

FESSION

WEEK 04

LEAD. JUNKYU LEE

- 01** 과제 점검 + 4 ~ 6주차 세션 개요
- 02** CSS Module + 여러분이 공부해야 할 방향
- 03** 고급 스타일링
- 04** 프론트엔드 코드잼! (MBTI 테스트 만들기)
- 05** Figma 기초
- 06** 과제 공지





과제... 잘 하셨나요?

많이 어려웠나요? 힘들었나요? 할만했나요? 어땠나요?

진짜 착각임 페이지 재활용 아님 진짜임!!!!!!

통합 세션이 사라졌습니다! 사실 완전 사라진건 아니고... 하긴 하는데...

백엔드 세션이 생각보다 촉박하다고해서, 백엔드 파트장님과 함께 새롭게 일정을 조율해봤어요!

자바스크립트 기초(1)

자바스크립트라는 언어와 기초 문법

자바스크립트 기초(2)

객체 & 함수와 클래스, 이벤트 핸들링

React

React 기본과 모던 웹 패러다임의 이해

고급 스타일링 + Figma

가상 클래스 / 선택자 + 애니메이션 / 반응형

과제) 나만의 일정 관리 어플리케이션 디자인

Open API를 활용한 Data Fetching

나만의 일정 관리 어플리케이션 API 테스트

2주 휴강

과제) 나만의 일정관리 어플리케이션 퍼블리싱 + 개발

나만의 일정 관리 어플리케이션 완성

프론트 SEMI Hackthon! => 완성 및 서비스 배포

최종 과제) 백엔드 짹지가 만들어준 API로 마이그레이션

아니 저는 아직 **React** 잘 모르겠는데요 뭔 어플리케이션을 완성하래

오케이 좋아요... 그러니까 오늘 리액트 찐으로 뿐셔봅시다!

스타일링은 덤으로!!!! (원래 스타일링이 메인이었으나 님들 과제해오는거 보니까 강 개 고수 같음...)

라이브 코드잼으로 진행할거니까 무서워 말자구용

오늘 우리는 MBTI테스트를 만들어볼겁니다. 시간이 없어요 빨리 복습하고 배울거 배우고 빨리 실습하러 넘어갈겁니다. 그게 어떻게 되냐고 생각해도... 우린 항상 말도안되는 세션을 다 해온 프론트였자나요



CSS의 치명적인 단점을 극복해보자

CSS Module

개발자가 가장 싫어하는 것 1위) 이름 짓기... 왜?

```
#nav > li a { color: red }
# nav > li a.selected { color : blue }
strong { color : blue }
```

NestedNestedNestedNestedNestedNested

```
#nav > li a { color: red }
# nav > li a.selected { color : blue }
strong { color : blue }
#name > span { color : blue }
#link-first > span { color : blue }
#link-second > a { color : blue }
#link-third > a { color : blue }
```

타자연습이냐고



CSS의 악마 소환진

CSS의 치명적인 단점을 극복해보자

CSS Module

개발자가 가장 싫어하는 것 1위) 이름 짓기... 왜?

모던 어플리케이션은 대개 컴포넌트를 기준으로 개발된다.
하지만, Global Scope와 Specificity에 부숴지는 작고 귀여운 CSS

아니 우리는 이제 문
서공유가 아니라 어플리케이션을 만들
고싶다구요
예암풀킥통보이

Global Scope

- 수정한 코드가 모든 곳에 영향
- 사용된 이름을 피해야 함

Cascade Specificity

- CSS 순서에 의한 우선순위
- 선택자의 복잡성 증가
- !import 지옥

Scope를 지역을 기준으로 제한하도록, 모듈화 하자!

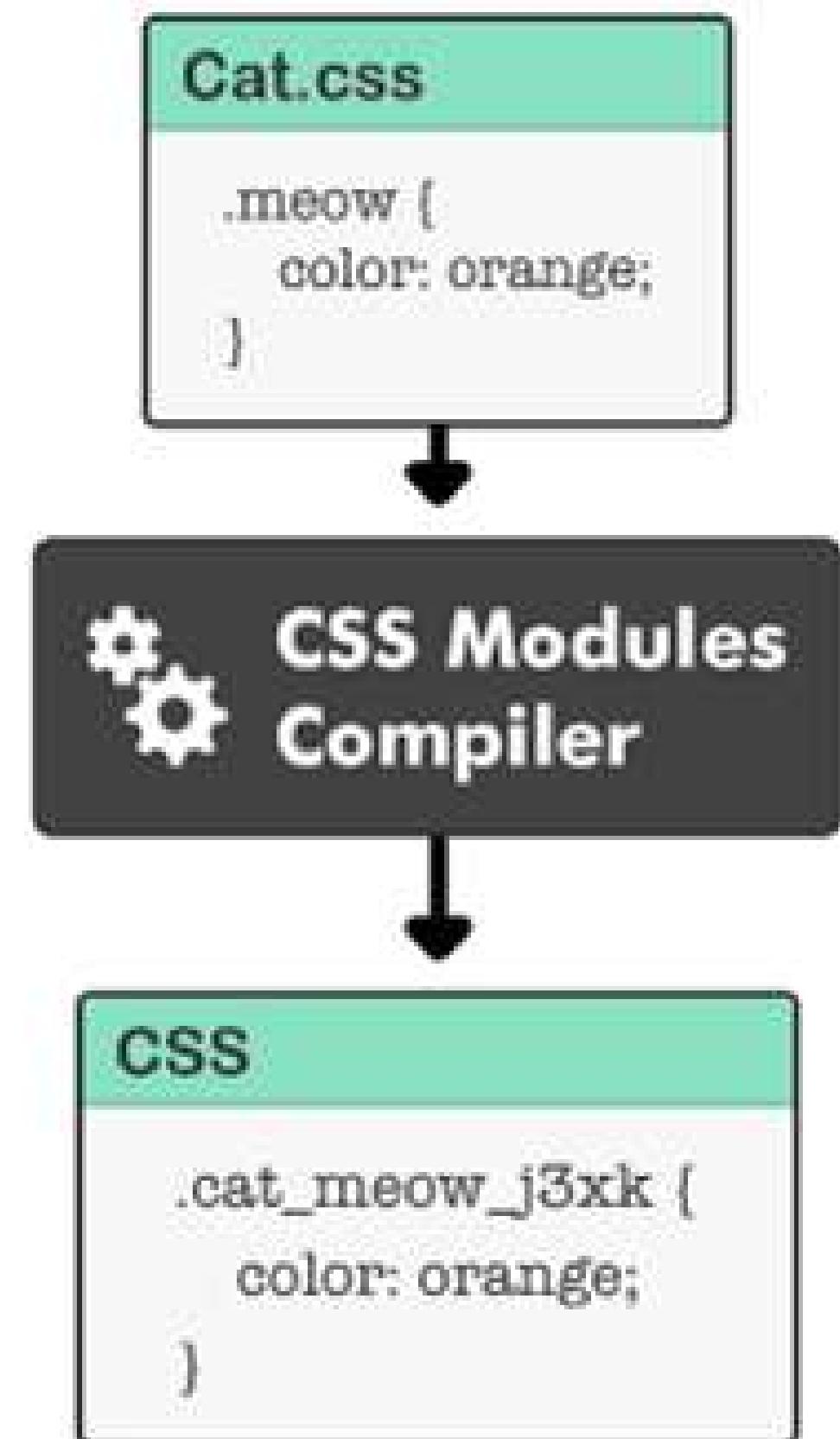


CSS Modules (1세대 CSS-In-JS)

CSS의 치명적인 단점을 극복해보자

CSS Module

자! 이제 마음껏 중복 이름 써봐라 마!



