

CSS 심화

Session 1

NEXT 김지성

5분 넥코톡

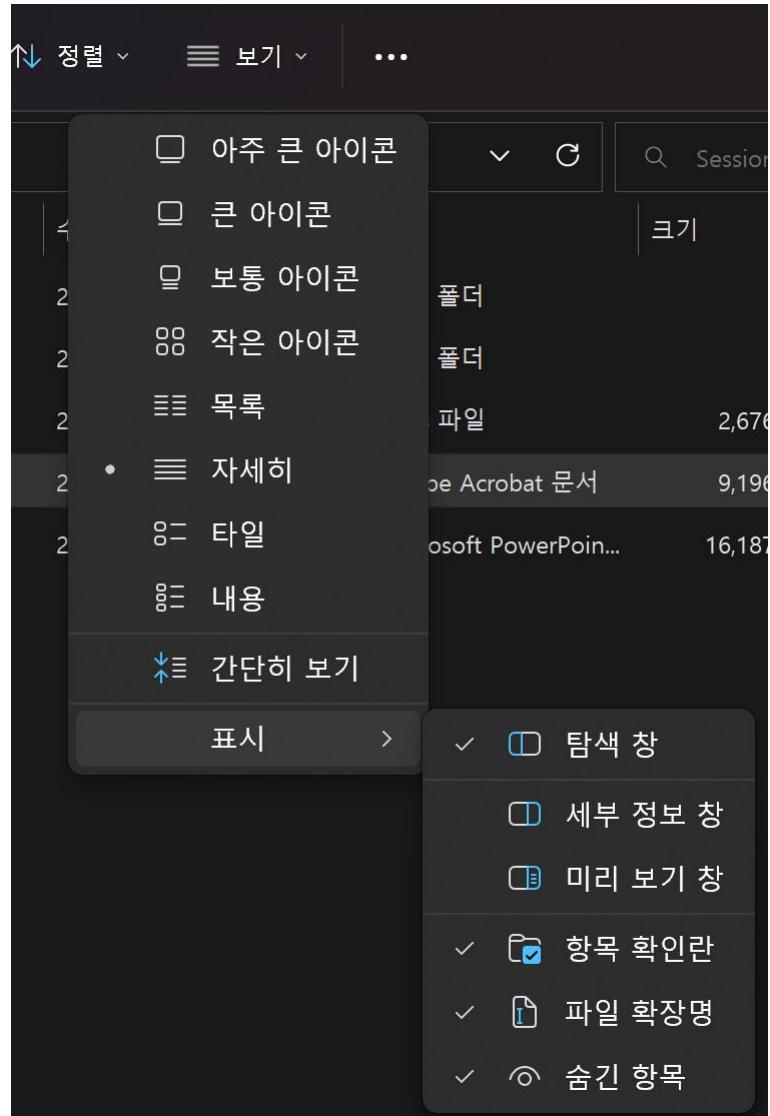
VS Code 단축키

Ctrl - Cmd / Alt - Option

주석처리	Ctrl + /
파일 삭제	Ctrl + x
행 삭제	Ctrl + delete
행 위아래 이동	Alt + 방향키
위아래에 행 복제	Alt + shift + 방향키
동일 변수명 동시 선택	변수명 선택 후 Ctrl + D

5분 넥코톡

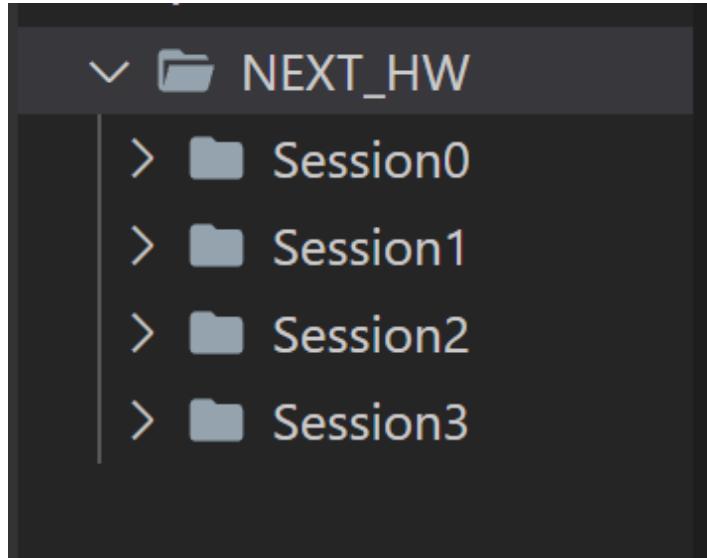
파일 관리법



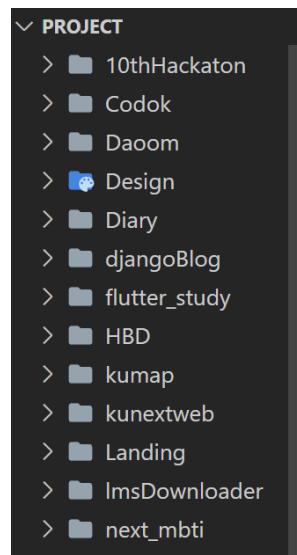
앞으로 지겹도록 보게 될 .git(git init 했을 때 생겨날 폴더) 폴더가 숨김 폴더니까 보기에서 숨긴 항목 보기 활성화 해 주세요!

