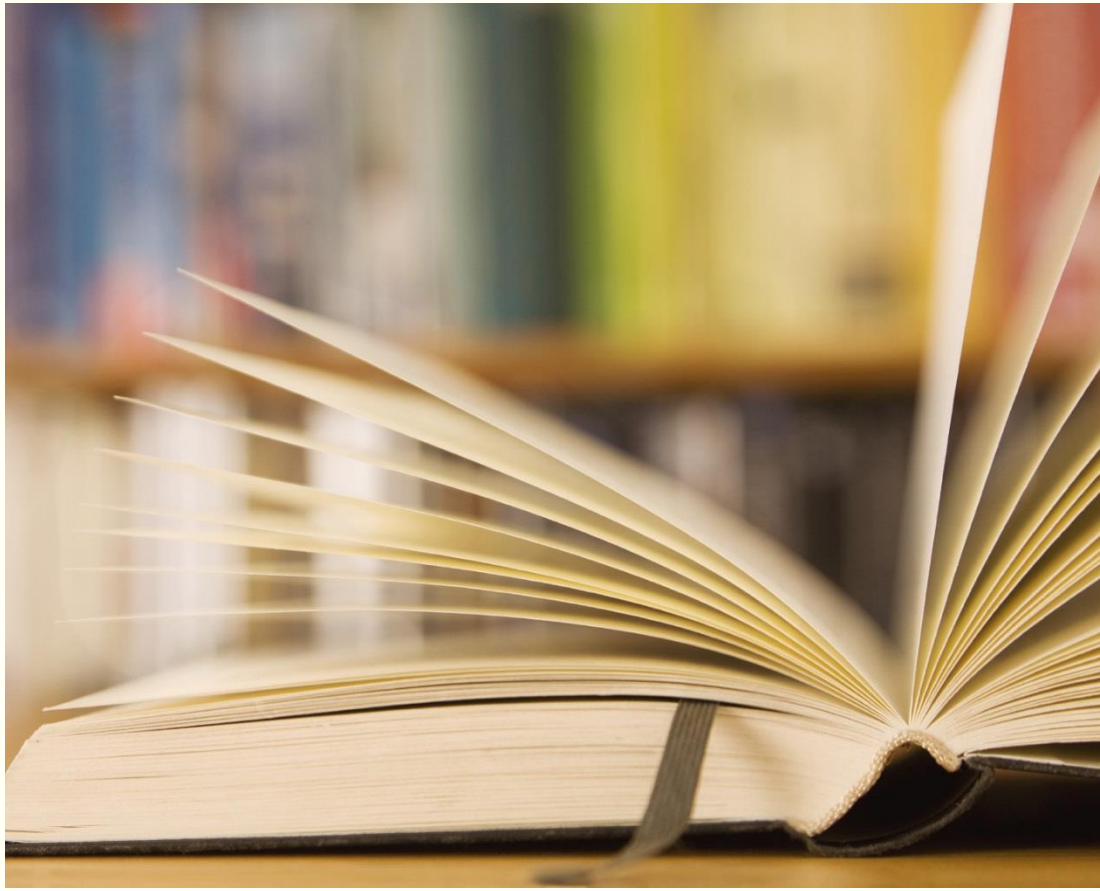
REACT FUNDAMENTALS

Aj.Amonpan Chomklin
Information Technology 1/2563



Ref:https://reactnative.dev/docs/intro-react

React Native คืออะไร

- React Native เป็น JavaScript Framework ตัวนึงที่พัฒนาโดย Facebook ซึ่งจะช่วยให้เราสามารถ เขียน Mobile Application แบบ Cross platform ได้ ช่วยทำให้เราเขียนโค้ดเพียงครั้งเดียว ก็สามารถสร้าง Application ทั้งของ Android OS และ IOS

Native App

- Native App คือ การพัฒนาแอพลิเคชั่นที่ใช้รูปแบบการพัฒนาและชุดคำสั่งต่าง ๆ ตามที่ผู้พัฒนาอุปกรณ์ได้จัดทำขึ้น เช่น
  - iOS สำหรับ iPhone, iPad, Apple Watch จะใช้ภาษา Object C หรือ Swift โดยการพัฒนาจะต้องใช้โปรแกรม XCode
  - Android จะใช้ภาษา Java และใช้โปรแกรม Android Studio ในการพัฒนา
  - Window Phone ใช้ภาษา C# และใช้โปรแกรม Visual Studio ในการพัฒนา

จะเขียน React Native ต้องรู้อะไรบ้าง?