CSS의 치명적인 단점을 극복해보자

CSS Module

테스트 해볼까요?



우리는 프론트니까 두 눈으로 똑똑히 봐야됨

CSS의 치명적인 단점을 극복해보자

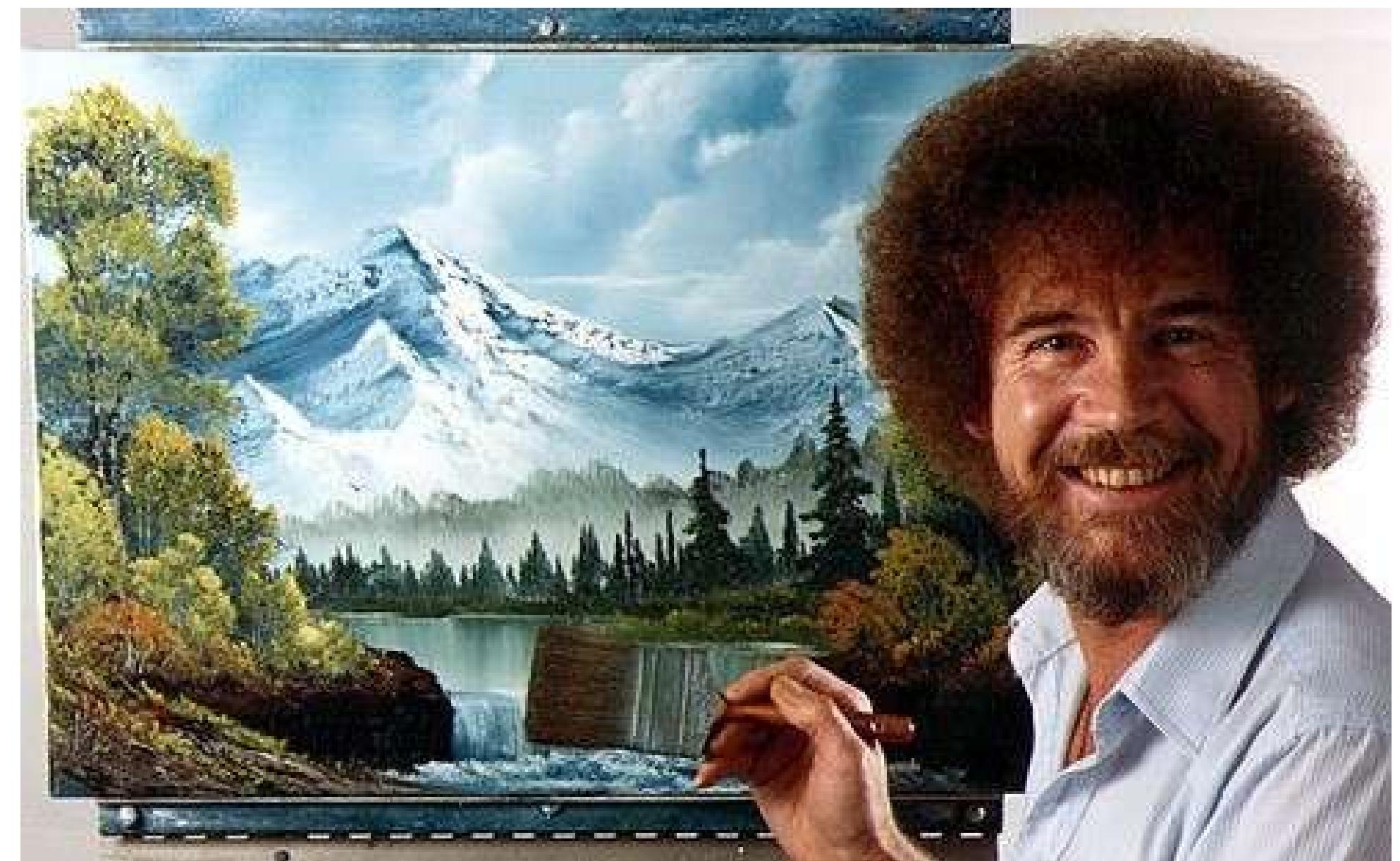
CSS Module

그래서 어떻게 쓰는데?

import **styles** from "CSS 파일 경로"



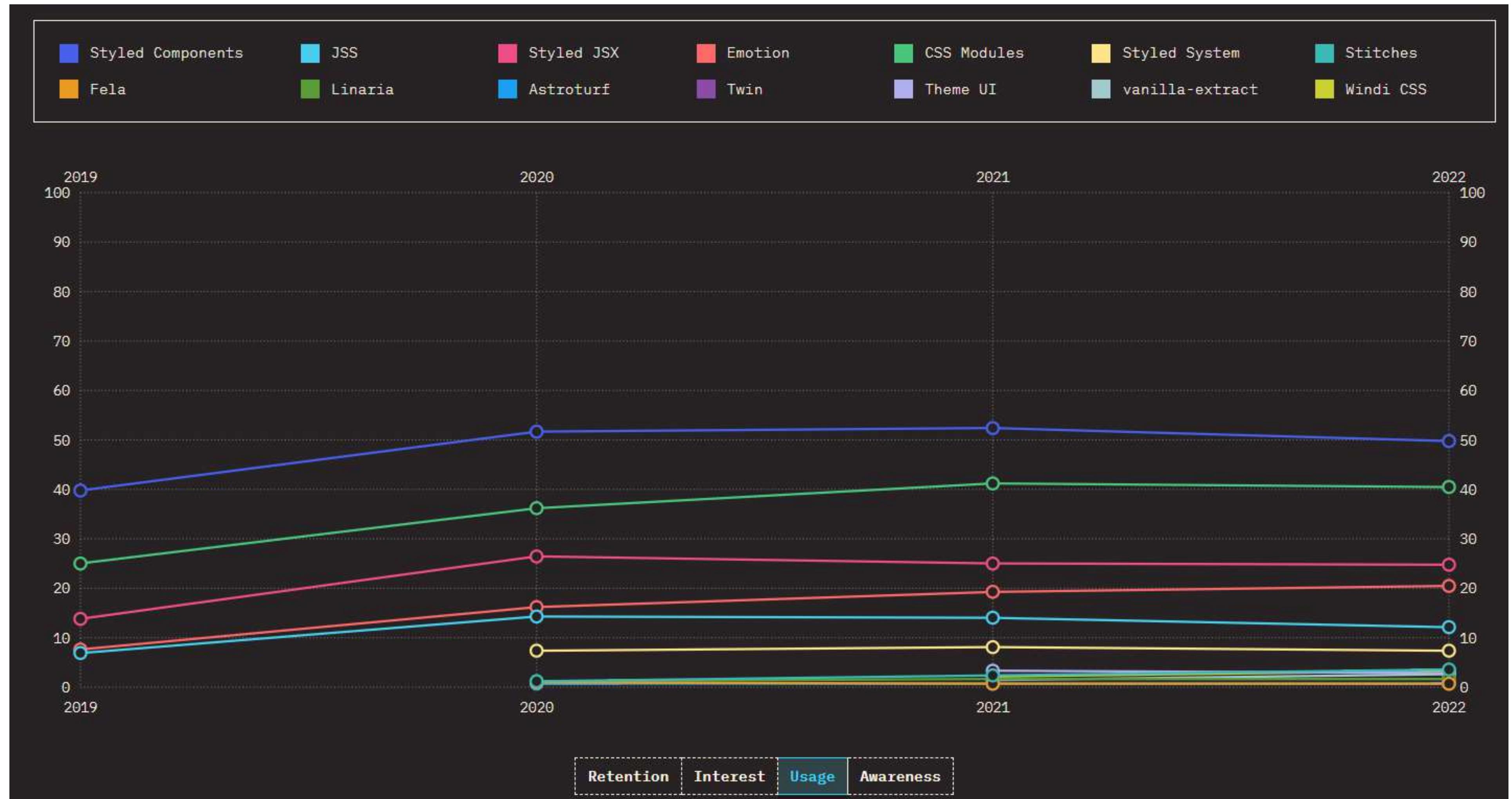
className="styles.클래스명"



CSS의 치명적인 단점을 극복해보자

CSS Module

망할 일 없음



CSS Module은 장점이 뚜렷한 만큼, 한계도 명확해요! 덤으로 스타일링 생태계는 빠르게 변화한다는 점까지...

