


실증 아카데미 리액트 특강

React.js로 스타일링 하기

일시 2021년 01월 17일

주최 동아대학교



CONTENTS—

01 HTML에서 스타일 적용
인라인 스타일
스타일 태그
.css 파일 로드

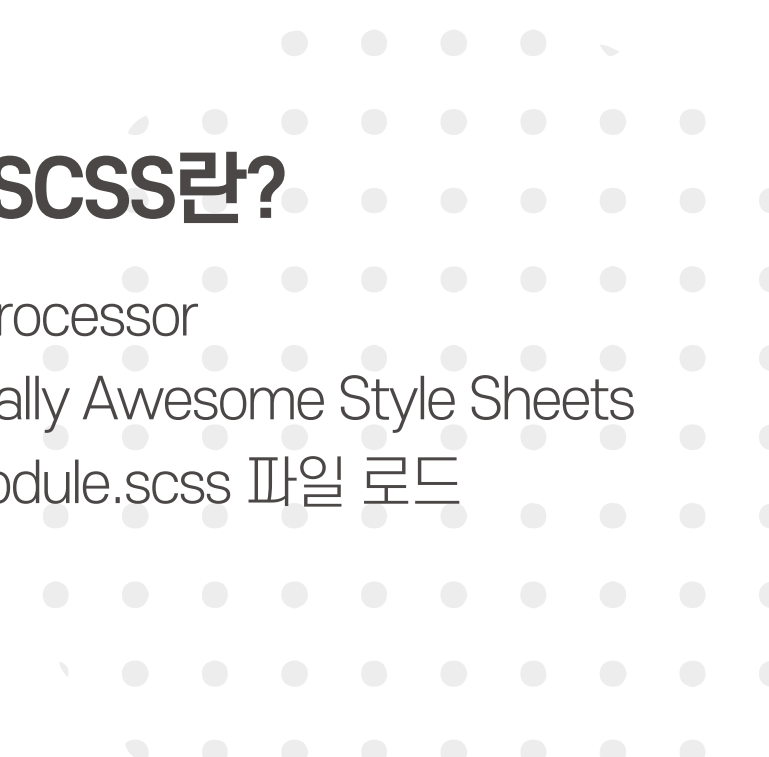
02 React.js 인라인 스타일
스타일 어트리뷰트
식별자 네이밍 규칙

