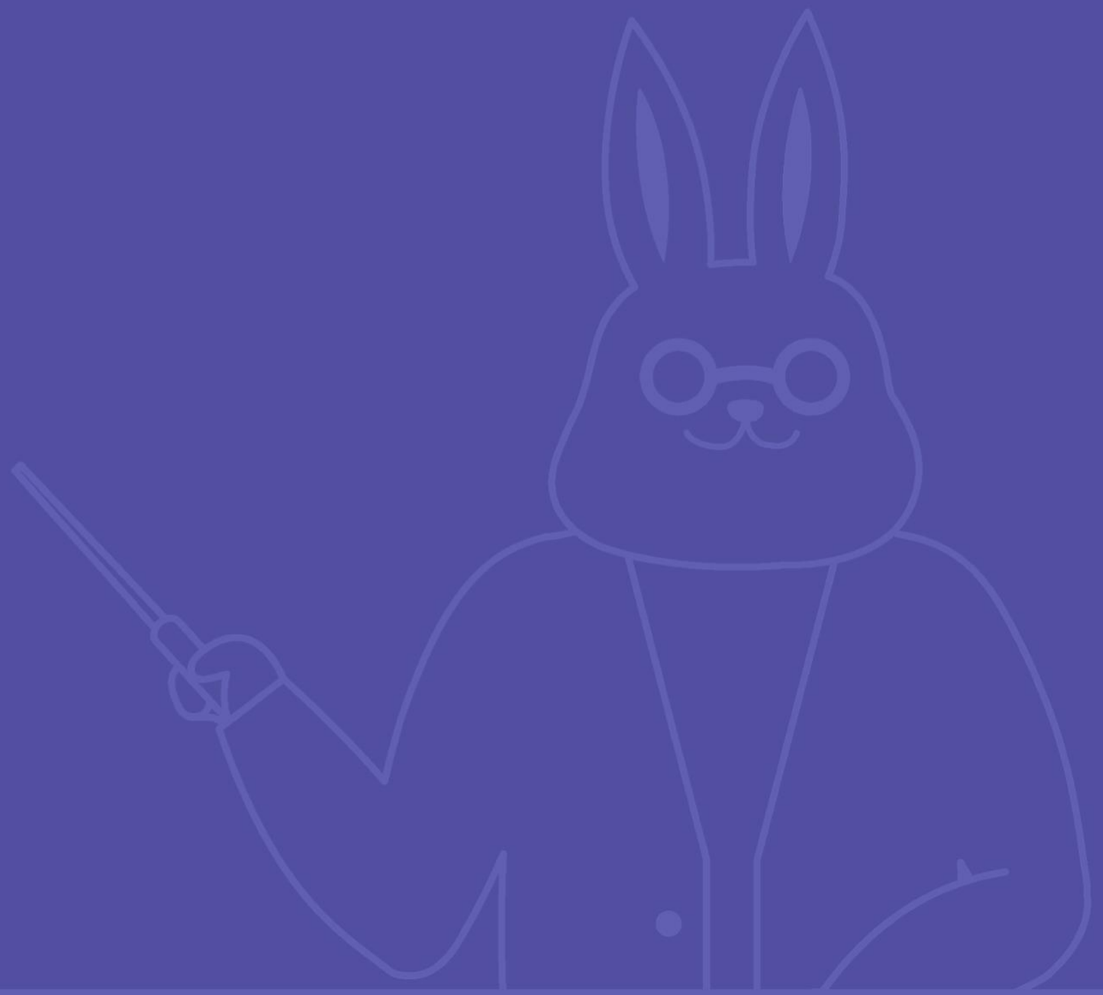
React 심화 I

00 수업 소개



/* elice */

커리큘럼

React 스타일링

React 앱에서의 스타일링 방법을 알고, 각 방법을 실습하며 익혀봅니다.

SPA와 라우팅

SPA에 대한 개념을 학습하고, SPA에서의 라우팅을 실습합니다.

커리큘럼

비동기통신과 Promise

자바스크립트의 비동기 처리에 대해 학습하고, 서버로 데이터를 요청합니다.

상태 관리

React 앱에서 순수하게 상태를 관리하는 방법을 알고 직접 실습합니다.

💡 추천대상

1. HTML, CSS, JS의 기본 문법과 내용을 이해하고 있는 분

HTML/CSS를 이용해 정적 페이지를 구성하고 JS를 이용해 이벤트를 달거나 DOM element를 검색해 동적 처리를 해보신 분

2. React를 이용해 간단한 UI를 구성할 수 있는 분

Virtual DOM, JSX, React Component, React hooks의 개념을 알고 간단한 UI를 구성해본 경험을 활용하고 싶은 분

3. React 관련 라이브러리를 자세히 배우고 싶은 분

axios, jest, react-router, redux, styled-components 등의 라이브러리를 들어보기는 했지만, 사용법을 자세히 익혀보고 싶은 분

🎯 수강목표

1. React 관련 도구로 원하는 기능을 구현할 수 있다.

React 관련 라이브러리를 알고, 그것들을 이용해 원하는 기능을 구현할 수 있다.

2. React 관련 기술 중 목적에 맞는 기술을 선택할 수 있다.

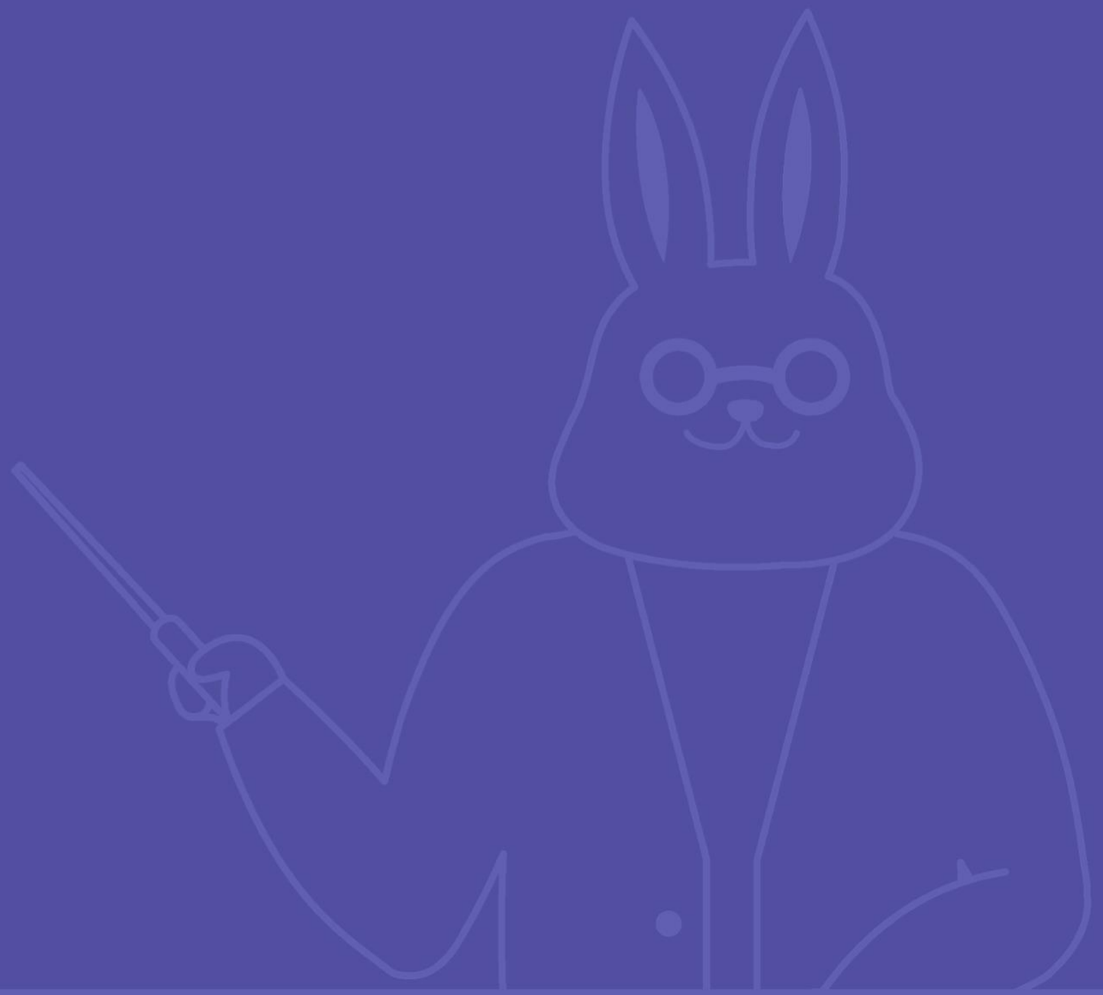
React 앱을 구성하는 여러 기술을 이해하고, 구현하고자 하는 목적에 맞는 기술을 선택할 수 있다.

3. 각 React 관련 기술이 왜 필요한지 이해한다.

React 앱을 구성하는 여러 도구가 어떤 문제를 해결하기 위한 것인지 파악할 수 있다.

