React Native

Touch interface

and

Lifecycle

Buttons

- Renders
  - blue label on iOS
  - blue rounded rectangle with white text on Android.
- "onPress" function
  - Can call an event handler
  - Can be anonymous function
  - in this case displays an alert popup.
  - can specify a "color" prop to change the color of your button.

```
<Button
  onPress={() => {
    Alert.alert('You tapped the button!');
  }}
  title="Press Me"
/>
```

Button 2

This is Home.js

```
import React, { Component } from 'react';
import { AppRegistry, Text, TextInput, View, Button, Keyboard } from 'react-native’;
export default class Home extends Component {
  constructor(props){
    super(props);
    this.state = { myState: this.props.origText, myState2: this.props.origText2,
myState3:'Waiting for text'};
  }
  updateState = () => {
       this.setState({ myState3: this.state.myState + '\n' + this.state.myState2})
  };
```

Continued on next slide

```
  render() {
    return (
    <View style={{padding: 10, marginTop: 100}}>
       <TextInput
        style={{height: 40}}
        placeholder="Type here to translate!"
        onChangeText={(myState) => this.setState({myState})}
        onSubmitEditing={Keyboard.dismiss}
      />
      <TextInput
        style={{height: 40}}
        placeholder="This is the second input line: type more!"
        onChangeText={(myState2) => this.setState({myState2})}
        onSubmitEditing={Keyboard.dismiss}
      />
      <Text style={{padding: 10, fontSize: 12}}>
        {this.state.myState3}
      </Text>
      <Button onPress = {this.updateState}
           color="#841584"
           title="Click to update"
           accessibilityLabel=”Update button"/>
    </View>      );
}    }
```

color is the color of the text on iOS
And the color of the background on Android

accessibilityLabel is to associate text with a button for accessibility

Create your own button

- build your own button using any of the "Touchable" components
- "Touchable" components provide the capability to capture tapping gestures,
- "Touchable" components can display feedback when a gesture is recognized.
- No provide any default styling,
  - Must provide your own styling.

Touchable components

- use **`TouchableHighlight`** anywhere you would use a button or link on web.
  - The view's background will be darkened when the user presses down on the button.

- Use **`TouchableNativeFeedback`** on Android to display ink surface reaction ripples that respond to the user's touch.

- **`TouchableOpacity`** can be used to provide feedback by reducing the opacity of the button, allowing the background to be seen through while the user is pressing down.

- If you need to handle a tap gesture but you don't want any feedback to be displayed, use **`TouchableWithoutFeedback`**.

Touchable components

- Examples: see facebook tutorial:

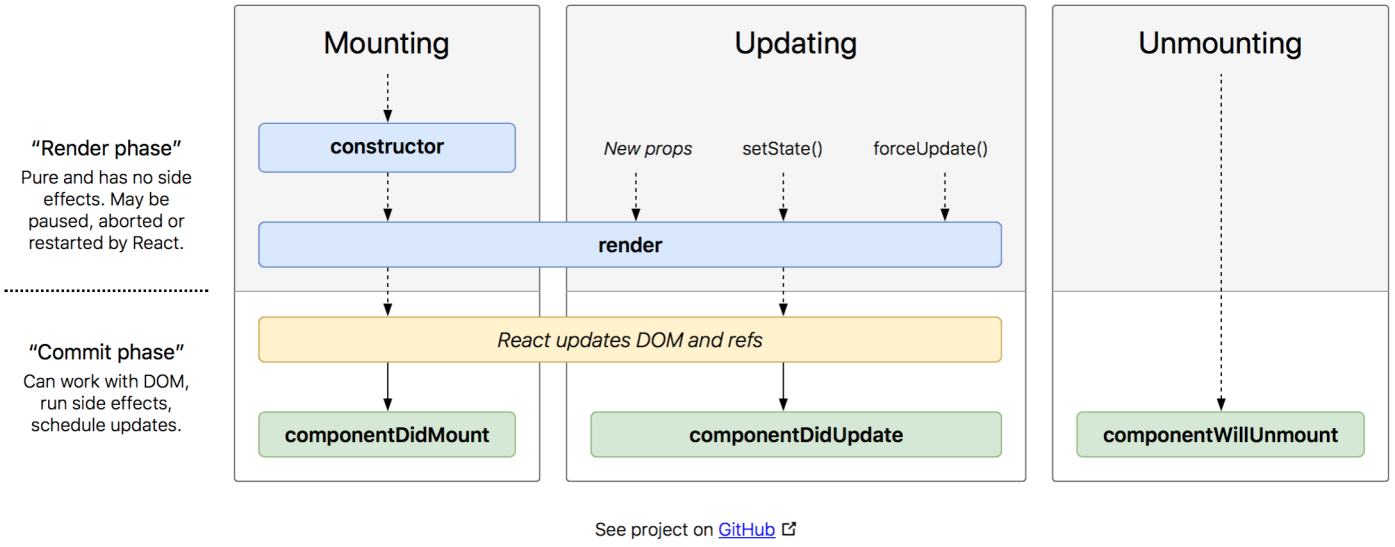
https://facebook.github.io/react-native/docs/handling-touches

Lifecycle

- iOS and Android apps go through a lifecycle
  - As the app changes state and as views are shown/hidden different events occur
  - Both platforms have lifecycle functions that an app can implement
  - These functions are called at the appropriate time.
- React Native offers some of these lifecycle functions
  - Also has a few of its own

Component Lifecycle

- Each component has several “lifecycle methods” that you can override to run code at particular times in the process.



See project on GitHub

Mounting

- These methods are called in the following order when an instance of a component is being created and inserted into the DOM:
  - **constructor()**
  - static getDerivedStateFromProps()
  - **render()**
  - **componentDidMount()**

- This method is considered legacy and you should avoid it in new code:
  - UNSAFE_componentWillMount()

Updating

- An update can be caused by changes to props or state.
- These methods are called in the following order when a component is being re-rendered:
  - static getDerivedStateFromProps()
  - shouldComponentUpdate()
  - **render()**
  - getSnapshotBeforeUpdate()
  - **componentDidUpdate()**
- These methods are considered legacy and you should avoid them in new code:
  - UNSAFE_componentWillUpdate()
  - UNSAFE_componentWillReceiveProps()