

# day45-RN-props

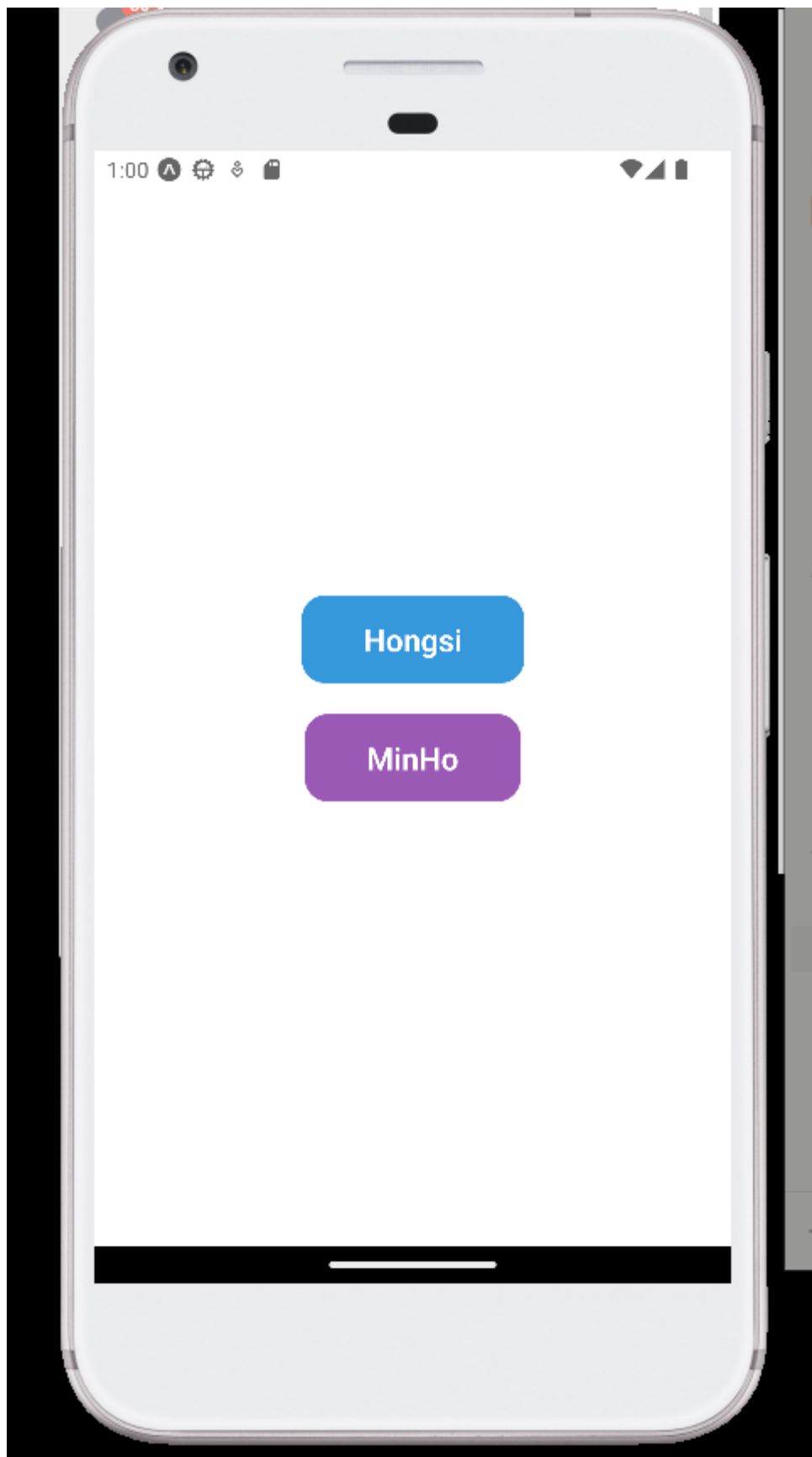
태그

날짜

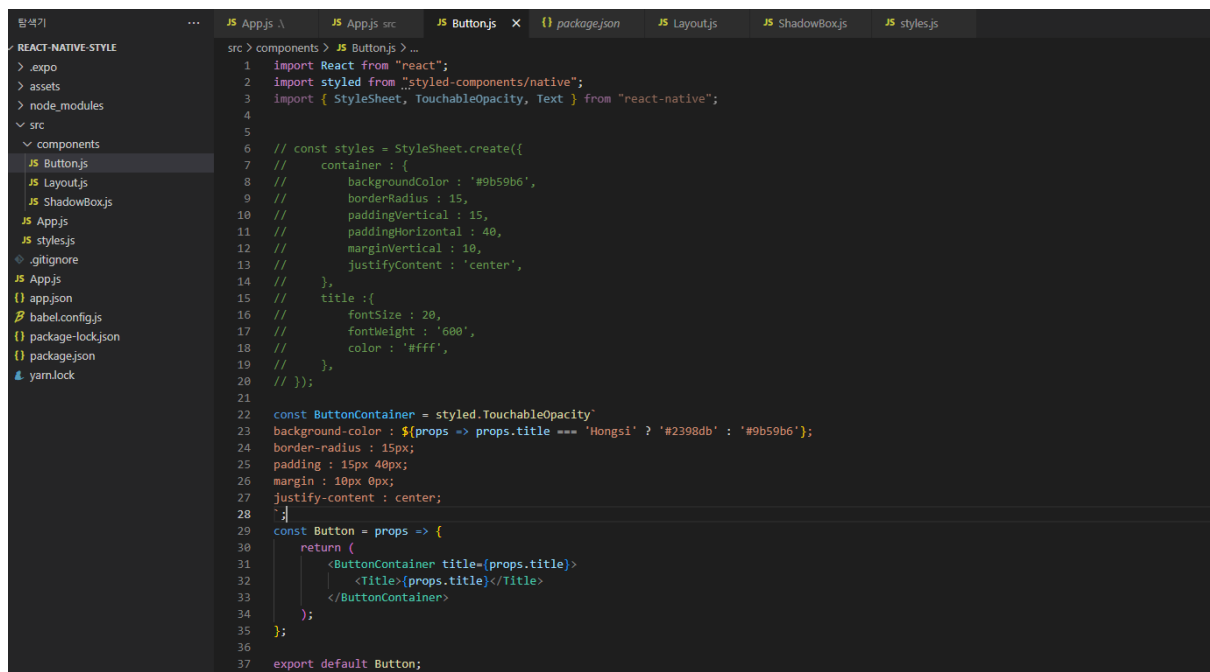
@2022년 12월 5일

앞에서 작성한 Button 컴포넌트에서 props로 전달되는 title의 값이 hongsi인 경우 바탕색을 다르게 표현하고 싶다면 어떻게 해야할까. 기존 방식으로 작성하면 스타일 시트 안에서 props에 접근할 수 있는 방법이 없으므로 다음과같이 작성한다.

```
src > components > JS Button.js > Button > backgroundColor
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: "bold",
18     color: "white",
19   },
20 });
21
22 const Button = props => {
23   return (
24     <TouchableOpacity style={([styles.container, { backgroundColor: props.title === 'Hongsi' ? '#3498db' : '#9b59b6'}, ])}
25       <Text style={styles.title}>{props.title}</Text>
26     </TouchableOpacity>
27   );
28 }
29
30 export default Button;
```



스타일드 컴포넌트에서는 스타일을 작성하는 백틱 안에서 props에 접근할 수 있다는 장점을 이용해 스타일을 작성하는 곳에서 조건에 따라 스타일을 변경할 수 있다.



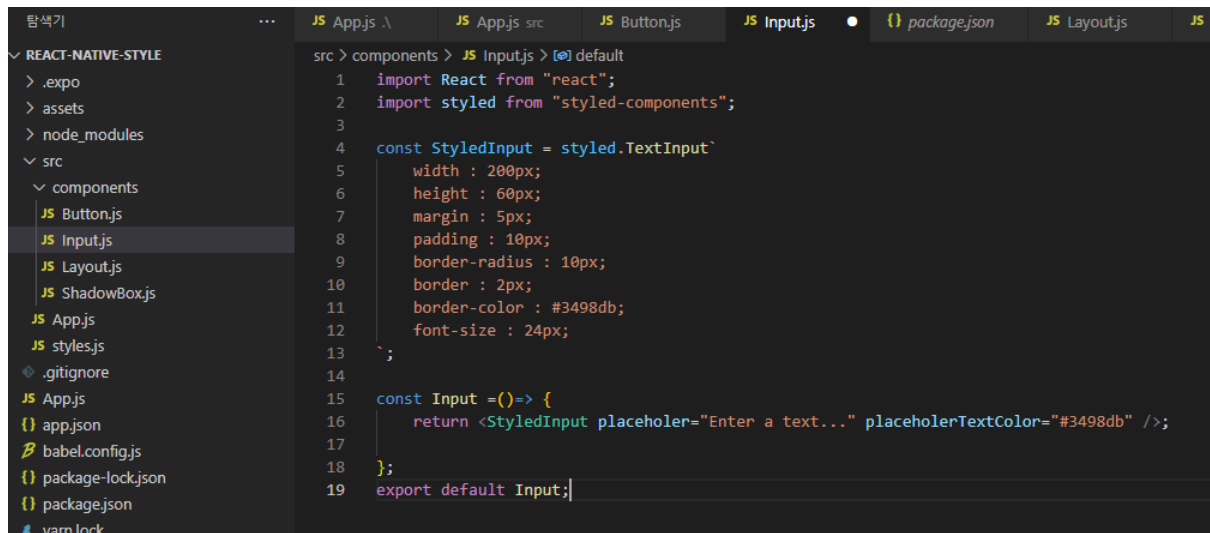
```
src > components > JS Button.js > ...
1  import React from "react";
2  import styled from "styled-components/native";
3  import { StyleSheet, TouchableOpacity, Text } from "react-native";