React 심화 I

01 React 스타일링

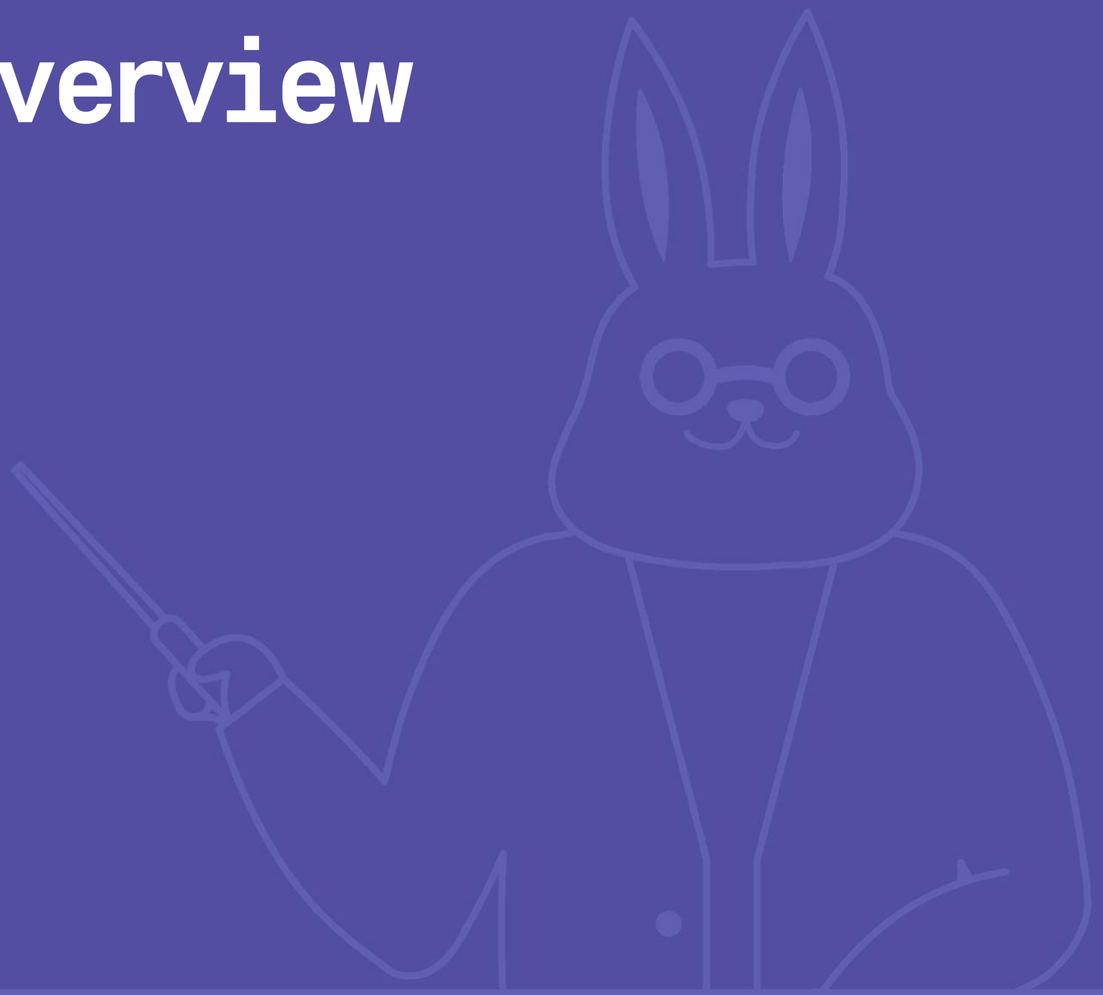


/* elice */

목차

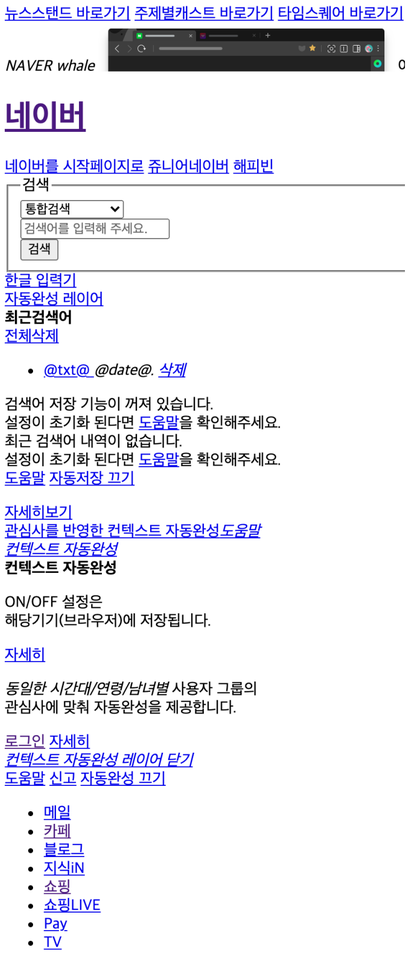
01

React 앱에서의 스타일링 Overview

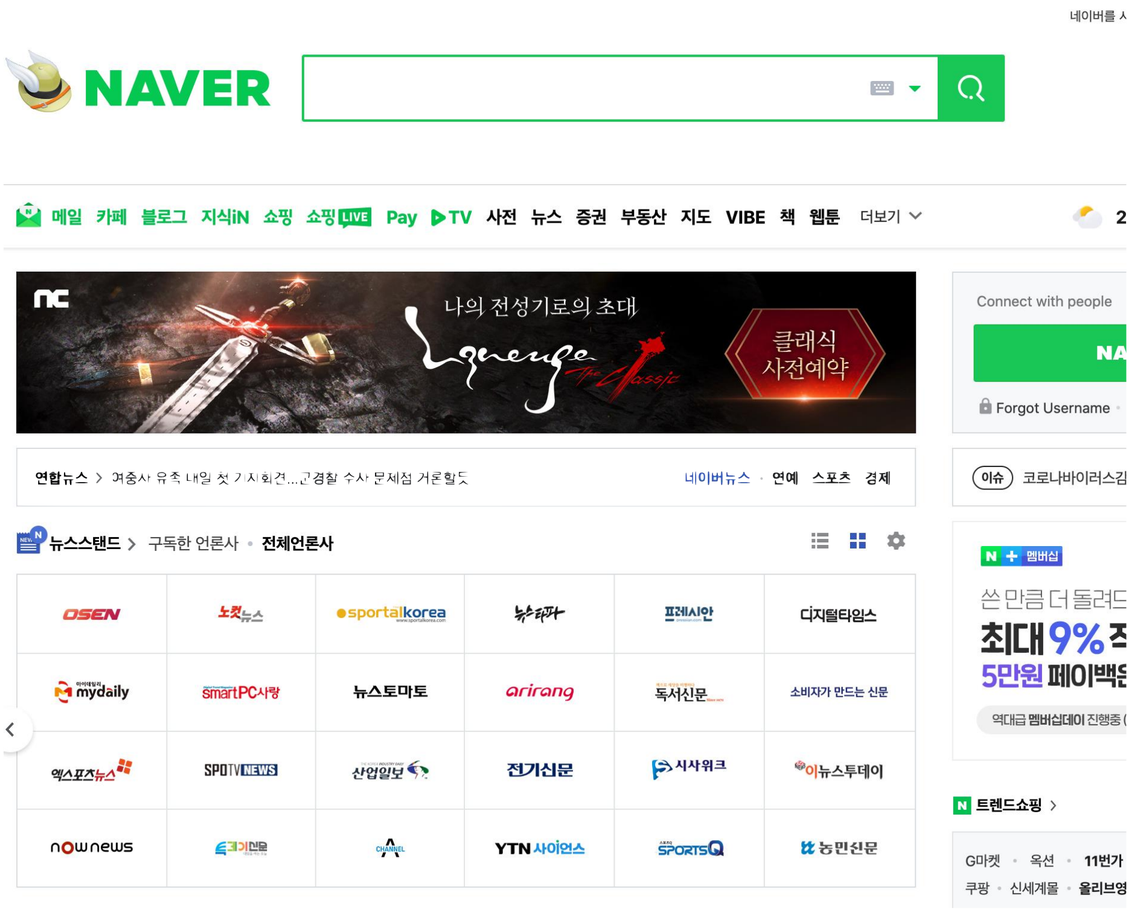


/* elice */

스타일링은 왜 중요할까?



스타일을 제거한 페이지



스타일이 있는 페이지

좋은 앱을 만들려면?

- 번들 사이즈에 대한 고려
- 앱 성능에 대한 고려
- 사용자에게 유리한 UI/UX를 고려
- 자바스크립트를 이용한 다양한 스타일 기법
- 유지보수가 용이하고 확장 가능한 코드를 작성

좋은 앱을 만들려면?

- 번들 사이즈에 대한 고려

> CSS 코드가 차지하는 사이즈는 무척 중요한 요소.

- 앱 성능에 대한 고려

> animation, transition 등 유저와의 상호작용에서 스타일 코드의 성능이 중요 요소.

- 사용자에게 유리한 UI/UX를 고려

> 스타일링에 대한 지식으로, 고급 테크닉을 적용하여 더 나은 UI/UX를 반영.

좋은 앱을 만들려면?

