3장

component

- 재사용할 수 있는 조립 블록으로 화면에 나타나는 UI 요소
- 컴포넌트란 재사용이 가능한 조립 블록으로 화면에 나타나는 UI 요소입니다.
- 컴포넌트는 단순히 UI 역할만 하는 것이 아니라 부모로부터 받은 속성(props)이나 자신의 상태(state)에 따라 표현이 달라지고 다양한 기능을 수행한다.
- 리액트 네이티브는 데이터와 UI 요소의 집합체라고 할 수 있는 컴포넌틀를 이용하여 화면을 구성하게 된다.

```
App.js
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Open up App.js to start working on your app!</Text>
      <StatusBar style="auto"/>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

→ JSX는 객체 생성과 함수 호출을 위한 문법적 편의를 제공하기 위해 만들어진 확장 기능으로 리액트 프로젝트에서 사용됩니다. JSX는 가독성이 높고 작성하기도 쉬울 뿐만 아니라, XML과 유사하다는 점에서 중첩된 구조를 잘 나타낼 수 있다는 장점이 있다.

null과 undefined

- 조건에 따라 출력하는 값을 변경하다 보면 컴포넌트가 null이나 undefined를 반환하는 경우가 있습니다. JSX의 경우 null은 허용하지만 undefined는 오류가 발생하는데 주의해야 합니다.

props와 state

- props와 state는 컴포넌트가 UI뿐만 아니라 다양한 기능을 담당할 수 있도록 하여 더욱 다양한 역할을 수행할 수 있도록 해줍니다.
- props란 properties를 줄인 표현으로, 부모 컴포넌트가 자식 컴포넌트의 props를 설정하면 자식 컴포넌트에서 해당 prop를 사용할 수 있지만 변경하는 것은 불가능합니다. prop의 변경이 필요한 경우 prop를 설정 및 전달한 부모 컴포넌트에서 변경해야 한다.
- propTypes
    - 프로젝트의 크기가 커지면서 컴포넌트에 prop를 전달할 때 잘못된 타입을 전달하거나, 필수로 전달해야 하는 값을 전달하지 않아서 문제가 생길 수 있습니다. 혹은 협업하는 다른 개발자가 잘못 전달할 수도 있습니다. 이런 상황에서 잘못된 props가 전달되었다는 것을 경고 메시지를 통해 알리는 방법으로 PropTypes를 사용하는 방법이 있습니다.
- props는 부모 컴포넌트에서 받은 값으로 변경할 수 없는 반면, state는 컴포넌트 내부에서 생성되고 값을 변경할 수 있으며 이를 이용해 컴포넌트 상태를 관리합니다. 상태란 컴포넌트에서 변화할 수 있는 값을 나타내며, 상태가 변하면 컴포넌트는 리렌더링 됩니다.

이벤트

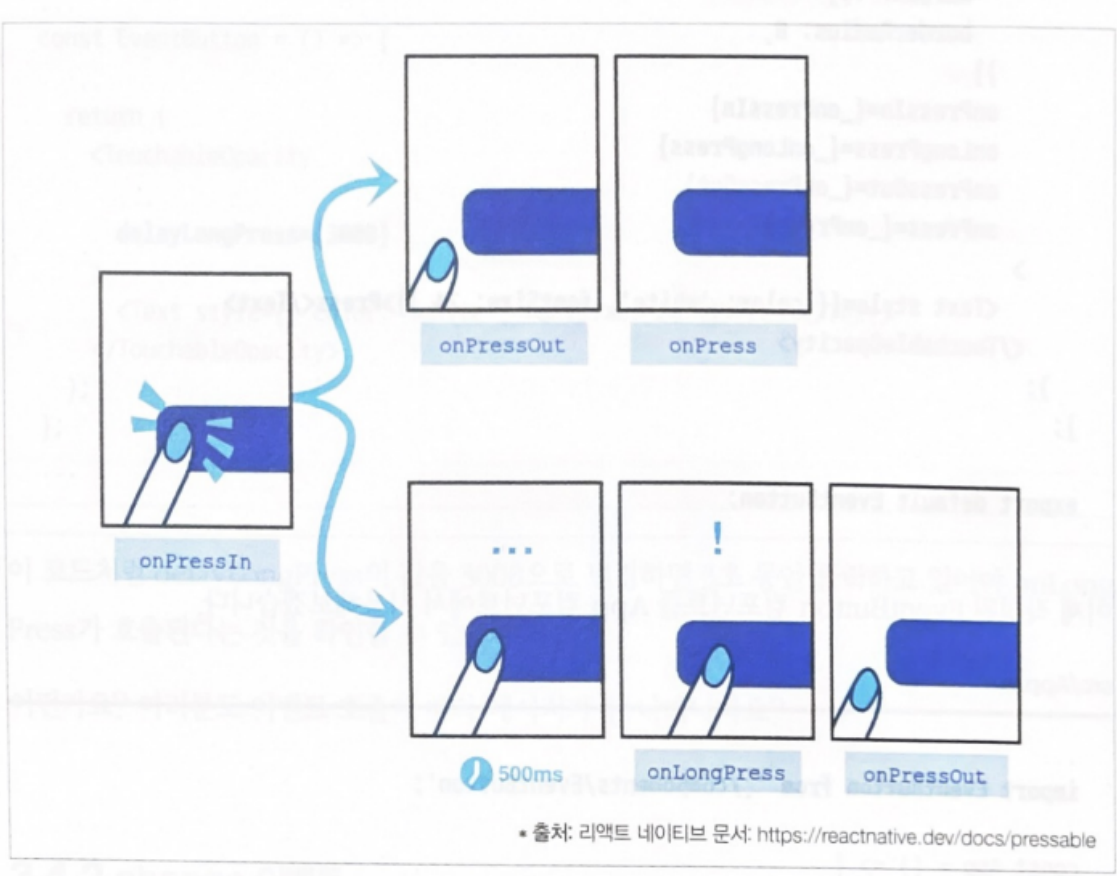

그림 3-14 Press 이벤트

```
src/components/EventButton.js

import React from 'react';
import { TouchableOpactity, Text } from 'react-native';

const EventButton = () => {
  const _onPressIn = () => console.log('Press In !!!\n');
  const _onPressOut = () => console.log('Press Out !!!\n');
  const _onPress = () => console.log('Press !!!\n');
  const _onLongPress = () => console.log('Long Press !!!\n');

  return (
    <TouchableOpacity
      style={{
        backgroundColor: '#f1c40f',
        padding: 16,
        margin: 10,
        borderRadius: 8,
      }}
      onPressIn={_onPressIn}
      onLongPress={_onLongPress}
      onPressOut={_onPressOut}
      onPress={_onPress}
    >
      <Text style={{ color: 'white', fontSize: 24 }}>Press</Text>
    </TouchableOpacity>
```

```
  );
};

export defualt EventButton;
```

```
src/App.js
...
import EventButton from './components/EventButton';

const App = () => {
  return (
    <View
      ...
    >
      <EventButton />
    </View>
  );
};
...
```