1. เขียน Android และ IOS ได้บ้าง
2. CSS , HTML
3. JavaScript — JSX , ES6
4. Java
5. Objective C

React Fundamentals

- React Native runs on React, a popular open source library for building user interfaces with JavaScript. To make the most of React Native, it helps to understand React itself. This section can get you started or can serve as a refresher course.

- We’re going to cover the core concepts behind React:
  - **components**
  - JSX
  - props
  - state

Overview

- What Is React?
  - React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.
  - React has a few different kinds of components, but we’ll start with React.Component subclasses:

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

Your first component

- The rest of this introduction to React uses cats in its examples: friendly, approachable creatures that need names and a cafe to work in. Here is your very first Cat component:

```
1. import React, { Component } from 'react';
2. import { Text } from 'react-native';
3. export default class Cat extends Component {
4.   render() {
5.     return (
6.       <Text>Hello, I am your cat!</Text>
7.     );
8.   }
9. }
```

Your first component

- You additionally import Component from React:

```
import React, { Component } from 'react';
```

- Your component starts as a class extending Component instead of as a function:

```
class Cat extends Component {}
```

- Class components have a render() function. Whatever is returned inside it is rendered as a React element:

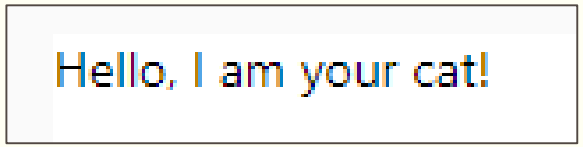
```
class Cat extends Component {
  render() {
    return <Text>Hello, I am your cat!</Text>;
  }
}
```

Your first component

- And as with function components, you can export your class component:

```
export default class Cat extends Component {
  render() {
    return <Text>Hello, I am your cat!</Text>;
  }
}
```

- Now take a closer look at that return statement. <Text>Hello, I am your cat!</Text> is using a kind of JavaScript syntax that makes writing elements convenient: JSX.
- Output:

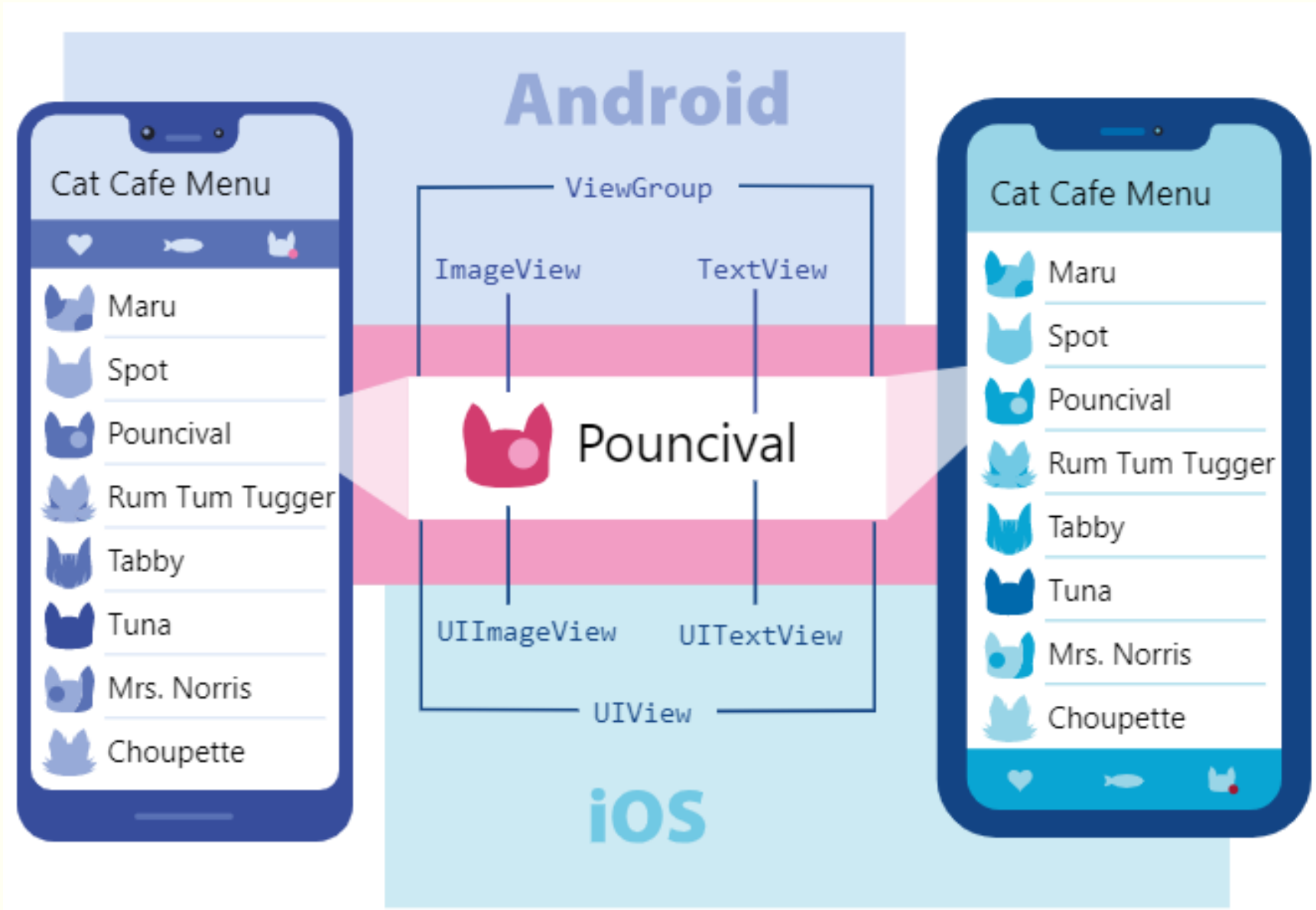


Core Components and Native Components

- React Native is an open source framework for building Android and iOS applications using React and the app platform’s native capabilities.
- With React Native, you use JavaScript to access your platform’s APIs as well as to describe the appearance and behavior of your UI using React components: bundles of reusable, nestable code.
- You can learn more about React in the next section. But first, let’s cover how components work in React Native

Views and mobile development

- In Android and iOS development, a view is the basic building block of UI: a small rectangular element on the screen which can be used to display text, images, or respond to user input.
- Even the smallest visual elements of an app, like a line of text or a button, are kinds of views. Some kinds of views can contain other views.
- It’s views all the way down!



Just a sampling of the many views used in Android and iOS apps.

Core Components

- React Native has many Core Components for everything from form controls to activity indicators. You can find them all documented in the API section. You will mostly work with the following Core Components:

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
`<View>`	`<ViewGroup>`	`<UIView>`	A non-scrollling `<div>`	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
`<Text>`	`<TextView>`	`<UITextView>`	`<p>`	Displays, styles, and nests strings of text and even handles touch events
`<Image>`	`<ImageView>`	`<UIImageView>`	`<img>`	Displays different types of images
`<ScrollView>`	`<ScrollView>`	`<UIScrollView>`	`<div>`	A generic scrolling container that can contain multiple components and views
`<TextInput>`	`<EditText>`	`<UITextField>`	`<input type="text">`	Allows the user to enter text

Custom Components

- You’ve already met React Native’s Core Components. React lets you nest these components inside each other to create new components. These nestable, reusable components are at the heart of the React paradigm.
- For example, you can nest Text and TextInput inside a View below, and React Native will render them together:

ตัวอย่างที่ 1

- View, TextInput, Button Component

React Fundamentals

- React Native runs on React, a popular open source library for building user interfaces with JavaScript. To make the most of React Native, it helps to understand React itself. This section can get you started or can serve as a refresher course.