5분 넥코톡

파일 관리법



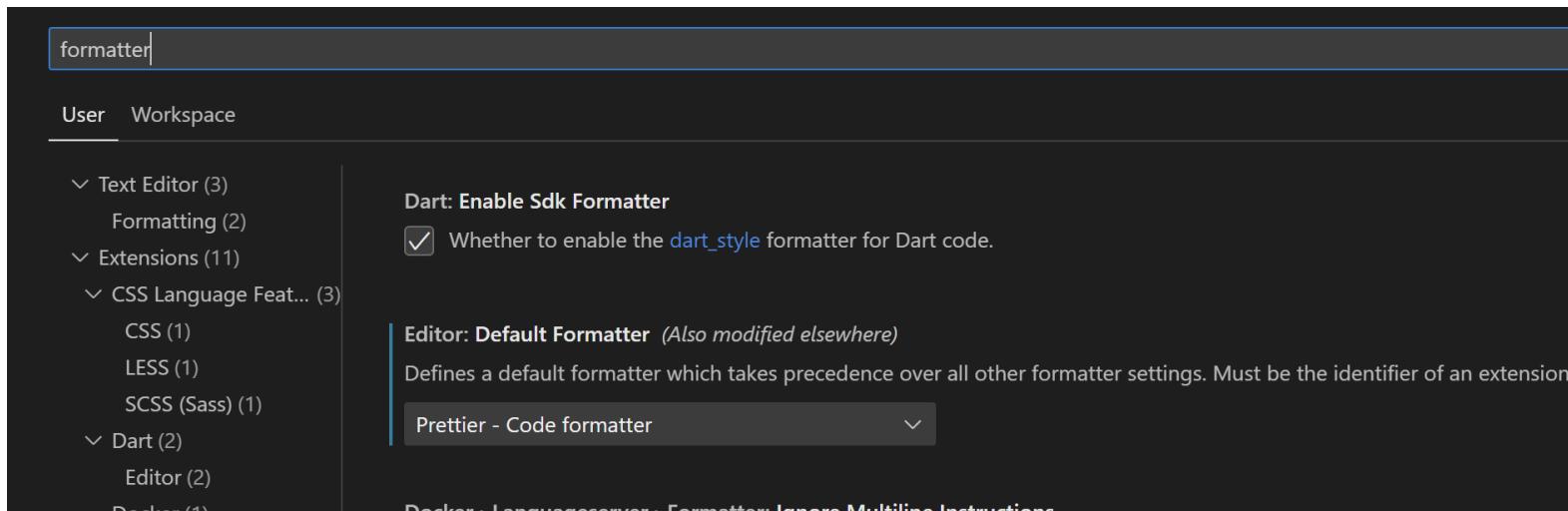
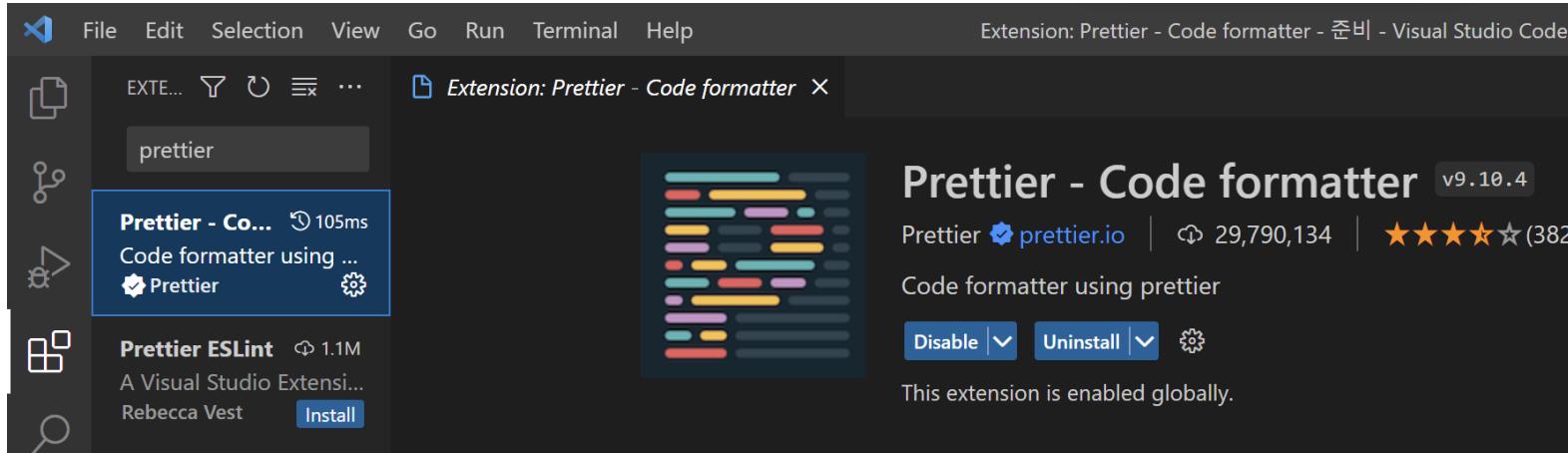
과제용 폴더를 만드시고 그 안에 Session 폴더를 만들어주세요
Github과 함께 사용할 때 파일이 구분없이 널려있으면 무수한 에러가
생길 확률이 높답니다...
아무데나 git init 하면 무조건 에러터집니다!!



취향차이지만 저는 프로젝트 관리할 때에도 이렇게 관리하고 있습니다!

5분 넥코톡

Prettier



Prettier 설치

아마 여러분의 코딩인생이 132배쯤
편해지실 예정...

Formatter 설정

Ctrl + , 로 메뉴 열고 formatter 검색!
Default Formatter 설정하기

복습: CSS 기본문법

선택자 {속성: 값;}

선택자
selector

p {

값
value

color: red;

property
속성

}

복습: CSS 선택자(Selector)

*

전체 선택자 (Universal Selector)

#id

ID 선택자 (ID Selector)

tag

태그 선택자 (Type Selector)

.class

클래스 선택자 (Class Selector)

NEW!

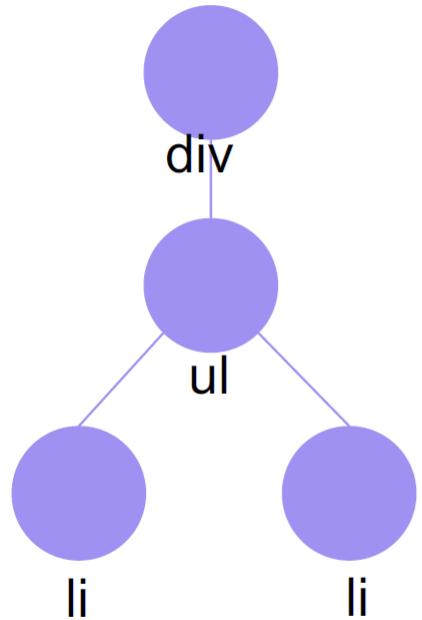
복합 선택자 (Combinator)

가상 클래스 선택자
(Pseudo-Class Selector)

자식과 부모, 조상과 후손의 개념

Tree 구조는 알아야 FE 개발자임

Tree 구조



```
<div class="ul의 부모이자 li의 조상">
  <ul class="div의 자식이자 li의 부모">
    <li class="ul의 자식이자 div의 후손">
      </li>
    <li class="위 li의 형제">
      </li>
    </ul>
  </div>
```

자식도 후손(하위) 요소다!!!