4
5
6  // const styles = StyleSheet.create({
7  //   container: {
8  //     backgroundColor: '#9b59b6',
9  //     borderRadius: 15,
10 //     paddingVertical: 15,
11 //     paddingHorizontal: 40,
12 //     marginVertical: 10,
13 //     justifyContent: 'center',
14 //   },
15 //   title: {
16 //     fontSize: 20,
17 //     fontWeight: 'bold',
18 //     color: 'fff',
19 //   },
20 // });
21
22 const ButtonContainer = styled.TouchableOpacity`
23   background-color: ${props => props.title === 'Hongsi' ? '#2398db' : '#9b59b6'};
24   border-radius: 15px;
25   padding: 15px 40px;
26   margin: 10px 0px;
27   justify-content: center;
28 `;
29
30 const Button = props => {
31   return (
32     <ButtonContainer title={props.title}>
33       <Title>{props.title}</Title>
34     </ButtonContainer>
35   );
36 };
37
38 export default Button;
```

ButtonContainer 컴포넌트에 props로 title을 전달하여 배경색을 설정하는 곳에서 title의 값에 따라 다른색으로 지정하도록 수정했다. 스타일드 컴포넌트를 이용하면 이 코드처럼 스타일이 작성되는 부분과 컴포넌트가 사용되는 부분을 구분하여 코드를 조금 더 깔끔하게 관리할 수 있다.

