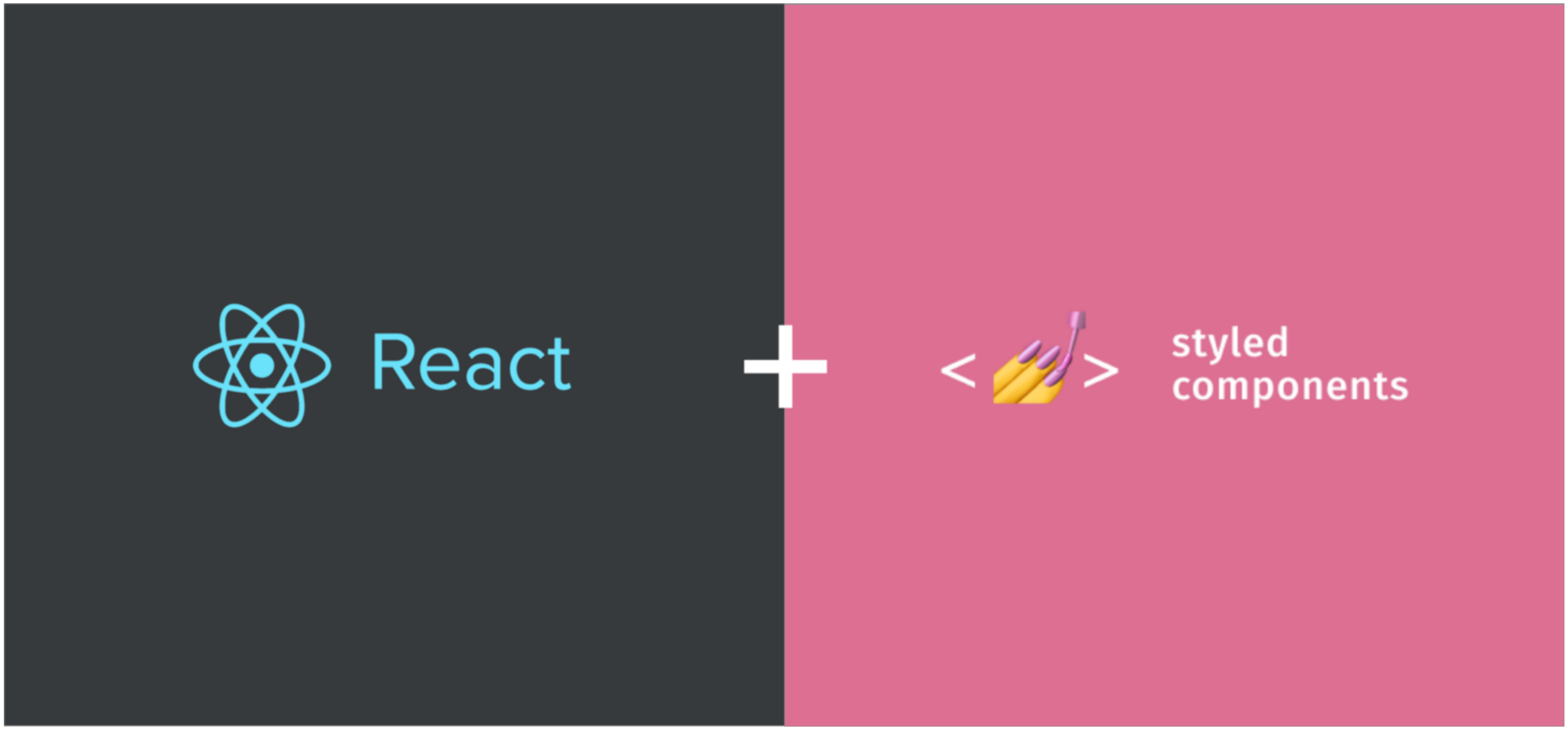
만들면서 배우는 React UI

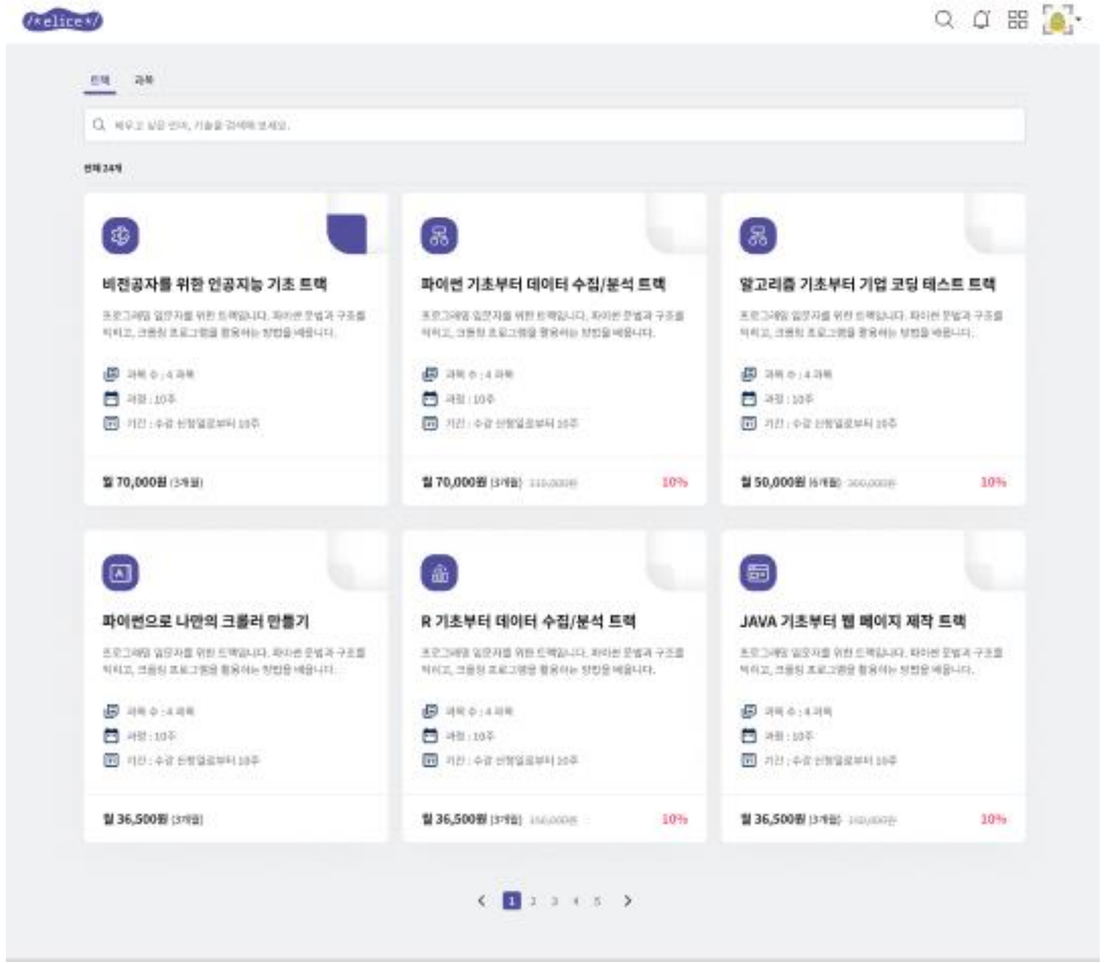
00 수업 소개

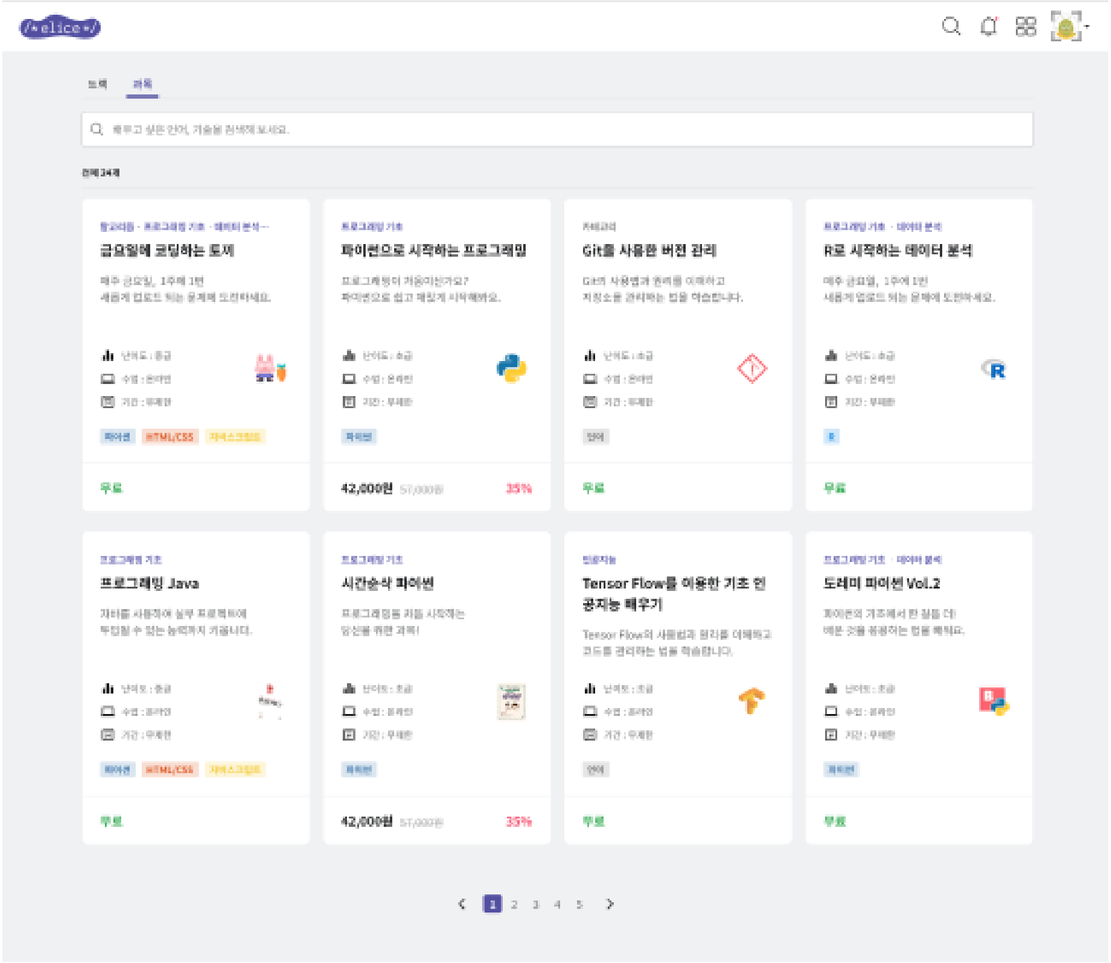
/* elice */

React + styled component



목표 화면 UI


/*elice*/
트랙
과목
비전공자를 위한 인공지능 기초 트랙
파이썬 기초부터 데이터 수집/분석 트랙
알고리즘 기초부터 기업 코딩 테스트 트랙
파이썬으로 나만의 크롤러 만들기
R 기초부터 데이터 수집/분석 트랙
JAVA 기초부터 웹 페이지 제작 트랙


/*elice*/
트랙
과목
금요일에 코딩하는 토끼
파이썬으로 시작하는 프로그래밍
Git을 사용한 버전 관리
R로 시작하는 데이터 분석
프로그래밍 Java
시간순삭 파이썬
Tensor Flow를 이용한 기초 인공지능 배우기
도레미 파이썬 Vol.2

React + styling 도구 목록

CSS MODULES

CSS module

styled components

CSS in JS library

Ant Design

UI framework

w3.css

Compact but complete CSS framework

CSS framework

추천대상

웹 개발 경험자

HTML, CSS의 기본 동작 방식을 숙지하신 분

React 경험자

React 작동방식의 기본기를 숙지하신 분