복합 선택자

2개 이상의 선택자 요소가 모인 선택자

일치 선택자 **a.b**

: a 와 b의 조건을 동시에 만족하는 요소 선택

자식 선택자 **a > b**

: a의 자식 요소인 b를 선택

후손 선택자 **a b**

: a의 하위 요소인 b를 선택

인접 형제 선택자 **a + b**

: a의 바로 다음 형제 요소인 b 하나만 선택

일반 형제 선택자 **a ~ b**

: a의 다음 형제 요소 b 모두 선택

가상 클래스 선택자(Pseudo Class Selector)

임의로 가상의 선택자를 지정해 선택하는 것

:link 방문한 적이 없는 링크

:first 첫번째 요소

:visited 방문한 적이 있는 링크

:last 마지막 요소

:hover 마우스를 둘오버 했을 때

:first-child 첫번째 자식

:active 마우스를 클릭했을 때

:last-child 마지막 자식

:focus 포커스 되었을 때 (input 태그 등)

:nth-child(2n+1) 홀수 번째 자식

Ex)

```
.box1:hover {  
    background-color: black;  
}
```

복습: Cascading

상속에 대해 더 깊이 알아봅시다

Cascading Style Sheet



위에서 아래로 흐르는, **상속** 또는 종속하는

복습: Cascading

상속의 개념

상속이란?

상위 (부모나 조상) 요소에 적용된 프로퍼티를 하위 (자식이나 후손)이 물려받는 것!

CSS에는 상속이 되는 속성이 있고, 되지 않는 것이 있음.

CSS의 기본 속성입니다!

복습: Cascading

외울 필요 없어요!!!

property	inherit
width/height	no
margin	no
padding	no
border	no
box-sizing	no
display	no
visibility	yes
opacity	yes
background	no
font	yes
color	yes
line-height	yes
text-align	yes
vertical-align	no
text-decoration	no
white-space	yes
position	no
top/right/bottom/left	no
z-index	no
overflow	no
float	no

복습: Cascading Order

CSS의 적용 우선 순위

중요도

명시도

선언순서

중요도

CSS의 적용 우선 순위

CSS가 **어디에 선언 되었는지**에 따라서 우선순위가 달라진다

1. head 요소 내의 style 요소
2. head 요소 내의 style 요소 내의 @import 문
3. <link>로 연결된 CSS 파일
4. <link>로 연결된 CSS 파일 내의 @import 문
5. 브라우저 디폴트 스타일시트

명시도

CSS의 적용 우선 순위

대상을 **명확하게 특정할수록** 명시도가 높아지고 우선순위가 높아진다

1. ! important
2. in-line style
3. ID selector
4. class selector
5. tag 선택자
6. 전체 선택자

웬만하면 쓰지 마세요!!!!

**!important를 써야하는 상황은 이미 CSS 구조가 잘못 짜였다는 말이고,
상당한 맥락을 무시하고 적용되기에 결국 모든 CSS가 !important 범벽이
될 수 있습니다.**

선언 순서

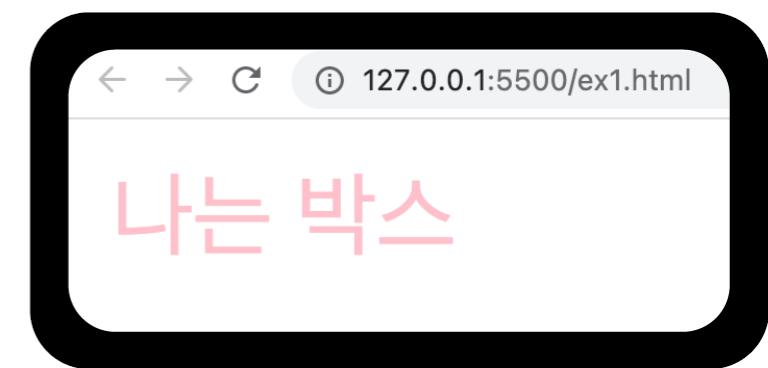
CSS의 적용 우선 순위

나중에 선언된 스타일이 우선 적용된다!

```
</head>
<body>
    <div class="box">나는 박스</div>
</body>
</html>
```

```
.box {
    color: ■purple;
}

.box {
    color: ■pink;
}
```



복습: 속성

많이 쓰이는 속성들

width 가로 길이

height 높이

color 텍스트의 색

background-color 배경 색

font-size 폰트 크기 조절

border 테두리

margin 바깥 여백

padding 안쪽 여백

복습 끝!!

이제 CSS 마스터?



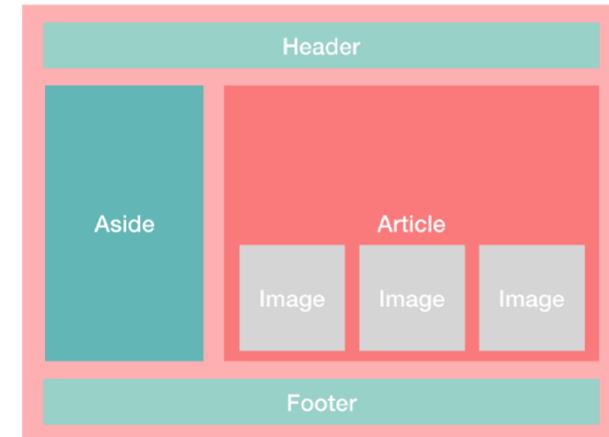
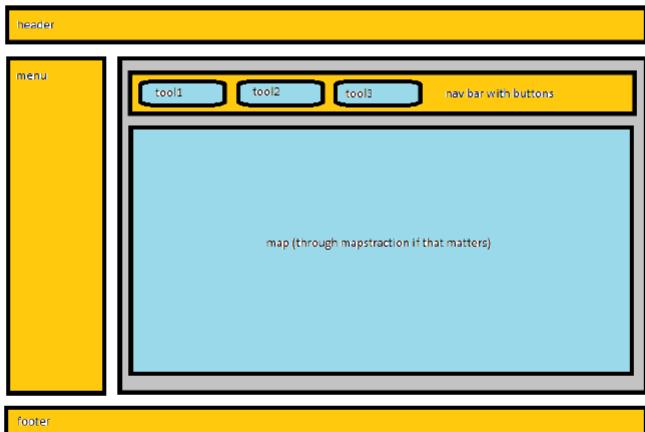
응, 아니야

레이아웃

Position? Flex?

레이아웃이란?

페이지에 있는 요소들을 어떻게 배치할지를 정하는 방법!



우리는 position 이랑 flex 활용할 것!

position

Position

Position? 그거 왜 쓰는데?

position 이란?

기존의 배치 위치에서 벗어나 다른 위치로 이동시킬 수 있게 한다!

```
.box1 {  
    position: relative;  
}
```

```
.box2 {  
    position: absolute;  
}
```

Position의 type들

Position에는 이런 녀석들이 있습니다

static

relative

absolute

fixed

sticky

Position: static;

Super Normal.