- We’re going to cover the core concepts behind React:
  - components
  - **JSX**
  - props
  - state

What is JSX

- JSX ย่อมาจาก (JavaScript Syntax eXtension) ซึ่งหมายถึงส่วนเสริมของ Javascript ถูกพัฒนาขึ้นมาใช้กับ React โดยเฉพาะ โดย Syntax นั้นมีความคล้ายคลึงกับ HTML เป็นอย่างมาก

- รูปแบบของ JSX นั้นเหมือนกับ HTML เป็นอย่างมาก แต่ที่เห็นได้ชัดคือจะเห็นการเขียนโค้ดในลักษณะนี้ในอยู่ในไฟล์ Javascript แทนที่จะเป็นไฟล์ HTML

JSX Elements

- JSX นั้นถือว่าเป็น JavaScript Expression กล่าวคือทุกๆที่ ที่สามารถเขียน Expression ได้ ก็สามารถเขียน JSX ได้เช่นกัน หมายความว่า เราสามารถบันทึก JSX element ไว้ในตัวแปรต่างๆ,ส่งผ่านไปยังฟังก์ชัน,เก็บไว้ในออบเจกต์ หรืออาร์เรย์ได้ เพียงแค่เรากำหนดตัวแปรให้กับมัน

- ตัวอย่างการเก็บ JSX element ไว้ในตัวแปร

```
const navBar = <nav>I am a nav bar</nav>;
```

JSX

- React and React Native use JSX, a syntax that lets you write elements inside JavaScript like so: <Text>Hello, I am your cat!</Text>.
- Here you are declaring a name for the cat, name, and embedding it with curly braces inside <Text>.

```
1. import React from 'react';
2. import { Text } from 'react-native';
3. export default function Cat() {
4.   const name = "Maru";
5.   return (
6.     <Text>Hello, I am {name}!</Text>
7.   );
8. }
```

JSX is included in the React library, it won't work if you don't have import React from 'react' at the top of your file!

JSX Elements

- ตัวอย่างการเก็บ JSX elements ไว้ในออบเจกต์

```
const myTeam = {
  center: <li>Benzo Walli</li>,
  powerForward: <li>Rasha Loa</li>,
  smallForward: <li>Tayshaun Dasmoto</li>,
  shootingGuard: <li>Colmar Cumberbatch</li>,
  pointGuard: <li>Femi Billon</li>
};
```

Attributes In JSX

- ใน JSX เองสามารถมี Attribute ได้เหมือนกับ HTML อีกทั้งยังมีความคล้ายคลึงกันมาก โดยการประกาศ Attribute สามารถทำได้ดังนี้

```
my-attribute-name="my-attribute-value"
ชื่อ attributes เครื่องหมายเท่ากับ ตามด้วย "ค่า" หรือ 'ค่า'
```

- ตัวอย่างการใช้งาน JSX elements ร่วมกับ Attributes

```
const title = <h1 id="title">Introduction to React.js: Part I</h1>;
```

- ในหนึ่ง Element สามารถมีได้หลาย Attributes เหมือนกันกับ HTML

```
const panda = <img src="images/panda.jpg" alt="panda" width="500px"
height="500px" />;
```

แบบฝึกหัด

1. ประกาศค่าคงที่ (const) ชื่อ p1 ขึ้นมาสำหรับเก็บ JSX element และมีข้อความข้างในคือ foo
2. ในบรรทัดต่อมาให้ประกาศค่าคงที่เช่นเดิม แต่เปลี่ยนชื่อเป็น p2 และมีข้อความข้างในเป็น bar
3. และเพิ่ม Attributes id ขึ้นมาโดยกำหนดให้ p1 มี id=”learge” และ p2 มี id=”small ”

Nested JSX

- เราสามารถเขียน JSX element ซ้อนๆกันได้เหมือนใน HTML ดังตัวอย่างต่อไปนี้

```
<a href="https://www.example.com">
  <h1>Click me!</h1>
</a>
```

- ซึ่งถ้า JSX expression มีมากกว่าหนึ่งบรรทัด จำเป็นต้องใส่วงเล็บ () ครอบไว้ด้วย ดังตัวอย่าง

```
const theExample = (
  <a href="https://www.example.com">
    <h1> Click me! </h1>
  </a>
);
```

แบบฝึกหัด

1. ให้ประกาศตัวแปร myDiv ขึ้นมาและกำหนดให้ตัวแปรเก็บค่า JSX element ไว้
2. โดยใน tag มี tag <h1></h1> ซ้อนอยู่ข้างใน และให้ tag <h1> มีข้อความ Hello world

```
const myDiv = (
 <div>
   <h1>Hello world</h1>
 </div>
 )
```

JSX Outer Elements

- ในรูปแบบการเขียน JSX นั้นมีอีกกฎนึงที่ยังไม่ได้พูดถึง ให้ลองสังเกตุความแตกต่างของโค้ดทั้งสองอันนี้
- ตัวอย่างโค้ดที่สามารถทำงานได้

```
const paragraphs = (
   <div id="i-am-the-outermost-element">
      <p>I am a paragraph.</p>
      <p>I, too, am a paragraph.</p>
   </div>
);
```