## attrs 사용하기

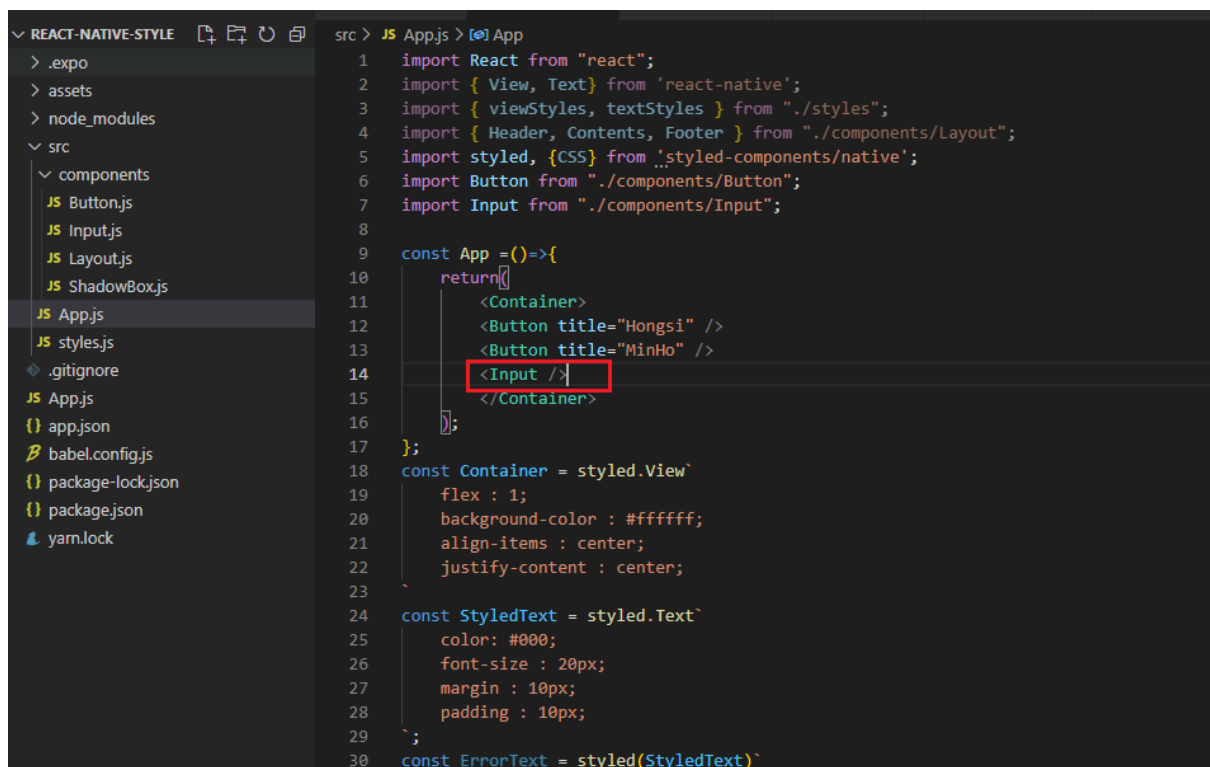
스타일드 컴포넌트를 이용하면 스타일을 작성하는 곳에서 컴포넌트 속성도 설정할 수 있다. 그리고 속성을 설정할 때도 전달된 props를 이용할 수 있으므로 props의 값에 따라 속성을 변경할 수 있다. 이번에는 스타일드 컴포넌트에서 속성을 설정할 때 사용하는 attrs의 사용법에 대해 알아보겠다.

components폴더안에 input.js 파일을 생성하고 아래 처럼 작성하자.