트렌디한 웹 개발자 꿈나무

Styled Component를 이용해 트렌디한 웹 페이지를 만들어 보고 싶으신 분

커리큘럼

Style 도구 소개, styled component 소개

React와 같이 사용하기 좋은 styling 툴들을 소개하고, 그 도구 중 대중적인 도구 중 하나인 styled component를 간략히 학습합니다.

styled component를 이용하여 과목 카드 만들어 보기

샘플 UI를 중 과목 카드를 만들면서, 스타일드 컴포넌트와 CSS사용법을 학습합니다.

커리큘럼

트랙 카드, 탭, 검색 창 등 여러 UI만들기

styled component를 이용하여 과목카드, 탭, 검색창 등 목표 UI의 컴포넌트들을 만들며 styled component와 CSS사용법을 학습합니다.

API를 통해 가져온 데이터와 UI를 연동하여 목표 UI 완성하기

useEffect훅 안에서 API를 호출하고, 탭, 검색 창, 페이지네이션 등의 UI 상태와 API호출을 연동하여 목표 UI를 완성합니다.

🎯 수강목표

Styling 도구 이해하기

React와 함께 사용할 수 있는 styling 도구들의 전반적인 작동방식을 이해할 수 있다.

Styled Component 숙지

React와 Styled Component를 이용하여 트렌디한 웹 페이지를 만들 수 있다.

React 기본기 증진

React를 이용한 웹 페이지를 개발하며, React를 더 능숙하게 다룰 수 있다.

