WhiteHat Jr
{ Live Online Coding for Kids }

| **Topic** | **Component Lifecycle and State** | |
|---|---|---|
| **Class Description** | **Students learn about Component Lifecycle and State of a component. They build a simple counter app using the property of state of a component and component lifecycle.** | |
| **Class** | **C59** | |
| **Class time** | **45 mins** | |
| **Goal** | ● Learn about the component lifecycle and the functions which are called at different stages of the component lifecycle.<br>● Learn about the state of a react component and how to set the state.<br>● Build a simple Counter App.<br>● Change the color of a button to randomly generated color. | |
| **Resources Required** | ● Teacher Resources<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Android/iOS Smartphone with Expo App installed<br>○ Expo Snack account<br><br>● Student Resources<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Android/iOS Smartphone with Expo App installed<br>○ Expo Snack Account | |
| **Class structure** | **Warm Up**<br>**Teacher-led Activity**<br>**Student-led Activity**<br>**Wrap up** | **5 mins**<br>**15 min**<br>**15 min**<br>**5 min** |

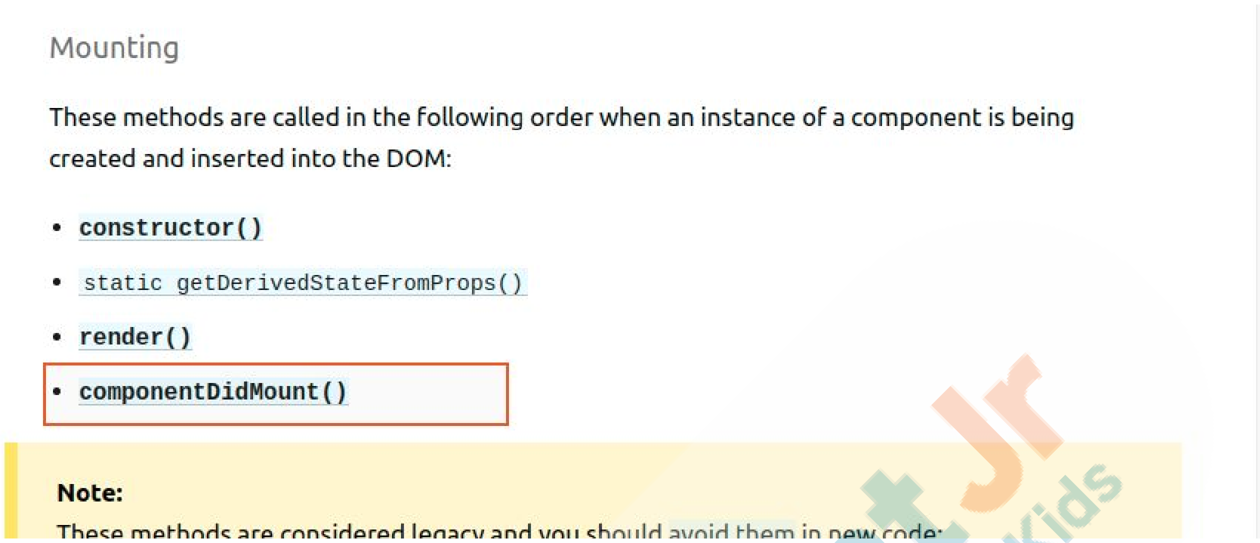
**CONTEXT**

- **Set stage for understanding the state of react components and their life cycle.**

| **Class Steps** | **Teacher Action** | **Student Action** |
|---|---|---|
| **Step 1:**<br>**Warm Up**<br>**(5 mins)** | We ended the last class with the creation of a quiz buzzer app.<br><br>This app allows users to choose their teams from four different teams and press a buzzer.<br><br>The user pressing a button is recorded in the database along with the timestamp.<br><br>What more do we need to make the app practically useful? | ESR:<br>We talked about a Quiz admin app which displays the names of the teams that press the button and it displays the team names in the order in which they pressed the buttons. |
| | In order to build the Quiz Admin App, we need to understand two very important concepts in React Native:<br>- Lifecycle of a React Component<br>- State of a React Component | . |
| | Earlier, we have explored the props in a React Component.<br><br>Can you recall with an example what props are? | ESR:<br>Props are properties of a React Component. They are similar to attributes in HTML.<br><br>For example: color, title, onPress are the props of the Button Component. They can be accessed by 'this.props.color' or 'this.props.title' in code for the Component class. |

