

day43-RN-stylecomponent

☰ 태그	
📅 날짜	@2022년 12월 2일

웹 프로그래밍과 유사한 만큼 차이가 있는 부분에서는 쉽게 혼동하거나 실수하는 등 불편하게 느껴지는 부분도 존재한다.

하이픈을 사용하지 않고 카멜 표기법으로 작성해야 한다는 차이.

몇몇 값들은 웹프로그래밍에서 익숙한 css의 속성과 이름이 같지만 타입이 다르거나 단위가 생략.

width, height처럼 입력되는 값에 맞춰 타입이 달라지는 경우

이런 불편한 점들은 스타일드 컴포넌트(styled-components로 해소할 수 있다.

- 스타일드 컴포넌트 : <http://styled-components.com/>

스타일드 컴포넌트는 자바스크립트 파일 안에 스타일을 작성하는 css-in-js라이브러리이며, 스타일이 적용된 컴포넌트라고 생각하면 이해하기 쉽다.

- `npm install styled-components`

설치하기

에러나면

<https://jiny-dongle.tistory.com/13> 참고하자

```

src > JS App.js > App
1  import React from "react";
2  import { View, Text } from 'react-native';
3  import { viewStyles, textStyles } from "../styles";
4  import { Header, Contents, Footer } from "../components/Layout";
5  import ShadowBox from "../components/ShadowBox";
6  import styled from 'styled-components/native';
7
8  const App = () => {
9    return(
10     <View style={viewStyles.container}>
11       <ShadowBox />
12
13       <Text>123</Text>
14
15     </View>
16   );
17 };
18
19
20 const MyTextComponent = styled.Text`
21   color : #fff;
22 `;
23
24
25
26 export default App;

```

“styled.[컴포넌트 이름]” 형태 뒤에 백틱(`)을 사용하여 만든 문자열을 붙이고 그 안에 스타일을 지정하면 된다.

이러한 문법을 태그드 템플릿 리터럴(Tagged Template Literals)라고 한다.

주의할 점은 styled 뒤에 작성하는 컴포넌트의 이름은 반드시 존재하는 컴포넌트를 지정해야 한다는 것이다.

이 코드는 리액트 네이티브의 Text컴포넌트에서 글자 색이 흰색으로 스타일링된 MyTextComponent라는 새로운 컴포넌트가 된다.

스타일을 작성하다 보면 많은 스타일이 중복되어 재사용 가능한 코드를 분리해내는 상황이 있습니다. 스타일드 컴포넌트에서는 css를 이용하여 재사용 가능한 코드를 관리할 수 있다.

```

1  import React from "react";
2  import { View, Text } from 'react-native';
3  import { viewStyles, textStyles } from "../styles";
4  import { Header, Contents, Footer } from "../components/Layout";
5  import ShadowBox from "../components/ShadowBox";
6  import styled, { CSS } from 'styled-components/native';
7
8  const App = () => {
9    return (
10     <View style={viewStyles.container}>
11       <ShadowBox />
12
13       <Text>123</Text>
14
15     </View>
16   );
17 };
18
19
20 const MyTextComponent = styled.Text`
21   color : #fff;
22 `;
23
24 const whiteText = CSS`
25   color : #fff;
26   font-size : 14px;
27 `;
28
29 const MyBoldTextComponent = styled.Text`
30   ${whiteText}
31   font-weight : 600;
32 `;
33
34 const MyLightTextComponent = styled.Text`
35   ${whiteText}
36   font-weight : 200;
37 `;