CSS Module은 3세대 스타일링 기법입니다. 그럼에도 많이 쓰이는건, 레거시를 유지하고 있기 때문 + 아직 생태계가 정립되지 않았기 때문



CSS 전처리기(SCSS)

CSS의 슈퍼셋 문법으로, 이전까지 불편했던 점들을 보완함.
성능상의 이슈가 있음 (전처리 문제)



CSS-In-CSS CSS 전처리기의 등장

공통 요소 / 반복문을 변수나 함수로 대체
개발 시간 감소
CSS 파일을 여러개로 나누어서 유지보수성 향상
구조화된 코드로 유지 및 관리해서 유지보수성 향상

```
#nav > li a{  
    color : red  
    &.selected { color : $primary }  
}  
strong { color : $primary }
```

CSS In JS

자바스크립트에서 스타일링까지 해결해버리기



CSS-In-JS 일자리 잃은 CSS, 해결사 CSS-In-JS 등장

Global namespace : 글로벌 명명 규칙 이슈
Dependencies : CSS간 의존 관계 이슈
Dead Code Elimination : 미사용 코드 이슈
Minification : 클래스명 오버플로 이슈
Sharing Constants : JS와 CSS간 상태 공유 이슈
Non-deterministic Resolution : CSS 로드 순서 이슈
Isolation : CSS와 JS 상속에 따른 격리 필요 이슈
...
CSS 이슈 다 해결해드림!

일단 이름짓기 먼저 해결해보자!

BEM (Block Element Modifier)

CSS 방법론

CSS Class Naming Convention



현재 CSS 방법론의 최종 승자는 BEM

구조 표현
종속 관계 X
Specificity가 하나로 관리됨
Convention이 단순

과연...?



일단 이름짓기 먼저 해결해보자!

BEM (Block Element Modifier)

The diagram shows a card component with the following structure:

```
<article class="card">
  <h1 class="card_title">Some engaging title</h1>
  <p class="card_text">There are many variations....</p>
  <p class="card_text card_text--secondary">The majority....</p>
  <a class="card_button button" href="#">Read more</a>
</article>
```

Annotations with orange arrows point from UI elements to their corresponding BEM classes:

- A curved arrow points from the **SOME ENGAGING TITLE** heading to the **card_title** class.
- A curved arrow points from the first paragraph of text to the **card_text** class.
- A curved arrow points from the secondary paragraph of text to the **card_text--secondary** class.
- A curved arrow points from the **READ MORE** button to the **card_button** class.

진짜 쉬움ㅋㅋ



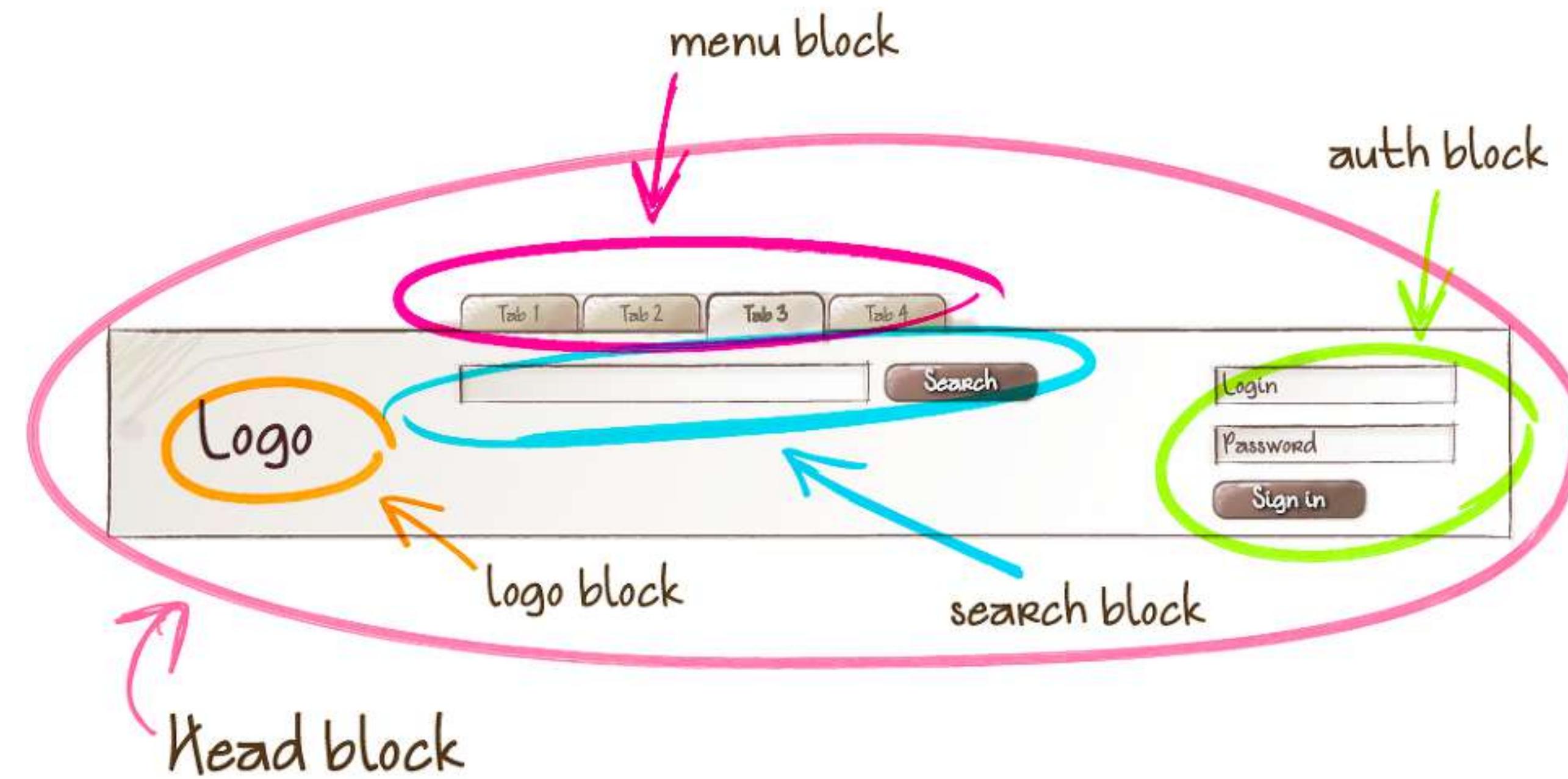
이걸로 구분!

어떻게 보이는가가 아닌, 어떤 목적으로 사용되는가! (e.g. 에러 메시지에 red가 아닌 error라는 이름을)



일단 이름짓기 먼저 해결해보자!

