컴포넌트 스타일링

일반 CSS

컴포넌트 이름-클리스 이름 형태

```
.App-logo {
}
.App-header {
}
.App-link {
}
```

CSS Selector

최상위 html요소에는 컴포넌트의 이름을 사용

```
.App .logo {
}
.App header {
}
.App a {
}
```

Sass(Syntacically Awesome Style Sheets)

- 복잡한 작업을 쉽게 할 수 있도록 해줌
- 스타일 코드의 재활용성을 높혀 줌
- 코드의 가독성을 높혀줌
- scss는 css와 거의 같은 문법으로 sass기능을 지원
    - {}(중괄호)와 ;(세미콜론)의 유무

.sass vs .scss

.sass

```
$font-stack: Helvetica, sans-serif
$primary-color: #333

body
    font: 100% $font-stack
    color: $primary-color
```

.scss

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

.scss문법이 더 자주 사용 됨

사용

```
$ yarn add node-sass
```

특징

- 파일을 분리하여 import가능

```
@import "./style/utils.scss";
```

sass-loader 커스터마이징

- **create-react-app**은 세부 설정이 모두 감춰져 있음
- `yarn eject` 명령어로 설정을 꺼냄
    - git에 변화가 있으면 안됨

webpack.config.js

```
{
    ...
    use: getStyleLoaders({
        importLoaders: 3,
        sourceMap: isEnvProduction && shouldUseSourceMap,
    }).concat({
        loader: require.resolve('sass-loader'),
        options: {
        prependData:`@import 'utils';`,
        sassOptions: {
            includePaths: [paths.appSrc + '/styles'],
            sourceMap: isEnvProduction && shouldUseSourceMap,}
        }
    }), sideEffects: true,
},
```

node_modules에서 불러 오기

- **include-media** 반응형 디자인을 쉽게 만들어 줌
- **open-color** 편리한 색상 팔레트

```
$ yarn add open-color include-media
```

```
@import '~include-media/dist/include-media';
@import '~open-color/open-color';
.SassComponent {
  display: flex;
  background: $oc-gray-2;
  @include media('<768px') {
      background: $oc-gray-9;
  }
  ...
}
```

CSS Module

- **[파일 이름]_[클래스 이름]__[해시값]** 을 자동으로 만듦
- **.module.css**로 사용하면 됨
- 고유성에 대한 고민을 하지 않아도 됨
    - 우리가 만든 스타일을 직접 불러온 컴포넌트 내부에서 만 작동
    - `:global` 키워드를 사용하면 해당 웹페이지 내에서 전역적으로 적용

./CSSModule.module.css

```
.wrapper {
}
.inverted {
}
:global .something {
}
```

```
import styles from './CSSModule.module.css';

const CSSModule = () => {
    return (
        <div className={styles.wrapper}>
        // <div className={`${styles.wrapper} ${styles.inverted}`}>
            Hi, <span className="something">CSS Module!</span>
        </div>
    )
```

`{styles.wrapper}`가 `{wrapper: "CSSModule_wrapper__1sbdQ}`로 바뀜

classnames

- 조건부 CSS 클래스 설정에 유용한 라이브러리
- `yarn add classnames`
- mad-talk에서도 사용 중

```
import classNames from "classnames";
const myClass = "six";
classNames("one", "two", { three: false }, ["four", "five"], myClass);
```

classnames 사용

```
const MyComponent = ({ highlighted, theme }) => (
  <div className={classNames("MyComponent", { highlighted }, theme)}>Hello</div>
);
```

classnames 미사용

```
const MyComponent = ({ highlighted, theme }) => (
  <div className={`MyComponent ${theme} ${highlighted ? "highlighted" : ""}`}>
    Hello
  </div>
);
```

styled-components

- **CSS-in-JS**: js 파일 안에 스타일을 선언하는 방식
- 개발자들이 선호하는 **styled-components**를 사용
- .css 또는 .scss파일을 만들지 않아도 됨

```
import styled, { css } from 'styled-components';
const Box styled.div`
    background: ${props => props.color || 'blue'};;
`;
export default () => (
    <Box color="black">
    </Box>
)
```

React에서 제공하는 방법

```
    render() {
        const { title, size } = this.props;
        const styles = {
            container: {
                fontSize: size
            }
        };
        return <span style={styles.container}>{title}</span>;
    }
```

Mad Talk(webapp)

- `./sass`에 .scss파일들이 있음
- `_`를 이용해 모듈화 적용
  - `_filename`을 이용하면 부분파일로 인식되며 css파일로 생성되지 않음
- `root.jsx`에서 최상위 scss인 `sass/style.scss`를 불러 옴
- 그외 컴포넌트 자체 scss 사용하는 경우도 있음

결론

css	기본적인 css
sass	가독성, 상속, 재활용성 등을 높힌 css
css modules	불러온 컴포넌트 내부에서만 사용하는 css
classnames	조건부 css 적용에 유용한 라이브러리
CSS-in-JS	자바스크립트 파일 안에서 스타일 선언