```

재사용 가능한 스타일 코드를 관리하는 방법 외에 완성된 컴포넌트를 상속받아 이용하는 방법도 있다.

```

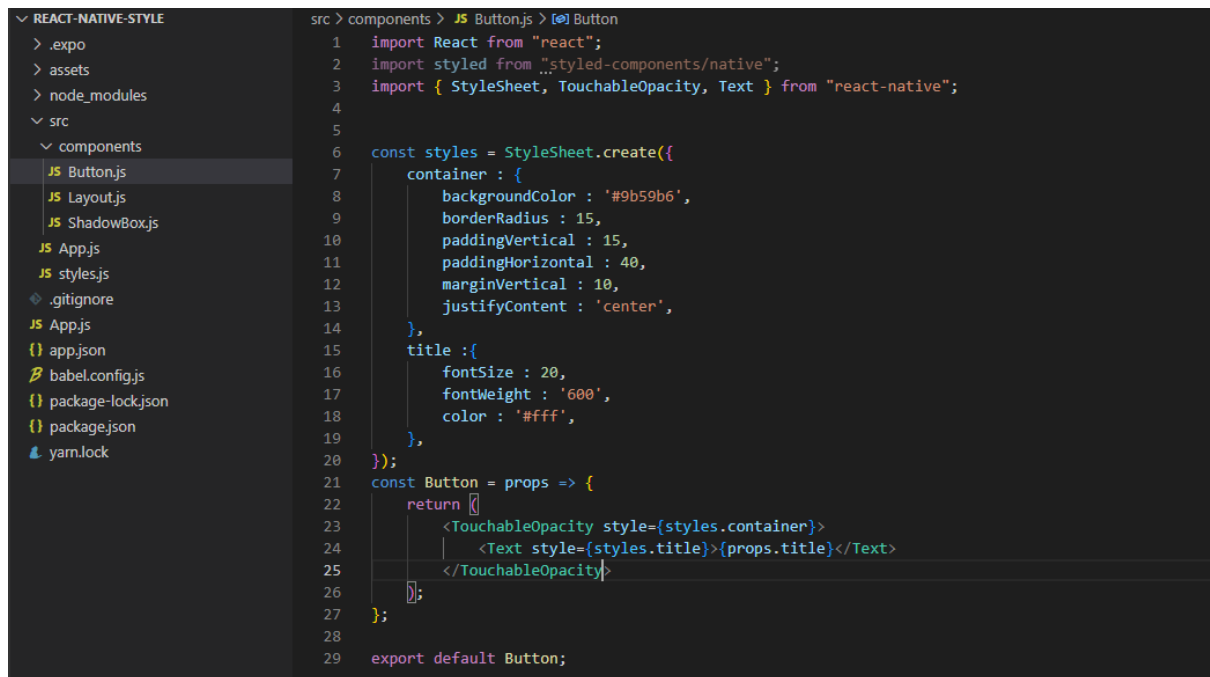
const StyledText = styled.Text`
  color: #000;
  font-size : 20px;
  margin : 10px;
  padding : 10px;
`;

const ErrorText = styled(StyledText)`
  font-weight : 600;
  color : red;
`;

```

ErrorText 컴포넌트는 StyledText컴포넌트의 스타일을 그대로 상속받은채로 글자의 두께와 색만 변경된 새로운 컴포넌트가 된다.

스타일 컴포넌트에서 이미 작성된 스타일을 상속받아 새로운 스타일드 컴포넌트를 만들 때는 기존에 사용하던 styled[컴포넌트 이름] 이 아니라 styled(컴포넌트 이름)처럼 소괄호로 감싸야한다.



```
src > components > JS Button.js > [0] Button
1  import React from "react";
2  import styled from "styled-components/native";
3  import { StyleSheet, TouchableOpacity, Text } from "react-native";
4
5
6  const styles = StyleSheet.create({
7    container : {
8      backgroundColor : '#9b59b6',
9      borderRadius : 15,
10     paddingVertical : 15,
11     paddingHorizontal : 40,
12     marginVertical : 10,
13     justifyContent : 'center',
14   },
15   title :{
16     fontSize : 20,
17     fontWeight : '600',
18     color : '#fff',
19   },
20 });
21 const Button = props => {
22   return (
23     <TouchableOpacity style={styles.container}>
24       <Text style={styles.title}>{props.title}</Text>
25     </TouchableOpacity>
26   );
27 };
28
29 export default Button;
```

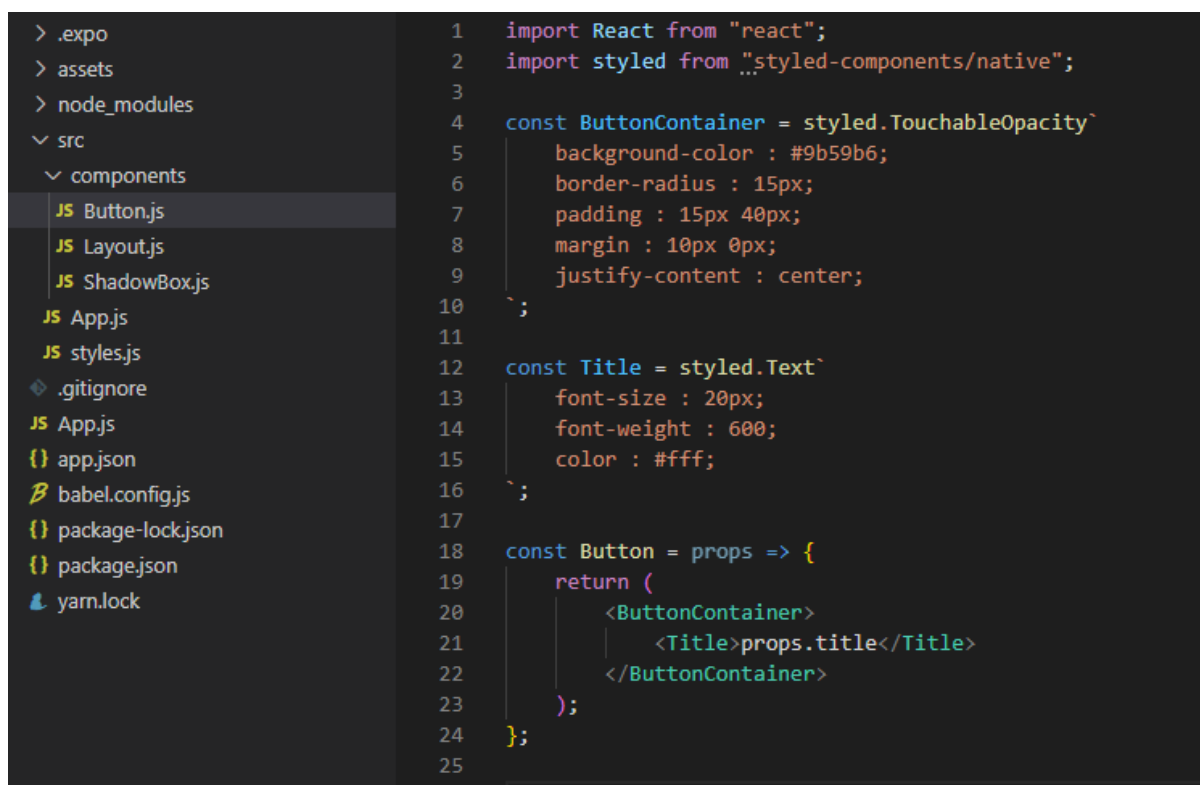
가장 먼저 보이는 차이는 카멜 표기법으로 표기되는 스타일 속성과 하이픈 형태의 스타일 속성이다.