03 CSS in JavaScript
CSS-in-JavaScript란?
CSS-in-JavaScript 라이브러리

04 @emotion 사용하기
@emotion 설치
@emotion 코드 작성

05 React.js CSS 로드
.css 파일 로드
.module.scss 파일 로드

06 SASS, SCSS란?
CSS Preprocessor
Syntactically Awesome Style Sheets
.scss, .module.scss 파일 로드



01

HTML에서 스타일 적용

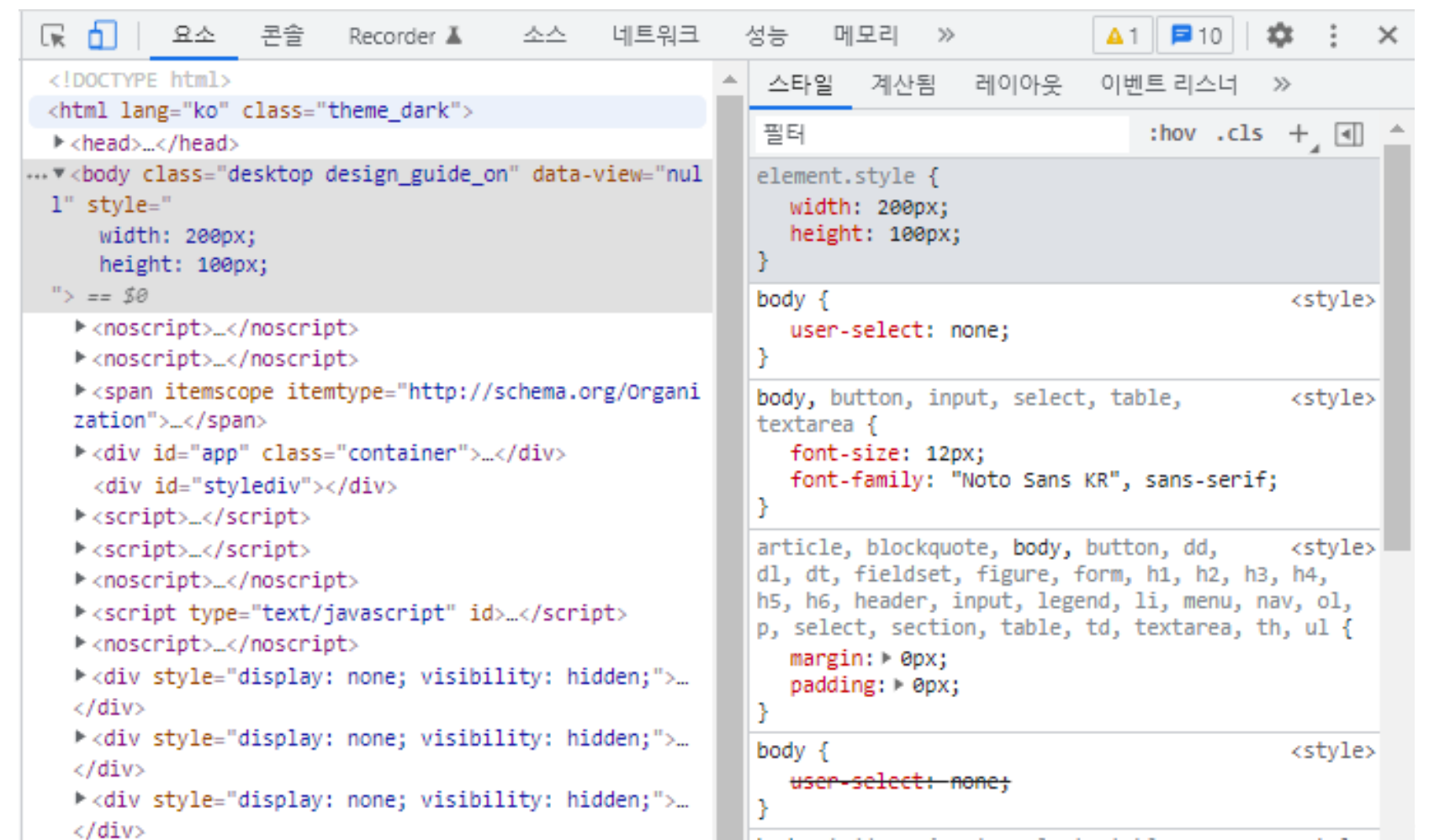
인라인 스타일
스타일 태그
.css 파일 로드

인라인 스타일

HTML 문서의 스타일을 지정할 태그에 직접 적용

```
<div style="width: 100px; height: 100px"></div>
<span style="width: 100px; height: 100px"></span>
```

- HTML 요소 style 어트리뷰트에 직접 기입하는 방법
- 큰 따옴표("") 내에 스타일 코드를 작성
- 단일 HTML 요소에만 적용되므로 재사용성은 굉장히 떨어짐
 - ※ 모두 적용하기 위해서 각각의 태그에 모두 코드를 작성하는 것은 낭비
- 일반적인 경우에선 가장 높은 우선순위로 적용
- 개발자 도구에서 디버깅/테스트 용으로 활용하기 좋음



스타일 태그

HTML 문서의 스타일을 모아두어 관리

```
<style>
```

```
div { width: 100px; }
```

```
.class-name { width: 100px; height: 50px; }
```

```
#id-name { width: 400px; height: 40px; }
```

```
</style>
```

- style 태그 안에 적용할 스타일 코드를 작성하는 방법
- HTML 문서 어디에 작성하여도 동일한 효과 지정
- 만일 같은 스타일 지정식이 있으면 뒤에 작성한 코드가 적용
- 한 HTML 문서에 종속된 지정 방식
- HTML 문서 내에 있으므로 JavaScript를 통해 조작 가능

※ 스타일 코드 작성법

선택자 { 속성: 값; }

```
/* 전체 선택자: 모든 요소를 선택 */
* {
  ...;
}
/* 태그 선택자: 모든 p 태그 요소를 모두 선택 */
p {
  ...;
}
/* id 선택자: id 값이 'foo'인 요소를 모두 선택 */
#foo {
  ...;
}
/* class 선택자: class 값이 'foo'인 요소를 모두 선택 */
.foo {
  ...;
}
/* 어트리뷰트 선택자: input 요소 중에 type 어트리뷰트 값이 'text'인 요소를 모두 선택 */
input[type="text"] {
  ...;
}
/* 후손 선택자: div 요소의 후손 요소 중 p 요소를 모두 선택 */
div p {
  ...;
}
/* 자식 선택자: p 요소의 자식 요소 중 p 요소를 모두 선택 */
div > p {
  ...;
}
/* 이전 형제 선택자: p 요소의 형제 요소 중에 p 요소 바로 뒤에 위치하는 ul 요소를 선택 */
p + ul {
  ...;
}
/* 일반 형제 선택자: p 요소의 형제 요소 중에 p 요소 뒤에 위치하는 ul 요소를 모두 선택 */
p ~ ul {
  ...;
}
/* 가상 클래스 선택자: hover 상태인 a 요소를 모두 선택 */
a:hover {
  ...;
}
/* 가상 요소 선택자: p 요소의 콘텐츠 앞에 위치하는 공간을 선택
일반적으로 content 프로퍼티와 함께 사용 */
p::before {
  ...;
}
```