| | | |
|---|---|---|
| | So far, the app that we have built mostly uses the power of Props.<br><br>Our apps will become more powerful once we understand about the States and the Lifecycle of React Components.<br><br>So let's get started. | - |
| **Teacher Initiates Screen Share** | | |
| **CHALLENGE**<br>● **Build a simple counter which increments on the press of a button.** | | |
| **Step 2: Teacher-led Activity (15 min)** | Have you heard about Lifecycles?<br>What is the lifecycle of a Butterfly? | ESR:<br>Egg -> Larva -> Pupa -> Adult Butterfly. |
| | Great! Every React Component rendered on the screen also has a lifecycle.<br><br>A React Component has the following stages in its lifecycle:<br>● **Mounting:** This is when the react components are created and rendered on the screen.<br>● **Updating**: This is when the components are updated. For example: Their prop values are changed.<br>● **Unmounting:** This is when the components are removed from the screen. | ESR:<br>varied |

| | | |
|---|---|---|
| | Now, how do you think these different React component lifecycle stages would be important for creating an app? | |
| | Each Lifecycle stage has certain methods defined on them.<br><br>Look at **Student/Teacher Activity 1** to see the different methods defined on the different Lifecycle stages of React Components.<br><br>These methods get called automatically when the React Component reaches that particular lifecycle stage. | The student looks at the reference link to see the different methods defined for different Lifecycle stages of a React Component. |
| | Let's look at an example.<br><br>You can see that 'componentDidMount()' and 'render()' are two functions which automatically get called at the Mounting Stage of a component.<br><br>You have already seen the render() function.<br><br>Let's look at the componentDidMount() function. Let's call the function and inside it let's console log something to say that the Component has been mounted.<br><br>This function gets called as soon as the Component is mounted. | The student listens and asks questions |



Mounting

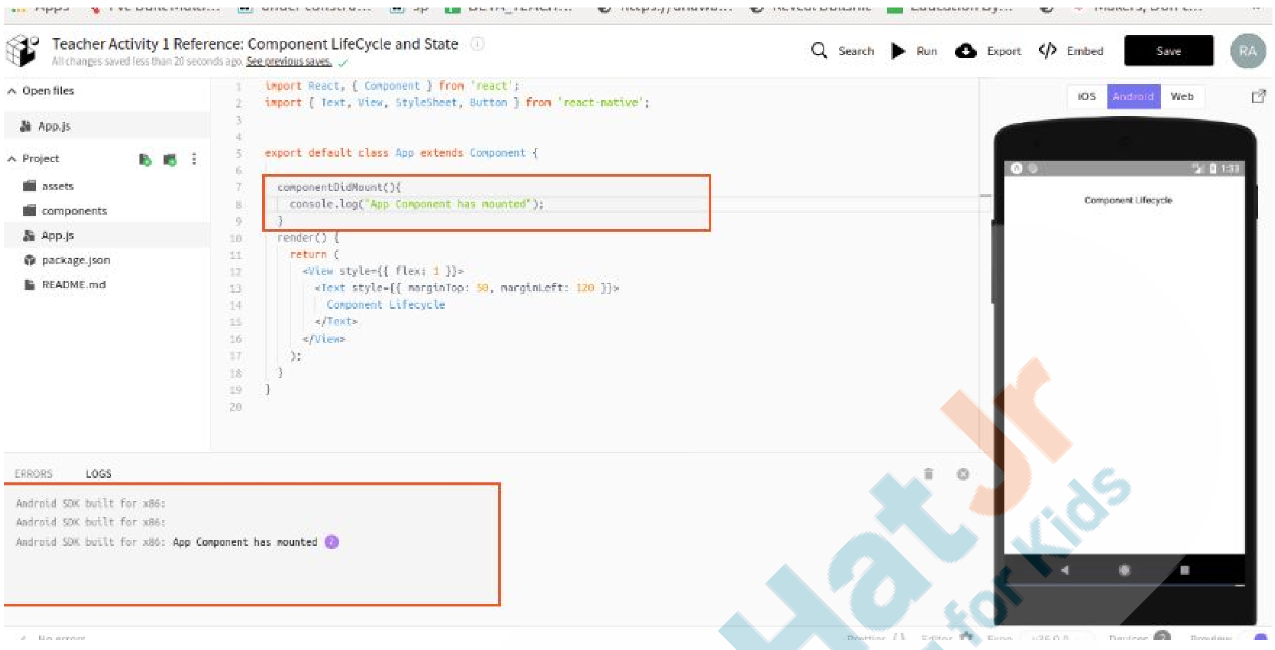
These methods are called in the following order when an instance of a component is being created and inserted into the DOM:

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

**Note:**

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

| | | |
|---|---|---|
| | Teacher opens **<u>Teacher Activity 2.</u>** She writes a console log message inside 'componentDidMount()' function and runs the code on the device.<br><br>The message gets logged in the console log as soon as the component is mounted.<br><br>Note: The console log is at the bottom bar of the expo snack. Click on it to expand the log column. | The student observes the code and the output. |

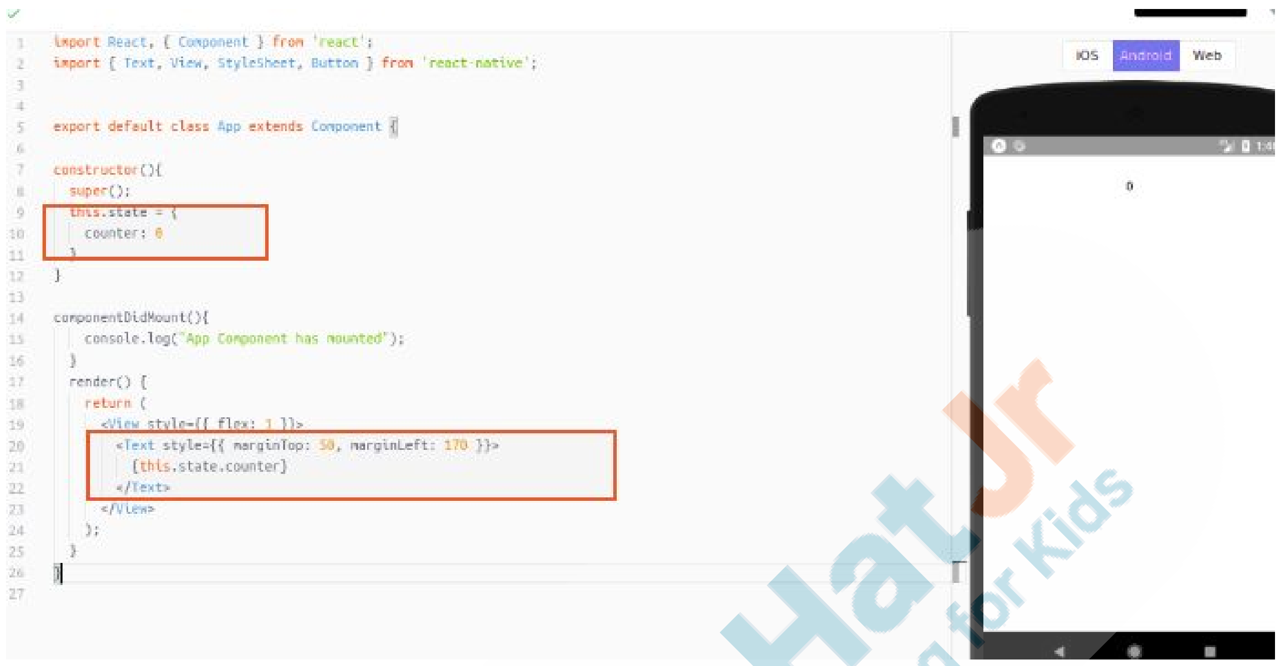

Teacher Activity 1 Reference: Component LifeCycle and State

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


export default class App extends Component {

  componentDidMount(){
    console.log("App Component has mounted");
  }
  render() {
    return (
      <View style={{ flex: 1 }}>
        <Text style={{ marginTop: 50, marginLeft: 120 }}>
          Component Lifecycle
        </Text>
      </View>
    );
  }
}
```