스타일드 컴포넌트를 이용하면 PaddingVERTICAL이나 PaddingHorizontal처럼 익숙하지 않은 속성보다 조금 더 익숙한 이름과 익숙한 적용방법으로 값을 설정할 수 있다.

※ 리액트 네이티브에서 PaddingVertical은 paddingTop과 paddingBottom를 모두 설정하는 것과 같고 PaddingHorizontal은 paddingLeft과 paddingRight를 모두 설정하는 것과 같다. margin의 경우에도 같은 방식으로 동작한다.

스타일 적용하기

components폴더에 Button.js을 만들어서 스타일드 컴포넌트를 이용해 작성하자



```
1  import React from "react";
2  import styled from "styled-components/native";
3
4  const ButtonContainer = styled.TouchableOpacity`
5    background-color : #9b59b6;
6    border-radius : 15px;
7    padding : 15px 40px;
8    margin : 10px 0px;
9    justify-content : center;
10 `;
11
12 const Title = styled.Text`
13   font-size : 20px;
14   font-weight : 600;
15   color : #fff;
16 `;
17
18 const Button = props => {
19   return (
20     <ButtonContainer>
21       <Title>props.title</Title>
22     </ButtonContainer>
23   );
24 };
25
```

TouchableOpacity 컴포넌트에 스타일이 적용된 ButtonContainer 라는 이름의 컴포넌트를 만들고, Text 컴포넌트에 스타일이 적용된 Title컴포넌트를 만들었다.

마지막으로 스타일 컴포넌트로 만들어진 컴포넌트를 이용해서 Button컴포넌트를 만들었다.

스타일드 컴포넌트를 사용하면 이렇게 역할에 맞는 이름을 지정할 수 있다는 장점이 있다.

완성된 버튼 컴포넌트를 app컴포넌트에 적용하고, app컴포넌트에서 사용하는 view 컴포넌트도 스타일드 컴포넌트를 이용해 수정해보자.

```
✓ REACT-NATIVE-STYLE  [Icons]
  > .expo
  > assets
  > node_modules
  ✓ src
    ✓ components
      JS Button.js
      JS Layout.js
      JS ShadowBox.js
      JS App.js
      JS styles.js
    .gitignore
    JS App.js
    {} app.json
    {} babel.config.js
    {} package-lock.json
    {} package.json
    {} yarn.lock

src > JS App.js > ...
1  import React from "react";
2  import { View, Text } from 'react-native';
3  import { viewStyles, textStyles } from "../styles";
4  import { Header, Contents, Footer } from "../components/Layout";
5  import styled, { CSS } from 'styled-components/native';
6  import Button from "../components/Button";
7
8  const App = () => {
9    return(
10     <Container>
11       <Button title="Hongsi" />
12       <Button title="MinHo" />
13     </Container>
14   );
15 };
16 const Container = styled.View`
17   flex : 1;
18   background-color : #ffffff;
19   align-items : center;
20   justify-content : center;
21 `;
22 const StyledText = styled.Text`
23   color: #000;
24   font-size : 20px;
25   margin : 10px;
26   padding : 10px;
27 `;
28 const ErrorText = styled(StyledText)`
29   font-weight : 600;
30   color : red;
31 `;
32
33
34 export default App;
```