.css 파일 로드

.css 파일을 통해 스타일을 외부에서 관리

```
<head>
  <link rel="stylesheet" href="/expand_style.css">
</head>
```

- .css 파일에 적용할 스타일 들을 모아서 작성하는 방법
- link 태그에 .css 파일 경로를 입력하여 불러옴
- HTML과 CSS를 분리하여 관리할 수 있는 장점
- 여러 HTML 문서에 사용 가능한 지정 방식

```
.div {
  width: 100px;
  height: 200px;
}
```

```
.div > p {
  font-size: 12px;
  color: #f0f0f0;
}
```

- /expand_style.css 파일 예제



02

React.js 인라인 스타일

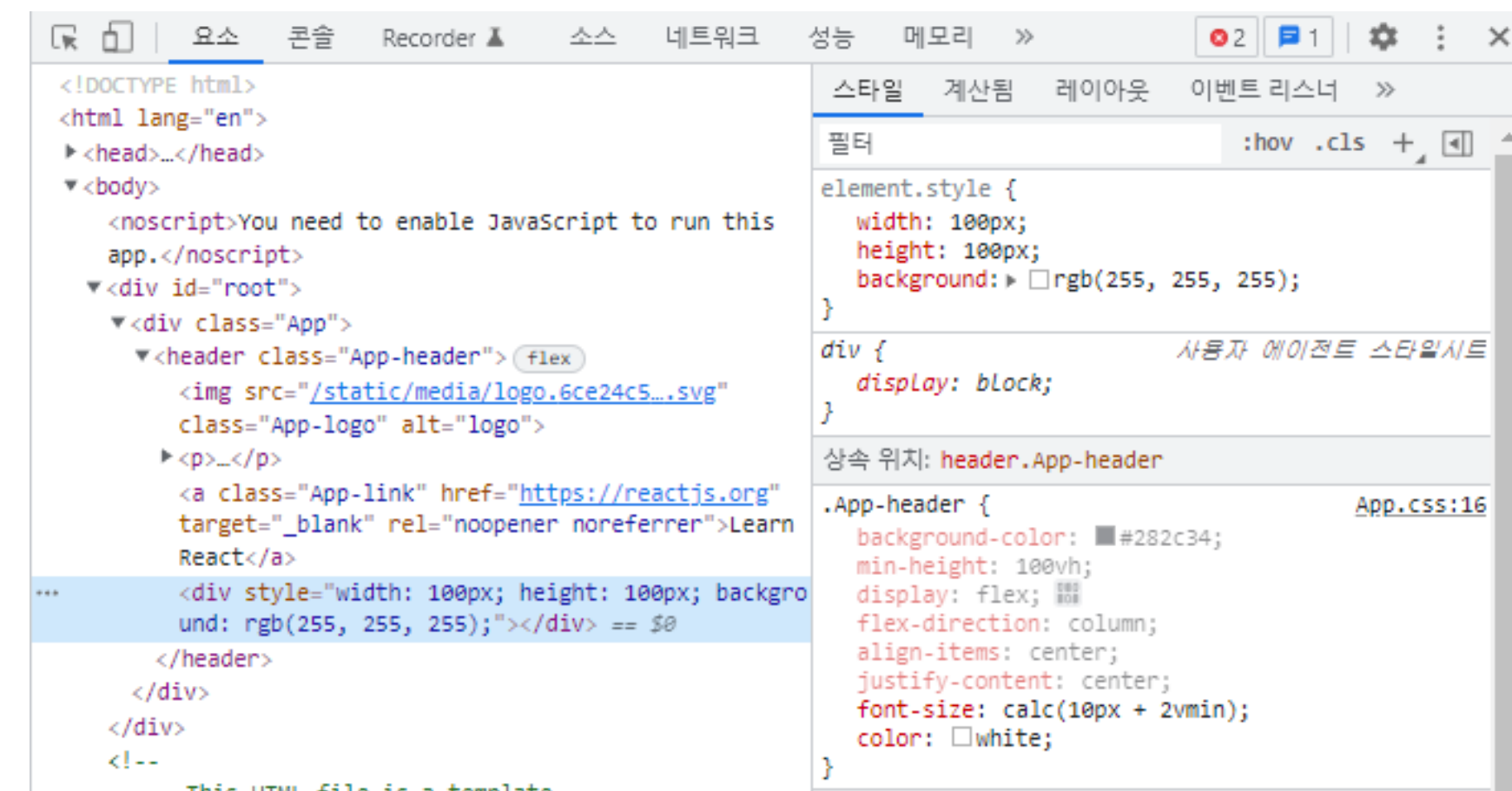
스타일 어트리뷰트
식별자 네이밍 규칙

스타일 어트리뷰트

JSX의 HTML 요소 또한 style 어트리뷰트 존재

```
const ExampleComponent = () => {
  return (
    <div
      style={{
        width: 100,
        height: "100px",
        backgroundColor: 12,
        // background: "#fff",
        background: "rgba(0, 0, 0, 0.6)"
      }}
    />
  );
};
```