모든 요소에 주어지는 기본값!

아무런 변화 없음..

position: relative;

나의 측은 너였다.

static 이었을 때의 위치를 기준으로 이동 가능!

top, right, bottom, left, z-index 등의 속성을 쓸 수 있음

하지만 보통 position: relative는 그 자체로 사용되기보다는,
다음에 나올 position: absolute의 부모요소이자 기준으로 많이 활용됩니다!!!

position: relative;

Relative 활용 방법

```
.box {  
    height: 200px;  
    width: 200px;  
    background-color: #purple;  
    position: relative;  
    top: 50px;  
    left: 50px;  
}
```

top: 50px; top을 기준으로 50px 만큼 이동해라

left: 50px; left을 기준으로 50px 만큼 이동해라

이런 방식은 이후에 나올 absolute와 fixed에서도 적용되니 잘 알아두세요!

position: absolute;

질풍노도의 시기를 겪는 중학교 2학년 같은 녀석

position: static;을 가지고 있지 않은 부모를 기준으로 이동!

여기서 ‘부모’는 아까 설명했던 tree 구조를 다시 생각해보면 됨!

position: absolute;

Position: static이 아닌 부모를 기준으로 이동

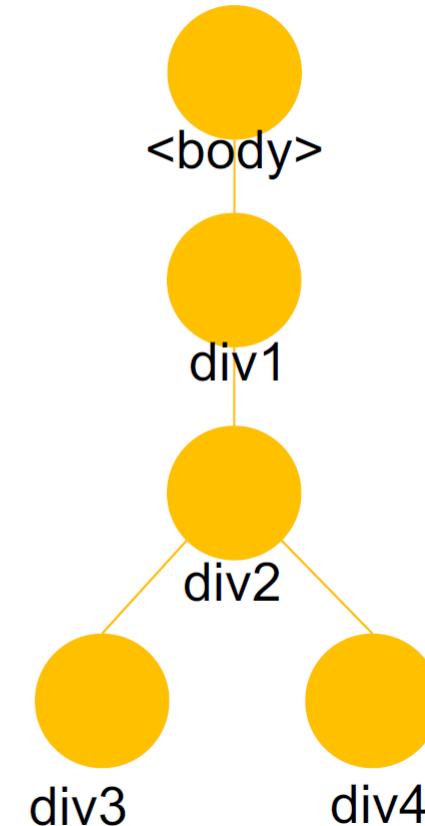
position: static;이 아닌 부모가 있을 때

부모를 기준으로 움직인다!

position: static;이 아닌 부모가 없을 때

<body>를 기준으로 움직인다!

Tree 구조



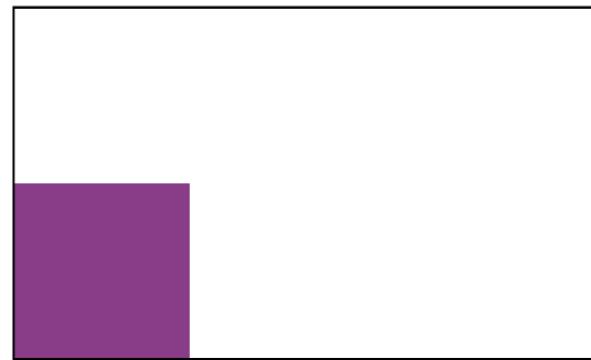
position: absolute;

Position: static이 아닌 부모를 기준으로 이동

position: static이 아닌 부모가 있을 때

```
.box_container {  
    border: 2px solid black;  
    height: 300px;  
    width: 500px;  
    position: relative;  
}  
  
.box {  
    height: 150px;  
    width: 150px;  
    background-color: #rgb(136, 61, 136);  
    position: absolute;  
    top: 150px;  
}
```

부모를 기준으로 움직인다!



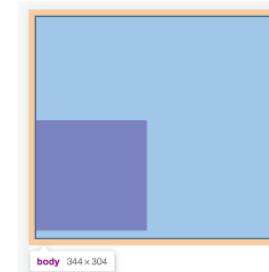
position: absolute;

Position: static이 아닌 부모를 기준으로 이동

position: static;이 아닌 부모가 없을 때

```
.box_container {  
    border: 2px solid black;  
    height: 300px;  
    width: 500px;  
}  
  
.box {  
    height: 150px;  
    width: 150px;  
    background-color: #rgb(136, 61, 136);  
    position: absolute;  
    top: 150px;  
}
```

<body>태그를 기준으로 움직인다!



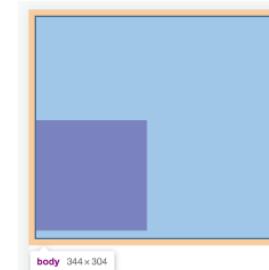
position: absolute;

Position: static이 아닌 부모를 기준으로 이동

position: static;이 아닌 부모가 없을 때

```
.box_container {  
    border: 2px solid black;  
    height: 300px;  
    width: 500px;  
}  
  
.box {  
    height: 150px;  
    width: 150px;  
    background-color: #rgb(136, 61, 136);  
    position: absolute;  
    top: 150px;  
}
```

<body>태그를 기준으로 움직인다!



position: absolute;

Position: static이 아닌 부모를 기준으로 이동

따라서, 부모를 기준으로 움직이고 싶다면?

부모에는 static이 아닌 값을 position에!

해당 태그에는 position: absolute;

| **position: fixed;**

부모요소와 관계없이 무조건 뷰포트를 기준으로 움직인다.

position: fixed;는 해당 위치에 고정된다!

스크롤을 해도 위치에 고정되어 있다

position: sticky;

위치 고정

position: sticky;는 스크롤을 하다가 부딪치는 순간 그 위치에 고정된다!

fixed와 헷갈리지 않게 조심!

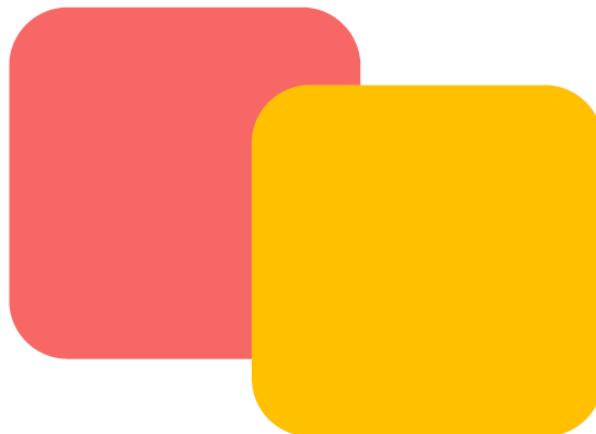
예시: <https://www.maisonmargiela.com/ko-kr/>

Z-index

쌓임 맥락에 영향을 미치는 요소

한 요소 위에 다른 요소를 쌓을 때 필요!