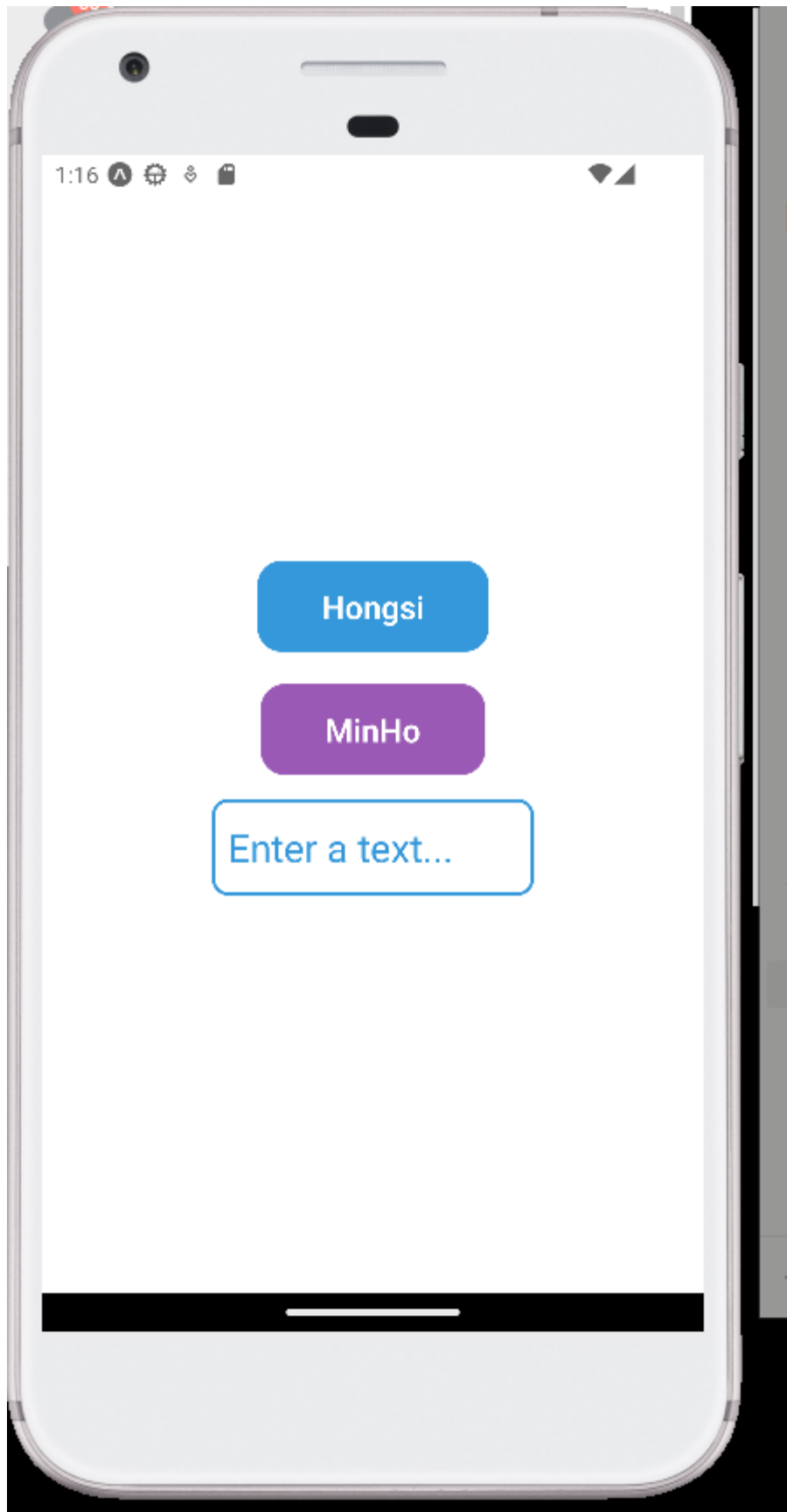


```
src > components > JS Input.js > [0] default
1  import React from "react";
2  import styled from "styled-components";
3
4  const StyledInput = styled.TextInput`
5      width : 200px;
6      height : 60px;
7      margin : 5px;
8      padding : 10px;
9      border-radius : 10px;
10     border : 2px;
11     border-color : #3498db;
12     font-size : 24px;
13 `;
14
15 const Input = () => {
16     return <StyledInput placeholder="Enter a text..." placeholderTextColor="#3498db" />;
17 };
18
19 export default Input;
```

TextInput 컴포넌트를 이용해서 StyledInput 컴포넌트를 만들고 placeholder와 placeholderTextColor 속성을 설정했다. input컴포넌트를 app.js에 적용해보자.



```
src > JS App.js > [0] 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 styled, { CSS } from "styled-components/native";
6  import Button from "../components/Button";
7  import Input from "../components/Input";
8
9  const App = () => {
10     return(
11         <Container>
12             <Button title="Hongsi" />
13             <Button title="MinHo" />
14             <Input />
15         </Container>
16     );
17 };
18
19 const Container = styled.View`
20     flex : 1;
21     background-color : #ffffff;
22     align-items : center;
23     justify-content : center;
24 `;
25
26 const StyledText = styled.Text`
27     color: #000;
28     font-size : 20px;
29     margin : 10px;
30     padding : 10px;
31 `;
32
33 const ErrorText = styled(StyledText)`
```