```
Android SDK built for x86:
Android SDK built for x86:
Android SDK built for x86: App Component has mounted
```

| | | |
|---|---|---|
| | Great!<br><br>So, for example, we want to run some piece of code as soon as the app renders on the screen (like - fetch data from the database), we can do that inside the 'componentDidMount()' function.<br><br>Similarly if we want to give instructions to do something as soon as the app gets unmounted (like - saving/writing data to the database), we can do that inside the 'componentDidUnmount()' function. | The student listens and asks questions. |
| | You also know that a Component Updates itself whenever the prop of the component changes. | |

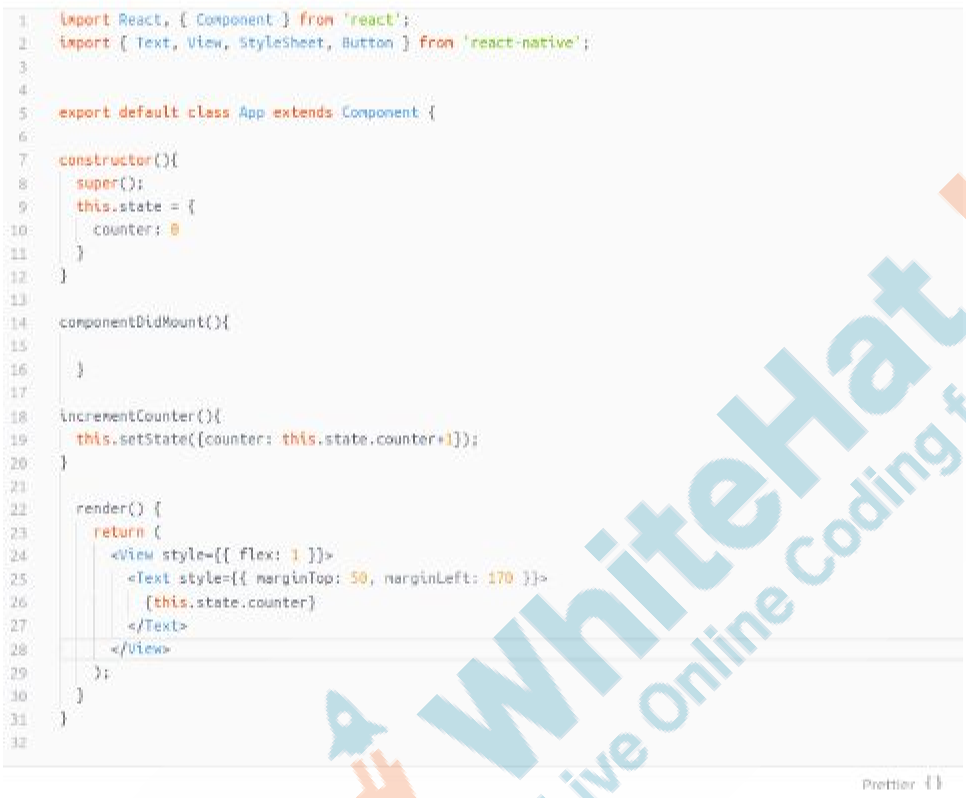
| | | |
|---|---|---|
| | There is another way a component can update itself - whenever its State is changed.<br><br>We have not learned about State of a component till now - but we are going to do it now and it is a very powerful concept. | |
| | Each component can hold a state object. The object can hold any number of keys and values.<br><br>The state of the component can be accessed by 'this.state.<name of the key>'.<br><br>Let me quickly show you how.<br><br>Teacher writes code to show how the state of a component is declared and accessed.<br>- State of a component is declared inside the constructor.<br>- 'super()' is used in the constructor to inherit the properties of the Component Class. | The student observes the code and asks questions. |