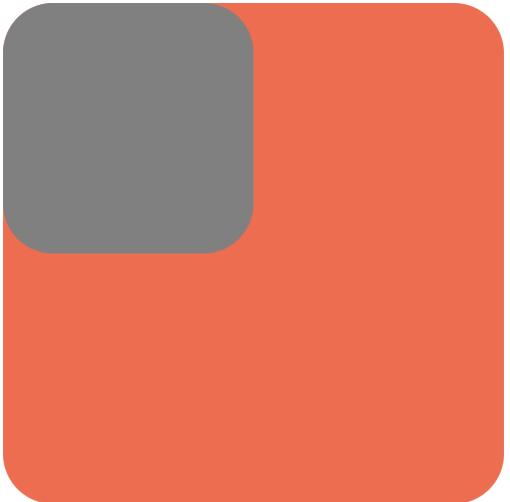
position: static이 아니면 z-index를 조절할 수 있다.
z-index 값이 작을수록 아래에 쌓이고, 값이 클수록 위에 쌓인다!



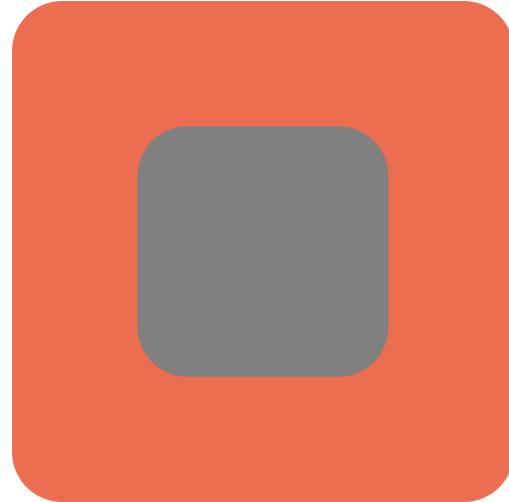
*z-index 값을 따로 지정하지 않으면 문서에 먼저 삽입하는 요소가
z-index: 1 값을 가지며, 후에 삽입되는 요소들은 점점 커짐

실습 1. position 이해하기

실습 1 폴더



실습 1 폴더의 이 녀석을

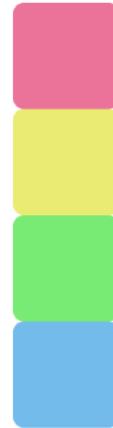


이렇게 만들어보세요

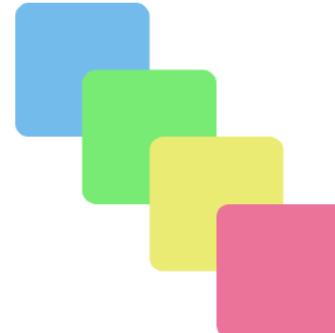
실습 2. position 이해하기

실습 2 풀더

아래의 이미지처럼 만들어보기!



before



after

***핑크색 박스가 가장 앞에 와야 함!

`display: flex;`

| display 속성

개념 이해

display 속성이란?

화면에 요소를 어떻게 표시할지를 선택하는 속성!

| display 속성

| **display: inline;**

| **display: inline-block;**

| **display: block;**

| **display: none;**

display: inline

가로로 쭈우우우우우우

inline은 다음 요소를 줄 바꿈 하지 않는 것을 의미한다!

쉽게 말하면 다음 요소가 가로로 쌓임

inline인 width와 height를 지정할 수 없고 (해도 무시됨),
margin의 상하 간격 조절도 안된다.

, , <i>, <a> 등

| display: block

세로로 쭈우우우우우우

block은 다음 요소를 줄 바꿈 하는 것을 말한다!

쉽게 말하면 다음 요소가 세로로 쌓임

width, height 속성을 지정할 수 있다.

<div>, <p>, <h>, 등

display: inline-block

inline과 block 사이 어딘가

(**inline**처럼) 줄 바꿈이 되지 않지만,
(block처럼) **width**와 **height**를 지정할 수 있다!

margin 상하 조절도 가능함

display: none;

사라진다. visibility: hidden과는 다른 것!

요소를 화면에 표시하지 않는다!

display: flex;

CSS계의 본좌. 유일한 빛. 인민의령도자. 그 이름 flex.



그 전에 단위 공부부터!!

앞으로 px은 쓰지 마세요

절대 단위

px

상대 단위

%

em

rem

vh, vw

그 전에 단위 공부부터!!

px

px이란?

절대 단위로, 1/96th of 1 in

디바이스의 해상도에 따라서 다르기 때문에,
여러 디바이스 모두를 고려하는 상황에서는 부적절하다.

그 전에 단위 공부부터!!

%

%란?

상대 단위로,
부모 요소를 기준으로 비율이 표현된다

그 전에 단위 공부부터!!

em

em이란?

상대 단위로,
부모 요소의 폰트 사이즈에 대한 상대적인 사이즈를 의미한다!

예) 부모 요소의 폰트 크기가 14px라면,
 $1\text{em} = 14\text{px}$, $1.2\text{em} = (14 * 1.2)\text{px} = 16.8\text{px}$

부모 요소의 폰트 크기가 정해져 있지 않다면,
 $1\text{em} = 16\text{px}$

그 전에 단위 공부부터!!

rem

rem이란?

상대 단위로,
최상위 요소의 크기를 기준으로 하는 단위!

rem = root em

그 전에 단위 공부부터!!

vh, vw

vh와 vw란?

상대 단위로,
viewport를 기준으로 하는 단위!

vw (viewport width): viewport 너비의 1/100
vh (viewport height): viewport 높이의 1/100

display: flex;

다 flex 했어요

flex (유연성)의 뜻처럼, 요소들을 자유자재로 위치 시킬 수 있다!