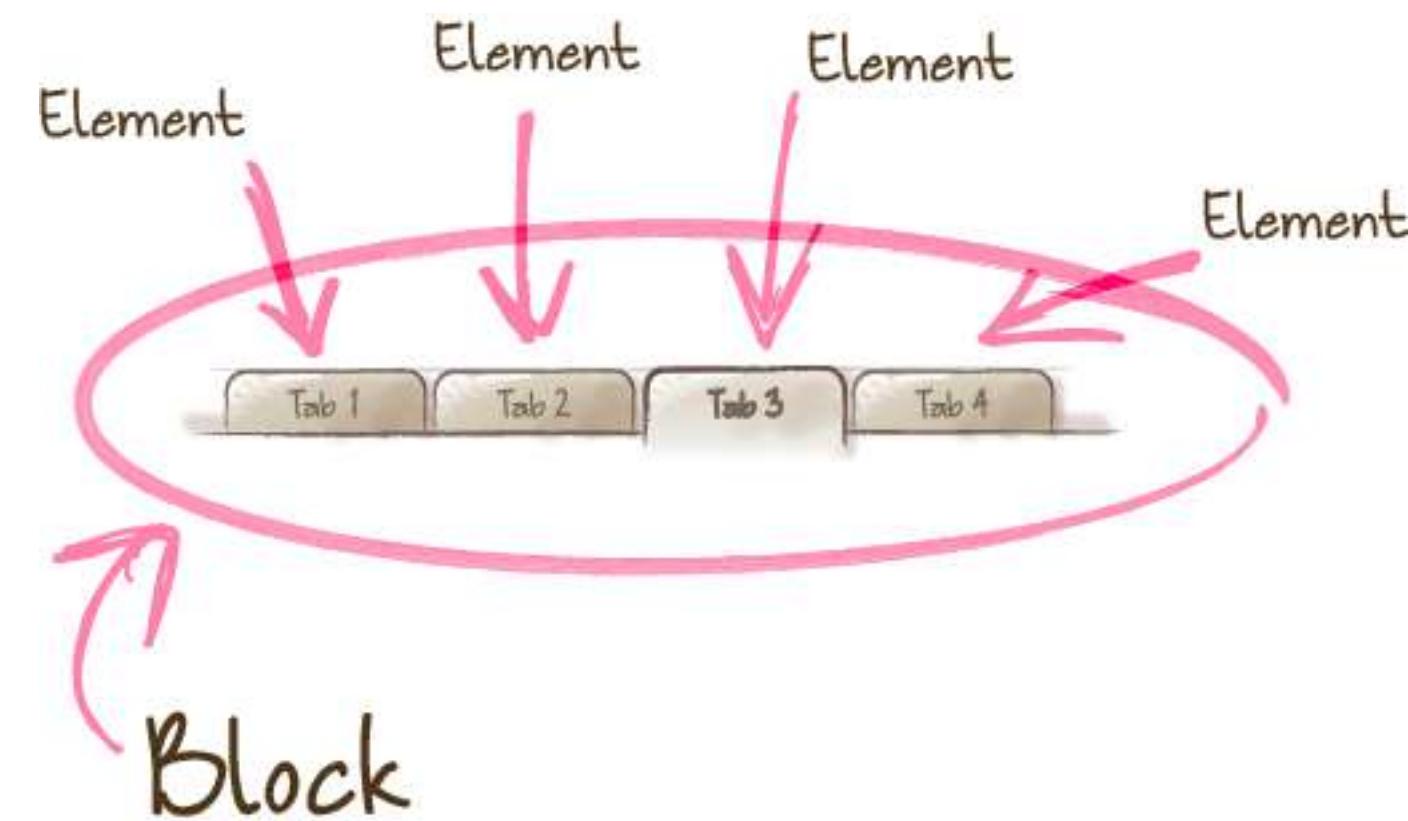
Block



재사용이 가능한 기능적으로 독립적인 컴포넌트를 블럭(block)이라 불러요.
독립된 컴포넌트이기 때문에, 여러 페이지에서 사용할 수 있습니다!

일단 이름짓기 먼저 해결해보자!

Element

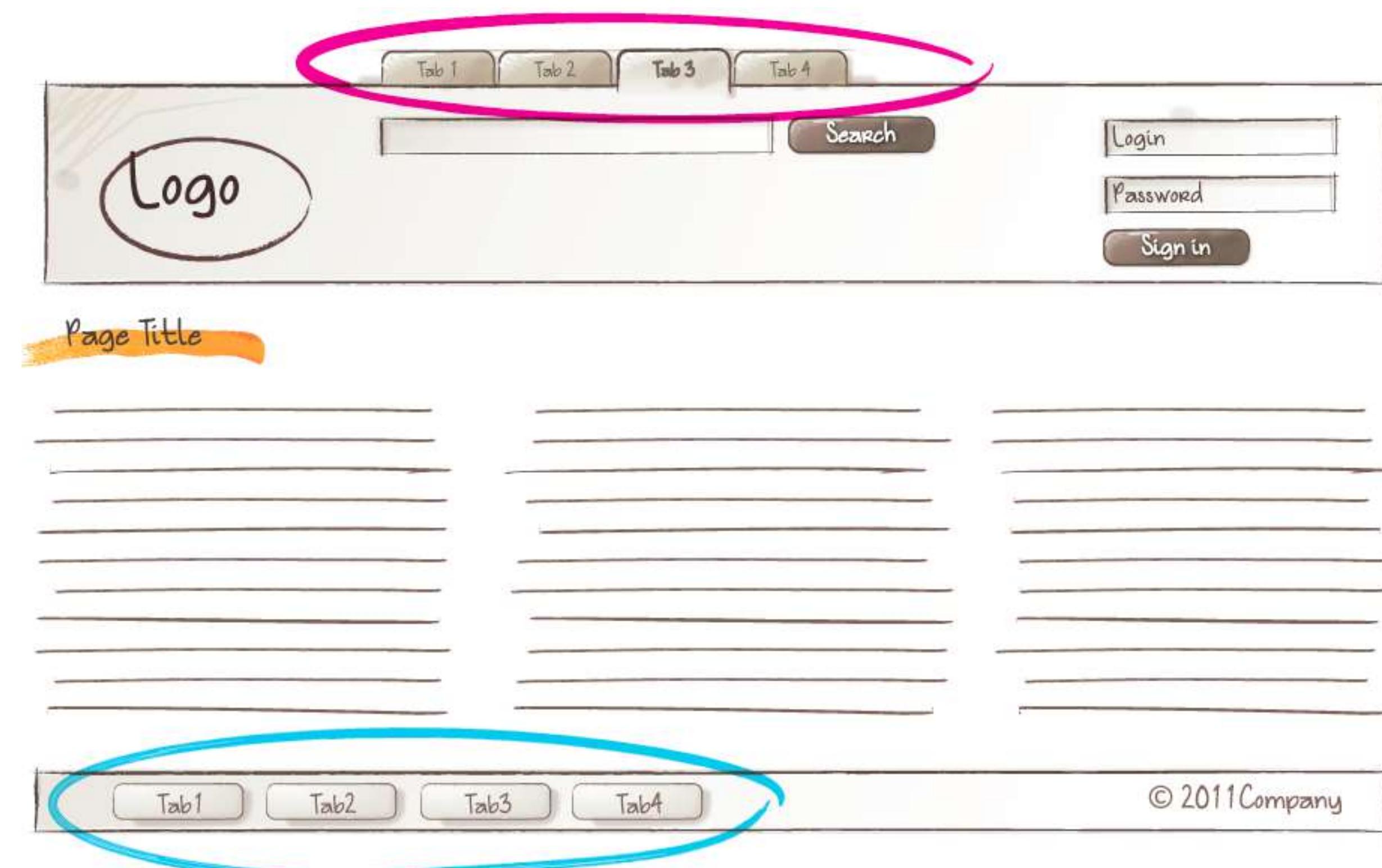


블럭을 구성하는 작은 요소들을 Element라고 불러요!