- 자바스크립트를 이용한 다양한 스타일 기법

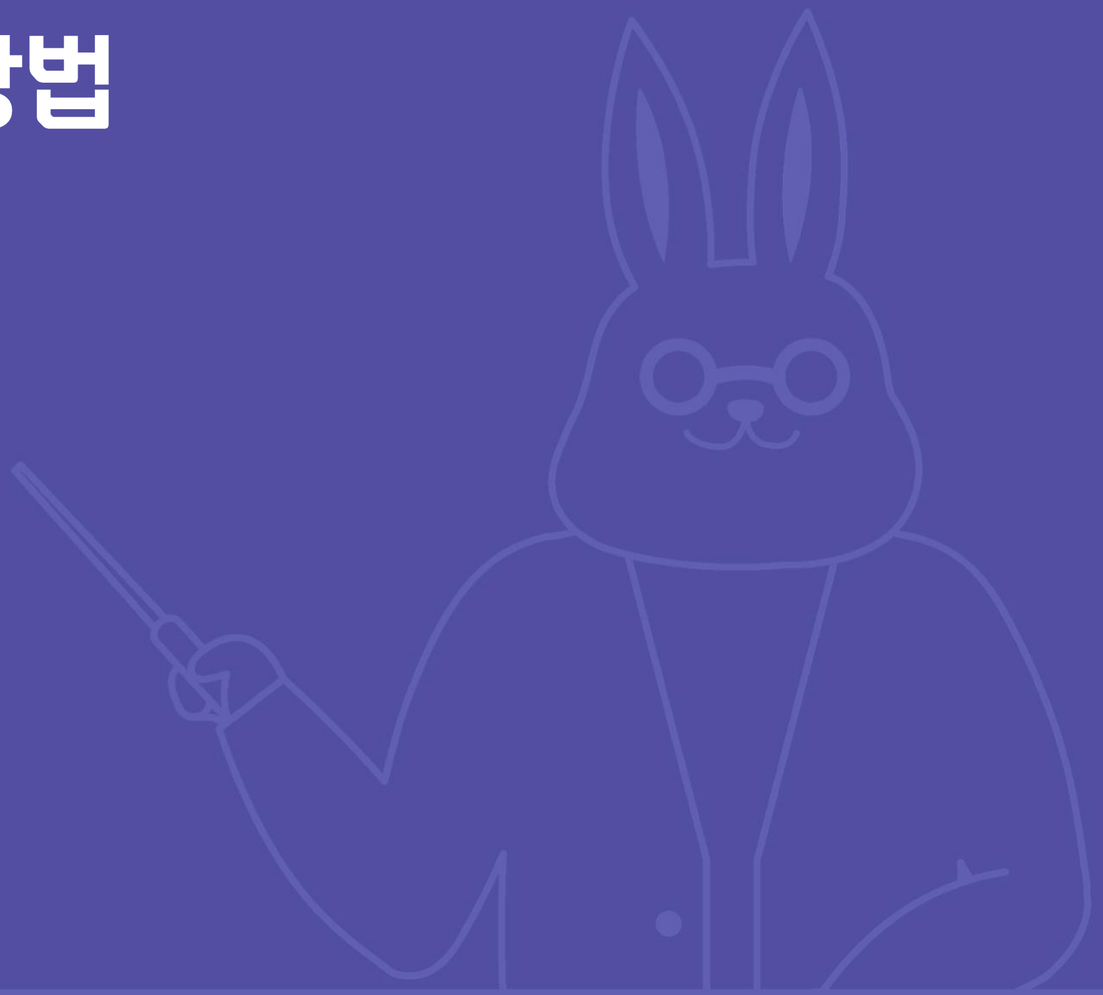
> UI 토글링, 애니메이션, 다크모드, 복잡한 UI 컴포넌트 등은
> 자바스크립트에 대한 지식만으로 구현하기 힘듦.

- 유지보수가 용이하고 확장 가능한 코드를 작성

> 스타일에 관련된 코드를 어떻게 작성하고 관리하는가에 대한 지식이 필요.

02

React 앱에서의 스타일링 방법



스타일을 어떻게 적용할까?

CSS import

CSS module

CSS-in-js

CSS import

- CSS(혹은 SCSS, Sass) 파일을 import 해서 사용.
- 필요한 모든 CSS 스타일을 하나의 파일에 작성하여,
  자바스크립트 파일과 코드 분리 가능.

CSS import

Button.jsx

```jsx
import 'button.css'

function Button({ children }) {
  return (
    <button className="button">
      {children}
    </button>
  )
}
```

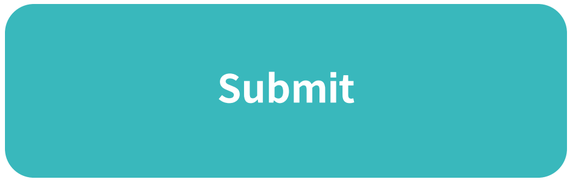
button.css

```css
.button {
  background-color: orangered;
  color: white;
  width: 140px;
  height: 40px;
}
```

CSS import

App.jsx

```
import Button from './Button'

function App() {
  return (
    <div>
      <Button>Submit</Button>
    </div>
  )
}
```



CSS import - 장/단점

- 단순히 CSS 파일만을 import 하여 사용할 수 있어 편리.
- 컴포넌트가 많지 않을 경우, 하나의 CSS 파일에 코드를 관리하는 것도 가능함.
- CSS 파일은 분리할 수 있으나, namespace를 나눌 수 없음.
- 만일 스타일이 겹칠 경우 cascading rule에 따라, 마지막에 나온 룰이 덮어씌워짐.

CSS import - 문제점

InputWithButton.jsx

```
import "./input-with-button.css";

export function InputWithButton() {
  return (
    <div className="container">
      <input type="text" name="text" className="input" />
      <button className="button">test</button>
    </div>
  );
}
```

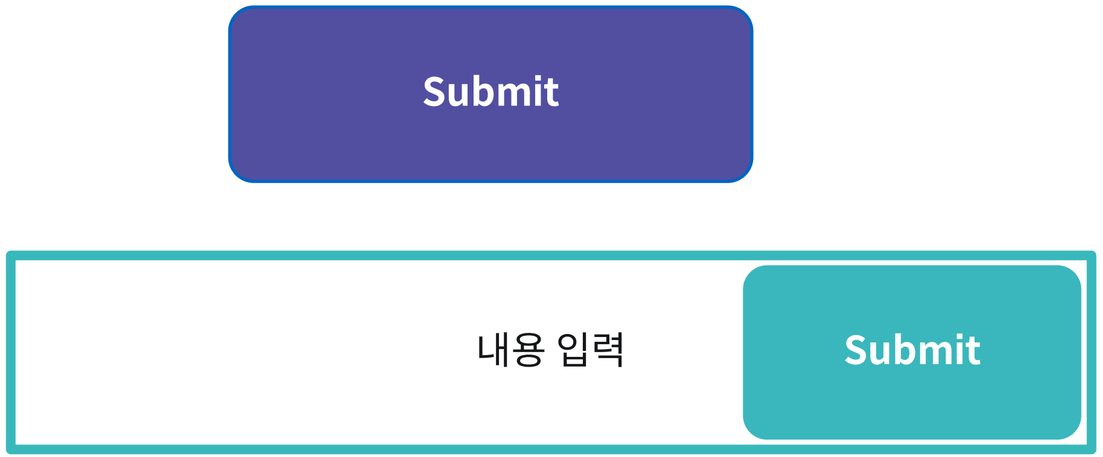
CSS import - 문제점

input-with-button.css

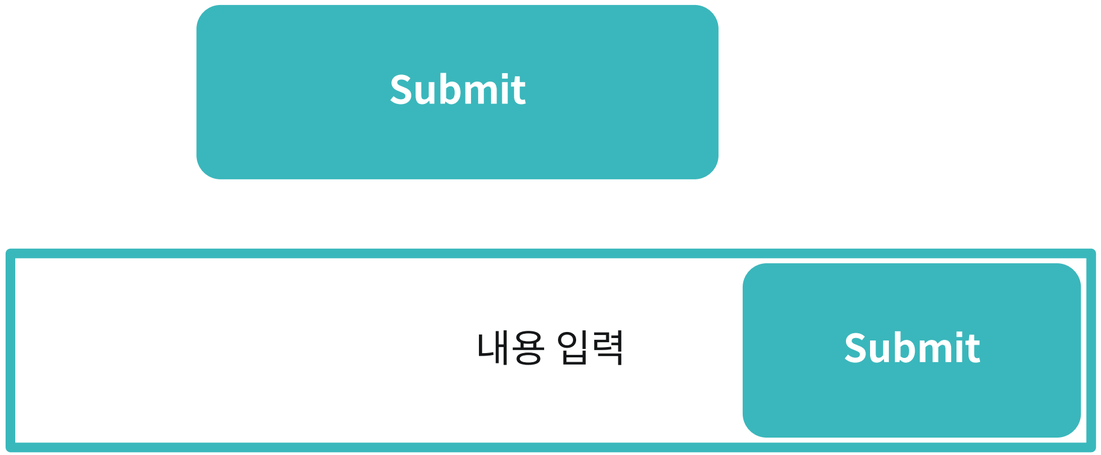
```css
.button {
  background-color: blue;
  color: white;
  border: none;
  border-radius: 5px;
  height: 40px;
  width: 140px;
}
```

```css
.container {
  background: rgba(0, 0, 0, 0.05);
  margin: 10px;
  padding: 5px;
}

.input {
  outline: none;
  border: none;
  background: white;
  border-radius: 2px;
  color: rgba(0, 0, 0, 0.8);
  height: 40px;
}
```

CSS import - 문제점



예상



결과

CSS module

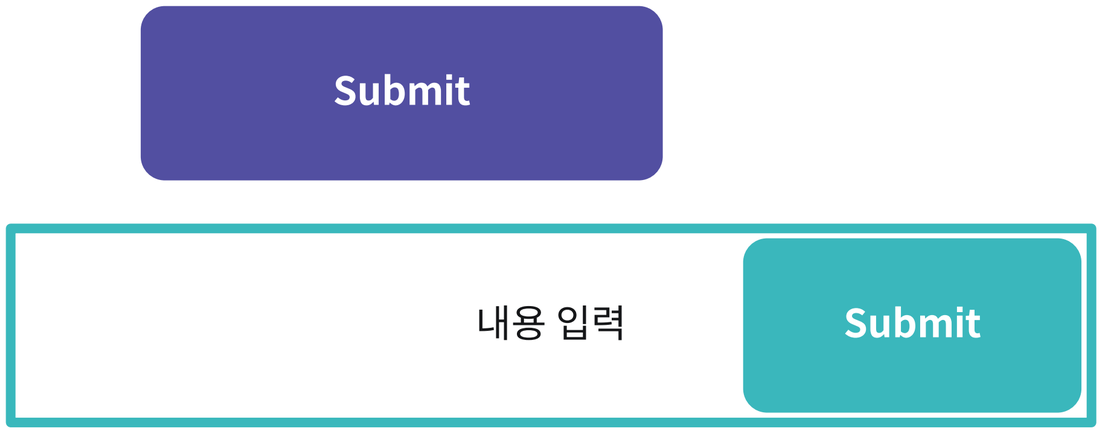
- 하나의 CSS module 파일 안에 작성한 스타일은 하나의 파일 namespace로 관리.
- class name 뒤에 겹치지 않는 hash를 붙임.
- 스타일이 겹치는 상황을 해결.
- 두 단어 이상의 경우, class 명을 camelCase로 이름을 지음.