- style 어트리뷰트에는 style 객체를 지정해줌
- 스타일에 알맞지 않은 프로퍼티는 파싱하지 않음
- 프로퍼티 네이밍 규칙을 준수



식별자 네이밍 규칙

JavaScript의 객체 키 값으로 사용하기 적절한 네이밍 방법

- React는 식별자 네이밍 규칙을 준수하지 않는 스타일 속성의 이름을 규칙을 준수하는 이름으로 변환하여 지정할 수 있도록 함
- 대체적으로 camelCase로 명명하며 이는 HTML 어트리뷰트에도 적용
Ex) onclick => onClick, background-color => backgroundColor

- 특수문자를 제외한 문자, 숫자, 언더스코어(_), 달러 기호(\$) 포함 가능
- 숫자로 시작하는 것은 불가능
- 예약어는 식별자로 사용 불가 (strict mode)

function, try, catch, break, async, await, class, const, for, delete, do, if, else 등등

※ ES5부터 유니코드 문자를 허용하기 때문에 한글, 일본어, 한자도 가능하지만, 권장하지 않음



03

CSS in JavaScript

CSS-in-JavaScript란?
CSS-in-JavaScript 라이브러리

CSS-in-JavaScript란?

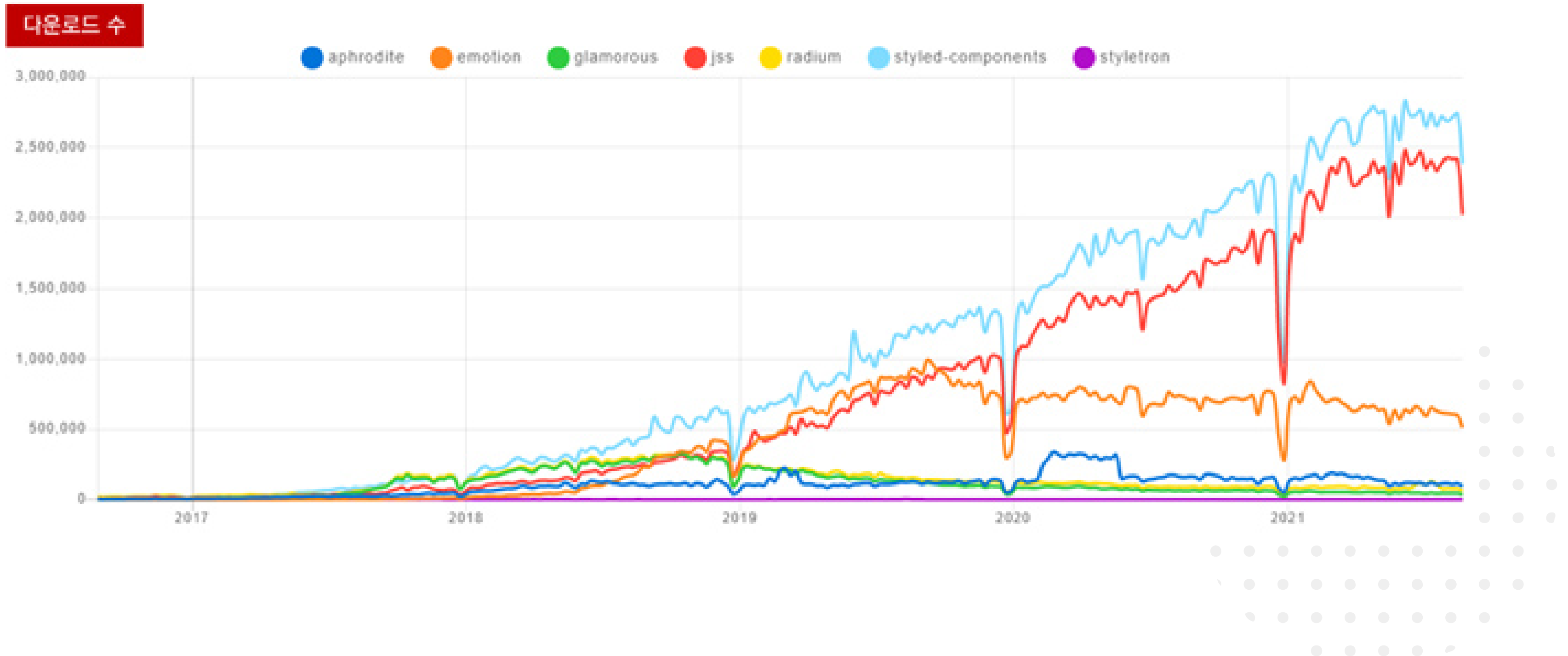
JavaScript 코드에서 CSS를 작성하는 방식

- React에서는 CSS-in-JavaScript를 지원하는 라이브러리를 통해 마치 하나의 컴포넌트를 작성하는 것처럼 사용할 수 있음
- 때문에 실제 로직이 작성되는 컴포넌트로부터 스타일을 담당하는 영역, 인라인 스타일과 className까지 모두 분리하여 사용할 수 있음
- 그 후 실제 렌더링 될 때는 HTML 내의 <style> 태그로 파싱됨

- JavaScript 환경을 최대한 활용
- JavaScript 상수, 함수 쉽게 공유 가능
- 짧은 이름의 클래스로 코드 경량화
- 러닝 커브가 가파른 단점
- 새로운 의존성(모듈)이 발생
- 라이브러리로 인한 번들 크기가 커짐
- CSS 파일 로드 방식에 비해 느린 성능



CSS-in-JavaScript 라이브러리 현황



< 🛠️ > styled-components

Style your React.js apps without stress

styled-components

- 가장 유명한 CSS-in-JavaScript 라이브러리
- 문서의 양이 방대함
- 태그의 이름을 식별하기 어려움

@emotion

- styled-components에 비해 가벼움
- div에 CSS 코드를 주입하는 방식(컴포넌트 형도 가능)
- 개발자 친화적이며, 고성능 및 유연성을 갖추