기능적으로 독립되지 않았기 때문에, 이를 떼어내서 다른 곳에서 사용할 수 없어요.

일단 이름짓기 먼저 해결해보자!

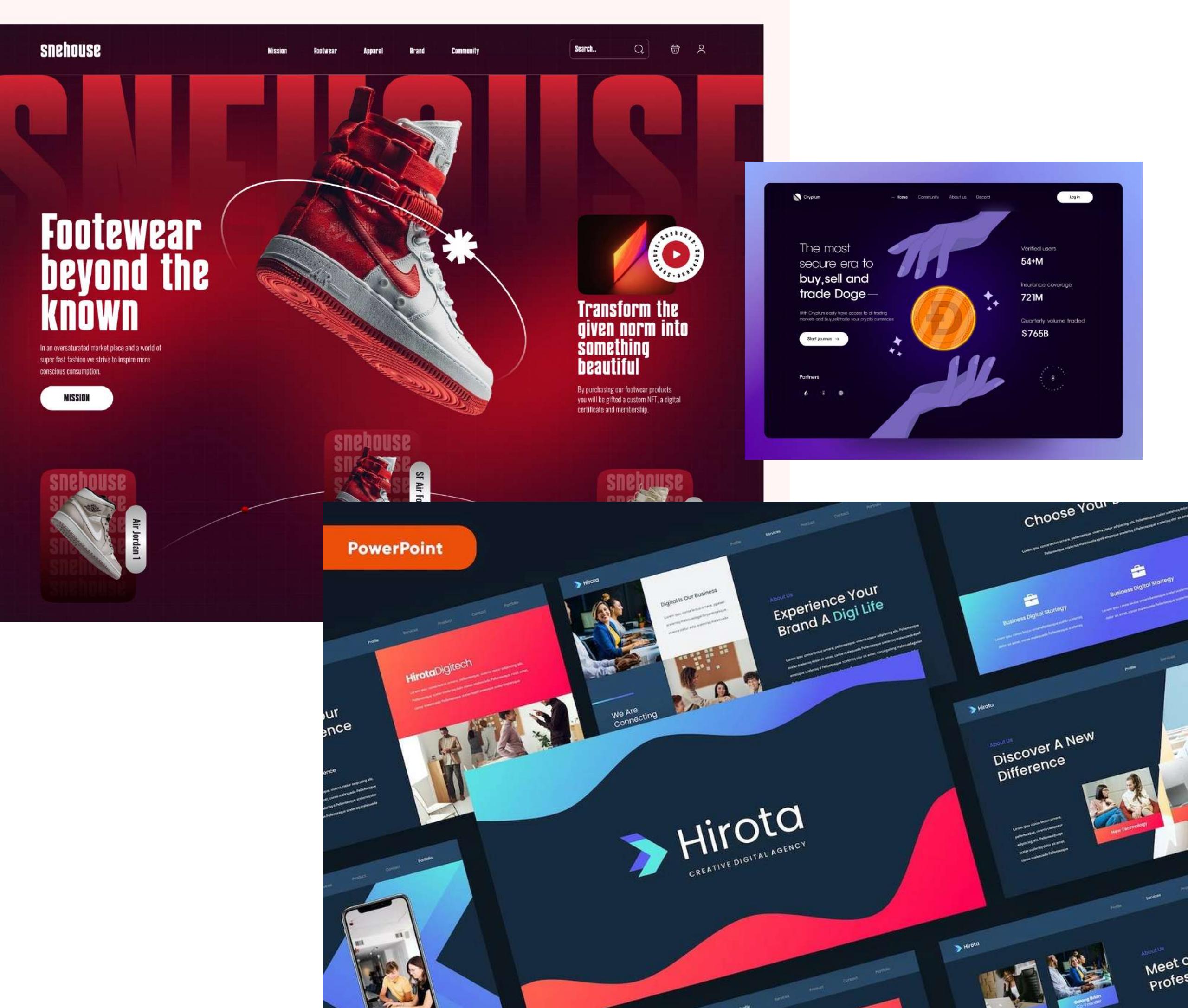
Modifier



블럭 내부의 Element들 중에서, 스타일이 조금 다르거나 다르게 동작하는 요소를 Modifier라고 해요!

고급 스타일링 마스터리 북

복합(일치 / 자식 / 후손) 선택자



이런 복잡한 스타일
CSS로 구현이 가능하긴한가요..?
제가 한 CSS는 레이아웃 나누기 뿐인데 | ...

고급 스타일링 마스터리 북

복합(일치 / 자식 / 후손) 선택자

이건 구현 못해요...



죄다 구현 못한다는 개발자가 되지 않기 위해 배워야하는, 선택자들!

고급 스타일링 마스터리 북

일치 선택자

```
<span>
  <div>1번 div</div>
  <div class="selectItem">2번 div</div>
  <div>3번 div</div>
  <p class="selectItem">p태그</p>
</span>
```

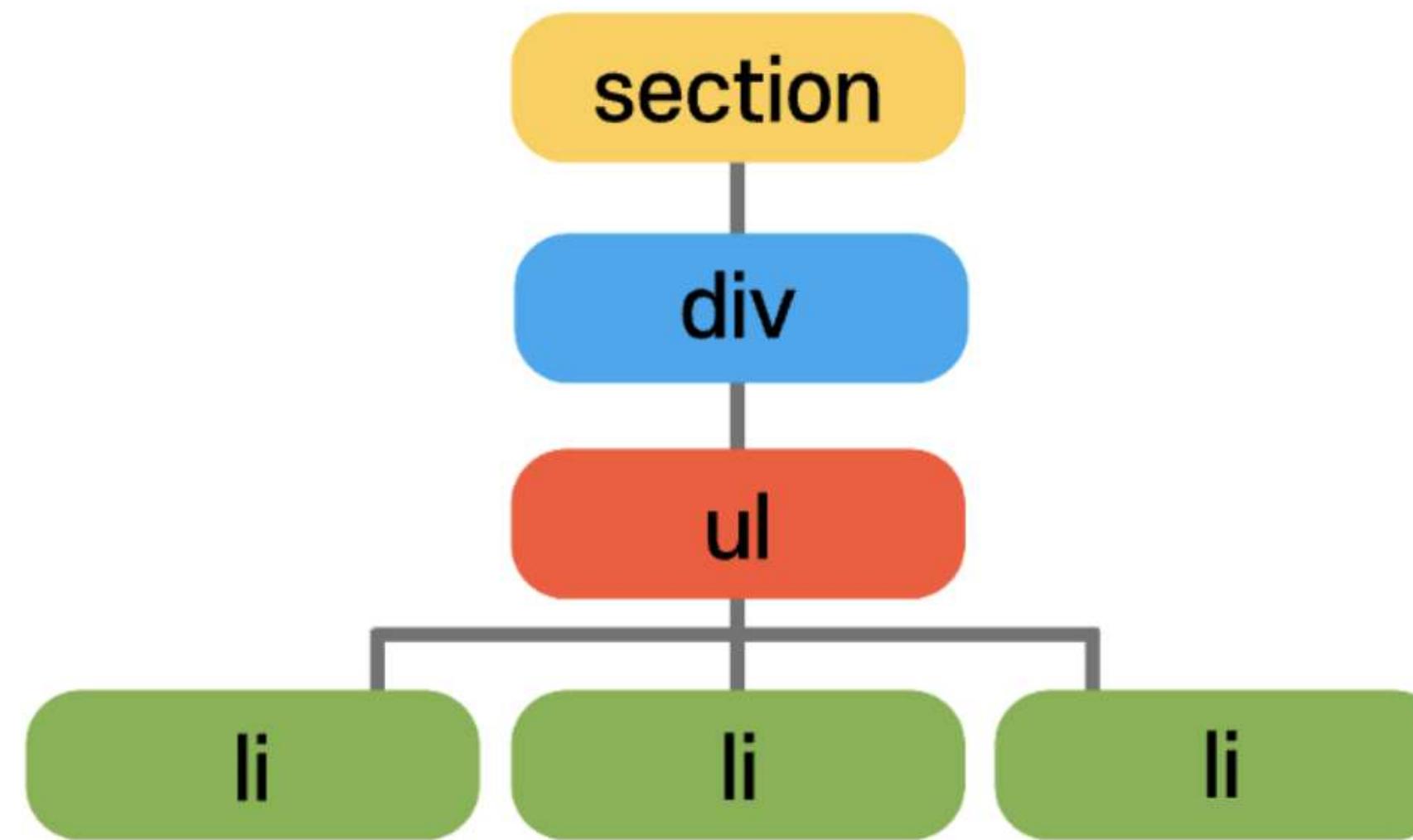
```
<style>
  div.selectItem {
    color: red;
  }
</style>
```