```
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(){
    console.log('App Component has mounted');
  }
  render() {
    return (
      <View style={{ flex: 1 }}>
        <Text style={{ marginTop: 50, marginLeft: 170 }}>
          {this.state.counter}
        </Text>
      </View>
    );
  }
}
```

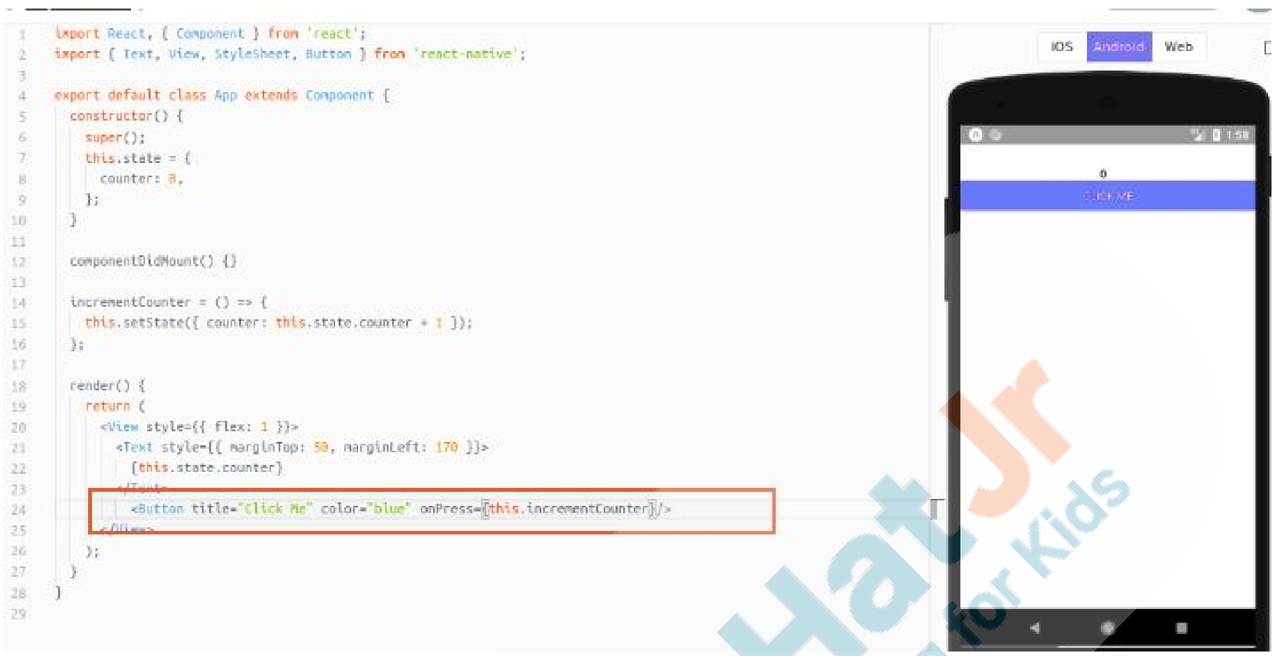
| | | |
|---|---|---|
| | State of a component can be updated by only using the function 'this.setState()'. It takes the new state object as its argument. Directly changing the state of a component results in error.<br><br>Let's create a button. On press of this button, the counter state of the component increments by 1.<br><br>Whenever the state of the component will change, the component will update and 'componentDidUpdate()' will be called. *(Refer to lifecycle methods.)*<br><br>Can you help me write the code to test this? | ESR:<br>yes |

| | | |
|---|---|---|
| | Let's first write a function which changes the state of the counter by incrementing the current counter state by 1. | The student guides the teacher in writing the code for this. |



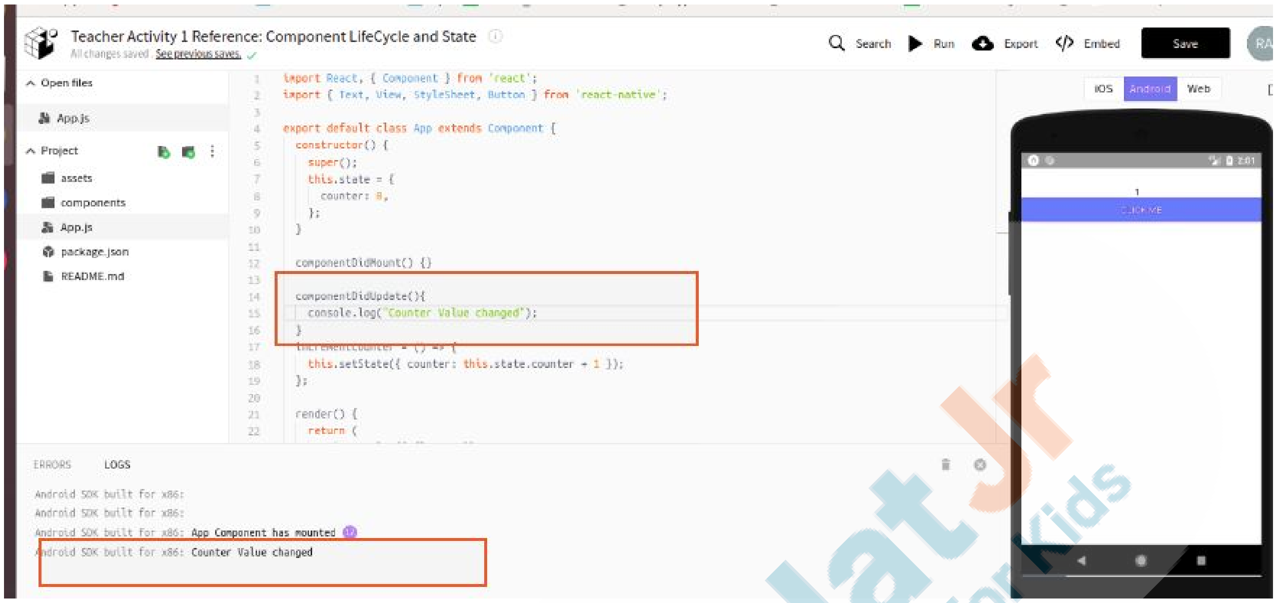
```
import React, { Component } from 'react';
import { Text, View, StyleSheet, Button } from 'react-native';