CSS module

InputWithButton.js

```
import styles from "./input-with-button.module.css";

export function InputWithButton() {
  return (
    <div className={styles.container}>
      <input type="text" name="text" className={styles.input} />
      <button className={styles.button}>Submit</button>
    </div>
  );
}
```

CSS module - 결과




```
▼<div class="App">  flex
    <button class="button">Submit</button>
  ▼<div class="input-with-button_container__2cwID">
      <input type="text" name="text" class="input-with-button_input__3K3Fz">
      <button class="input-with-button_button__1G2gr">Submit</button> == $0
    </div>
  </div>
...
```

결과

로드된 스타일

CSS-in-JS

- 별도의 CSS 파일을 만들지 않고 하나의 컴포넌트 파일 안에서 스타일을 작성.
- 자바스크립트 문법을 그대로 활용하여 코드를 작성.
- React 컴포넌트를 사용하는 것처럼 사용.
- Sass 문법 활용 가능.

CSS-in-JS

InputWithButton.js

```
import styled from "styled-components";

const Container = styled.div`
  background: rgba(0, 0, 0, 0.05);
  margin: 10px;
  padding: 5px;
`;
```

```
const Input = styled.input`
  border: none;
  background: white;
  border-radius: 2px;
  color: rgba(0, 0, 0, 0.8);
  height: 40px;
`;

const Button = styled.button`
  background: blue;
  color: white;
  border: none;
  border-radius: 5px;
  height: 40px;
  width: 140px;
`;
```

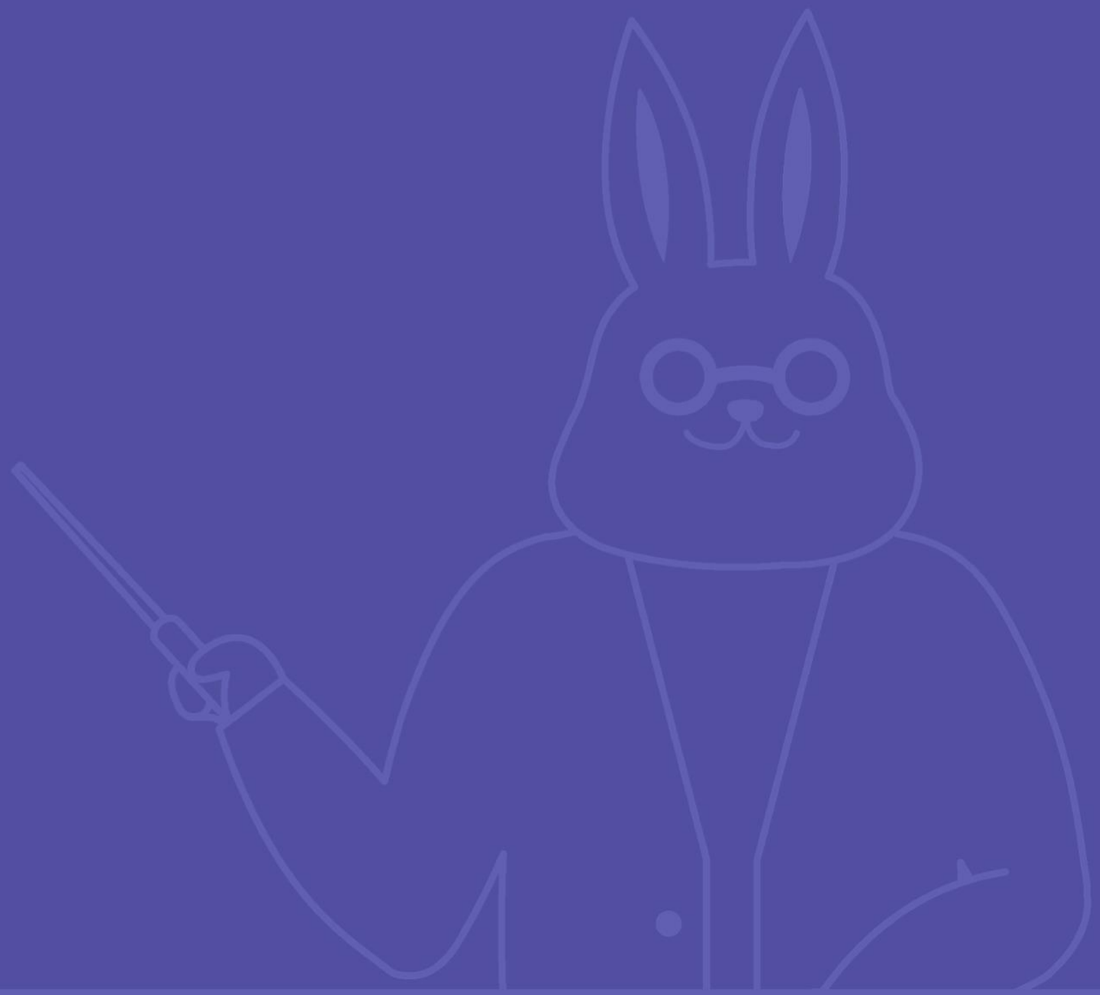
CSS-in-JS

InputWithButton.js

```
function InputWithButton() {
  return (
    <Container>
      <Input />
      <Button>Styled Button</Button>
    </Container>
  );
}
```

03

CSS, Sass



/* elice */

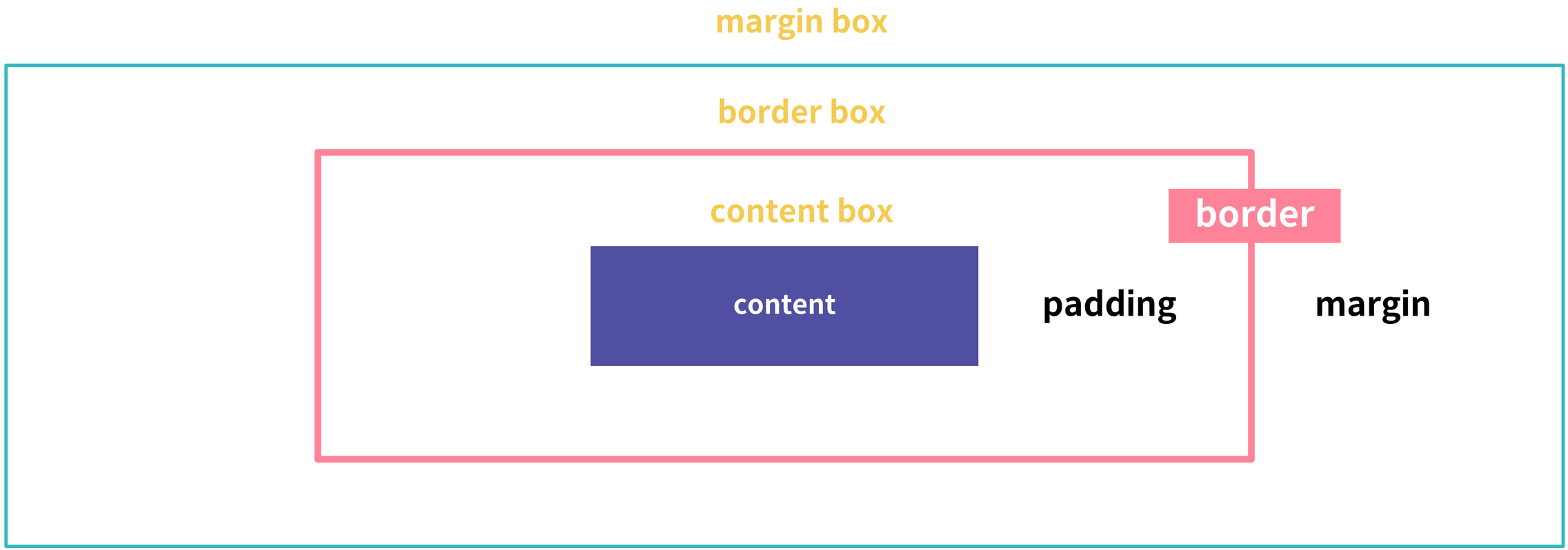
CSS Box Model

- CSS layout의 기본이 되는 모델.
- content-box, padding-box, border-box, margin-box 순으로 하나의 element를 감싸고 있음.
- box의 타입은 inline, block 두 가지.
- display:inline, display:inline-block, display:block 으로 서로 다른 box type을 적용함.

CSS Box Model - box-sizing

- width, height는 디폴트로 content-box의 크기를 정의.
- width: 100px 으로 정의 시, content의 크기만 100px이 되며,
  padding, border의 크기는 100px에 추가됨.
- box-sizing: border-box로 box sizing의 방식을 변경할 수 있음.
- border-box는 padding, border를 width, height에 포함.
- 보통 이해하기 쉬운 레이아웃을 정의하기 위해 box-sizing:border-box를 선호.

CSS Box Model



CSS Position

static	position의 default 값으로, element는 normal flow를 따라 위치함.
relative	normal flow를 따라 위치하되, 자기 자신에 상대적으로 위치함.
absolute	normal flow에서 벗어나 가장 가까운 ancestor에 상대적으로 위치함.
fixed	normal flow를 벗어나 viewport에 상대적으로 위치함.
sticky	normal flow에 따라 위치하되, 가장 가까운 scrolling ancestor에 상대적으로 위치함.

CSS Units

px, pt, cm, in	절대적인 길이를 표현하는 unit.
rem, em, %	특정 값에 상대적인 길이를 표현하는 unit.
vw, vh, vmin, vmax	viewport에 상대적인 길이를 표현하는 unit.