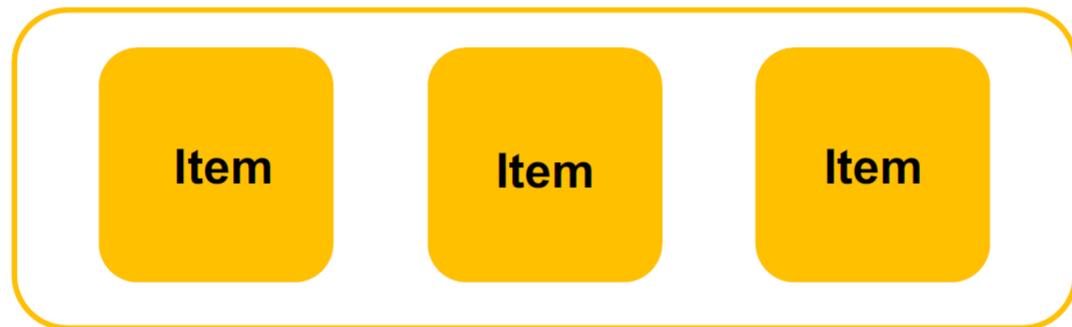
flex도 레이아웃 배치를 위한 것..!

display: flex;

flex-box의 구성요소

Flexbox는 “container”와 “items”로 이루어져 있다!

Container (item들을 감싸는 부모)



Container은 flex의 영향을 받는 전체 공간이고,
설정된 속성에 따라서 각각의 item이 어떤 형태로 배치 된다

display: flex;

flex-box의 구성요소

Flex의 속성들은 두 가지로 나뉜다!

1. Container에 적용하는 속성

display
flex-direction
flex-wrap
justify-content
align-content
align-items

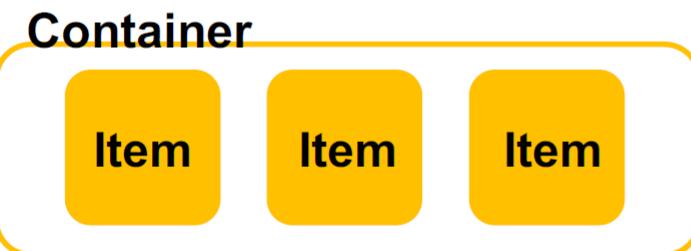
2. Item에 적용하는 속성

order
flex-grow
flex-shrink
align-self

display: flex;

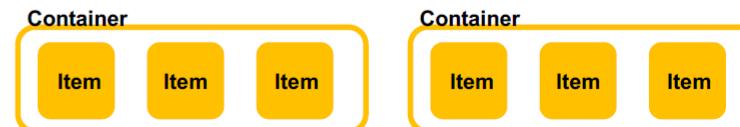
속성1 display

요소를 Container로 정의하는 것!



: 지정된 flex container은 block 요소의 성향을
가진다 (수직으로 쌓임)

반대로 display: inline-flex;는 inline 요소의
성향을 가진다 (수평으로 쌓임)



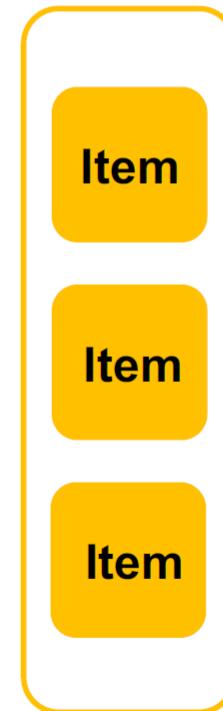
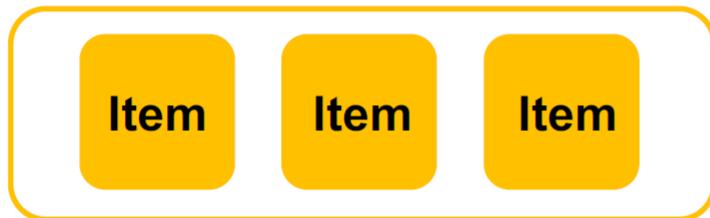
display: flex;

속성2 flex-direction

Items의 주 축 (main-axis)을 설정하는 것!

기본값

flex-direction: row;



flex-direction: column;

이외에도 column-reverse, row-reverse도 있음!

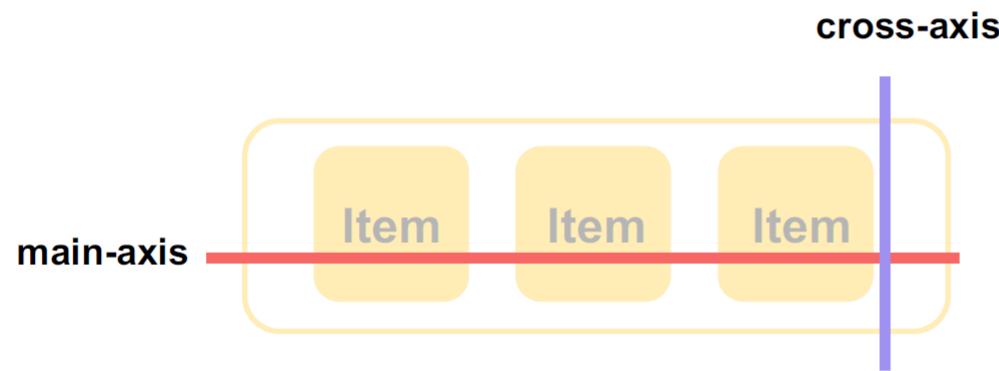
* column-reverse: column의 반대 축으로 표시

* row-reverse: row의 반대 축으로 표시

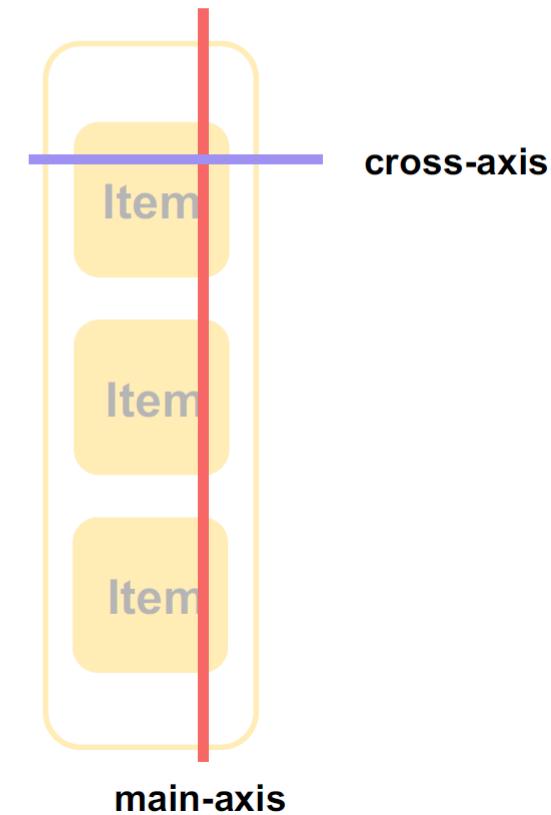
display: flex;

주축과 교차축의 개념

주 축 (main axis)와 교차 축 (cross axis)



main-axis: item이 배치되는 방향의 축
cross-axis: main-axis와 수직인 축



display: flex;

속성3 flex-wrap

flex-wrap이란?

flex item이 flex container 영역을 넘어갈 경우
줄 바꿈을 할지를 결정하는 속성이다!

nowrap (기본값): 줄바꿈 하지 않고 item은 한 줄로 배치됨

wrap: 줄 바꿈 되어 item이 배치됨

wrap-reverse: item이 wrap의 역순으로 배치됨

display: flex;

속성4 justify-content

justify-content란?