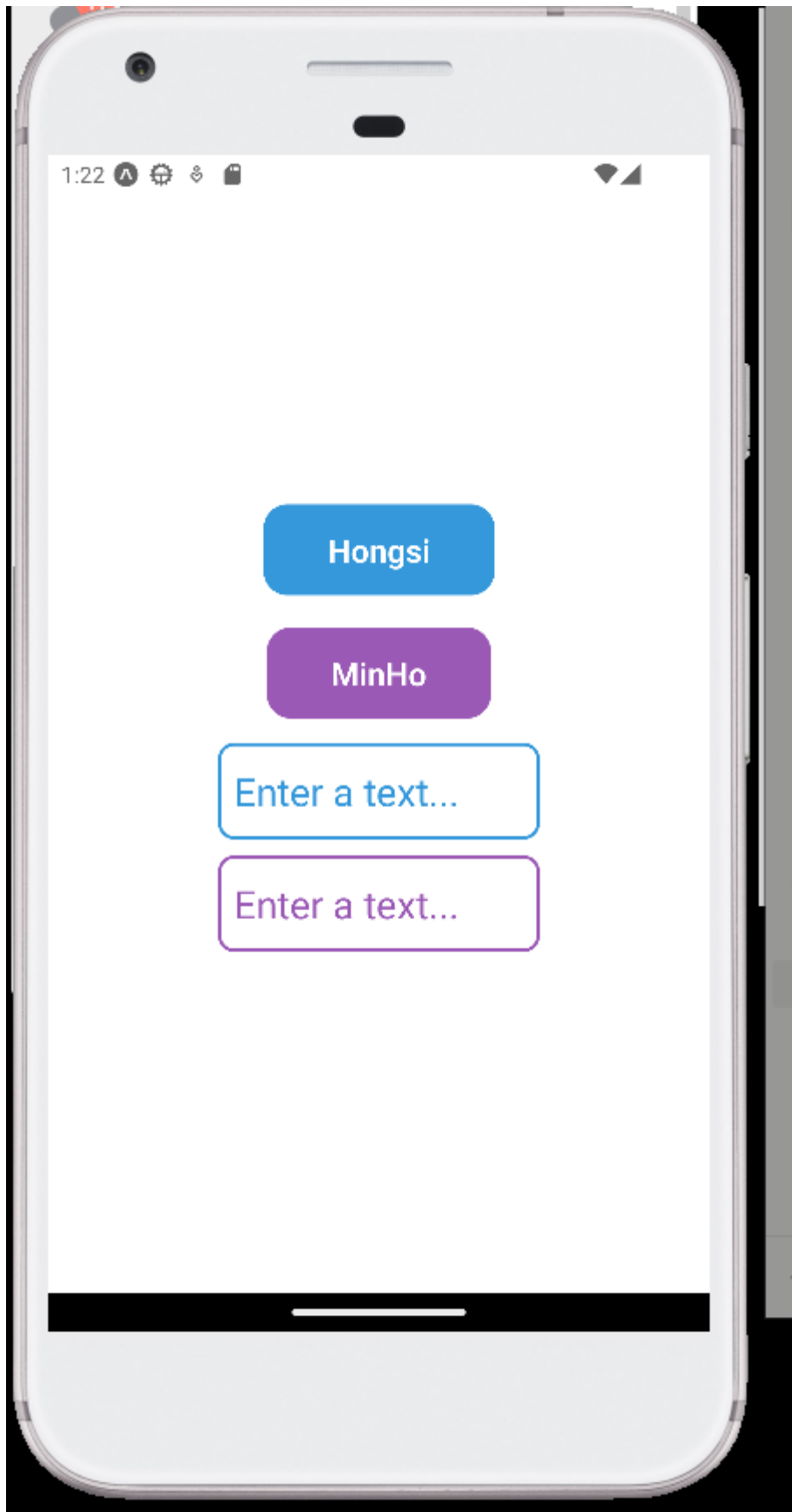
이번엔 스타일드 컴포넌트의 attrs를 이용해서 props로 전달된 borderColor값에 따라 input 컴포넌트의 디자인이 변경되도록 수정해보자

```
src > components > JS Input.js > [0] Input
1  import React from "react";
2  import styled from "styled-components";
3
4  const StyledInput = styled.TextInput.attrs(props => ({
5    placeholder : "Enter a text...",
6    placeholderTextColor : props.borderColor,
7  })))`
8    width : 200px;
9    height : 60px;
10   margin : 5px;
11   padding : 10px;
12   border-radius : 10px;
13   border : 2px;
14   border-color : #3498db;
15   font-size : 24px;
16   border-color : ${props => props.borderColor};
17   font-size : 24px;
18 `;
19
20 const Input = props => {
21   return <StyledInput borderColor={props.borderColor}/>;
22 };
23
24 export default Input;
```

```
src > JS App.js > [0] 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 styled, { CSS } from "styled-components/native";
6  import Button from "../components/Button";
7  import Input from "../components/Input";
8
9  const App = () => {
10   return (
11     <Container>
12       <Button title="Hongsi" />
13       <Button title="MinHo" />
14       <Input borderColor="#3498db" />
15       <Input borderColor="#9b59b6" />
16     </Container>
17   );
18 };
19
20 const Container = styled.View`
21   flex : 1;
22   background-color : #ffffff;
23   align-items : center;
24   justify-content : center;
25 `;
26
27 const StyledText = styled.Text`
28   color : #000;
29   font-size : 20px;
30   margin : 10px;
```

attrs를 이용하면 스타일을 설정하는 곳에서 props의 값에 따라 컴포넌트의 속성을 다 다르게 적용할 수도 있고 항상 일정한 속성을 미리 정의해놓을 수 있다.

하지만 `attrs`를 이용하여 속성을 설정하는 것이 항상 좋은 것만은 아니다. 컴포넌트가 어떻게 사용되는가에 따라 `attrs`를 이용하지 않고 컴포넌트에 직접 속성을 전달하는 것이 코드를 이해하는 데 더 도움이 되는 경우도 있다.



