: center;
  background-color: orange;
  margin: 8px;
}
div.container2 div.box3 {
  flex-shrink: 2;
}
```

flex-shrink: 1 (default value) for all flex items



flex-shrink: 2 on the third item



In the above example, the initial width of all the flex items is 200px. Since the container doesn't have available space, all the flex items **shrink proportionally**.

However, in the second container, the use of flex-shrink: 2 causes the third flex item to **shrink more compared to the others**, as there isn't enough space to accommodate all items at their natural sizes.

Flex Item의 공간 배분 전 기본 너비

flex-basis

auto

요소의 Content 너비

단위

px, em, rem 등 단위로 지정

The flex-basis property defines the starting size of the flex item before any wrapping, growing, or shrinking occurs.

It is an **alternative to the width and height** of the flex items.

```
<body>
  <h2>flex-basis: auto (default value for all flex items)</h2>
  <div class="container container1">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
    <div class="box box4">4</div>
    <div class="box box5">5</div>
  </div>
```

```
  <h2>flex-basis: 200px on the fourth item</h2>
  <div class="container container2">
    <div class="box box1">1</div>
    <div class="box box2">2</div>
    <div class="box box3">3</div>
    <div class="box box4">4</div>
    <div class="box box5">5</div>
  </div>
</body>
```

```
div.container {
  display: flex;
  flex-direction: row;
  background-color: purple;
}
div.box {
  width: 40px;
  height: 40px;
  text-align: center;
  background-color: orange;
  margin: 8px;
}
div.container2 div.box4 {
  flex-basis: 200px;
}
```

flex-basis: auto (default value for all flex items)



flex-basis: 200px on the fourth item



Here, flex-basis: 200px sets the length of the fourth flex item to 200 pixels in the second flex container.

JavaScript 선행 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

0 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;  
}
```



데이터를 저장하고 참조(사용)하는 데이터의 이름 **var, let, const**

변수

예약어

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

Reserved Word

```
let this = 'Hello!'; // SyntaxError
let if = 123; // SyntaxError
let break = true; // SyntaxError
```

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 ...

특정 동작(기능)을 수행하는 일부 코드의 집합(부분)function

함수

조건문

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

DOM API

Document Object Model, Application Programming Interface

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(낙
타 표기법)로 작성하시오
!

A

.

theQuickBrownFox

Q.

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

위 데이터를 활용해
'Banana'를 콘솔 출력하
시오!

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 };  
cons  
ole.l  
og(o
```

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, b**와 같이, 함수 호출에서 전달받은 **인수**를

함수 내부로 **전달**하기 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 출력하
시오!

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) 이벤트**를 **추가**해, 클릭 시 **'Hello~'**를 **콘솔 출력**하시오!

A.

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

Q.

```
<div>1</div>
```

```
<div>2</div>
```

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

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 속성의값으로
'active'가 포함되어 있으면,
'포함됨!'을 콘솔 출력하시오!

A.

```
if
```

```
    (boxEl.classList  
    .contains('activ  
e')) {  
    console.log('포  
함됨!');
```

```
}
```