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>
    );
  }
}

```

| | | |
|---|---|---|
| | Now let's create a Button which calls this function. | The student guides the teacher in writing the code for this. |



```
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>
        <Button title="Click Me" color="blue" onPress={this.incrementCounter}/>
      </View>
    );
  }
}
```

| | | |
|---|---|---|
| | Now let's run the app on our device and see the output. | The student and teacher run the program on their devices and check their outputs. |
| | Everytime, the state changes, you can see that the Text Component gets updated.<br><br>You can also log a message inside the 'ComponentDidUpdate()' to see if the component is updating.<br><br>Teacher writes the code and shows the console to the student. | |



```
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() {}

  componentDidUpdate(){
    console.log('Counter Value changed');
  }
  incrementCounter = () => {
    this.setState({ counter: this.state.counter + 1 });
  };

  render() {
    return (
```

```
Android SDK built for x86:
Android SDK built for x86:
Android SDK built for x86: App Component has mounted
Android SDK built for x86: Counter Value changed
```

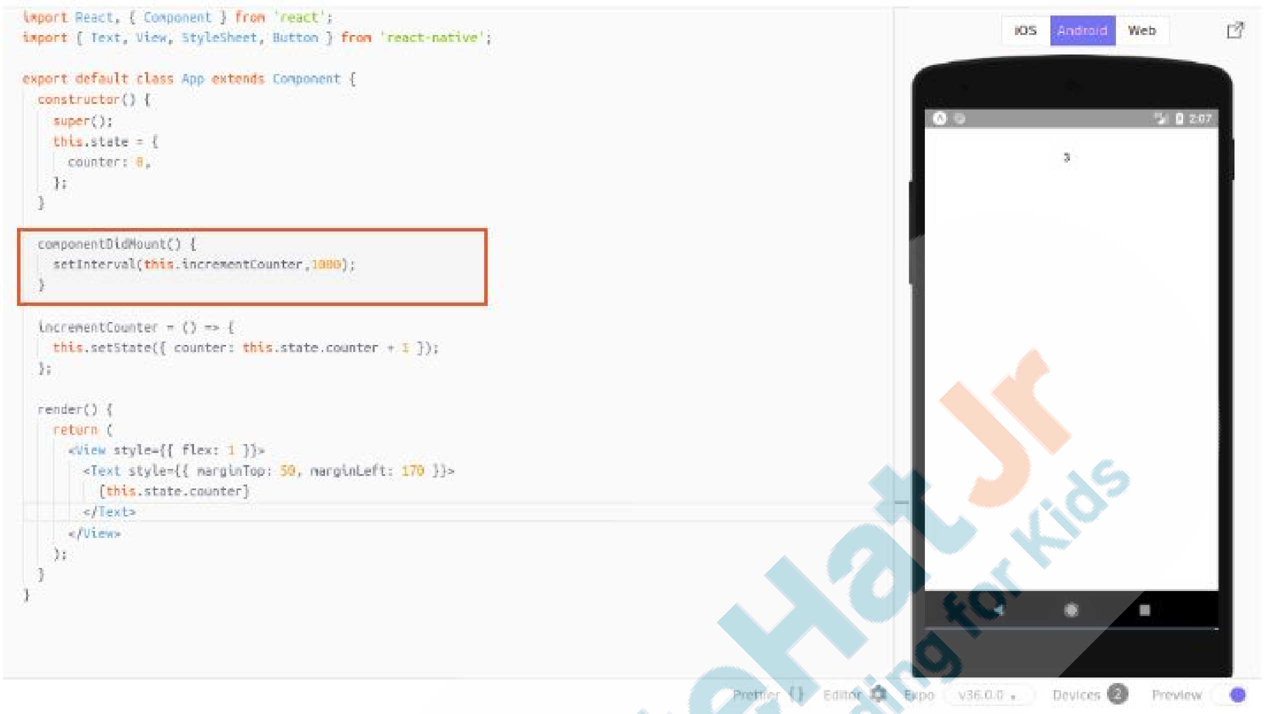
| | | |
|---|---|---|
| | When we are building our Quiz Admin app, we will see how the concept of states can be so powerful and help us in building the app.<br><br>For now, here is a challenge for you. Right now we need to click on a button to increment the counter. Can you write code to increment the counter on its own every 1s or 1000 ms? | The student takes up the challenge. |