만들면서 배우는 React UI

01 Style 도구 및 styled component 소개

/* elice */

목차

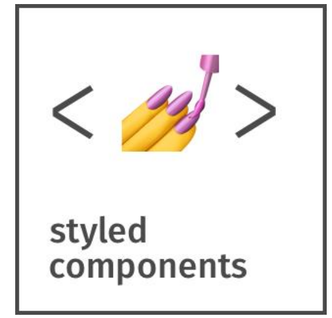
01

React styling 방법론 소개

React + styling 도구 목록

CSS MODULES

CSS module



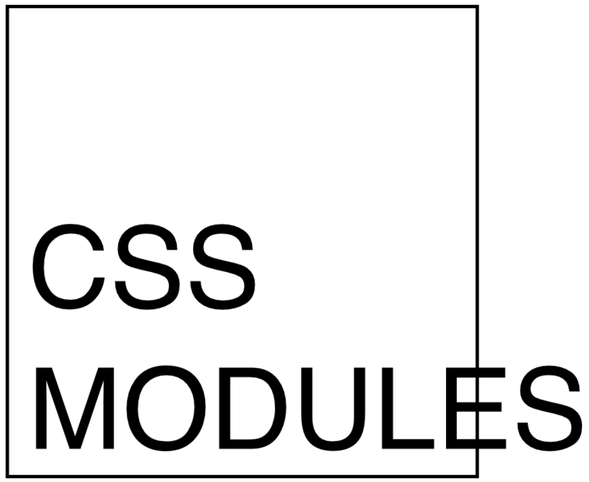
CSS in JS library

Ant Design

UI framework

CSS framework

CSS Module



CSS module

class, id 등에 random string을 달아주기 때문에 선택자가 겹칠 우려가 없음

스타일 충돌을 방지하고 코드를 격리하여 체계적으로 CSS 설계가 가능

스타일링 직접 하나하나 해야 함

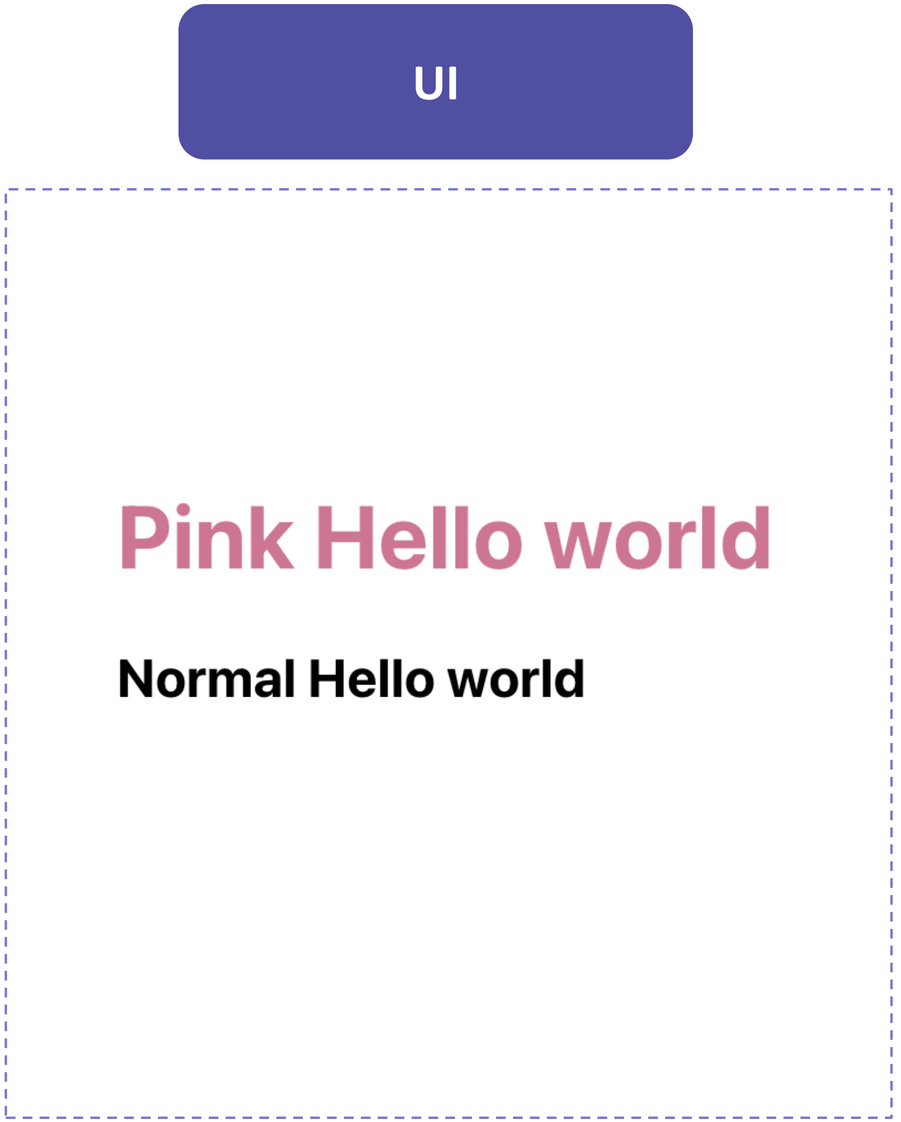
CSS Module 예시

App.jsx

```jsx
import styles from "./app.module.css";

export default function App() {
 return (
   <div>
     <h1 className={styles.title}>Pink Hello world</h1>
     <h1 className={"title"}>Normal Hello world</h1>
   </div>
 );
}
```

app.module.css

```css
h1 {
 font-size: 1.5rem;
}

.title {
 font-size: 2.5rem;
 color: palevioletred;
}
```



UI framework (ex. Ant Design, Material UI)