04

@emotion 사용하기

@emotion 설치

@emotion 코드 작성

@emotion 설치

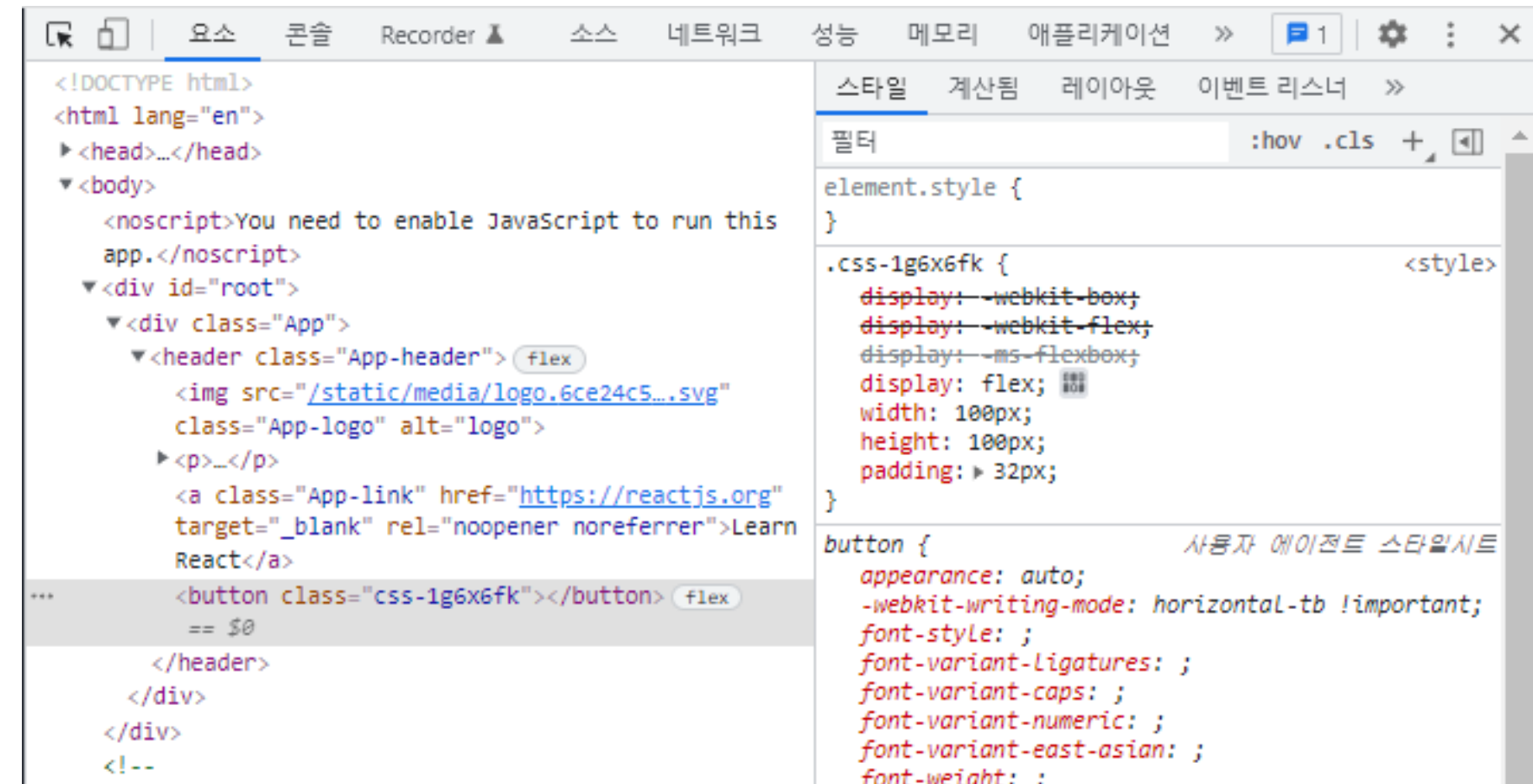
yarn add @emotion/react @emotion/styled

import styled from "@emotion/styled";

```
