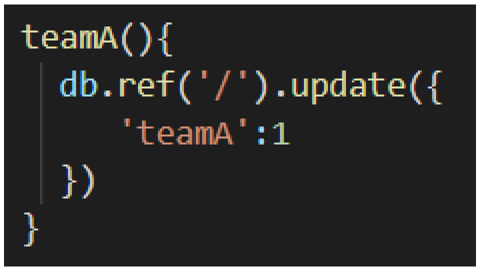
WhiteHat Jr
{ Live Online Coding for Kids }

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. ● Learn about the state of a react component and how to set the state. ● Build a simple Counter App. ● Change the color of a button to randomly generated color.	
Resources Required	● Teacher Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Android/iOS Smartphone with Expo App installed ○ Expo Snack account ● Student Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Android/iOS Smartphone with Expo App installed ○ Expo Snack Account	
Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **15 min** **5 min**

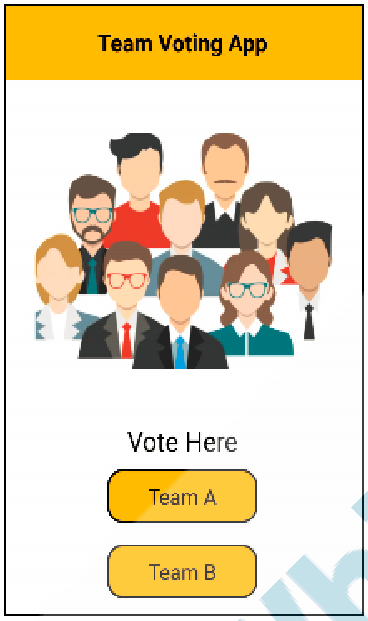
CONTEXT

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

Teacher starts slideshow from slides 1 to 10 Refer to speaker notes and follow the instructions on each slide.	
Activity details	**Solution/Guidelines**
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Run the presentation from slide 1 to slide 4** **Following are the WARM-UP session deliverables:** ● **Greet the student.** ● **Revision of previous class activities.** ● **Quizzes**	**ESR**: Hi, thanks, Yes I am excited about it! Click on the slide show tab and present the slides
QnA Session	
Question	**Answer**
Which is the correct way to export a database file? A. export default firebase.database ( ) B. export default firebase ( ) C. export firebase.database ( ) D. export default database ( )	**A**
What is the following block of code doing?	**D**



```
teamA(){
  db.ref('/').update({
      'teamA':1
  })
}
```