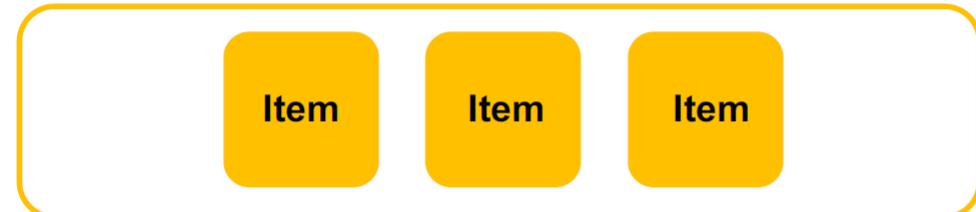
item의 main-axis를 정렬하는 속성!

기본값

justify-content: flex-start;



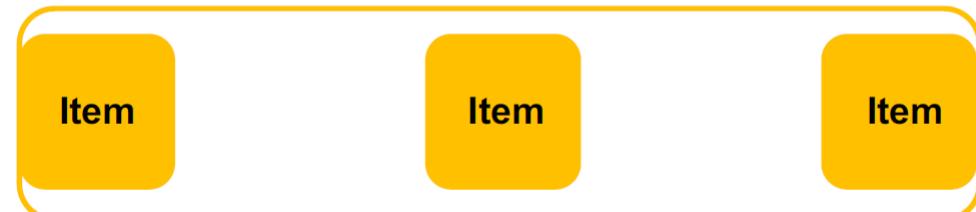
justify-content: center;



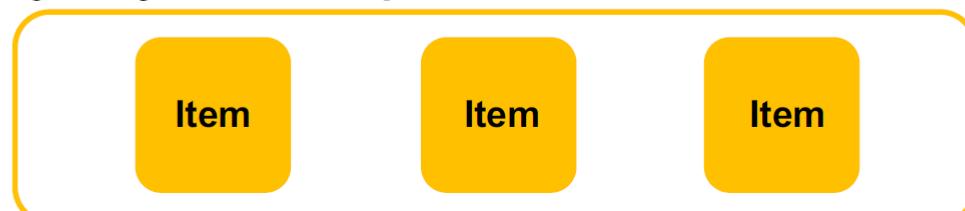
justify-content: flex-end;



justify-content: space-between;



justify-content: space-around;



display: flex;

속성4 justify-content

justify-content의 예시?
실제 홈페이지에서 찾아보자!



display: flex;

속성5 align-content

align-content란?

item의 cross-axis를 정렬하는 속성!

align-items와 비슷하지만,
item이 flex-wrap: wrap;으로 2줄 이상 나열되어 있을 때 사용!



stretch (기본값), flex-start, flex-end, flex-center, space-between, space-around NEXT X LIKELION

display: flex;

속성6 align-items

align-items란?

item의 cross-axis를 정렬하는 속성!

기본값

align-items: stretch;

container의 높이만큼 교차 축 방향으로
item을 늘려, 전체 높이를 채움

align-items: flex-end;

교차 축의 마지막 지점을 기준으로 item 정렬

align-items: flex-start;

교차 축의 시작 지점을 기준으로 item 정렬

align-items: center;

교차 축의 item을 가운데로 정렬

display: flex;

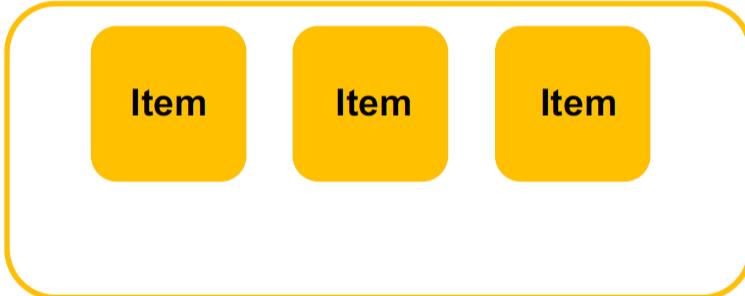
속성6 align-items

align-items란?

item의 cross-axis를 정렬하는 속성!

기본값

align-items: stretch;



align-items: flex-end;



align-items: flex-start;



align-items: center;



display: flex;

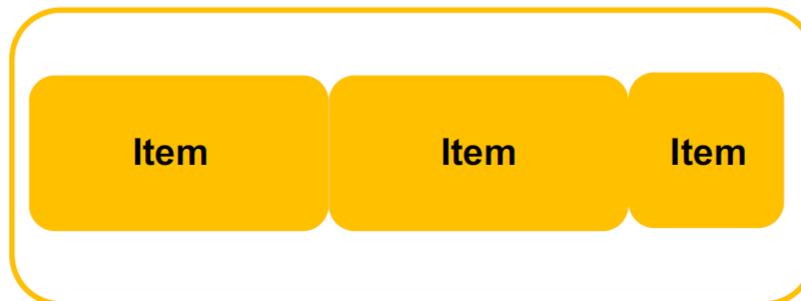
item 속성1 flex grow

flex-grow란?

flex item의 확장 너비 비율을 설정하는 속성!

기본값은 0이며, 숫자를 키울수록 너비가 확장된다

예) item 3개의 flex-grow가 각각 2, 2, 1일 때



display: flex;

item 속성2 flex shrink

flex-shrink란?

flex item의 축소 너비 비율을 설정하는 속성!

기본값은 0이며, 숫자를 키울수록 너비가 축소된다

display: flex;

item 속성3 align-self

align-self란?