ab : a와 b의 조건을 동시에 만족하는 요소 선택
(선택자가 공백 없이 붙어있는 형태)

고급 스타일링 마스터리 북

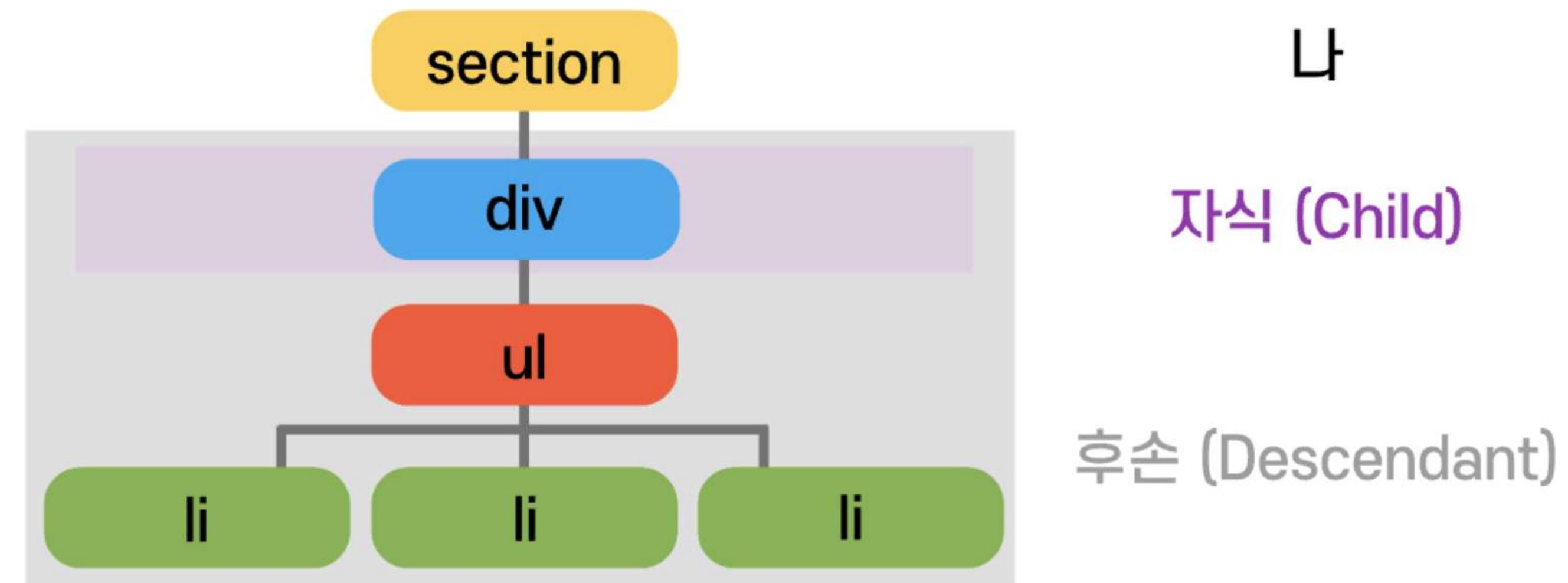
자식 선택자

```
<section>
  <div>
    <ul>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </div>
</section>
```



고급 스타일링 마스터리 북

자식 선택자



고급 스타일링 마스터리 북

자식 선택자

특정 요소 바로 밑에 있는 요소를 선택하는 선택자!

Depth가 1까지만 내려간다는 점 주의

```
<div>
  <ul>
    <li>child1</li>
    <li>child2</li>
    <li class="selectedChild">child3</li>
  </ul>
</div>
```

```
<style>
  ul > .selectedChild {
    color: orange;
  }
</style>
```

a > b : a의 자식 요소인 b를 선택

고급 스타일링 마스터리 북

후손 선택자

계층 구조에서 하위에 오는 자손 요소를 선택하는 선택자.
자식 요소 (depth 1)도 후손 요소에 포함된다.

```
<div>
  <ul>
    <li>li child1</li>
    <li>li child2</li>
    <li class="selectedChild">li child3</li>
  </ul>
  <div>div child1</div>
  <p>p child1</p>
  <div class="selectedChild">div child2</div>
</div>
```

```
<style>
  div .selectedChild {
    color: orange;
  }
</style>
```

a b : a의 하위 요소인 b를 선택
(선택자 사이 공백)

고급 스타일링 마스터리 북

의사 클래스

선택자에 추가하는 키워드, 선택요소가 특별한 상태여야 적용되는 스타일!

```
선택자:pseudo-class {
    속성 : 값;
}
```

표준 의사클래스만해도 양이 상당함.
정말 너무 많기 때문에 외우는건 불가능하고,
필요할 때 검색해서 쓰자!

- [:active](#)
- [:any-link_\(en-US\)](#)
- [:blank_\(en-US\)](#)
- [:checked](#)
- [:current_\(en-US\)](#)
- [:default](#)
- [:defined](#)
- [:dir\(\)\(en-US\)](#)
- [:disabled](#)
- [:drop](#)
- [:empty_\(en-US\)](#)
- [:enabled](#)
- [:first](#)
- [:first-child](#)
- [:first-of-type](#)
- [:fullscreen](#)
- [:future_\(en-US\)](#)
- [:focus](#)
- [:focus-visible_\(en-US\)](#)
- [:focus-within](#)
- [:host_\(en-US\)](#)
- [:host\(\)\(en-US\)](#)
- [:host-context\(\)](#)
- [:hover](#)
- [:indeterminate_\(en-US\)](#)
- [:in-range_\(en-US\)](#)
- [:invalid_\(en-US\)](#)
- [:is\(\)\(en-US\)](#)
- [:lang\(\)\(en-US\)](#)
- [:last-child](#)
- [:last-of-type_\(en-US\)](#)
- [:left_\(en-US\)](#)
- [:link](#)
- [:local-link_\(en-US\)](#)
- [:not\(\)](#)
- [:nth-child\(\)](#)
- [:nth-col\(\)\(en-US\)](#)
- [:nth-last-child\(\)\(en-US\)](#)
- [:nth-last-col\(\)\(en-US\)](#)
- [:nth-last-of-type\(\)\(en-US\)](#)
- [:nth-of-type\(\)\(en-US\)](#)
- [:only-child_\(en-US\)](#)
- [:only-of-type_\(en-US\)](#)
- [:optional_\(en-US\)](#)
- [:out-of-range_\(en-US\)](#)
- [:past_\(en-US\)](#)
- [:placeholder-shown_\(en-US\)](#)
- [:read-only_\(en-US\)](#)
- [:read-write_\(en-US\)](#)
- [:required_\(en-US\)](#)
- [:right_\(en-US\)](#)
- [:root](#)
- [:scope_\(en-US\)](#)
- [:state\(\)](#)
- [:target_\(en-US\)](#)
- [:target-within_\(en-US\)](#)
- [:user-invalid_\(en-US\)](#)
- [:valid_\(en-US\)](#)
- [:visited](#)
- [:where\(\)\(en-US\)](#)

고급 스타일링 마스터리 북

의사 클래스

선택자에 추가하는 키워드, 선택요소가 특별한 상태여야 적용되는 스타일!