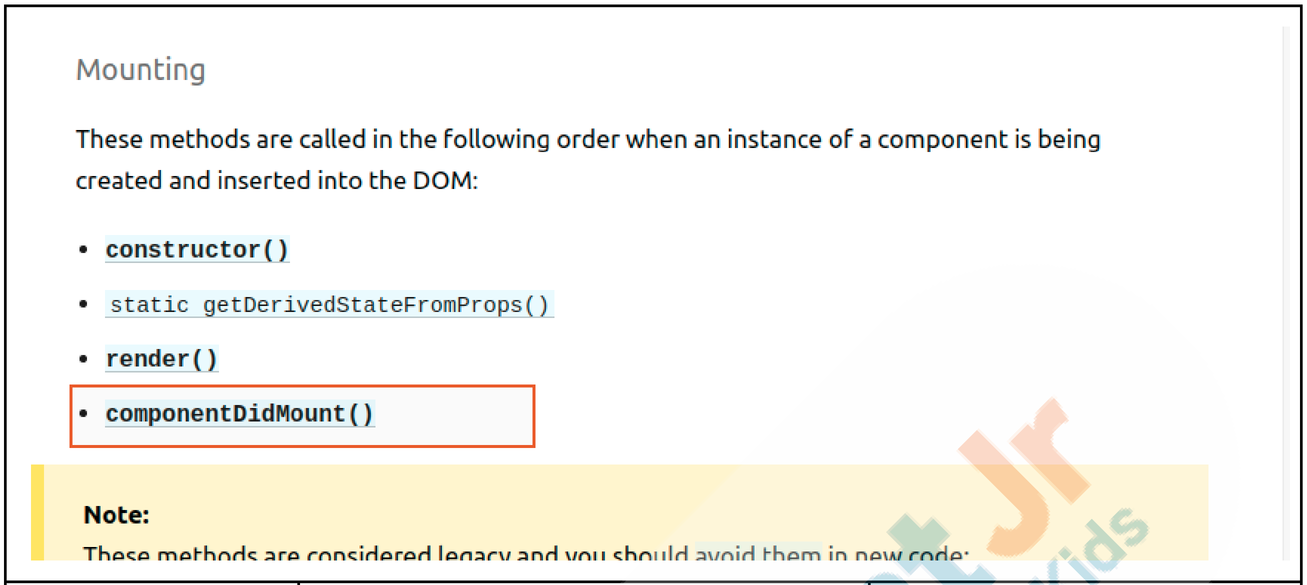


A. It is referring to the variable in code, and updating it to 1.
B. It is referring to the database, and updating the **teamA** node to 2.
C. It is referring to the database, and updating the **teamB** node to 1.
D. It is referring to the database, and updating the **teamA** node to 1.

Continue the WARM-UP session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 5 to slide 10 to set the problem statement. **Following are the WARM-UP session deliverables:** ● Appreciate the student. ● Explain Component lifecycle in React Native	Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.

Teacher ends slideshow		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
<u>CHALLENGE</u> ● **Build a simple counter which increments on the press of a button.**		
Step 2: Teacher-led Activity (15 min)	Have you heard about Lifecycles? What is the lifecycle of a Butterfly?	ESR: Egg -> Larva -> Pupa -> Adult Butterfly.
	Great! 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. Now, how do you think these different React component lifecycle stages would be important for creating an app?	ESR: varied

	Each Lifecycle stage has certain methods defined on them. Look at **Student/Teacher Activity 1** to see the different methods defined on the different Lifecycle stages of React Components. 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. You can see that 'componentDidMount()' and 'render()' are two functions which automatically get called at the Mounting Stage of a component. You have already seen the render() function. 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. This function gets called as soon as the Component is mounted.	The student listens and asks questions



Mounting

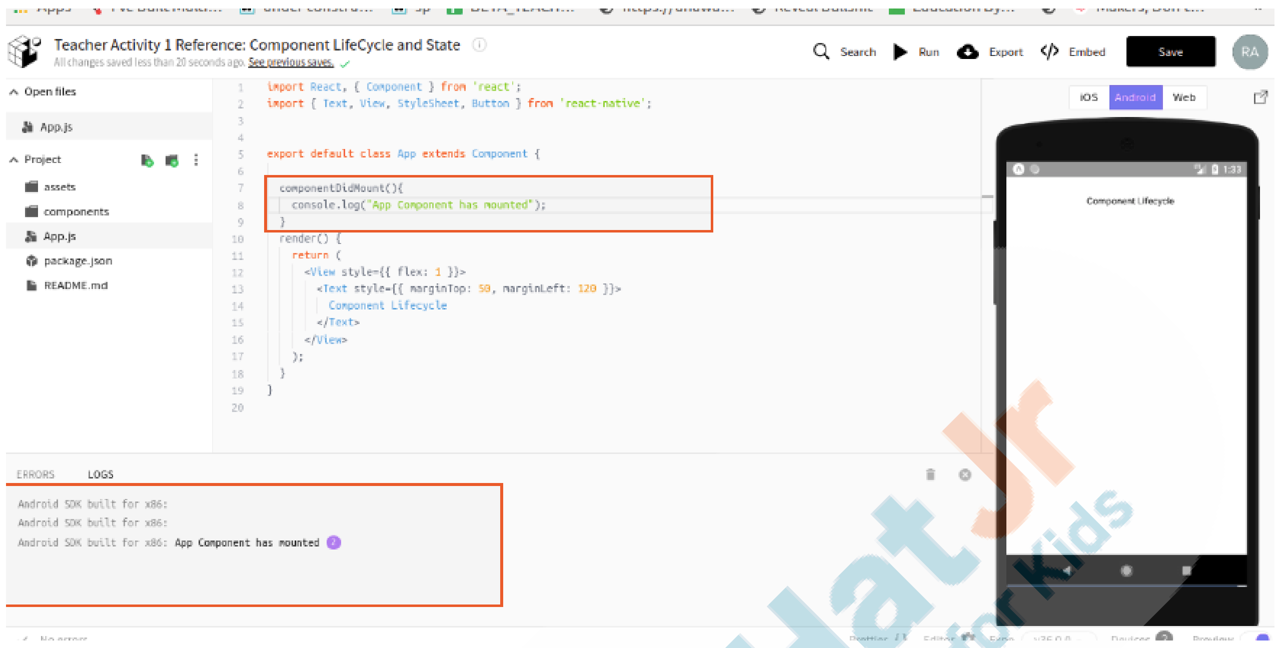
These methods are called in the following order when an instance of a component is being created and inserted into the DOM:

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

Note:

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

	Teacher opens **Teacher Activity 2.** She writes a console log message inside 'componentDidMount()' function and runs the code on the device. The message gets logged in the console log as soon as the component is mounted. 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.

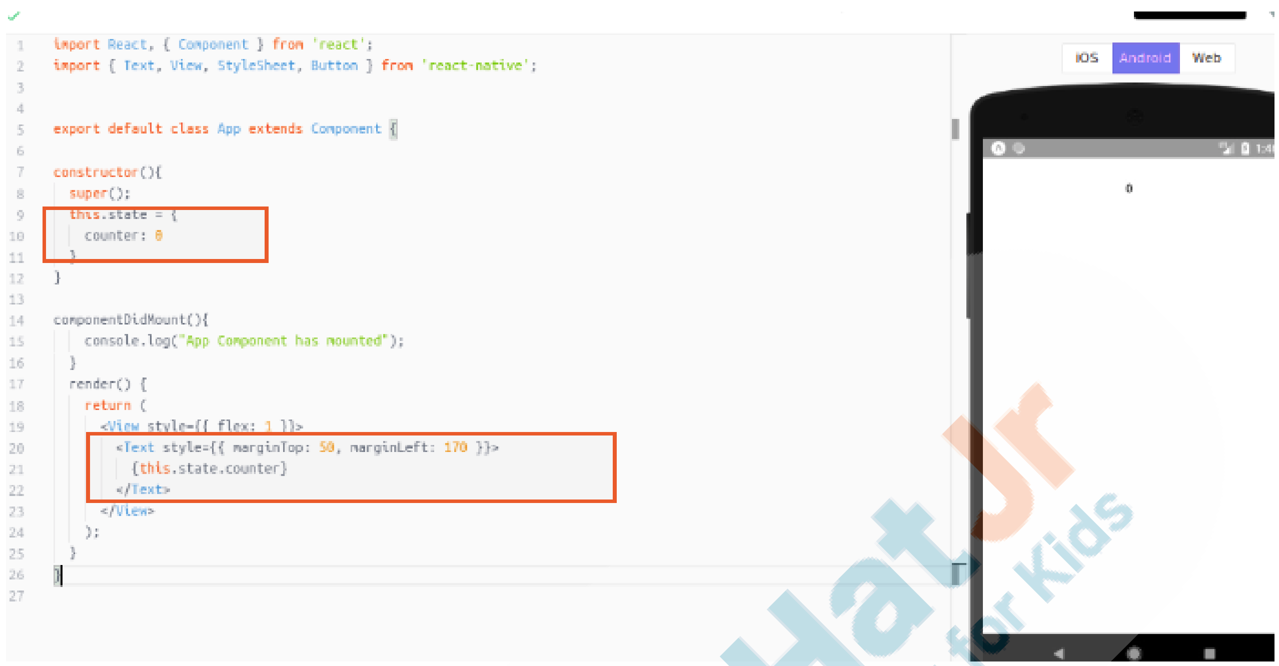


```
import React, { Component } from 'react';
import { Text, View, StyleSheet, Button } from 'react-native';


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! 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. 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.	

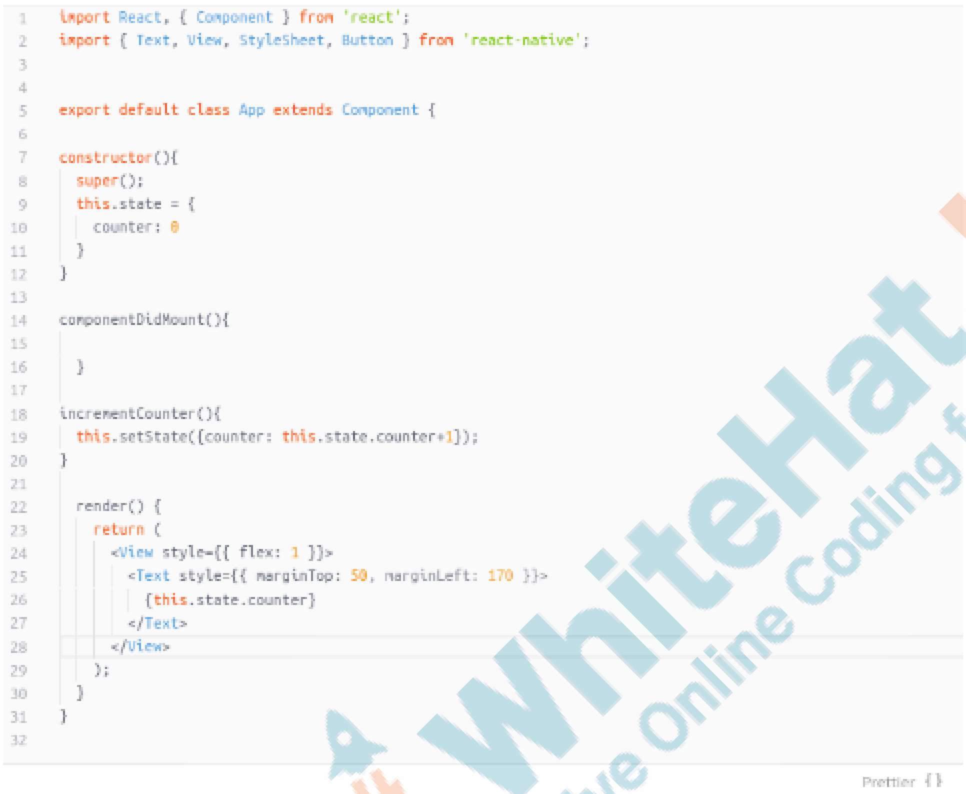
	There is another way a component can update itself - whenever its State is changed. We have not learned about State of a component till now - but we are going to do it now and it is a very powerful concept.	