UI framework

이미 다 만들어져 있어서 간편하고 쉽게 쓰기에 좋음

이미 다 만들어져 있어서 styling의 학습 및 훈련이 필요한 초심자들에게는 비추천

해당 framework의 디자인철학을 벗어나기 쉽지 않음

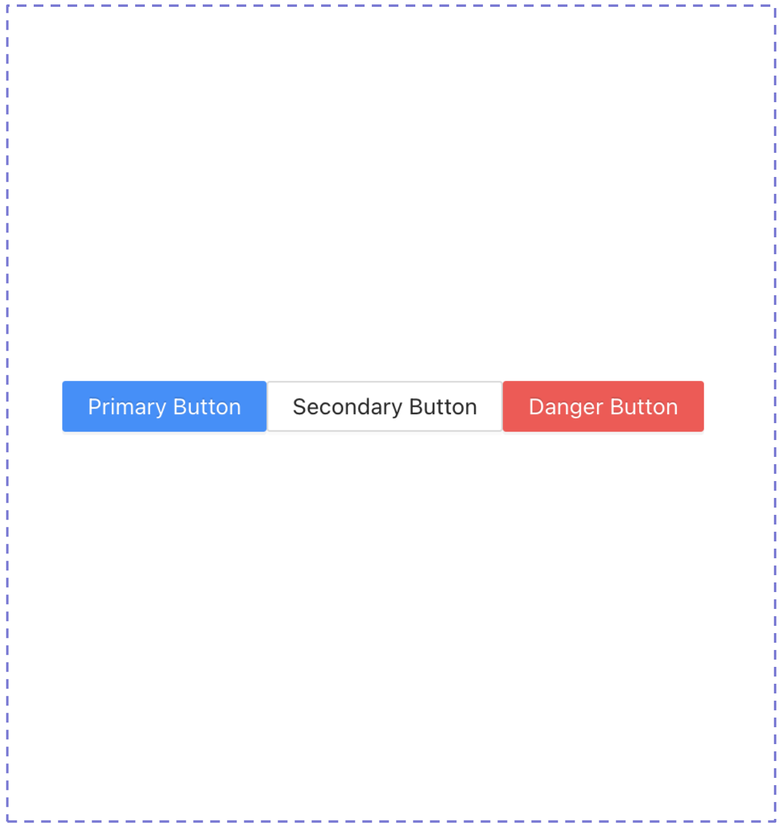
컴포넌트들을 커스터마이징 하기 어려움

UI framework (Ant Design) 예시

App.jsx

```jsx
import "antd/dist/antd.css";
import { Button } from "antd";

export default function App() {
 return (
   <div>
     <Button type="primary">Primary Button</Button>
     <Button type="secondary">Secondary Button</Button>
     <Button type="danger">Danger Button</Button>
   </div>
 );
}
```

UI



CSS framework (ex. W3CSS, TailwindCSS)