const StyleDiv = styled.button`
  display: flex;
  width: 100px;
  height: 100px;
  padding: 32px;
`;
```

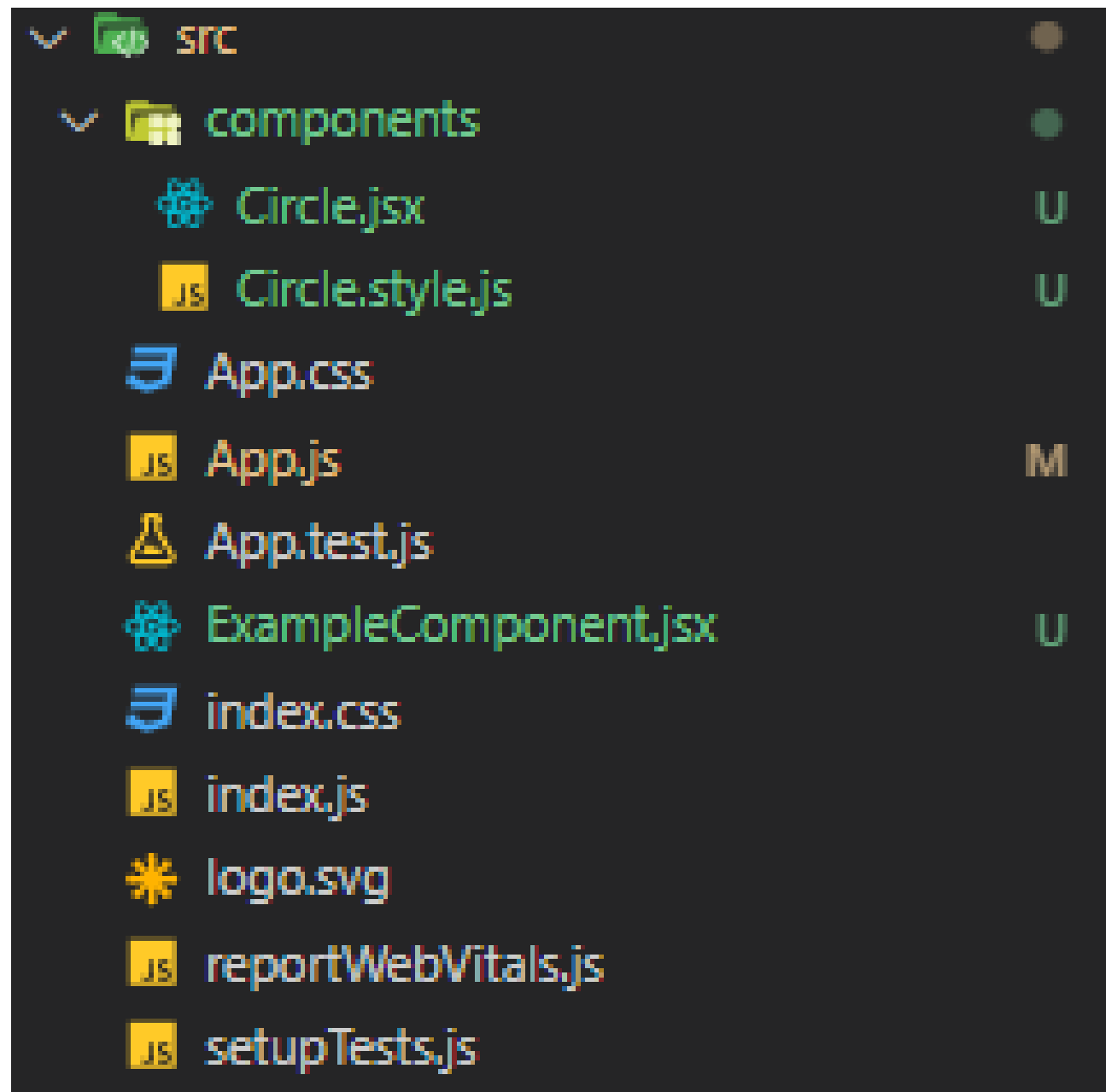
```
const ExampleComponent = () => {
  return <StyleDiv />;
};
```