	Each component can hold a state object. The object can hold any number of keys and values. The state of the component can be accessed by ‘this.state.<name of the key>’. Let me quickly show you how. Teacher writes 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.	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. Let's create a button. On press of this button, the counter state of the component increments by 1. Whenever the state of the component will change, the component will update and 'componentDidUpdate()' will be called. *(Refer to lifecycle methods.)* Can you help me write the code to test this?	ESR: yes

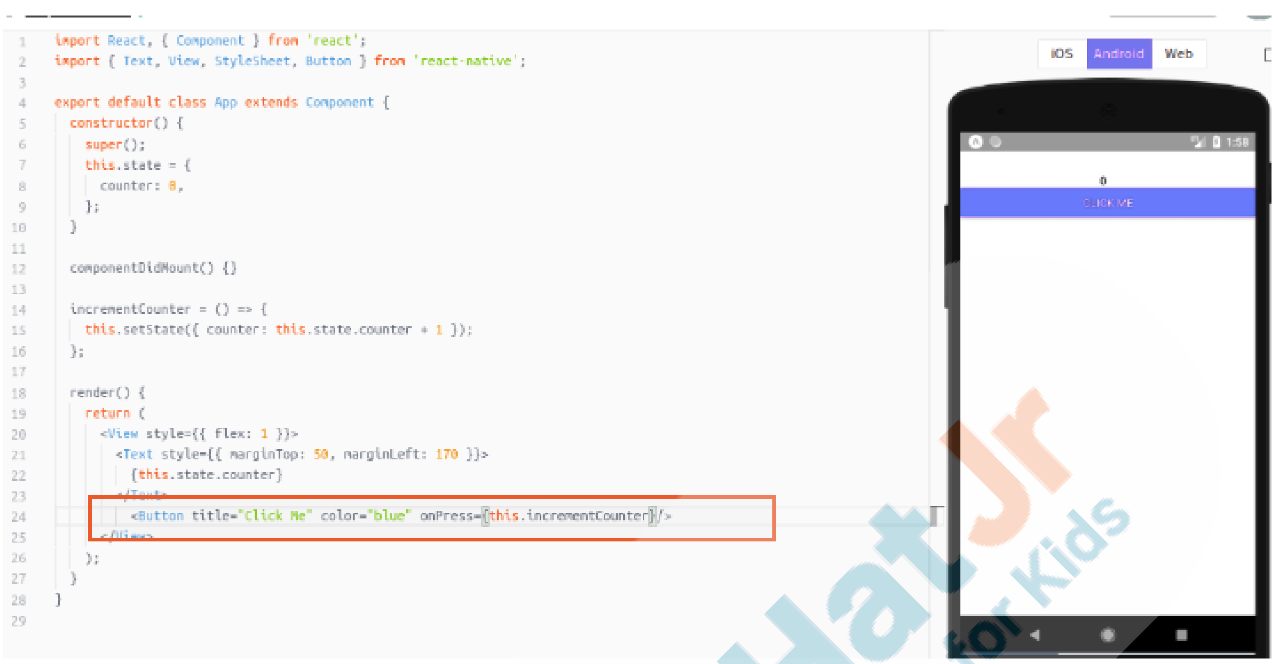
	Let's first write a function which changes the state of the counter by incrementing the current counter state by 1.	The student guides the teacher in writing the code for this.