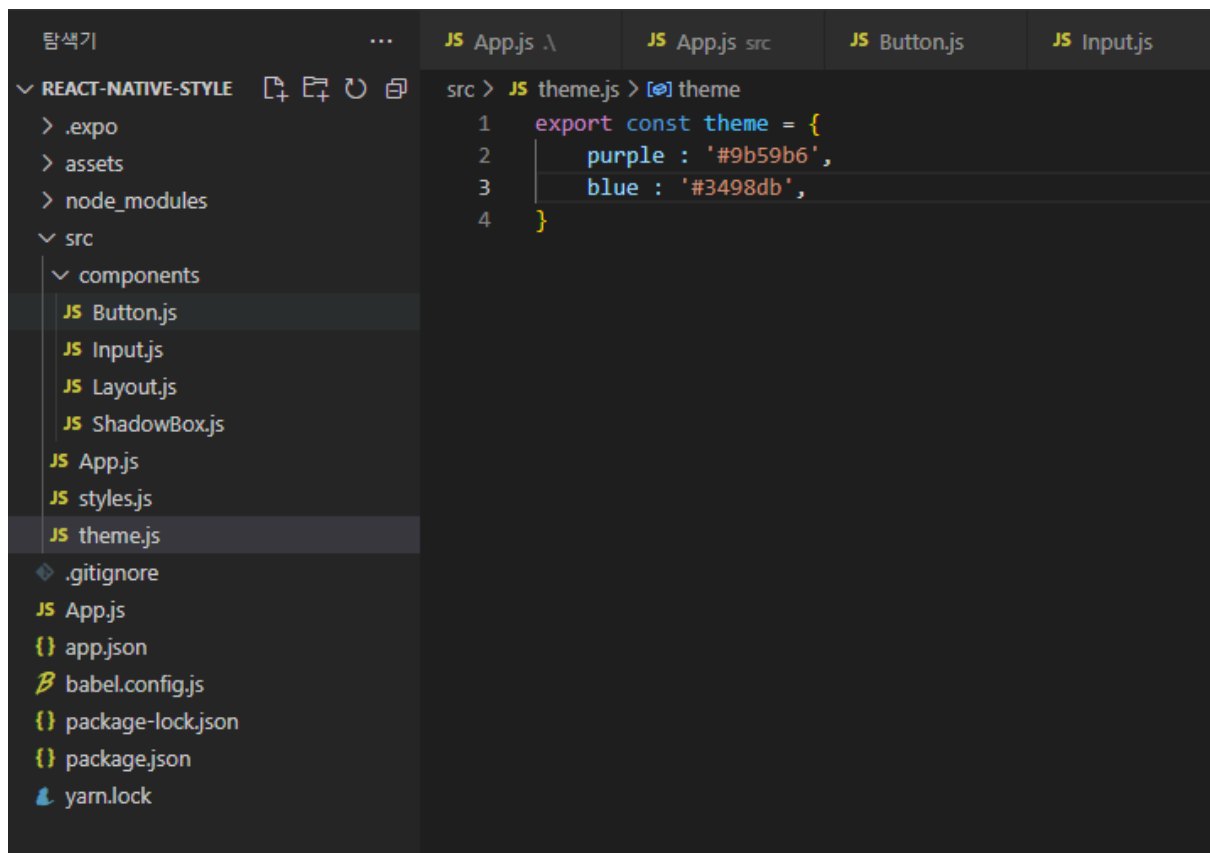


themeProvider

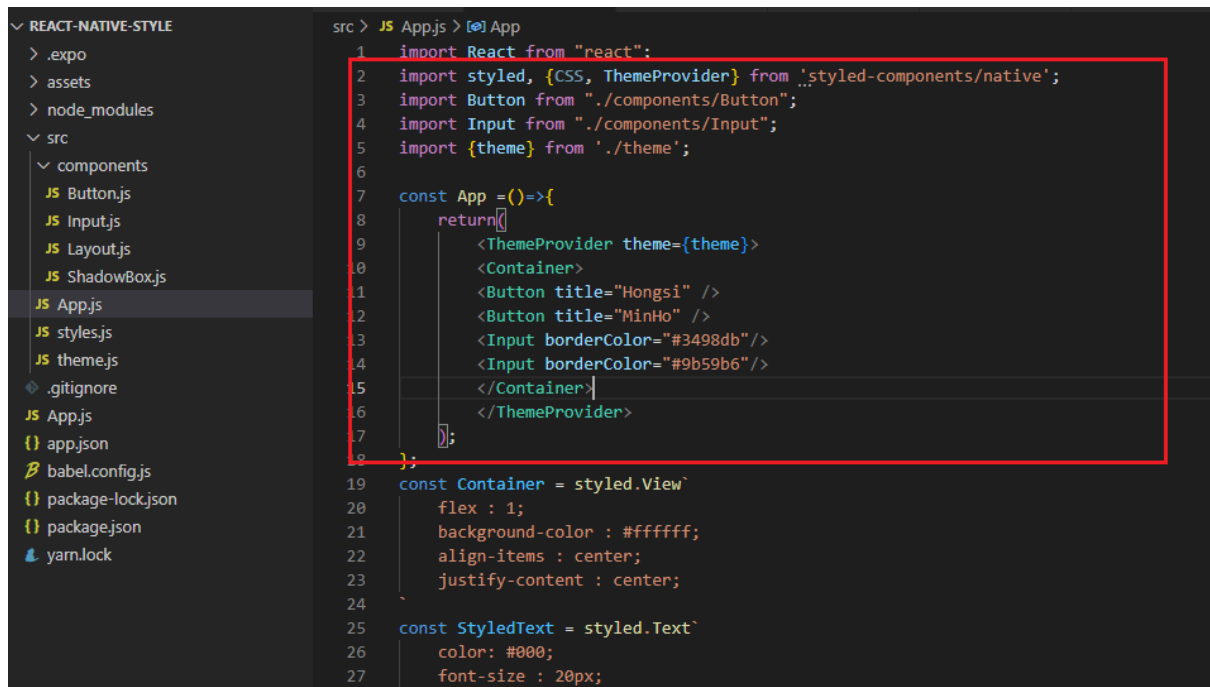
스타일드 컴포넌트의 themeProvider이다.

context API를 활용해 애플리케이션 전체에서 스타일드 컴포넌트를 이용할 때 미리 정의한 값들을 사용할 수 있도록 props로 전달한다. themeprovider를 이용하여 스타일을 정의할 때 미리 정의한 색을 사용하는 방법에 대해 알아보자.

src폴더 안에 theme.js파일을 생성하고 button 컴포넌트에서 사용했던 색을 정의하자.