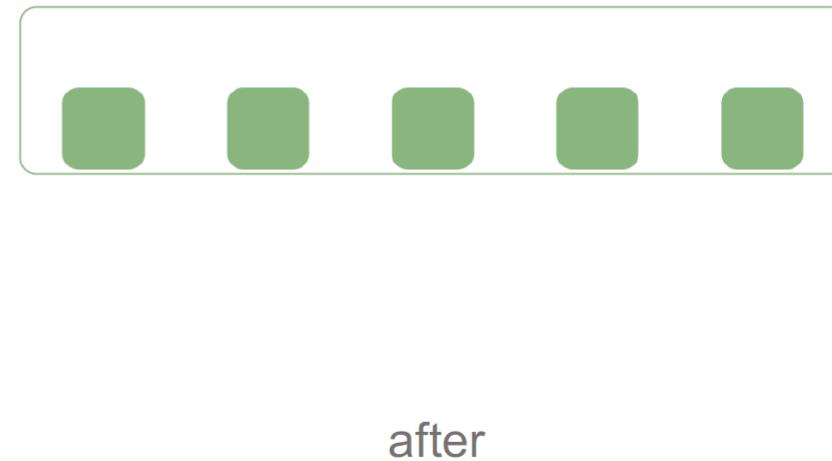
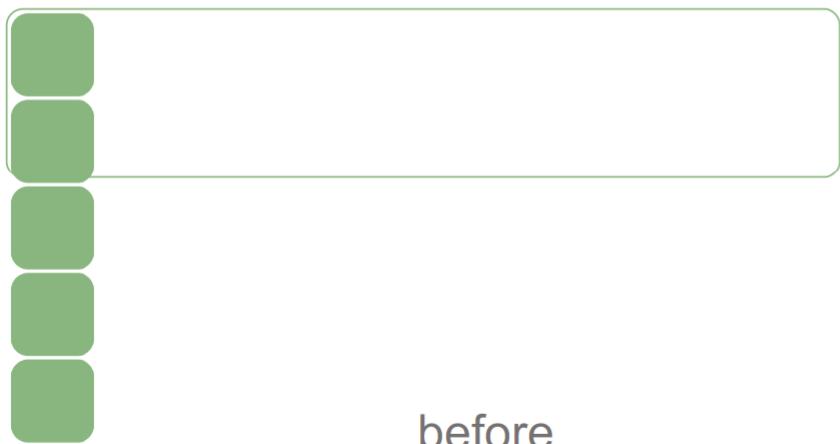
개별 item의 교차 축을 정렬하는 속성!

align-items 가 Container 안에 있는 모든 item의 교차 축을 정렬한다면,
align-self는 필요한 요소만 개별적으로 정렬하고 싶은 경우에 사용하면 된다

실습 3: flex

실습3 폴더

flex 활용한 실습 1



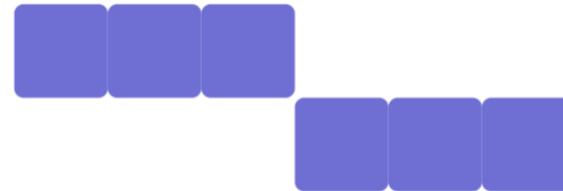
실습 4: flex

실습4 폴더

flex 활용한 실습 2



before

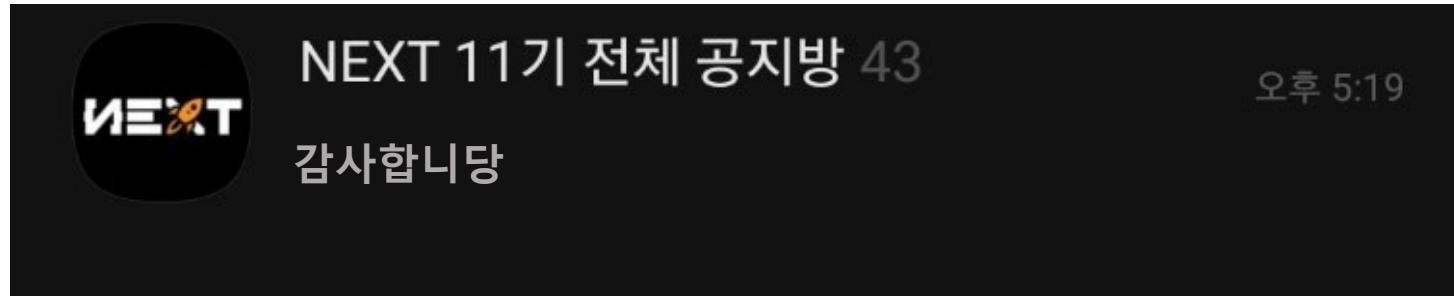


after

실습 5: 레이아웃

실습5 풀더

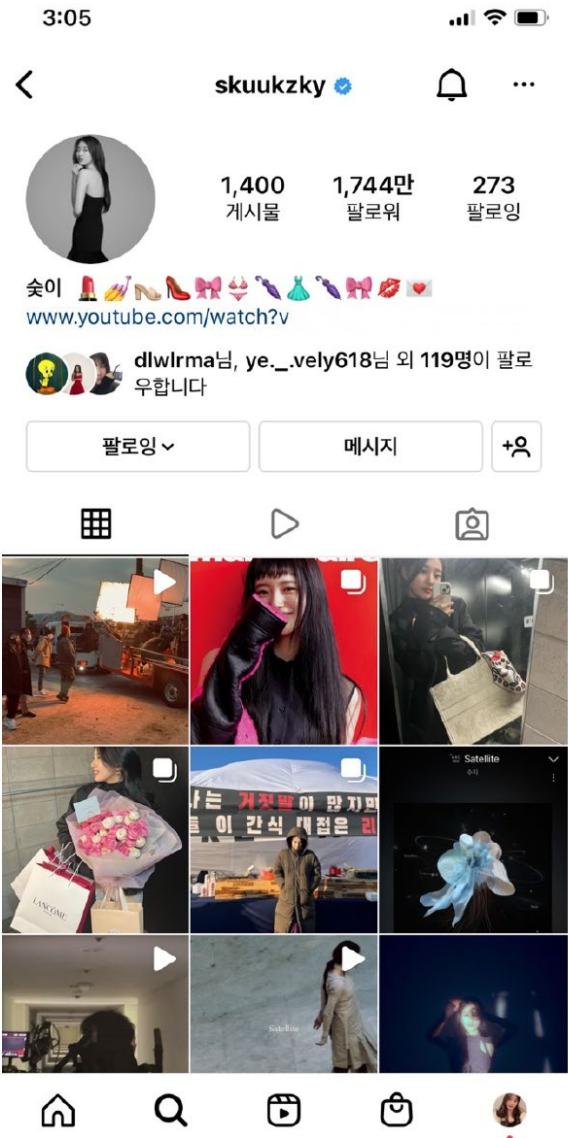
flex 활용한 실습 3 (심화)



자유롭게 만들어보세요!

과제: 인스타그램 클론코딩

실습3 풀더



아이콘은
fontawesome, bootstrap-icons 등 다양한
아이콘 라이브러리를 사용해보세요!