- 벤더 프리픽스가 자동으로 적용되는 것을 확인



@emotion 코드 작성

컴포넌트와 스타일 파일 생성



- /components 폴더 내에 Circle.jsx, Circle.style.js 파일 생성

```
import Circle from "../components/Circle";
```

```
function App() {  
  return <Circle />;  
}
```

```
export default App;
```

- App.js에서 Circle.jsx를 import



@emotion 코드 작성

컴포넌트, 스타일 코드 작성

```
import { StyleCircleBox, StyleCircleText } from "./Circle.style";
```

```
const Circle = () => {
  return (
    <>
      <StyleCircleBox>
        <StyleCircleText>Inner Box</StyleCircleText>
      </StyleCircleBox>
      <StyleCircleText red>Outer Box</StyleCircleText>
      <StyleCircleBox color="#0f6"></StyleCircleBox>
    </>
  );
};
export default Circle;
```

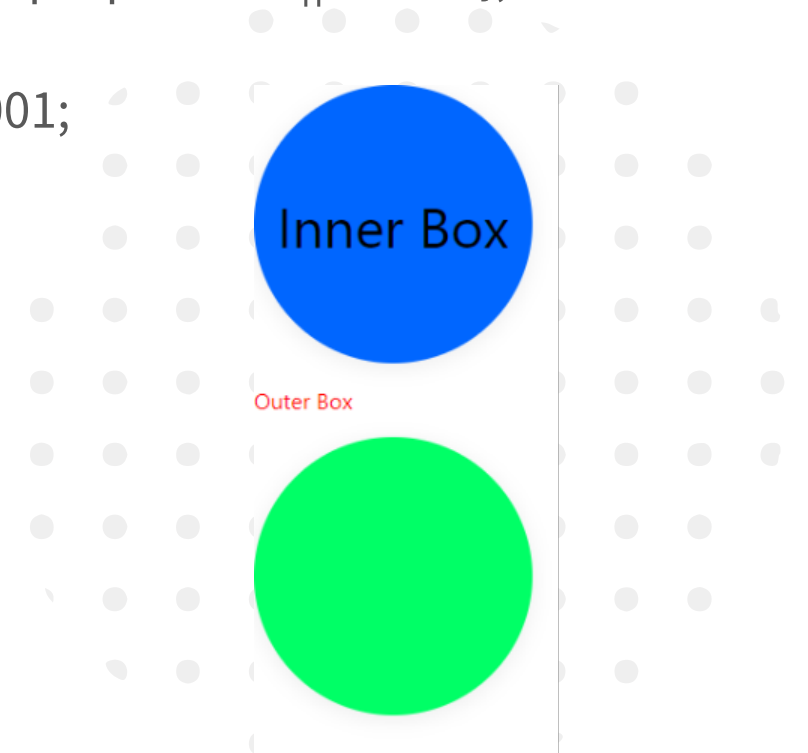
- Circle.jsx 파일

```
import styled from "@emotion/styled/macro";
```

```
export const StyleCircleText = styled.p((props) => ({
  color: props.red && "#f00",
}));
```

```
export const StyleCircleBox = styled.div`
  display: flex;
  align-items: center;
  justify-content: center;
  width: 200px;
  height: 200px;
  background-color: ${({props} => props.color || "#06f"};
  border-radius: 50%;
  box-shadow: 0px 3px 15px #0001;
  ${StyleCircleText} {
    font-size: 40px;
  }
`;
```

- Circle.style.js 파일





05

React.js CSS 로드

.css 파일 로드

.module.scss 파일 로드

.css 파일 로드

import를 사용해 로드

webpack이라는 도구의 css-loader를 사용하여 .css import 가능

import된 .css는 추후 경로가 매핑되어 HTML 문서 상단 <head> 영역에 link 태그를 통해 로드 됨.

```
App.js > App
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