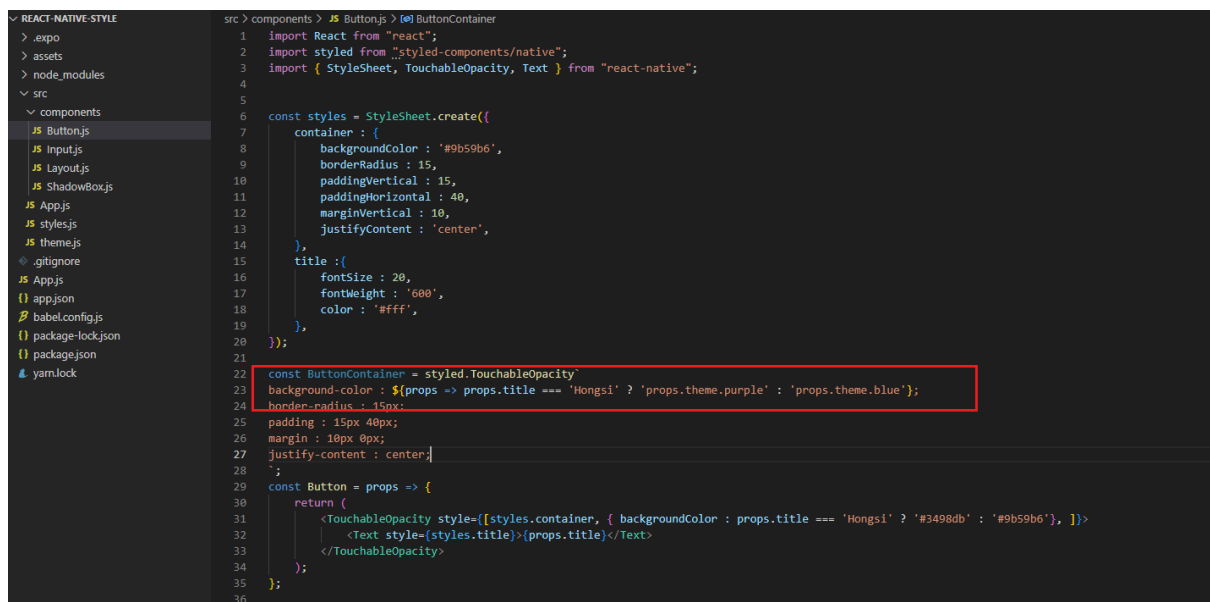


이제 모든 컴포넌트를 감싸는 최상위 컴포넌트로 themeprovider 컴포넌트를 이용하며, 앞에서 정의한 색을 themeprovider 컴포넌트의 theme속성에 설정하자. 그러면 themeprovider의 자식 컴포넌트에서는 스타일드 컴포넌트를 이용할 때 props로 theme를 전달받아 미리 정의된 색을 이용할 수 있다.



```
src > JS Appjs > [0] App
1 import React from "react";
2 import styled, {CSS, ThemeProvider} from 'styled-components/native';
3 import Button from "../components/Button";
4 import Input from "../components/Input";
5 import {theme} from '../theme';
6
7 const App = ()=>{
8   return(
9     <ThemeProvider theme={theme}>
10       <Container>
11         <Button title="Hongsi" />
12         <Button title="MinHo" />
13         <Input borderColor="#3498db"/>
14         <Input borderColor="#9b59b6"/>
15       </Container>
16     </ThemeProvider>
17   );
18 }
19
20 const Container = styled.View`
21   flex : 1;
22   background-color : #ffffff;
23   align-items : center;
24   justify-content : center;
25 `
26
27 const StyledText = styled.Text`
28   color: #000;
29   font-size : 20px;
```