```
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>
    );
  }
}

```

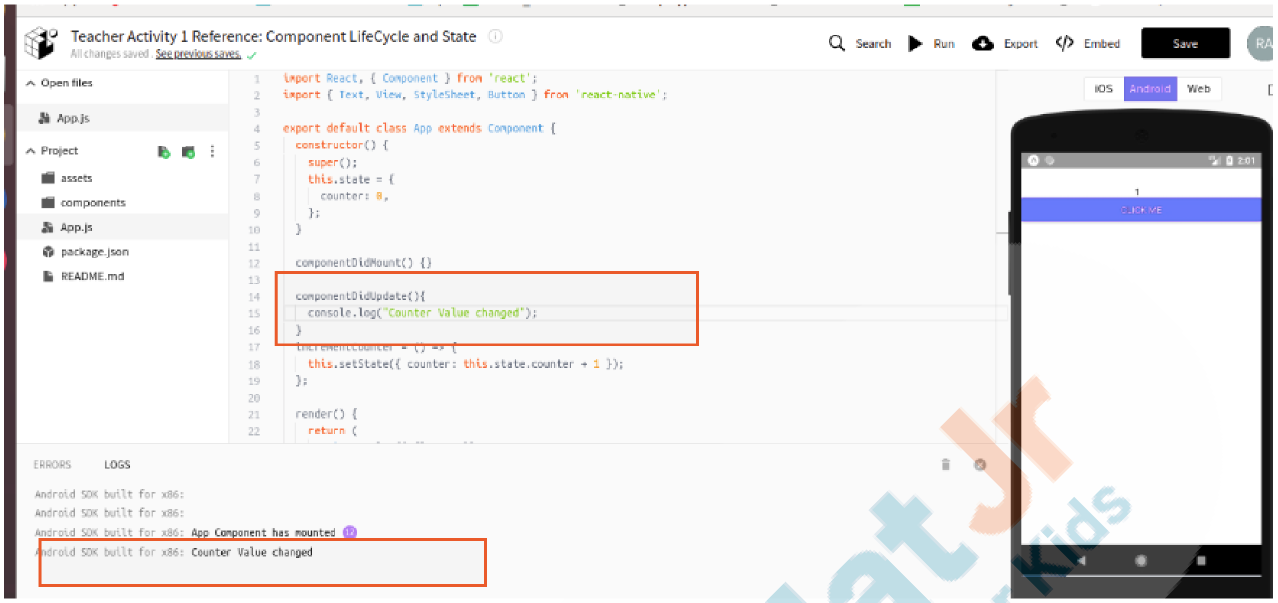
Prettier {}

	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. You can also log a message inside the 'ComponentDidUpdate()' to see if the component is updating. 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 (
```