- ตัวอย่างโค้ดที่ไม่สามารถทำงานได้

```
const paragraphs = (
      <p>I am a paragraph.</p>
      <p>I, too, am a paragraph.</p>
);
```

- ถ้าสังเกตุดีๆจะเห็นว่าโค้ดข้างล่างนี้ไม่มี tag ครอบอยู่ เหตุที่ต้องมีเพราะว่า JSX นั้นยอมให้มี root element ได้เพียงอันเดียวเท่านั้น ถ้ามี elements สองอันจะอยู่แยกกันไม่ได้ วิธีที่ง่ายที่สุดในการแก้ปัญหา คือครอบด้วย tag นั่นเอง

JSX In Depth

- Fundamentally, JSX just provides syntactic sugar for the
- **React.createElement(component, props, ...children)**
- function. The JSX code:

```jsx
<MyButton color="blue" shadowSize={2}>
  Click Me
</MyButton>
```

- compiles into:

```jsx
React.createElement(
  MyButton,
  {color: 'blue', shadowSize: 2},
  'Click Me'
)
```

JSX

- Any JavaScript expression will work between curly braces, including function calls like {getFullName("Rum", Tum", "Tugger")}:

```
1.  import React from 'react';
2.  import { Text } from 'react-native';
3.  export default function Cat() {
4.    function getFullName(firstName, secondName, thirdName) {
5.      return firstName + " " + secondName + " " + thirdName;
6.    }
7.    return (
8.      <Text>
9.        Hello, I am { getFullName("Rum", "Tum", "Tugger") } !
10.     </Text>
11.   );
12. }
```

Hello, I am Rum Tum Tugger!

Slide && ตัวอย่างที่ 2

- FlexBox & Layouts(deep drive)

React Fundamentals

- React Native runs on React, a popular open source library for building user interfaces with JavaScript. To make the most of React Native, it helps to understand React itself. This section can get you started or can serve as a refresher course.

- We’re going to cover the core concepts behind React:
  - components
  - JSX
  - **props**
  - state

Props

- Props is short for “properties.” Props let you customize React components. For example, here you pass each <Cat> a different name for Cat to render:
- Most of React Native’s Core Components can be customized with props, too. For example, when using Image, you pass it a prop named source to define what image it shows:
- Props คือ creation parameter ซึ่งสามารถถูกเรียกใช้งานได้ตอนที่ component นั้นๆกำลังจะถูกสร้างขึ้นมา สามารถอ้างถึงได้ตามรูปแบบการเรียกข้างล่าง
  - this.props.*myCreationParameter*

Props

- React Native มี component พื้นฐานที่ชื่อว่า Image ซึ่งมีตัวแปร props ชื่อ ‘source’ สำหรับกำหนดรูปที่จะแสดง

```
render() {
  let pic = {
    uri: 'https://url_to_image/test_image.jpg'
  };
  return (
    <Image source={pic} style={{width: 193, height: 110}}/>
  );
}
```

- Note — ใช้ braces { } ในการ embed ตัวแปรเข้าไปใน JSX และภายใต้ { } เราสามารถใส่ JavaScript expression เข้าไปได้

- นอกจากนี้เรายังสามารถใช้ props ใน component ที่สร้างขึ้นมาเองได้ ตัวอย่างเช่น

```
class Greeting extends Component {
  render() {
    return (
      <Text>Hello {this.props.name}!</Text>
    );
  }
}
```

```
1.  import React from 'react';
2.  import { Text, View } from 'react-native';
3.  function Cat(props) {
4.    return (
5.      <View>
6.        <Text>Hello, I am {props.name}!</Text>
7.      </View>
8.    );
9.  }
10. export default function Cafe() {
11.   return (
12.     <View>
13.       <Cat name="Maru" />
14.       <Cat name="Jellylorum" />
15.       <Cat name="Spot" />
16.     </View>
17.   );
18. }
```

```
Hello, I am Maru!
Hello, I am Jellylorum!
Hello, I am Spot!
```

React Fundamentals

- React Native runs on React, a popular open source library for building user interfaces with JavaScript. To make the most of React Native, it helps to understand React itself. This section can get you started or can serve as a refresher course.