Sass

- Syntactically Awesome Style Sheets. CSS Preprocessor.
- SCSS, Sass 문법을 지원함.
- 모듈, 믹스인, nested style, 변수, 조건문, 반복문 등의 기능으로 CSS를 프로그래밍 언어적으로 활용하도록 확장.
- styled-components는 Sass를 기본적으로 지원함.

Sass &

code

```
.reset-button {
  &.active {}
  &.disabled {}
  &:hover {}
  &:not(:first-of-type) {}
  & + & {}
  & ~ & {}
  & > button {}
}
```

- &는 자기 자신을 나타내는 placeholder.
- 기존 CSS의 selector 문법을 응용하여 복잡한 스타일을 적용.

Sass variable

code

```
$color-red: red;
$color-white: #fff;

.reset-button {
  color: $color-red;
  &:hover {
    color: $color-white;
  }
}
```

- 믹스인, partial 와 함께 Sass가 제공하는 코드 관리 방법 중 하나.
- 색상, 사이즈 등 자주 등장하는 값을 주로 변수로 사용함.

Sass nested style

code

```
$color-red: red;
$color-white: #fff;

.reset-button {
  color: $color-red;
  &:hover {
    color: $color-white;
  }
  > button {}
}
```

- 별도의 class를 정의할 필요 없이, 하나의 block 안에 여러 CSS를 적용할 수 있는 방법.
- CSS specificity가 그대로 적용되므로, 너무 깊게 nested되면 스타일 유지보수가 힘듦.

Sass mixins, import, include

font-styles.scss

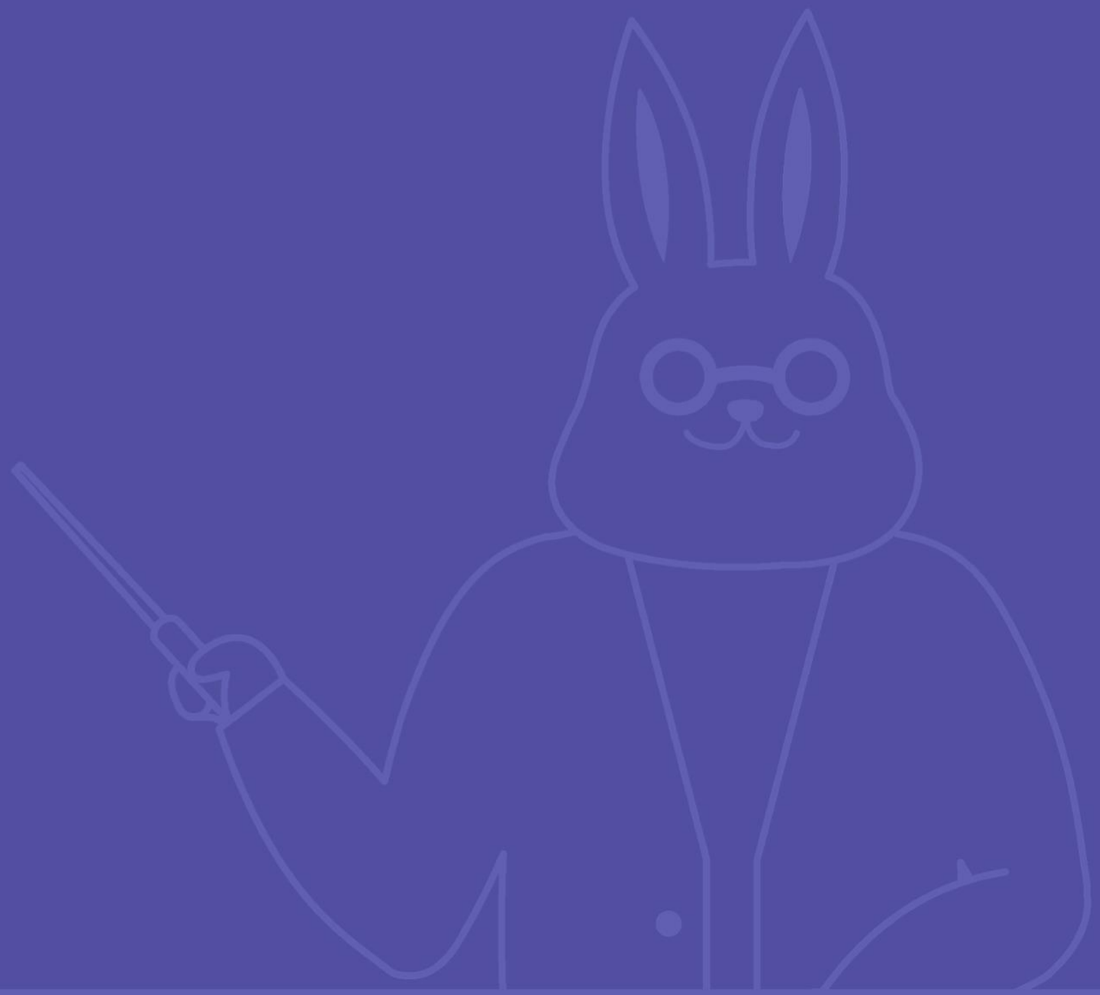
```scss
@mixins font-style-1 {
  font-size: 36pt;
  line-height: 1.5;
  font-weight: 700;
  letter-spacing: -0.05;
}
```

usage.scss

```scss
@import './font-styles.scss'

.button {
  @include font-style-1;
  background: red;
}
```

04

CSS Flexbox

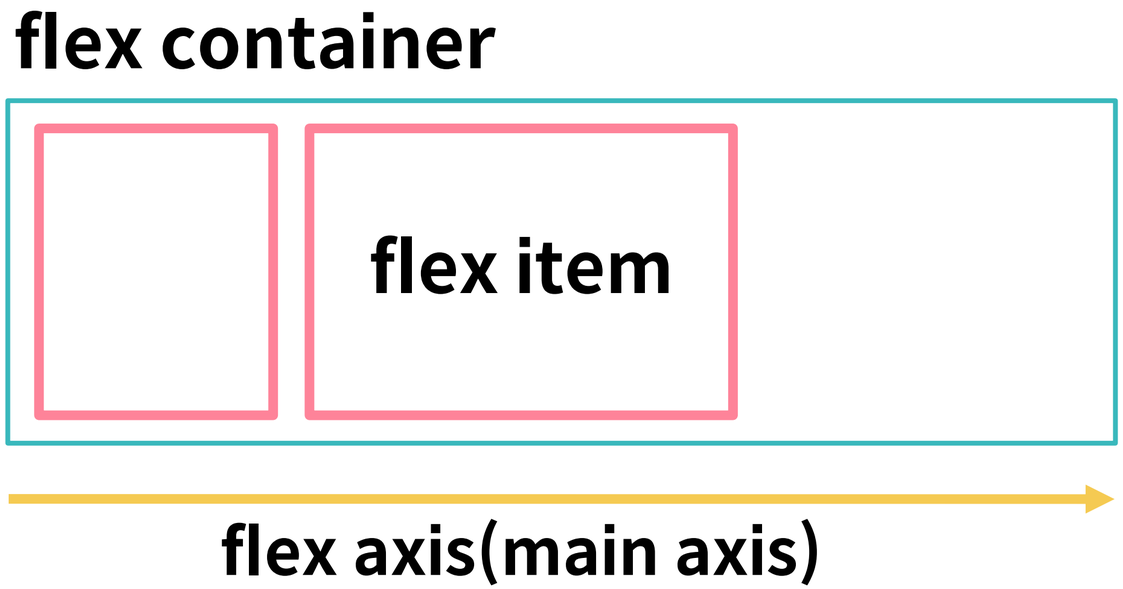


/* elice */

CSS Flexbox Model

- HTML element를 하나의 상자로 간주하고, 그 안에서 어떻게 내부 item을 배열할 것인가를 스타일 하는 모델.
- 1차원의 레이아웃을 디자인하는 데 사용.
- responsive design에 유리.
- 가운데 정렬, 비율로 정렬 등을 처리할 때 유리.

CSS Flexbox 기본 개념



- flex container - Flexbox 아이템을 담는 컨테이너.
- Flex Item - 컨테이너 안에 담긴 아이템.
- Flex axis - flex 아이템의 방향을 결정하는 축.

CSS Flexbox properties - container

flex-direction	row, column 등의 방향을 결정.
justify-content	main axis에서의 정렬을 결정
align-items	cross axis에서의 정렬을 결정.
flex-wrap	flex container가 내부 item의 width를 합친 것보다 작아질 때, 어떻게 정렬할 것인지를 결정.

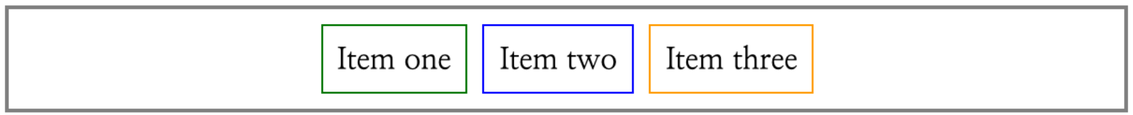
CSS Flexbox properties - item

flex-grow	flex container가 커질 때 item이 얼마만큼 늘어날 것인지를 결정.
flex-shrink	flex container가 줄어들 때 item이 얼마만큼 줄어들 것인지를 결정.
flex-basis	기준점이 되는 item의 크기.
justify-self	한 아이템을 main-axis에 따라 어떻게 정렬할 것인지를 결정.
align-self	한 아이템을 cross-axis에 따라 어떻게 정렬할 것인지를 결정.
order	flex container에서 item의 순서를 결정.