의사 클래스	설명
:active	선택된 요소가 활성화되었을 때 스타일을 지정합니다.
:hover	마우스 커서가 요소 위에 있을 때 스타일을 지정합니다.
:focus	요소가 포커스를 받았을 때 스타일을 지정합니다.
:visited	이미 방문한 링크를 나타내는 스타일을 지정합니다.
:link	방문하지 않은 링크를 나타내는 스타일을 지정합니다.
:first-child	요소의 첫 번째 자식 요소를 선택하여 스타일을 지정합니다.
:last-child	요소의 마지막 자식 요소를 선택하여 스타일을 지정합니다.
:nth-child(n)	요소의 n번째 자식 요소를 선택하여 스타일을 지정합니다.
:nth-last-child(n)	요소의 뒤에서부터 n번째 자식 요소를 선택하여 스타일을 지정합니다.
:first-of-type	동일한 태입의 요소 중 첫 번째 요소를 선택하여 스타일을 지정합니다.
:last-of-type	동일한 태입의 요소 중 마지막 요소를 선택하여 스타일을 지정합니다.
:nth-of-type(n)	동일한 태입의 요소 중 n번째 요소를 선택하여 스타일을 지정합니다.
:nth-last-of-type(n)	동일한 태입의 요소 중 뒤에서부터 n번째 요소를 선택하여 스타일을 지정합니다.
:only-child	부모 요소의 유일한 자식 요소를 선택하여 스타일을 지정합니다.
:only-of-type	동일한 태입의 요소 중 유일한 요소를 선택하여 스타일을 지정합니다.
:empty	자식 요소를 가지고 있지 않은 요소를 선택하여 스타일을 지정합니다.
:target	URL의 해시 대상이 되는 요소를 선택하여 스타일을 지정합니다.

요즘 세상이 좋아져서...

걔 지피티가 많이 쓰이는 의사 클래스 정리해줬음ㅋㅋ

:not(selector)	지정된 선택자(selector)에 해당하지 않는 요소를 선택하여 스타일을 지정합니다.
:enabled	활성화된 상태의 입력 요소를 선택하여 스타일을 지정합니다.
:disabled	비활성화된 상태의 입력 요소를 선택하여 스타일을 지정합니다.
:checked	선택된 상태의 체크 박스나 라디오 버튼을 선택하여 스타일을 지정합니다.
:first-line	요소의 첫 번째 줄을 선택하여 스타일을 지정합니다.
:first-letter	요소의 첫 번째 글자를 선택하여 스타일을 지정합니다.

고급 스타일링 마스터리 북

의사 클래스

선택자에 추가하는 키워드, 선택요소가 특별한 상태여야 적용되는 스타일!

```
<div>
  <button class="checkBtn">체크버튼</button>
</div>
```

```
<style>
  .checkBtn {
    margin: 10px;
    padding: 5px;
  }
  .checkBtn:hover {
    background-color: #aqua;
    font-weight: bold;
  }
</style>
```

체크버튼



체크버튼

고급 스타일링 마스터리 북

애니메이션 (@keyframe)

동적인 웹 어플리케이션의 생명은 애니메이션!

기본 동작 구조

애니메이션

: 기본적으로, 어떤 동작을 기반하여 이루어짐

2

우리는 그 동작을 구간별로 분리하여 keyframe 속성을 통해 애니메이션을 제작할 것
CSS를 통해 특정 구간에서 어떤 스타일이 적용될지 정의

3

선택자에게 animation 속성을 통해, 제작한 애니메이션을 전달
e.g. animation: fadeln 1s infinite;

고급 스타일링 마스터리 북

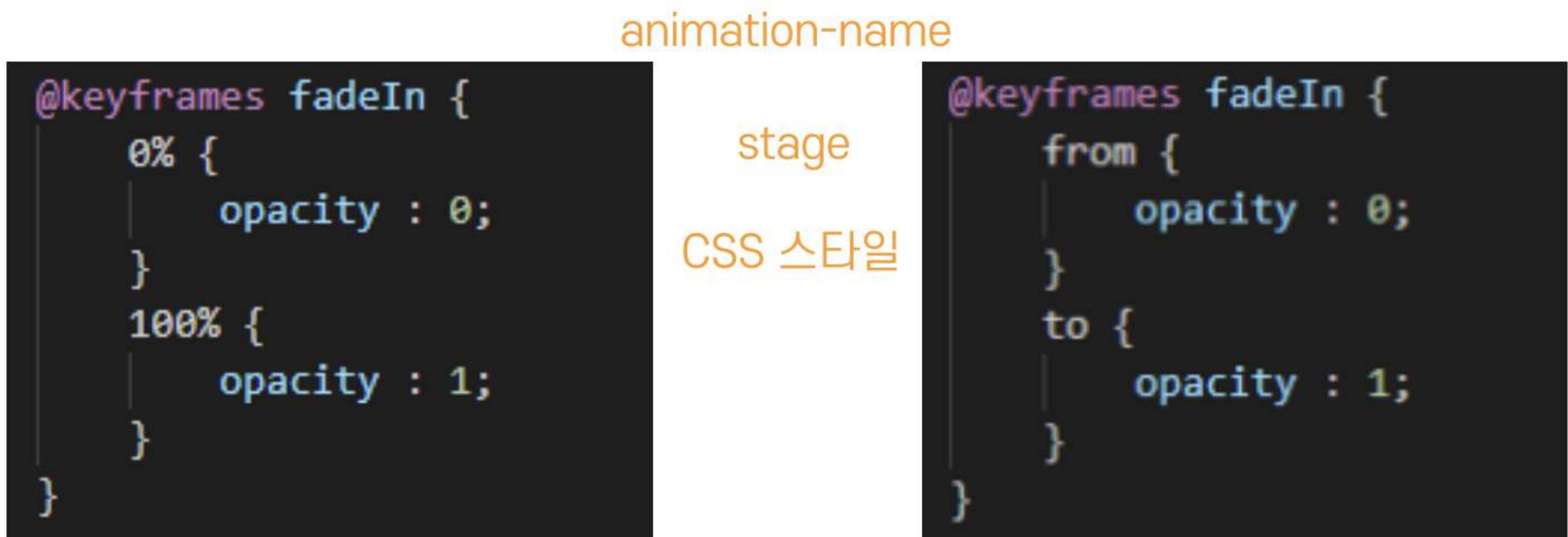
애니메이션 (@keyframe)

동적인 웹 어플리케이션의 생명은 애니메이션!

animation-name : 사용자가 직접 지정한 애니메이션 이름, @keyframes가 적용됨

스테이지 : from ~ to / 0% ~ 100% 의 구간

CSS 스타일 : 각 스테이지에 적용시킬 스타일



고급 스타일링 마스터리 북

animation 속성

애니메이션을 제작했으면 넣어야쥬

animation-name : @keyframes에서 지정해준 이름

animation-duration : 한 사이클의 애니메이션이 얼마동안 진행될지 지정

animation-delay : 엘리먼트가 로드된 뒤 애니메이션이 시작되는 시점을 지정

animation-direction : 애니메이션이 종료된 뒤 다시 시작되는 방향을 지정(정/역방향)

animation-iteration-count : 애니메이션의 반복 횟수 지정

animation-play-state : 애니메이션 멈춤 / 다시 시작 여부 지정

animation-timing-function : 중간 상태들의 전환을 어떤 시간 간격으로 진행할지 지정

animation-fill-mode : 애니메이션이 시작되기 전 또는 끝나고 난 뒤 어떤 값을 적용할지 지정

너무 귀찮아!