```
src > App.css > .App-header
1  .App {
2    text-align: center;
3  }
4
5  .App-logo {
6    height: 40vmin;
7    pointer-events: none;
8  }
9
10 @media (prefers-reduced-motion: no-preference) {
11   .App-logo {
12     animation: App-logo-spin infinite 20s linear;
13   }
14 }
15
16 .App-header {
17   background-color: #282c34;
18   min-height: 100vh;
19   display: flex;
20   flex-direction: column;
21   align-items: center;
22   justify-content: center;
23   font-size: calc(10px + 2vmin);
24   color: white;
25 }
26
27 .App-link {
28   color: #61dafb;
29 }
30
31 @keyframes App-logo-spin {
32   from {
33     transform: rotate(0deg);
34   }
35   to {
36     transform: rotate(360deg);
37 }
```

.module.css

CSS의 클래스 이름이 중첩되는 것을 방지하기 위한 기술

```
import logo from './logo.svg';
import styles from './App.module.css';
```

```
function App() {
  return (
    <div className={styles.App}>
      <header className={styles["App-header"]}>
        <img src={logo} className={styles["App-logo"]} alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
      </header>
    </div>
  );
}
export default App;
```

- 기존의 App.css 파일 명을 App.module.scss로 바꾼 후 다음과 같이 import

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      <div class="App_App__PTYEe">
        <header class="App_App-header__KAZ+5" flex>
          
        </header>
      </div>
    </div>
  </body>
</html>
```

중복되지 않도록 변경된 클래스 이름

06

SASS, SCSS란?

CSS Preprocessor

Syntactically Awesome Style Sheets

.scss, .module.scss 파일 로드

CSS Preprocessor

CSS 동작 전에 사용하는 전처리기



특수 구문
불편 기능 해소



CSS 코드
HTML 사용

- CSS 코드의 불편한 점을 해소하는 코드 작성
- 컴파일을 거쳐 CSS 코드로 변경
- 반복문, 조건문, 변수와 같은 프로그래밍 가능

- HTML에서 사용되는 CSS 코드로 출력
- 웹에서 동작하는 표준 형식에 맞춰짐

Syntactically Awesome Style Sheets

SASS, 가장 유명한 CSS 전처리기

- 처음에는 SASS라는 일반 CSS와는 조금 다른 문법의 전처리기로 등장(Python과 비슷하게 블록이 없었음)
- 그러나 이는 기존 CSS와의 차리오 인해 불편함을 가져와 SCSS를 통해 CSS 문법과 유사하면서도 전처리가 가능함
- 2006년부터 시작하여 높은 성숙도와 많은 커뮤니티를 가지고 있는 훌륭한 언어
- 컴파일 하기 위해서는 node-sass, SassMeister, Gulp, webpack, parcel 등의 컴파일러가 필요함
- // 주석을 사용할 수 있음
- 데이터 타입과 변수가 있음



SCSS	CSS
<pre>1 section { 2 height: 100px; 3 width: 100px; 4 5 .class-one { 6 height: 50px; 7 width: 50px; 8 9 .button { 10 color: #074e68; 11 } 12 } 13 }</pre>	<pre>1 section { 2 height: 100px; 3 width: 100px; 4 } 5 6 section .class-one { 7 height: 50px; 8 width: 50px; 9 } 10 11 section .class-one .button { 12 color: #074e68; 13 }</pre>

.scss, .module.scss 파일 로드

- 중첩 (Nesting)

```
$width: 100%;
.section {
  width: $width
  .list {
    padding: 20px;
    li {
      float: left;
    }
    font: {
      size: 10px; weight: bold;
    }
  }
  &.wrapper {
    width: $width / 2;
  }
}
```

- 코드를 중첩하여 하위에 지정할 클래스를 직관적으로 설정할 수 있음

- & 는 블록 자기 자신을 참조하며, 위 코드에서 &는 .section을 가리킴

- 두 번째 요소부터 margin을 적용하는 코드

```
.item {
  & + & {
    margin-top: 10px;
  }
}
```

- 가상 선택자를 쉽게 선택할 수 있음

```
.item {
  &:hover {
    background-color: #06f;
  }
  &::after {
    content: 'after';
  }
}
```


SCSS 문법

yarn add node-sass

- 파일 명을 .scss, .module.scss로 변경 후 import
- 우측과 같이 코드 작성하면 이전과 같이 적용됨
- 하지만 이전과는 다르게 클래스가 적용된 HTML 요소의 위치를 변경하면, 스타일이 제대로 적용되지 않음
- 중첩이 제대로 적용되었기 때문

```
.App {
  text-align: center;
  .App-header {
    background-color: #282c34;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    font-size: calc(10px + 2vmin);
    color: white;
  }
  .App-logo {
    height: 40vmin;
    pointer-events: none;
    @media (prefers-reduced-motion: no-preference) {
      animation: App-logo-spin infinite 20s linear;
    }
  }
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

실증 아카데미 리액트 특강

THANK
YOU

(주) 인바이즈 박철현
(주) 인바이즈 개발팀