| **Teacher Stops Screen Share** | | |
| | Now it's your turn. Please share your screen with me. | |

- **Ask Student to press ESC key to come back to panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

**ACTIVITY**

- **Build an automatic counter app.**

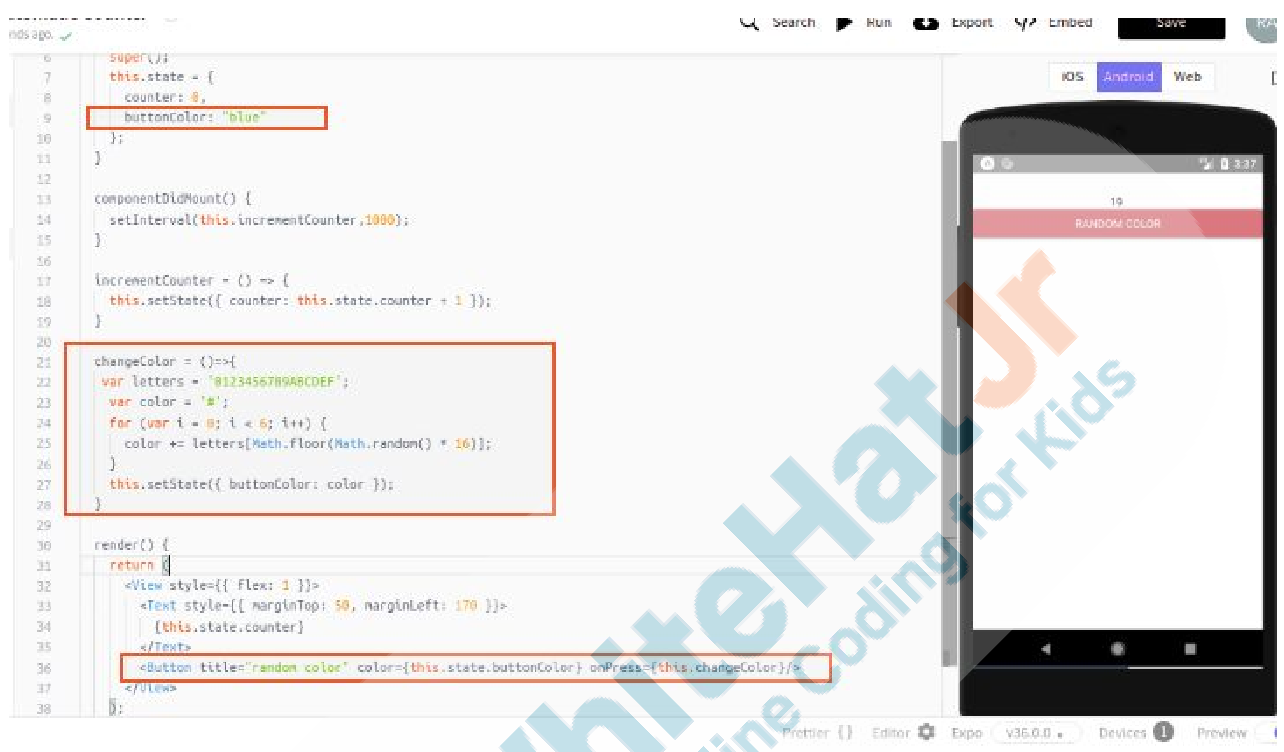
| | | |
|---|---|---|
| **Step 3: Student-Led Activity (15 min)** | Guide the student to open the Activity Link and create the counter state for the App Component. | The student opens **Student Activity 2** and creates the state for the component called counter. |
| | Guide the student to define an ‘incrementCounter()’ function which increments the counter by 1 and sets the new value as the counter state. | The student writes the ‘incrementCounter()’ function. |
| | Allow time for the student to think how to call the incrementCounter function automatically every second. | The student thinks about how to call incrementCounter function automatically every second. |
| | Guide the student to use the ‘setInterval()’ function to call the ‘incrementCounter’ every second.<br><br>- ‘setInterval’ calls a callback function after every given timeframe.<br>- We use the function inside of ‘componentDidMount()’ so that it is called since the app is rendered<br><br>Reason with the student on why we are calling the function inside componentDidMount().<br><br>Also check if the student understands how ‘setInterval’ function works. | The student writes the code to increment the counter automatically when the app renders. |



