Explanations of Basic Concepts

React Native Principles

Basic scaffold of a simple component

```
import React, { Component } from 'react';
import { View, Text, StyleSheet } from 'react-native';

export default class Example extends Component {
  render() {
    return (
      <View style={styles.container}>
        <Text>Hello World</Text>
      </View>
    )
  }
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
    },
})
```

1. The first two lines are importing React libraries and React Native core components.
2. `export default` is exposing the class, so that other files -- **parent** components can call and render it.
3. `class Example extends Component` is creating a class and inheriting from `React.Component` class.
4. `render()` function is mandatory for a React component, what **returns** from it will be displayed and should be written using `JSX` syntax which is a `HTML` -like syntax.
5. `View` or `Text` are built-in components of React Native as basic building blocks, as they accepts optional properties like `style` like HTML does. (Note that text strings can only be rendered inside `<Text>` tags)
6. `StyleSheet` allows writing CSS to style the components, but needs to be written as JavaScript object.

State & Props and Callback

State

1. `state` is a property encapsulated to and maintained by a component itself. Defined as a class variable `state = { key: value, ... }`, and can be accessed through `this.state.key`.

2. However, it **cannot** be changed by directly reassigning it like `this.state.key = newValue` as the change will not be reflected. Instead, `this.setState({ key: newValue })` should be called that can tell React to rerender (call `render()` again) the component to reflect the change. (So `setState()` cannot be inside `render()` function.)

```
export default class Example extends Component {
    state = {
        greeting: '',
    }

    componentDidMount() {
        this.setState({ greeting: 'Hello World' });
    }

    render() {
        return (
            <View>
                <Text>{this.state.greeting}</Text>
            </View>
        )
    }
}
```

Props

1. Different from `state`, `props` is passed down from **parent** components, and can be accessed using `this.props.xxx` from child components.
2. Change of `props` will cause components accepting it to rerender.

```
export default class Parent extends Component {
    render() {
        return (
            <View>
                <Child greeting='Hi there' />
            </View>
        )
    }
}

export default class Child extends Component {
    render() {
        return (
            <View>
                <Text>{this.props.greeting}</Text>
            </View>
        )
    }
}
```

Callback Function

1. For high reusability, some components should stay stateless and decoupled. It will display what is given in `props` and make change to parent component through callback function.

```
// TouchableOpacity is button in React Native
import { View, Text, TouchableOpacity } from 'react-native';

export default class Parent extends Component {
    state = {
        greeting: 'hi there'
    }

    // Arrow function will bind `this` automatically
    _handlePress = () => {
        this.setState({ greeting: 'good morning' })
    }

    render() {
        return (
            <View>
                <Text>{this.state.greeting}</Text>
                <Button onPress={this._handlePress} text='Change Text' />
            </View>
        )
    }
}

export default class Button extends Component {
    render() {
        return (
            <TouchableOpacity onPress={this.props.onPress}>
                <Text>{this.props.text}</Text>
            </TouchableOpacity>
        )
    }
}
```