CSS framework

거대한 CSS 파일 하나를 가져오는 것임

개발자가 따로 CSS 파일을 작성하지 않아도 HTML에 클래스만 적어주면 정해진 규칙대로 스타일링이 적용됨

CSS에 대한 이해력이 있어도 해당 framework를 사용하기 위한 학습을 또다시 해야 함

이미 다 만들어져 있어서 styling의 학습 및 훈련이 필요한 초심자들에게는 비추천

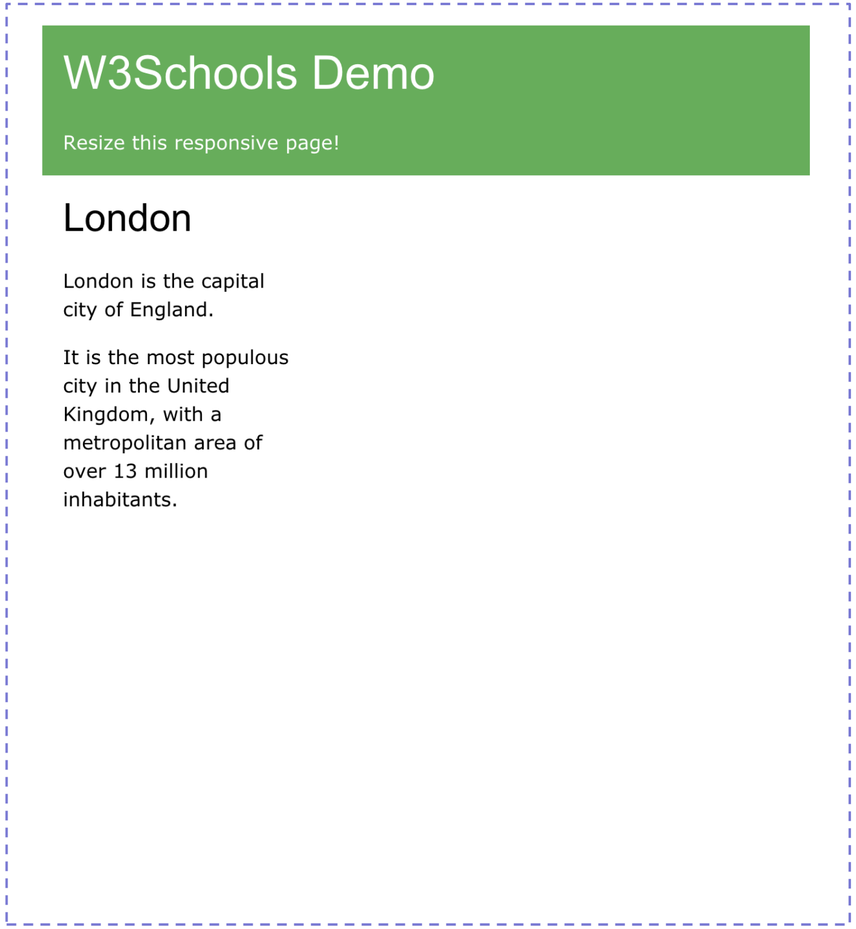
CSS framework (W3CSS) 예시

App.jsx

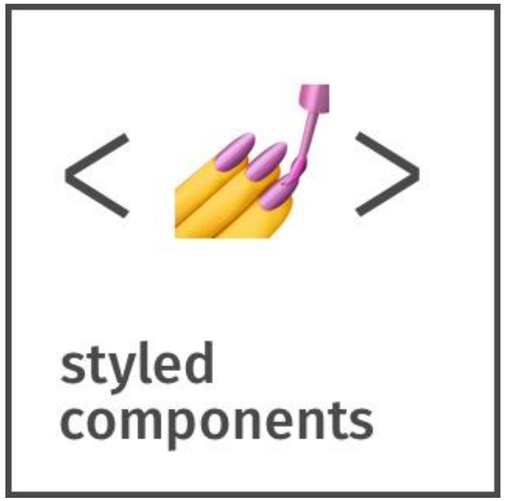
```
import { Helmet } from "react-helmet";

export default function App() {
 return (
   <div>
     <Helmet>
       <link
         rel="stylesheet"
         href="https://www.w3schools.com
                                    /w3css/4/w3.css"
       />
     </Helmet>
     <div className="w3-container w3-green">
       <h1>W3Schools Demo</h1>
       <p>Resize this responsive page!</p>
     </div>
     <div className="w3-row-padding">