```
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>
    );
  }
}
```

| | | |
|---|---|---|
| | Let's run the code and test it. | The student runs the code on a device and tests it. |
| | Another Challenge for you!<br><br>Can you change the color of a button with a random color, every time the button is clicked.<br><br>Hint: You might have to add another key in the state. | The student takes up the challenge. |
| | Remind the student that the color is represented with hexadecimal numbers.<br><br>The hexadecimal way of representing color is a 6 digit number starting with "#". | The student recalls the hexadecimal color representation and writes code to change the color of the button every time it is pressed. |

| | | |
|---|---|---|
| | Each digit contains any number from 0 to F. | |



```
    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>
    );
```

- **Initialize buttonColor state**
- **Write a function which generates random color. It should pick a random letter from the hexadecimal system, concatenate them to generate a random color.**
- **Set the color of the button equal to the buttonColor state**

**Teacher Guides Student to Stop Screen Share**

**FEEDBACK**

- **Encourage the student to experiment with different functions of component lifecycle.**
- **Encourage the student to make reflection notes in the markdown format.**
- **Complement the student for her/his effort in the class.**

| | | |
|---|---|---|
| **Step 4: Wrap-Up (5 min)** | I hope you had great fun with the lesson today.<br><br>Can you recall what we learned | The student recalls the concepts learned in the class. |

| | | |
|---|---|---|
| | today?<br><br>Some guiding questions:<br>- What are the different stages of the Component LifeCycle?<br>- What are the lifecycle methods associated with each stage?<br>- What are Props and States? | |
| | You get a “hats off”.<br><br>Next class, we are going to use the concepts covered in today's class to create a Quiz Master App. | Make sure you have given at least 2 Hats Off during the class for:<br> Creatively Solved Activities<br>+10<br>Great Question<br>+10<br>Strong Concentration<br>+10 |
| **Project Pointers and Cues** (5 min) | NEWSLETTER APP - 3<br><br>**Goal of the Project:**<br><br>Today you learned about various lifecycles and states of components and developed a counter app.<br><br>In today's project, you will complete the newsletter app by keeping a track of ratings of both likes and dislikes.<br><br>*This is a continuation of Projects 57 and 58. So make sure you complete them before you attempt this one.* | |

| | | |
|---|---|---|
| | **Story:**<br><br>In a poll that you ran, ninety percent of your friends said that they would really benefit from a Newsletter type of app!<br><br>You have already started building this awesome app for your friends. You have created different buttons for the user to quickly navigate to different screens. You have also connected this app to the Firebase database. Now you have to code to keep a track of ratings of likes and dislikes.<br><br>I am very excited to see your project solution and I know you both will do really well.<br><br>Bye Bye! | |
| | **Teacher Clicks** ✖ End Class | |
| **Additional Activities** | Encourage the student to write reflection notes in their reflection journal using markdown.<br><br>Use these as guiding questions:<br><br>● What happened today?<br>- Describe what happened<br>- Code I wrote<br>● How did I feel after the class?<br>● What have I learned about programming and developing games? | The student uses the markdown editor to write her/his reflection in a reflection journal. |

| | | |
|---|---|---|
| | • What aspects of the class helped me? What did I find difficult? | |

| Activity | Activity Name | Links |
|---|---|---|
| Teacher Activity 1 | Component Lifecycle documentation | https://reactjs.org/docs/react-component.html |
| Teacher Activity 2 | Class Link | https://snack.expo.io/@whitehatjr/pro-c59-teacher-activity-1:-component-lifecycle |
| Teacher Activity 3 | Reference 1 | https://snack.expo.io/@whitehatjr/pro-c59-teacher-activity-1-reference:-component-lifecycle-and-state |
| Student Activity 1 | Component Lifecycle documentation | https://reactjs.org/docs/react-component.html |
| Teacher Activity 4 | Reference 2 | https://snack.expo.io/@whitehatjr/pro-c-59-teacher-reference-2:-automatic-counter |
| Student Activity 2 | Class Link | https://snack.expo.io/@whitehatjr/pro-c59-teacher-activity-1:-component-lifecycle |