속기형

animation: 애니메이션 이름 한 주기의 길이 진행 속도 반복 횟수 시작 전 대기 시간 애니메이션 방향 (정/역) 시작 전 / 끝 후 스타일 재생 상태;

animation: example-animation 2s ease-in-out;

animation: example-animation 2s;

...

고급 스타일링 마스터리 북

반응형 (@media)

뷰포트 크기가 달라지는 경우에 변경되는 가변 스타일링 적용
자바스크립트의 if문과 유사하다고 생각하면 됨.

```
@media (조건) {  
    스타일  
}
```

좁은 화면 스타일링

max-width:

너비가 특정 크기보다 작다면, 스타일 적용!

max-height:

높이가 특정 크기보다 작다면, 스타일 적용!

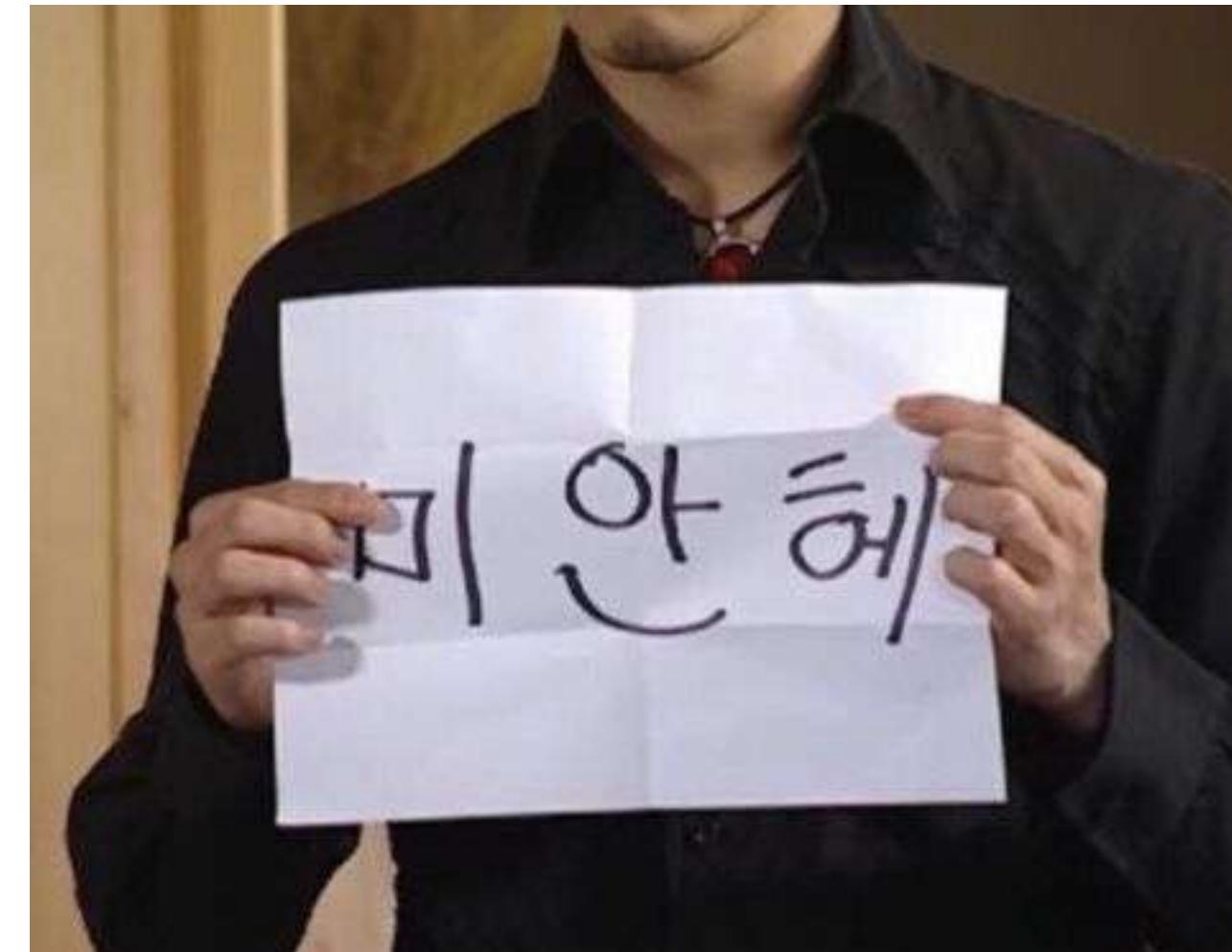
넓은 화면 스타일링

min-width:

너비가 특정 크기보다 크다면, 스타일 적용!

min-height:

높이가 특정 크기보다 크다면, 스타일 적용!



진짜 개빡셌죠...? 미안해요... 한 20분정도 푹 쉬고 실습 자 드가좌



간단한 MBTI 테스트 웹 앱 만들어보기!
진짜 너무 간단해서 한숨나옴!



어렵죠? 이해합니다... 어려운거 이해해요!
하지만 우리는 분명 진짜 미친듯한 속도로 성장중이에요 지금
일단 쉬었다 마지막 피그마 맛보고 끝냅시다ㄱㄱ



진짜진짜 간단하게 배워봅시다!

과제공지

나만의 일정 관리 어플리케이션 디자인!

그 동안 과제로 단련해온 퍼블리싱 실력을 뽐낼 때가 되었습니다.

나만의 일정관리 어플리케이션을 만들기 전, 내가 직접 디자인해보는 시간을 가질거예요!

어플리케이션에 들어가야할 기능에 맞춰서 디자인을 완료해주세요.

KYULANDER

나의 완벽한 일정관리를 위하여

닉네임을 입력해주세요.

비밀번호를 입력해주세요.

아직도 회원이 아니시라구요?



손에 잡힌듯 가볍게 확인하는 나의 일정

[일정 관리는, 굴린더에서 >](#)

예쁘게 좋지!

그치만 구현할 수 있게,,, 적당히 예쁘게 디자인 해오는 것을 추천드립니다!!!!!!

다음주 세션 때 피드백하고 수정하는 시간을 가질거예요.

회원가입 및 로그인 / 로그아웃

다른 추가정보는 받지 않아요. 아이디와 비밀번호만 입력해서 회원가입하고, 로그인합니다!

일정 작성

특정 날짜에, 간단한 일정을 작성해서 업로드할거예요.

일정 수정 및 삭제

특정 날짜에 작성된 일정 내용을 수정하거나 삭제할거예요.

일정 리스트

내가 업로드한 일정을 리스트 형태로 띄워줄 거예요.

일정 리스트

내가 업로드한 일정을 리스트 형태로 띄워줄 거예요.

일정 완료 및 후기 이모지 등록

일정을 완료했다면, 체크를 하고 해당 일정에 대한 이모지 하나를 등록할 거예요.