       <div className="w3-third">
         <h2>London</h2>
         <p>London is the capital city of
                                    England.</p>
         <p>
           It is the most populous city in the
             United Kingdom, with a metropolitan
            area of over 13 million inhabitants.
         </p>
       </div>
     </div>
   </div>
 );
}
```

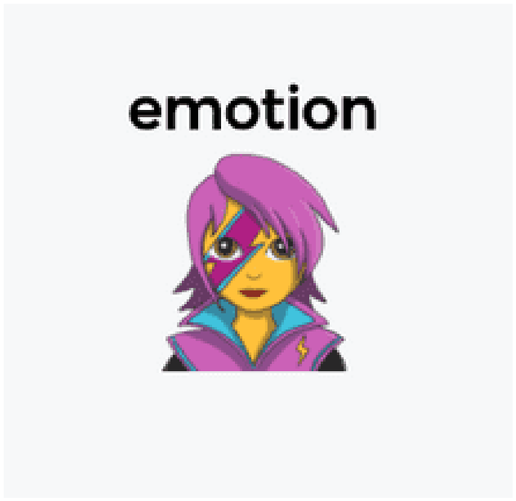
CSS framework (W3CSS) 예시

UI



CSS-in-JS library (ex. styled component, emotion)





CSS in JS library

따로 CSS 파일 만들 것 없이 JSX 파일 안에서 스타일링까지 해결 가능함

컴포넌트 재사용성이 쉬워짐

JS 값들을 props로 넘겨줘서 해당 JS 값에 의도된 styling을 바로 적용 할 수 있음

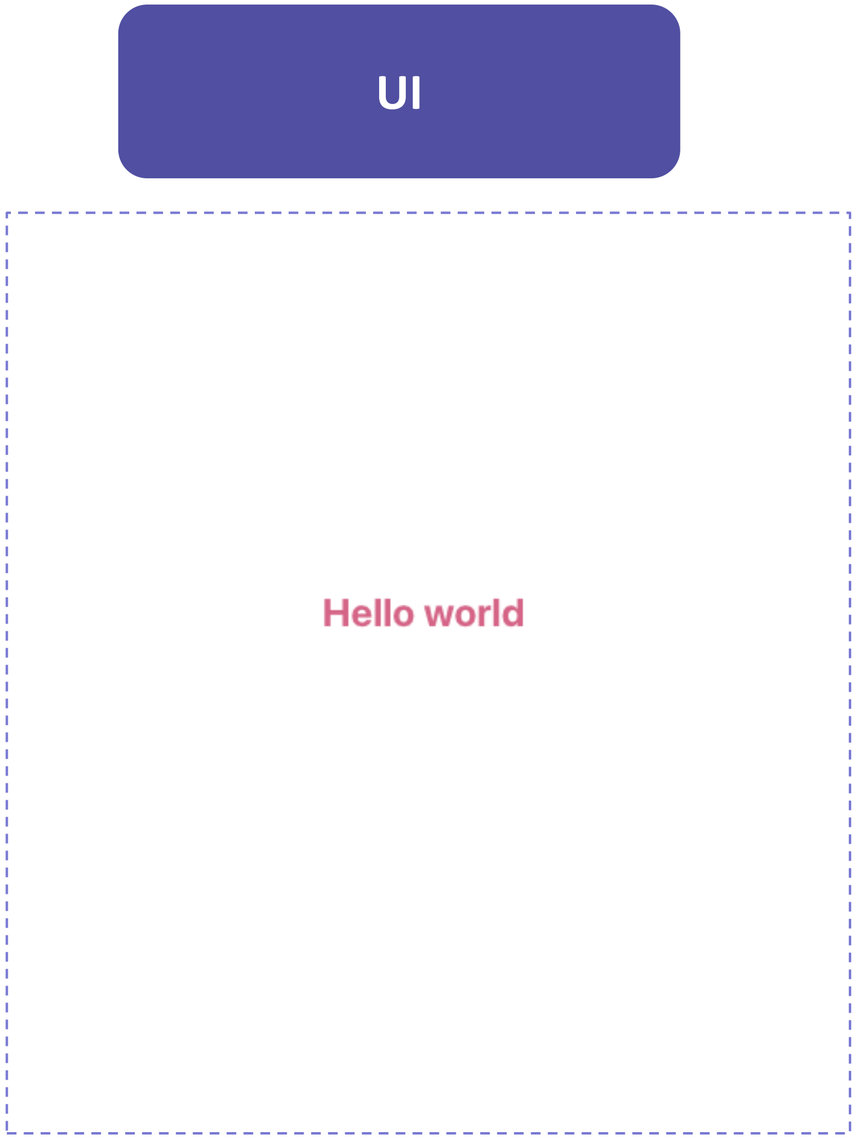
class이름에 random string이 생성되기 때문에 선택자 이름이 겹칠 우려가 없음

