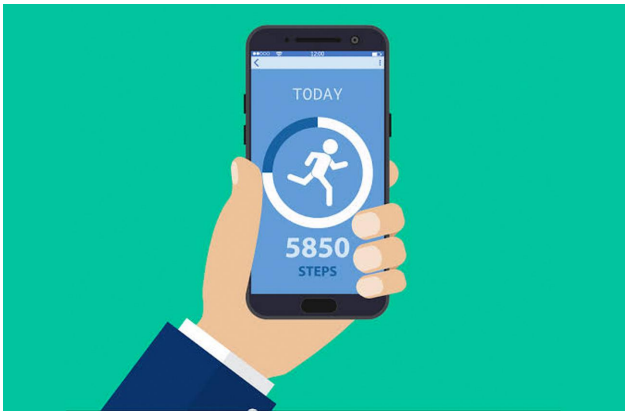
Component Lifecycle and State — Simple Counter App



What we did:

- Learned about the component lifecycle and the functions which are called at different stages of the component lifecycle.
- Learned about the state of a react component and how to set the state.
- Built a simple Counter App.
- Changed the color of a button to randomly generated color.

How we did it:

In order to build the Quiz Admin App, we need to understand two very important concepts in React Native:

- The lifecycle of a React Component
- State of a React Component

Every React Component rendered on the screen also has a lifecycle.

A React Component has the following stages in its lifecycle:

- Mounting: This is when the React Components are created and rendered on the screen.
- Updating: This is when the components are updated.
- For example, their prop values are changed.
- Unmounting: This is when the components are removed from the screen.

'componentDidMount()' and 'render()' are two functions which automatically get called at the Mounting Stage of a component.

Mounting

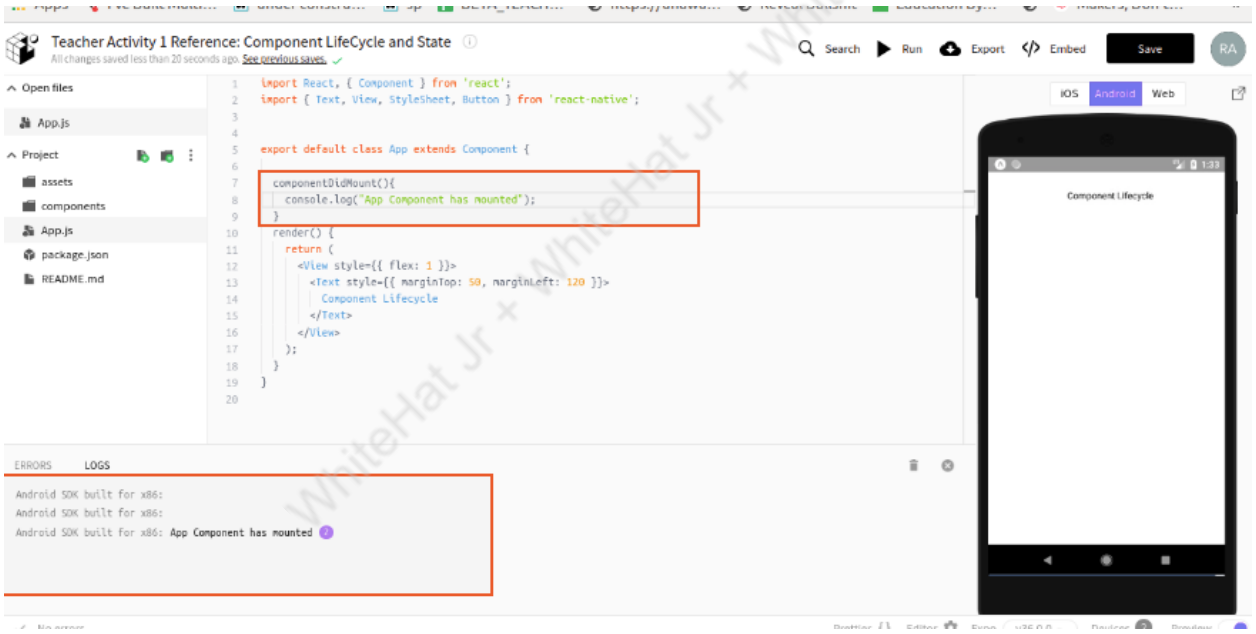
These methods are called in the following order when an instance of a component is being created and inserted into the DOM:

- `constructor()`
- `static getDerivedStateFromProps()`
- `render()`
- `componentDidMount()`

Note:

These methods are considered legacy and you should avoid them in new code:

We wrote a console log message inside 'componentDidMount()' function and ran the code



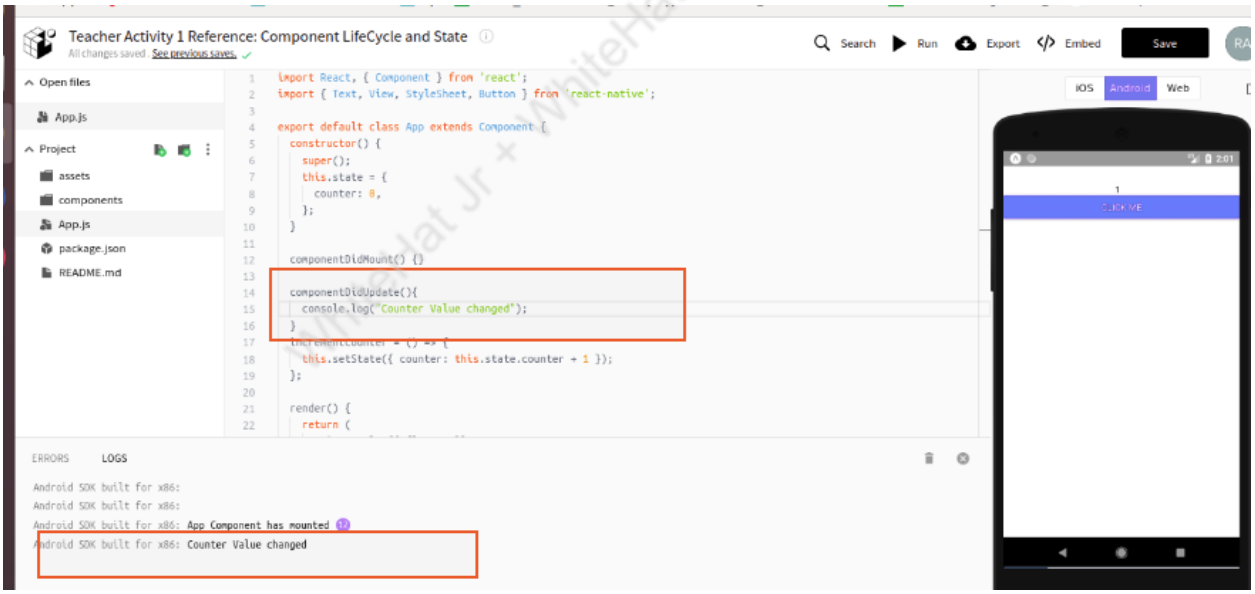
We wrote code to show how the state of a component is declared and accessed.
- State of a component is declared inside the constructor.
- 'super()' is used in the constructor to inherit the properties of the Component Class.

We wrote a function which changes the state of the counter by incrementing the current counter state by 1.