CSS Flexbox 예시

Flexbox Example

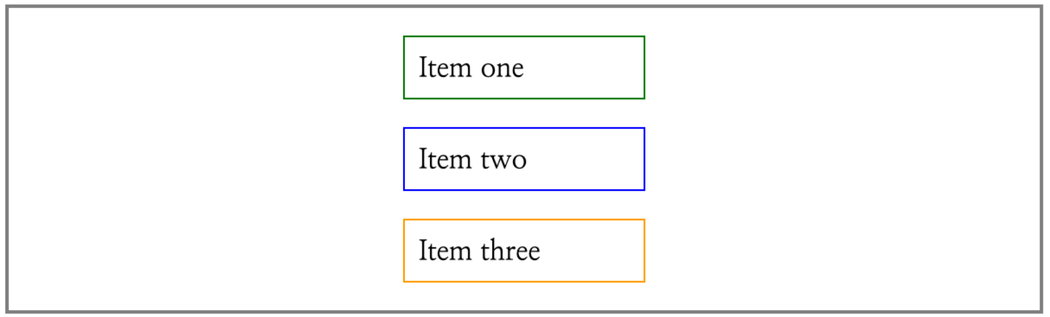
```
.container {
  display: flex;
  justify-content: center;
}
```



CSS Flexbox 예시

Flexbox Example

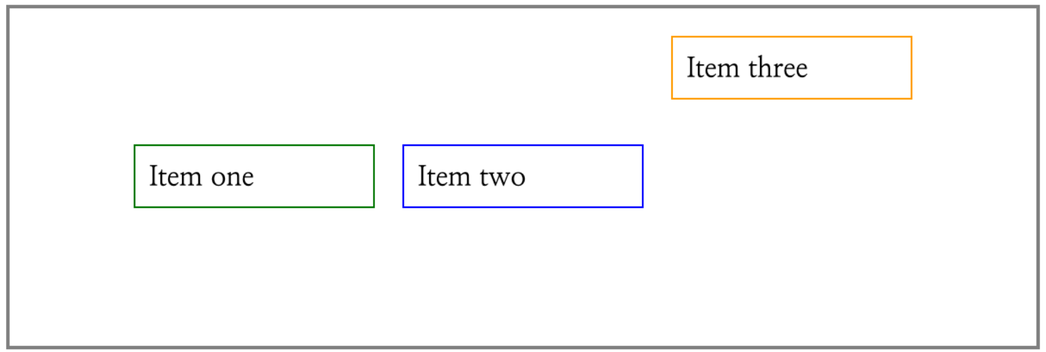
```
.container {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```



CSS Flexbox 예시

Flexbox Example

```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
}

.three {
  align-self: flex-start;
}
```



CSS Flexbox 예시

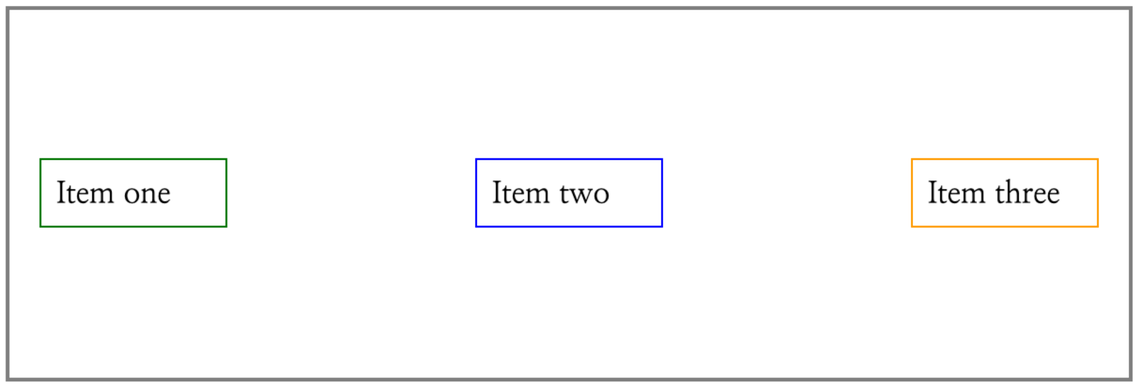
Flexbox Example

```
.container {
  flex-direction: column;
  justify-content: center;
}

.item {
  width: 100%;
}
```

CSS Flexbox 예시

Flexbox Example

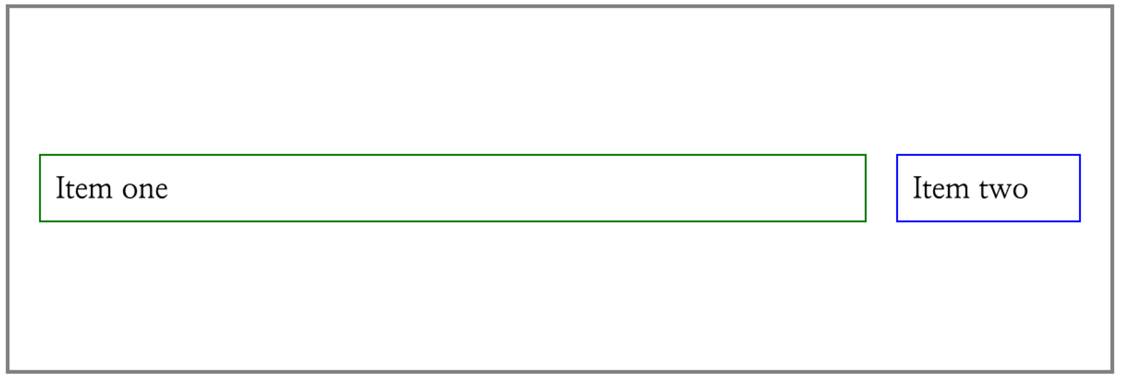
```
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```



CSS Flexbox 예시

Flexbox Example

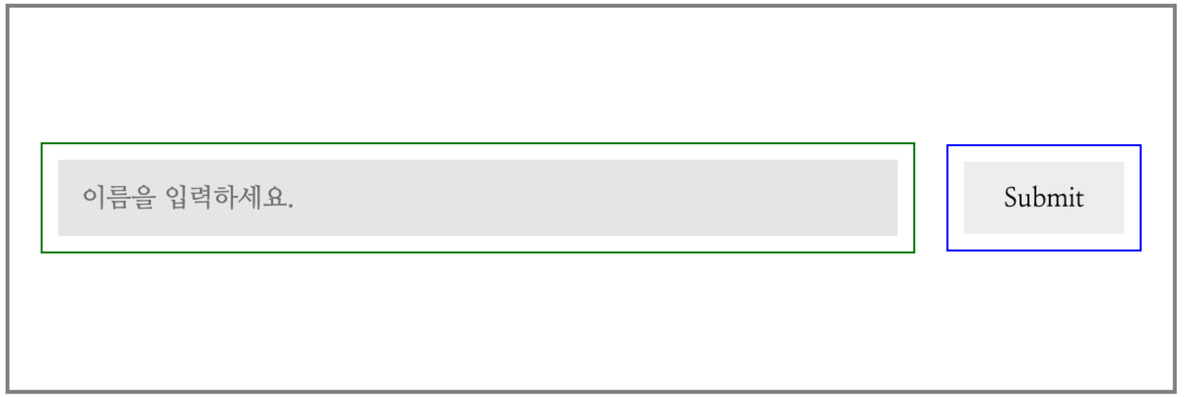
```
.container {
  display: flex;
  align-items: center;
}

.one {
  flex: 1;
}

.two {
  flex: 0 0 120px;
}
```



CSS Flexbox 예시

Flexbox Example

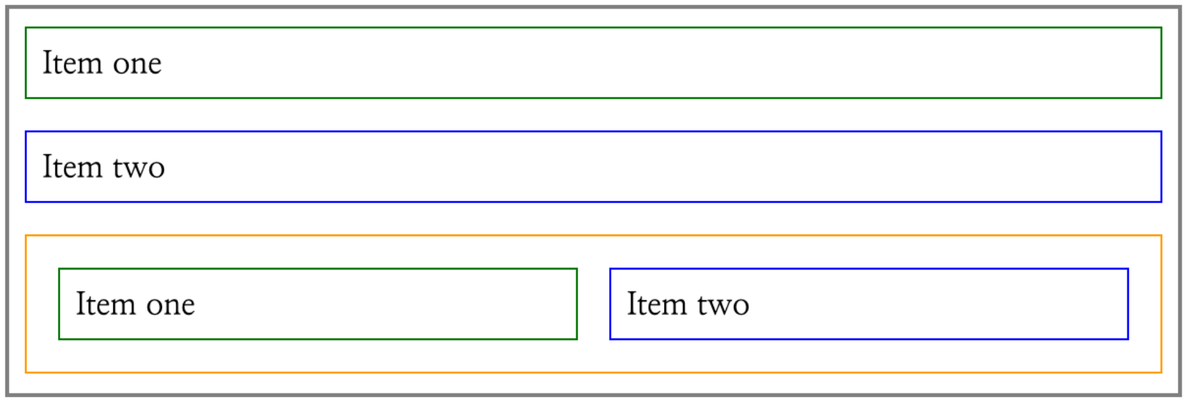
```
.container {
  display: flex;
  align-items: center;
}

.one {
  flex: 1;
}

.two {
  flex: 0 0 120px;
}
```