스타일링을 직접 개발자가 하나하나 해야 함

CSS-in-JS library (styled component) 예시

App.jsx

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

const Title = styled.h1`
 font-size: 1.5rem;
 text-align: center;
 color: palevioletred;
`;

export default function App() {
 return <Title>Hello World</Title>;
}
```



02

JavaScript template literal

JavaScript template literal

Template literals

템플릿 리터럴은 내장된 표현식을 허용하는 문자열 리터럴입니다.

```
`string text ${expression} string text`
```

쉽게 말해 문자열 안에서 JS 표현식을
사용할 수 있게 하는 문법이다.

JavaScript template literal

Template Literal

```
const string = "elice"
const message = `hello ${string}`

console.log(message)
// 결과 : "hello elice"
```

`${string}` Template literal 안에서 string 사용하기

JavaScript "template literal" vs "string concat"

string concat

```
const name = "elice"
const age = 32
const where = "seoul"
const message = "hello " + name +  ", I am " + age +
                          "year old, and I am living in " + where

console.log(message)
// 결과: hello elice, I am 32year old, and I am living in seoul
```

Template Literal

```
const name = "elice"
const age = 32
const where = "seoul"
const message = `hello ${name}, I am ${age}years old, and I am living
                                                    in ${where}`

console.log(message)
// 결과: hello elice, I am 32years old, and I am living in seoul
```

표현식이 섞인 문자열을 사용하고자 할 때,
String concat과 Template literal의 비교

JavaScript template literal with number

Template Literal

```
const number = 12345
const message = `hello ${number}`

console.log(message)
// 결과 : "hello 12345"
```

`${number}` Template literal 안에서 number 사용하기

JavaScript template literal with boolean

Template Literal

```
const boolean = true
const message = `hello ${boolean}`

console.log(message)
// 결과 : "hello true"
```

`${boolean}` Template literal 안에서 boolean 사용하기

JavaScript template literal with object

Template Literal

```
const object = { a: "apple" }
const message = `hello ${object}`

console.log(message)
// 결과 : "hello [object Object]"
```

`${object}` Template literal 안에서 object 사용하기

JavaScript template literal with ternary operator

Template Literal

```
const name = "chalie"
const gender = "male"
const message = `Hello ${gender === "male" ? "Mr." : "Mrs."}${name}, nice to meet you`

console.log(message)
// 결과 : "Hello Mr.chalie, nice to meet you"
```

`${expression}` Template literal 안에서 expression 사용하기

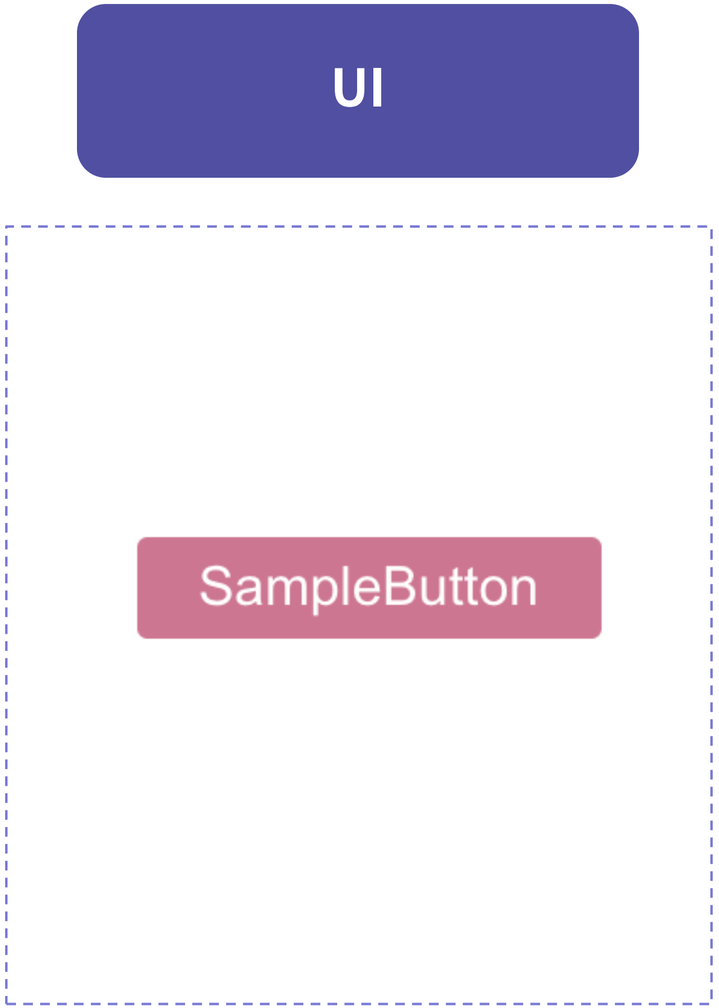
03

styled component 기본기

styled components

tag example