- We’re going to cover the core concepts behind React:
  - components
  - JSX
  - props
  - **state**

State

- While you can think of props as arguments you use to configure how components render, state is like a component’s personal data storage.
- State is useful for handling data that changes over time or that comes from user interaction.
- State gives your components memory!
- As a general rule, use props to configure a component when it renders. Use state to keep track of any component data that you expect to change over time.

State

- โดยทั่วไป state จะถูก Initialise ใน constructor ยกตัวอย่างเช่น

```
constructor(props) {
    super(props);
    this.state = { myStateVariable: true };
}
```

- และเปลี่ยนค่าเมื่อต้องการ โดยใช้ฟังก์ชัน setState เช่น

```
this.setState({ myStateVariable: false })
```

```
1.  import React, { Component } from "react";
2.  import { Button, Text, View } from "react-native";
3.  export class Cat extends Component {
4.    state = { isHungry: true };
5.    render(props) {
6.      return (
7.        <View>
8.          <Text>
9.            I am {this.props.name}, and I am
10.           {this.state.isHungry ? " hungry" : " full"}!
11.         </Text>
12.         <Button
13.           onPress={() => {
14.             this.setState({ isHungry: false });
15.           }}
16.           disabled={!this.state.isHungry}
17.           title={
18.             this.state.isHungry ? "Pour me some milk, please!" : "Thank you!"
19.           }
20.         />
21.       </View>
22.     );
23.   }
24. }
```

```
25. export default class Cafe extends Component {
26.   render() {
27.     return (
28.       <>
29.         <Cat name="Munkustrap" />
30.         <Cat name="Spot" />
31.       </>
32.     );
33.   }
34. }
```

State

- As always with class components, you must import the Component class from React:

```
import React, { Component } from 'react';
```

- In class components, state is stored in a state object:

```
export class Cat extends Component {
  state = { isHungry: true };
  //..
}
```

State

- As with accessing props with this.props, you access this object inside your component with this.state:

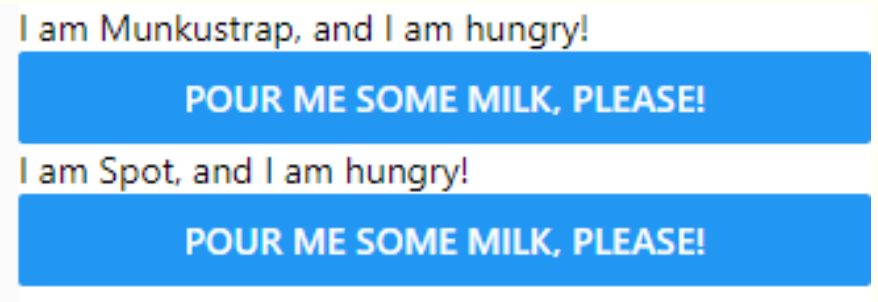
```
<Text>
  I am {this.props.name}, and I am
  {this.state.isHungry ? ' hungry' : ' full'}!
</Text>
```

- And you set individual values inside the state object by passing an object with the key value pair for state and its new value to this.setState():

```
<Button
  onPress={() => {
    this.setState({ isHungry: false });
  }}
  // ..
</Button>
```

```
1.  import React, { Component } from "react";
2.  import { Button, Text, View } from "react-native";
3.  export class Cat extends Component {
4.    state = { isHungry: true };
5.    render(props) {
6.      return (
7.        <View>
8.          <Text>
9.            I am {this.props.name}, and I am
10.           {this.state.isHungry ? " hungry" : " full"}!
11.         </Text>
12.         <Button
13.           onPress={() => {
14.             this.setState({ isHungry: false });
15.           }}
16.           disabled={!this.state.isHungry}
17.           title={
18.             this.state.isHungry ? "Pour me some milk, please!" : "Thank you!"
19.           }
20.         />
21.       </View>
22.     );
23.   }
24. }
```

```
25. export default class Cafe extends Component {
26.   render() {
27.     return (
28.       <>
29.         <Cat name="Munkustrap" />
30.         <Cat name="Spot" />
31.       </>
32.     );
33.   }
34. }
```



ตัวอย่างที่ 3 && Quiz

- Inline Styles & StyleSheet Objects
- Quiz