CSS Flexbox 예시

Flexbox Example

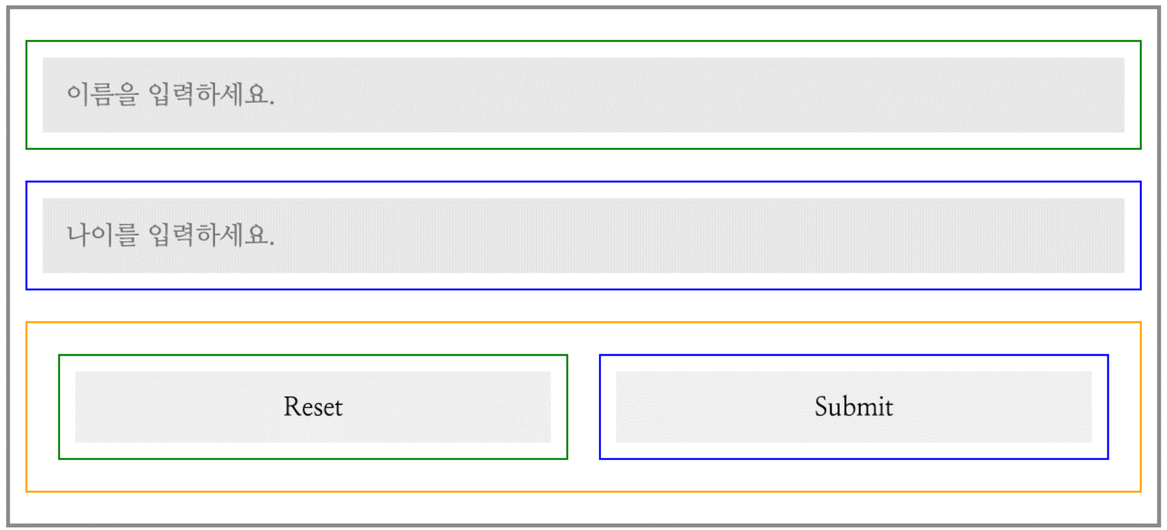
```
.container {
  display: flex;
  flex-direction: column;
}

.wrapper {
  width: 100%;
}

.three {
  display: flex;
}

.four, .five {
  flex: 1;
}
```



CSS Flexbox 예시

Flexbox Example

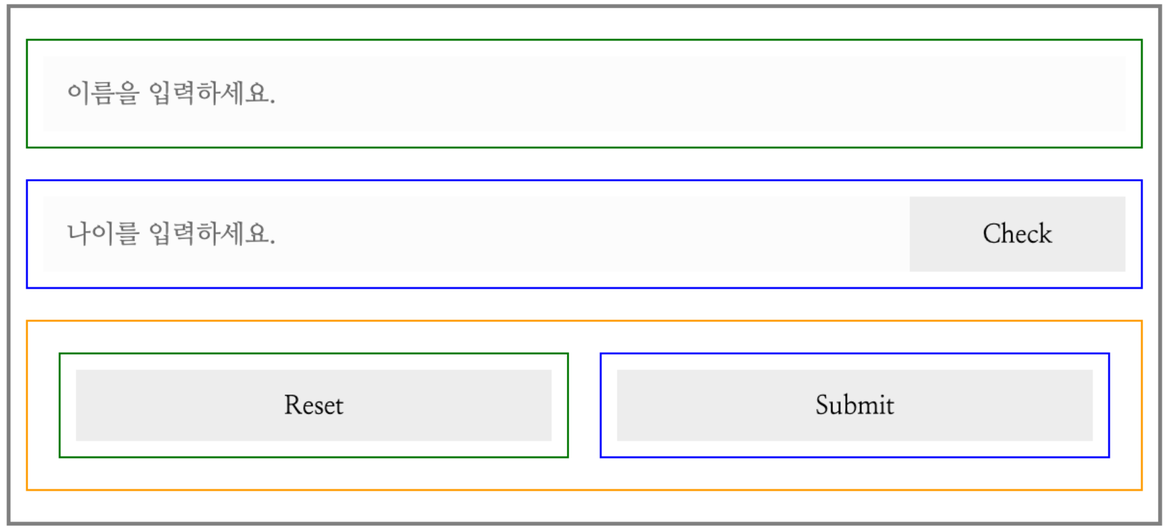
```
.container {
  display: flex;
  flex-direction: column;
}

.wrapper {
  width: 100%;
}

.three {
  display: flex;
}

.four, .five {
  flex: 1;
}
```



CSS Flexbox 예시

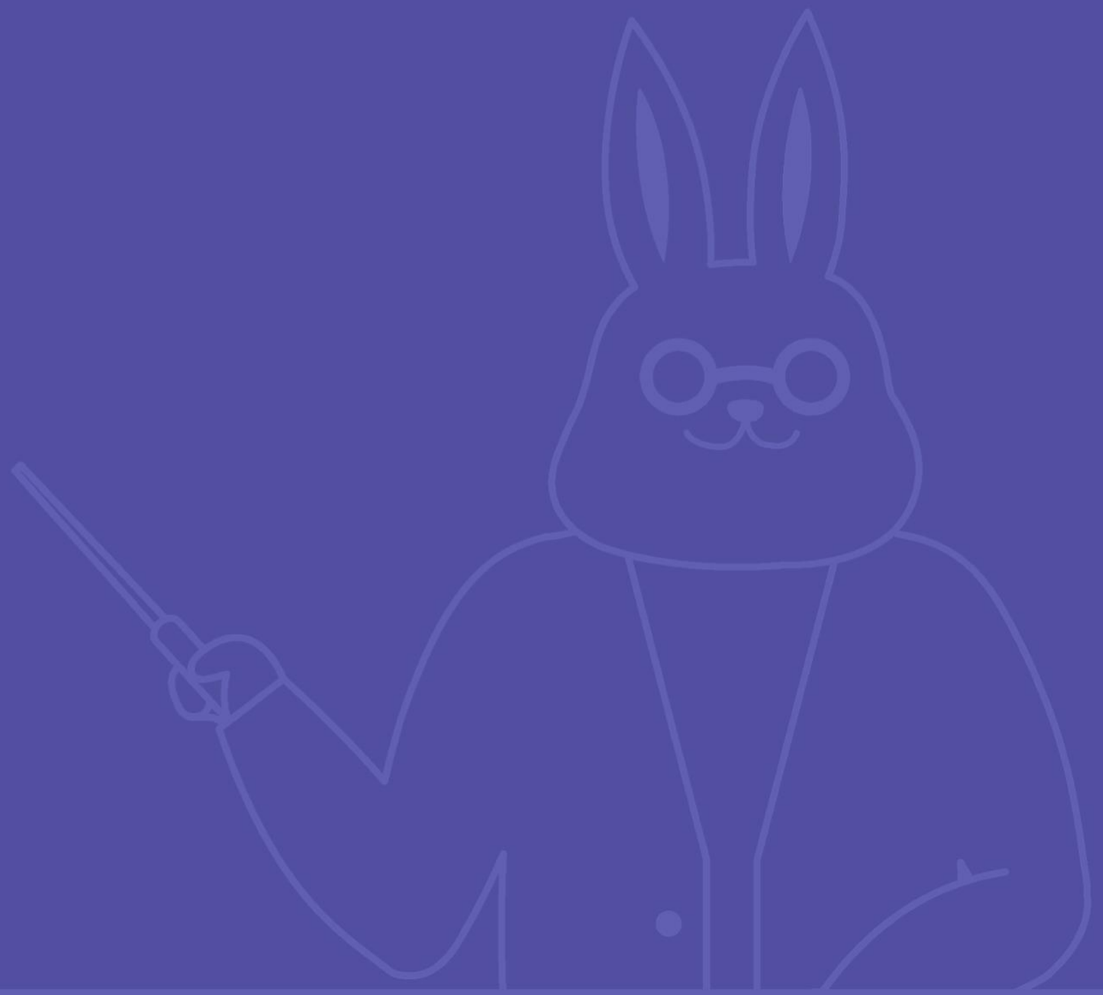
Flexbox Example

```
.container {
  display: flex;
  flex-direction: column;
}

.wrapper {
  width: 100%;
}

.three {
  display: flex;
}

.four, .five {
  flex: 1;
}
```



05

styled-components



styled-components

- 자바스크립트 파일 안에 스타일을 정의하고, React 컴포넌트처럼 활용.
- 자바스크립트 코드와 긴밀히 연계하여 다양한 코드를 작성할 수 있음.
- 별도의 CSS 파일을 만들지 않고 하나의 파일 안에 스타일을 관리하고 싶을 때 유리.
- 스타일 코드와 컴포넌트 코드 간의 결합을 나누고 싶을 때 유리.

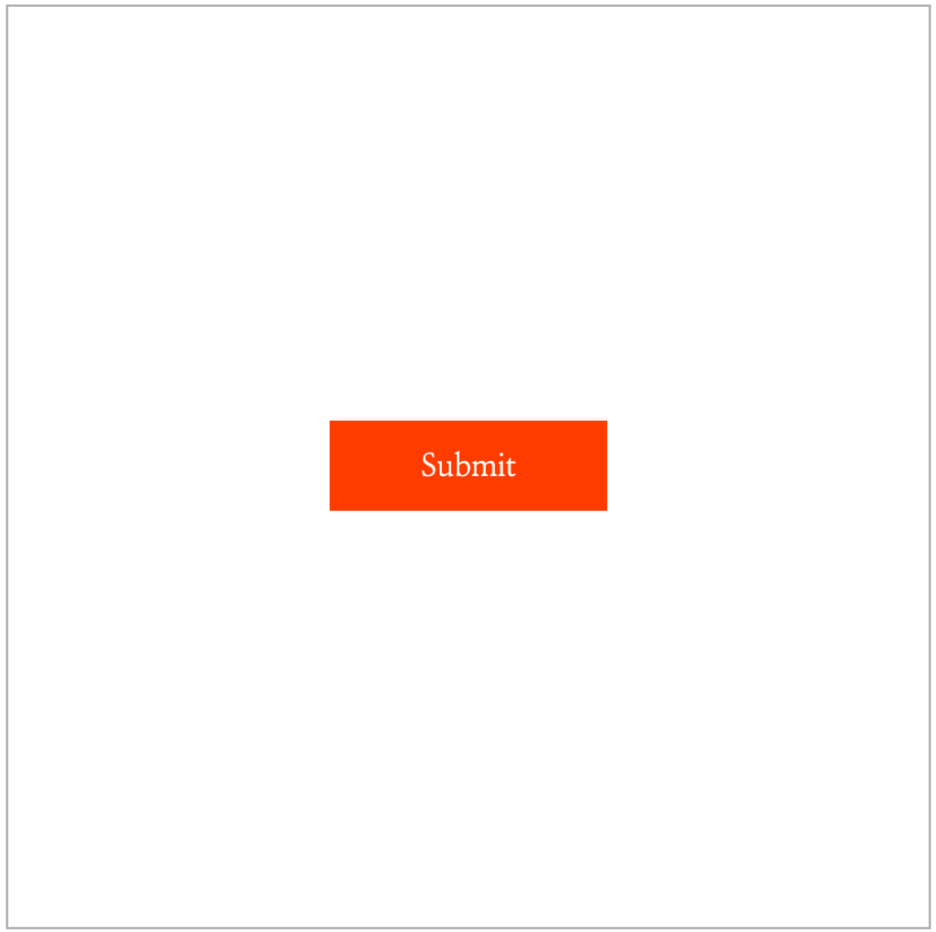
styled-components

- tagged template literal이라는 문법을 활용.
- CSS 코드에 post-css, minification, Sass 적용.
- CSS 코드를 겹치지 않게 처리.
  클래스 이름 자체가 hash.