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


export default class App extends Component {

constructor(){
  super();
  this.state = {
    counter: 0
  }
}

componentDidMount(){

  }

incrementCounter(){
  this.setState({counter: this.state.counter+1});
}

  render() {
    return (
      <View style={{ flex: 1 }}>
        <Text style={{ marginTop: 50, marginLeft: 170 }}>
          {this.state.counter}
        </Text>
      </View>
    );
  }
}
```

We created a Button which calls this function.

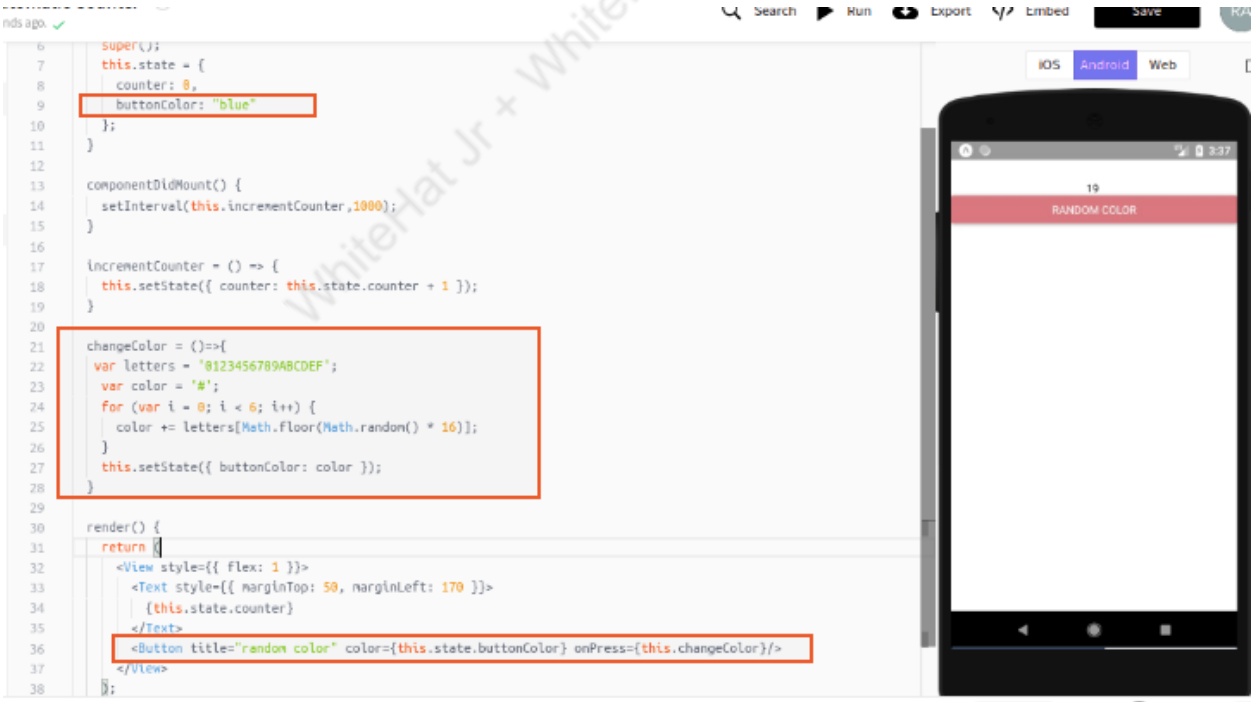
We logged a message inside the 'ComponentDidUpdate()' to see if the component is updating.



We wrote code to increment the counter on its own every 1s or 1000 ms using the 'setInterval()' function to call the 'incrementCounter' every second. We used the function inside of 'componentDidMount()'

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

export default class App extends Component {
  constructor() {
    super();
    this.state = {
      counter: 0,
    };
  }

  componentDidMount() {
    setInterval(this.incrementCounter,1000);
  }

  incrementCounter = () => {
    this.setState({ counter: this.state.counter + 1 });
  };

  render() {
    return (
      <View style={{ flex: 1 }}>
        <Text style={{ marginTop: 50, marginLeft: 170 }}>
          {this.state.counter}
        </Text>
      </View>
    );
  }
}
```

We changed the color of a button with a random color, every time the button is clicked.



```
    super();
    this.state = {
      counter: 0,
      buttonColor: "blue"
    };
  }

  componentDidMount() {
    setInterval(this.incrementCounter,1000);
  }

  incrementCounter = () => {
    this.setState({ counter: this.state.counter + 1 });
  }

  changeColor = ()=>{
   var letters = '0123456789ABCDEF';
    var color = '#';
    for (var i = 0; i < 6; i++) {
      color += letters[Math.floor(Math.random() * 16)];
    }
    this.setState({ buttonColor: color });
  }

  render() {
    return (
      <View style={{ flex: 1 }}>
        <Text style={{ marginTop: 50, marginLeft: 170 }}>
          {this.state.counter}
        </Text>
        <Button title="random color" color={this.state.buttonColor} onPress={this.changeColor}/>
      </View>
    );
```

What's next?:

In the next class, we are going to use the concepts covered in today's class to create a Quiz Master App.