**Styled-components**

1. CSS in JS 란 기술을 사용함. JS안에 CSS를 작성한다는 의미
2. styled-components는 현존하는 CSS in JS 관련 리액트 라이브러리 중에서 가장 인기있음
3. 대안으로는 emotion과 styled-jsx 가 있음

**Tagged Template Literal**

1. template literal : 문자열 조합을 쉽게해주는 ES6 문법

const name = ‘react’;

const message = `hello ${name}`;

console.log(message);

1. 다음과 같이 template literal 를 사용하면서, 함수를 사용하는 것이 tagged template literal

const red = ‘빨간색’;

const blue=’파란색’;

function favoriteColors(texts, …values) {

return texts.reduce((result, text, i) =>

`${result}${text}${values[i] ? `<b>${values[i]}</b>` : ‘’}`. ‘’

);  
};  
favoriteColors`제가 좋아하는 색은 ${red}과 ${blue}입니다.`

1. 위 같은 함수는 쓸 일이 없으나 styled-components에서 활용한다

const StyledDiv = styled`

background: ${props => props.color};

`;

**Styled-components 사용하기**

1. 설치하기

yarn add styled-components

1. styled-components 를 사용하면 스타일을 입력함과 동시에 해당 스타일을 가진 컴포넌트를 만들 수 있음.

div를 스타일링 하고 싶으면 styled.div, input을 스타일링 하고 싶으면 styled.input 이런식 사용하면됨

그리고 사용할 떄는, 지정한 이름으로 사용하면 됨.

ex) app.js

import styled from ‘styled-components’;

const Circle = styled.div`

width:5rem;

height:5rem;

background: black;

border-radius: 50%;

`;

function App() {

return <Circle />;

}

1. 다음처럼 props 를 줄 수도 있음

const Circle = styled.div`

width: 5rem;

height: 5rem;

background: ${props=> props.color || ‘black’};

border-radius: 50%;

`;

function App() {

return <Circle color=”blue” />;

}

1. 여러줄의 CSS 코드를 조건부로 보여주고 싶다면 css라는 걸 사용해야함

import styled, { css } from "styled-components";

const Circle = styled.div`

  width: 5rem;

  height: 5rem;

  background: ${(props) => props.color || "black"};

  border-radius: 50%;

  ${(props) =>

    props.huge &&

    css`

      width: 10rem;

      height: 10rem;

    `}

`;

function App() {

  return <Circle color="blue" huge />;

}

**버튼 만들기**

1. 특별한 거 없음

**polished 스타일 관련 유틸함수 사용하기**

1. Sass 의 lighten() 또는 darken()과 같은 함수를 사용할 때 씀
2. 설치 : yarn add polished
3. lighten, darken 사용

* 첫번째 파라미터가 어느정도, 두번째가 기준색이다.

//button.js

import { darken, lighten } from ‘polished’;

…

&:hover {

background: ${lighten(0.1, ‘#228be6’)};

}

&:active {

background: ${darken(0.1, ‘#228be6’)};

}

**ThemeProvider 로 색상코드를 전역적으로 설정하기**

1. ThemeProvider 내부에 렌더링된 컴포넌트로, ThemeProvider에 지정된 값을 조회할 수 있음
2. 조회할 때는 아래 예 같은 경우 props.theme 로 조회한다. 이름은 꼭 theme가 아니어도 된다.
3. 즉, 그냥 ThemeProvider로 지정된 건, 밑 컴포넌트의 props로 던져주는 것

//app.js

import styled, { ThemeProvider } from "styled-components";

…

function App() {

  return (

    <ThemeProvider

      theme={{

        palette: {

          blue: "#228be6",

          gray: "#405057",

          pink: "#f06595",

        },

      }}

    >

      <AppBlock>

        <Button color="gray">BUTTON</Button>

      </AppBlock>

    </ThemeProvider>

  );

}

//button.js

  /\* 색상 \*/

  ${(props) => {

    const selected = props.theme.palette[props.color];

    return css`

      background: ${selected};

      &:hover {

        background: ${lighten(0.3, "#228be6")};

      }

      &:active {

        background: ${darken(0.3, "#228be6")};

      }

    `;

  }}

  /\* 기타 \*/

  & + & {

    margin-left: 1rem;

  }

`;

function Button({ children, ...rest }) {

  return <StyledButton {...rest}>{children}</StyledButton>;

}

Button.defautProps = {

  color: "blue",

};

export default Button;