```
const Button = styled.button`
  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;
  background: palevioletred;
  color: white;
`

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

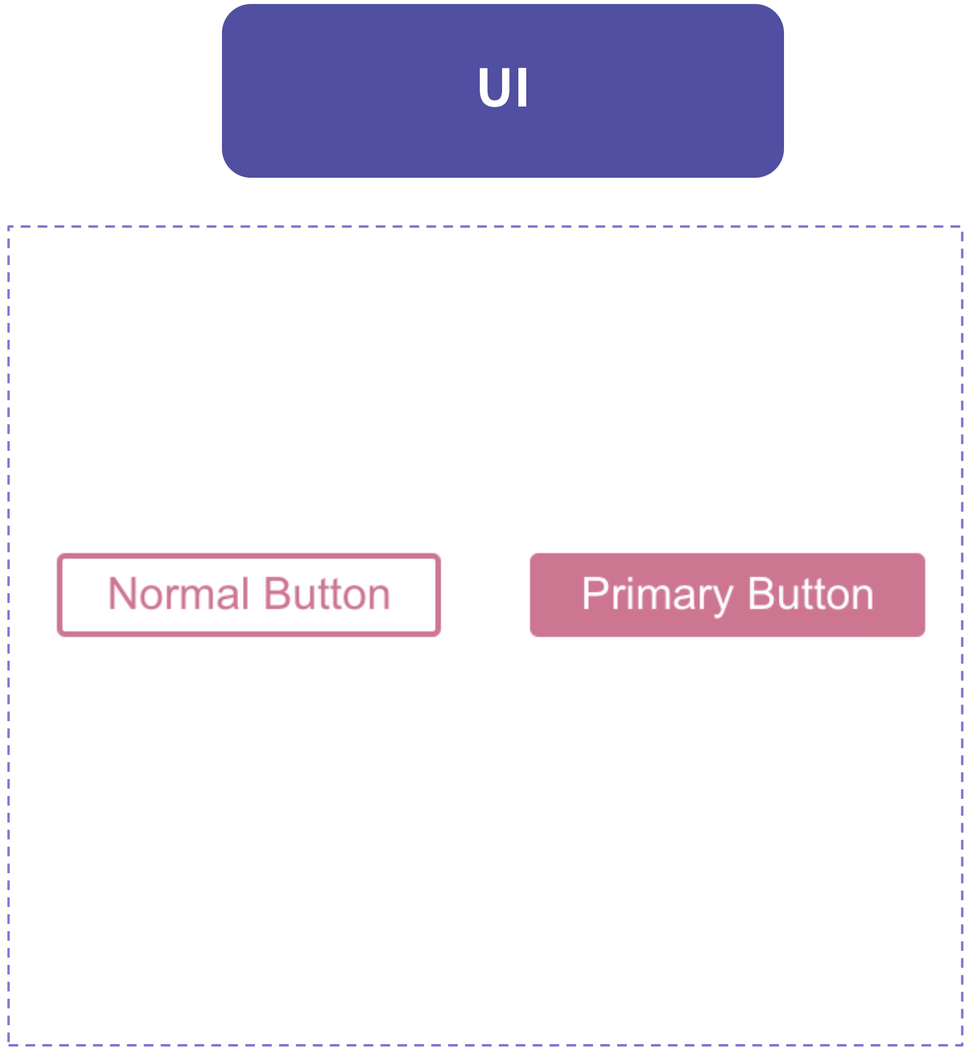


styled component button 만들기

styled components

props example

```
const Button = styled.button`
  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;
  background: ${props => props.primary ? "palevioletred" :
"white" };
  color: ${props => props.primary ? "white" :
"palevioletred" };
`
function App() {
  return (
    <div>
      <Button>Normal Button</Button>
      <Button primary>Primary Button</Button>
    </div>
  ) }
```



styled component props 활용

CSS와 SCSS 비교

CSS example

```css
.name.red {
  color: red;
}
.name .blue {
  color: blue;
}
.name .green {
  color: green;
}
.name .child .yellow {
  color: yellow;
}
.name + .name {
  color: purple;
}
```

SCSS example

```scss
.name {
  &.red {
    color: red;
  }
  .blue {
    color: blue;
  }
  .green {
    color: green;
  }
  .child {
    .yellow {
      color: yellow;
    }
  }
  + .name {
    color: purple;
  }
}
```

SCSS에서는 선택자 중복 회피 가능
이 외에도 SCSS variable 등 여러 추가 기능 가능

SCSS in styled component 예시

styled component example

```
const SCSSStyledDiv = styled.div`
  background-color: orange;
  color: white;

  div {
    background-color: red;
  }

  .purple {
    background-color: purple;
  }
`;
```

Component Use Case

```
function Test() {
  return (
      <SCSSStyledDiv>
        parent elem with className(orange)
        <div>child(red)</div>
        <div className={"purple"}>
          child elem with className(purple)
        </div>
      </SCSSStyledDiv>
  )
}
```

+, & 등 여러 선택자를 사용할 수 있음
중첩 선택자 스타일링 가능

크레딧

/* elice */

코스 매니저
이재성

콘텐츠 제작자
장재영

강사
장재영

감수자
장재영

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io