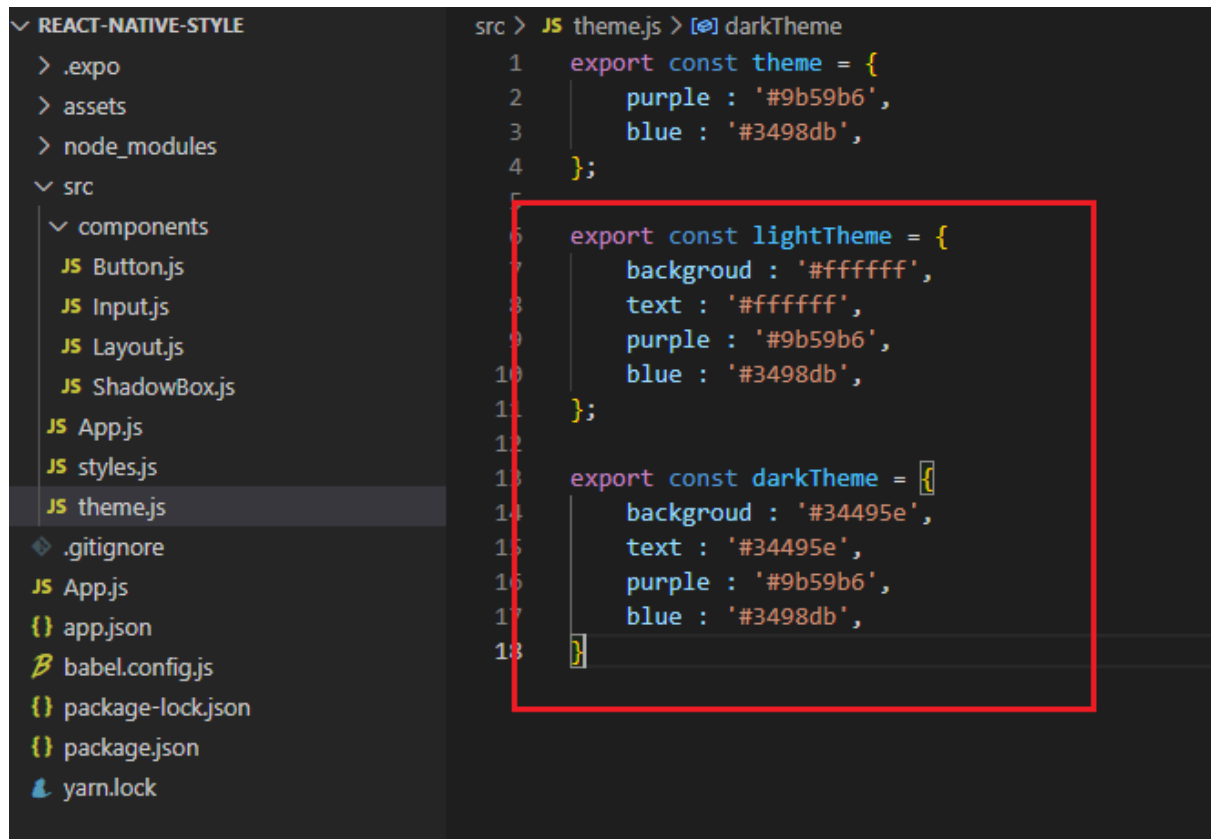
이제 button 컴포넌트에서 스타일을 정의할 때 props로 전달되는 theme을 이용하도록 수정해보자.



```
src > components > JS Buttonjs > [0] ButtonContainer
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 : 'bold',
18    color : 'fff',
19  },
20 });
21
22 const ButtonContainer = styled.TouchableOpacity`
23   background-color : ${props => props.title === 'Hongsi' ? 'props.theme.purple' : 'props.theme.blue'};
24   border-radius : 15px;
25   padding : 15px 40px;
26   margin : 10px 0px;
27   justify-content : center;
28 `;
29
30 const Button = props => {
31   return (
32     <TouchableOpacity style={styles.container, { backgroundColor : props.title === 'Hongsi' ? '#3498db' : '#9b59b6' }}>
33       <Text style={styles.title}>{props.title}</Text>
34     </TouchableOpacity>
35   );
36 }
```

스타일드 컴포넌트를 사용할 때 themeprovider를 활용하여 theme를 지정하면, 하나의 파일에서 미리 정의해둔 색을 하위 컴포넌트에서 사용할 수 있다. 하나의 파일에서 모든색을 관리하면 색의 사용이나 변경 등 유지보수에서 많은 이점을 얻을 수 있다.

두개의 색을 정의해두고 사용자의 선택에 따라 theme에 알맞은 값을 설정하는 것만으로 애플리케이션의 색 테마를 쉽게 변경할 수 있다.



```
src > JS theme.js > [⌘] darkTheme
1  export const theme = {
2    purple : '#9b59b6',
3    blue : '#3498db',
4  };
5
6  export const lightTheme = {
7    backgroud : '#ffffff',
8    text : '#ffffff',
9    purple : '#9b59b6',
10   blue : '#3498db',
11 };
12
13 export const darkTheme = {
14   backgroud : '#34495e',
15   text : '#34495e',
16   purple : '#9b59b6',
17   blue : '#3498db',
18 }
```

theme.js파일에 테마에 따라 사용할 색을 정의했다. 이제 app컴포넌트에서 테마를 변경할 수 있는 스위치를 추가하고, 테마에 따라 다른색이 되도록 수정해보자.

```
src > JS App.js > App
1 import React, {useState} from "react";
2 import {Switch} from 'react-native';
3 import styled, {CSS, ThemeProvider} from 'styled-components/native';
4 import Button from "../components/Button";
5 import Input from "../components/Input";
6 import {lightTheme, darkTheme} from '../theme';
7
8 const App = () => {
9   const [isDark, setIsDark] = useState(false);
10   const [_toggleSwitch] = () => setIsDark(!isDark);
11   return (
12     <ThemeProvider theme={isDark ? darkTheme : lightTheme}>
13       <Container>
14         <Switch value={isDark} onChange={_toggleSwitch} />
15         <Button title="Hongs-i" />
16         <Button title="MinHo" />
17         <Input borderColor="#3498db" />
18         <Input borderColor="#9b59b6" />
19       </Container>
20     </ThemeProvider>
21   );
22 };
23
24 const Container = styled.View`
25   flex : 1;
26   background-color : ${props => props.theme.background};
27   align-items : center;
28   justify-content : center;
29 `;
30
31 const StyledText = styled.Text`
```

useState를 이용해 테마의 상태를 관리할 isDark와 상태를 변경할 setIsDark 함수를 만들었다.

리액트 네이티브의 switch컴포넌트를 활용해 isDark와 상태를 변경할 수 있도록 화면을 구성했다.

상태에 따라 테마가 적용되도록 수정되었고 app컴포넌트에서 사용되는 container컴포넌트의 스타일을 정의하는 곳에서 props로 전달될 theme을 이용하여 배경색을 설정했다.