styled-components 예시

styled-components Example

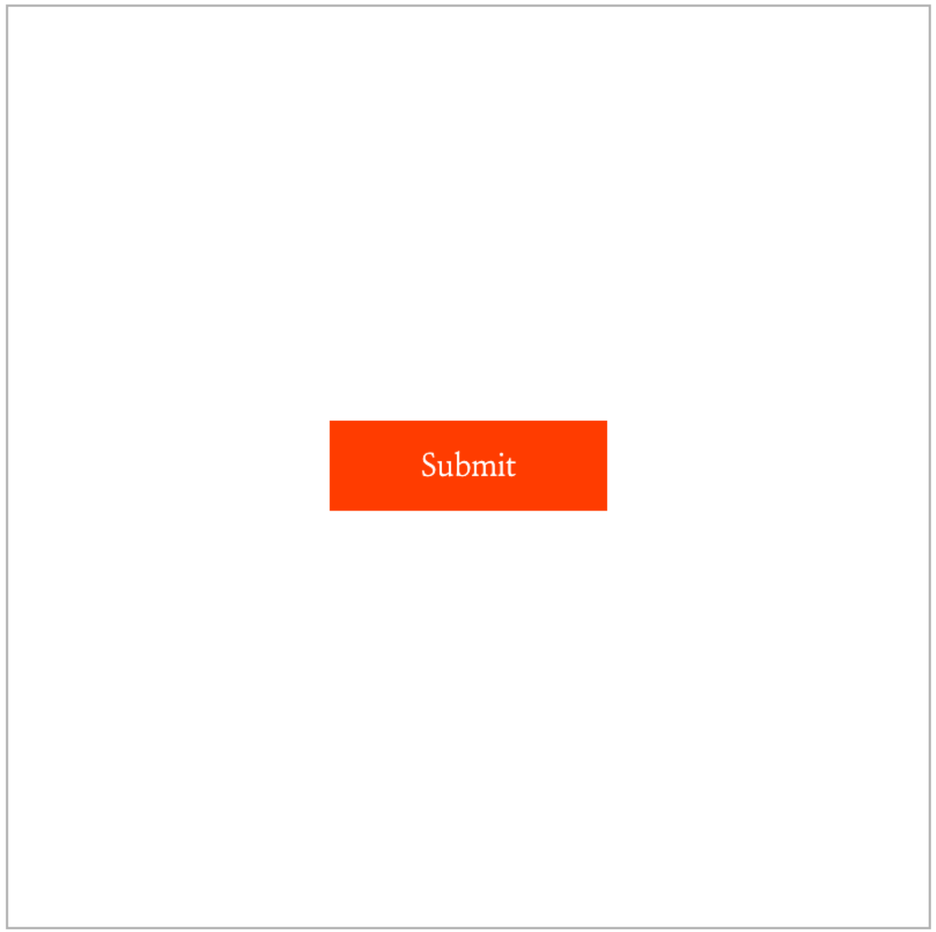
```
function Sample() {
  return (
    <Container>
      <Button>Submit</Button>
    </Container>
  );
}
```



styled-components 예시

styled-components Example

```
const Container = styled.div`
  width: 400px;
  height: 400px;

  display: flex;
  justify-content: center;
  align-items: center;
  border: 1px solid rgba(0, 0, 0, 0.3);
`;

const Button = styled.button`
  background: orangered;
  color: white;
  padding: 12px 40px;
  border: none;
`;
```



styled-components 예시

styled-components Example

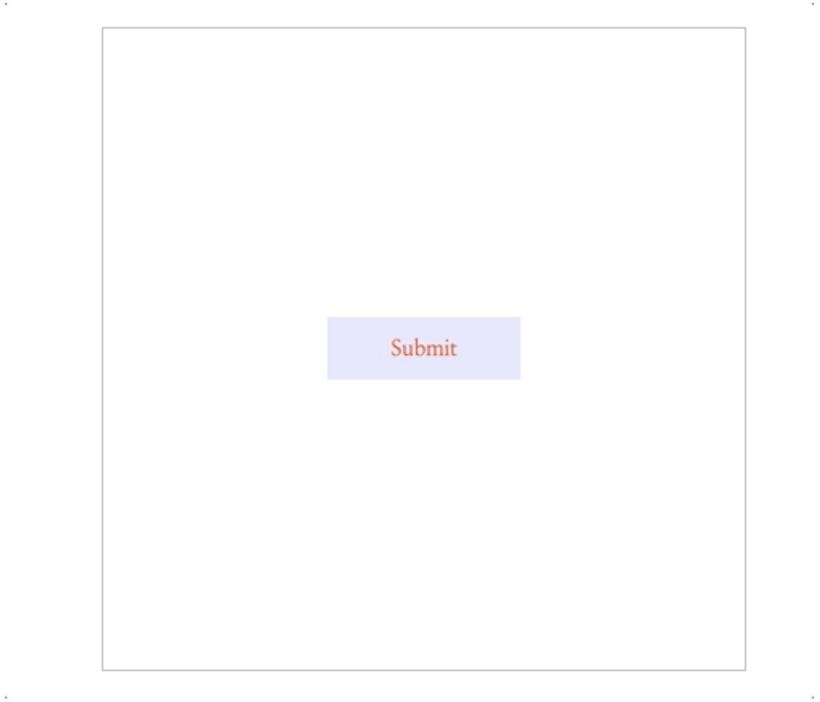
```
function Sample() {
  const [clicked, setClicked] = useState(false);

  return (
    <Container>
      <Button onClick={() => setClicked((bool) => !bool)} clicked={clicked}>
        Submit
      </Button>
    </Container>
  );
}
```

styled-components 예시

styled-components Example

```
const Button = styled.button`
  background: ${({ clicked }) => (clicked ? "orangered" : "lavender")};
  color: ${({ clicked }) => (clicked ? "lavender" : "orangered")};
  padding: 12px 40px;
  border: none;
`;
```

styled-components 예시



styled-components 예시

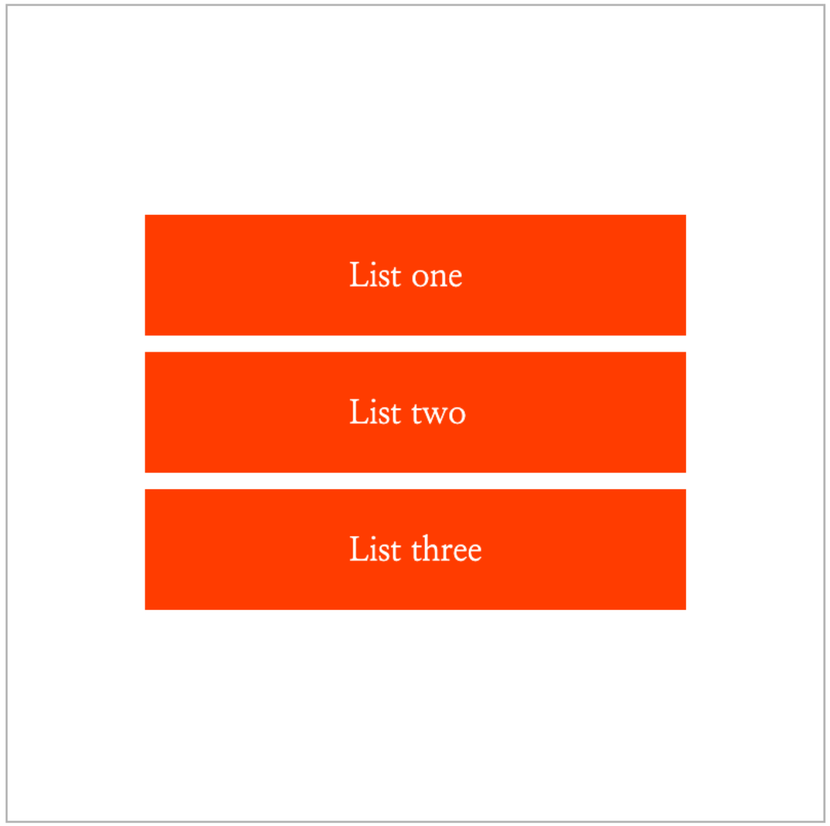
styled-components Example

```
function Sample2() {
  return (
    <List>
      <ListItem>List one</ListItem>
      <ListItem>List two</ListItem>
      <ListItem>List three</ListItem>
    </List>
  );
}
```

styled-components 예시

styled-components Example

```
const List = styled.ul`
  display: flex;
  flex-direction: column;
`;

const ListItem = styled.li`
  padding: 20px 100px;
  background: orangered;
  color: white;

  & + & {
    margin-top: 8px;
  }
`;
```



크레딧

/* elice */

코스 매니저
이재성

콘텐츠 제작자
김일식

강사
김일식

감수자
-

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io