ERRORS LOGS

```
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. 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

STUDENT-LED ACTIVITY - 25 mins

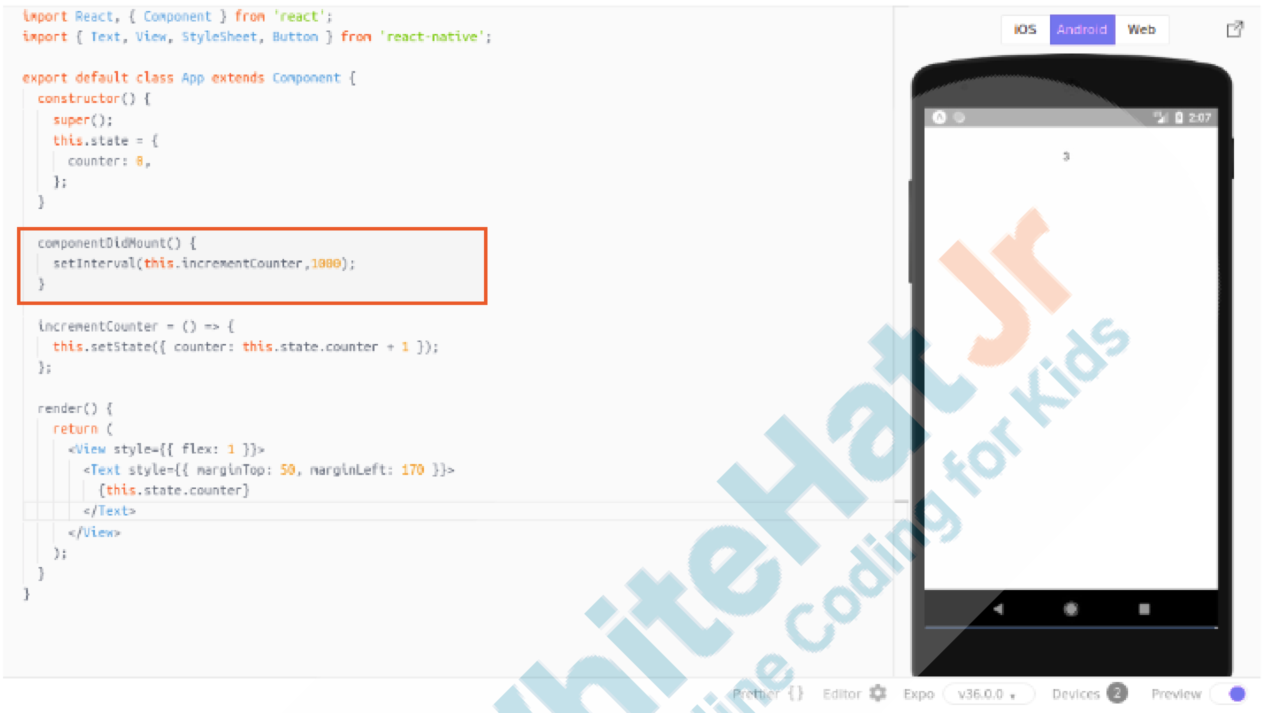
- **Ask Student to press ESC key to come back to panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

ACTIVITY

- **Build an automatic counter app.**

Teacher starts slideshow :Slide 11 to 12 Refer to speaker notes and follow the instructions on each slide.		
	Now it's your turn. Please share your screen with me.	
Teacher ends slideshow		
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. - ‘setInterval’ calls a callback function after every given timeframe. - We use the function inside of ‘componentDidMount()’ so that it is called since the app is rendered Reason with the student on why we are calling the function inside componentDidMount().	The student writes the code to increment the counter automatically when the app renders.

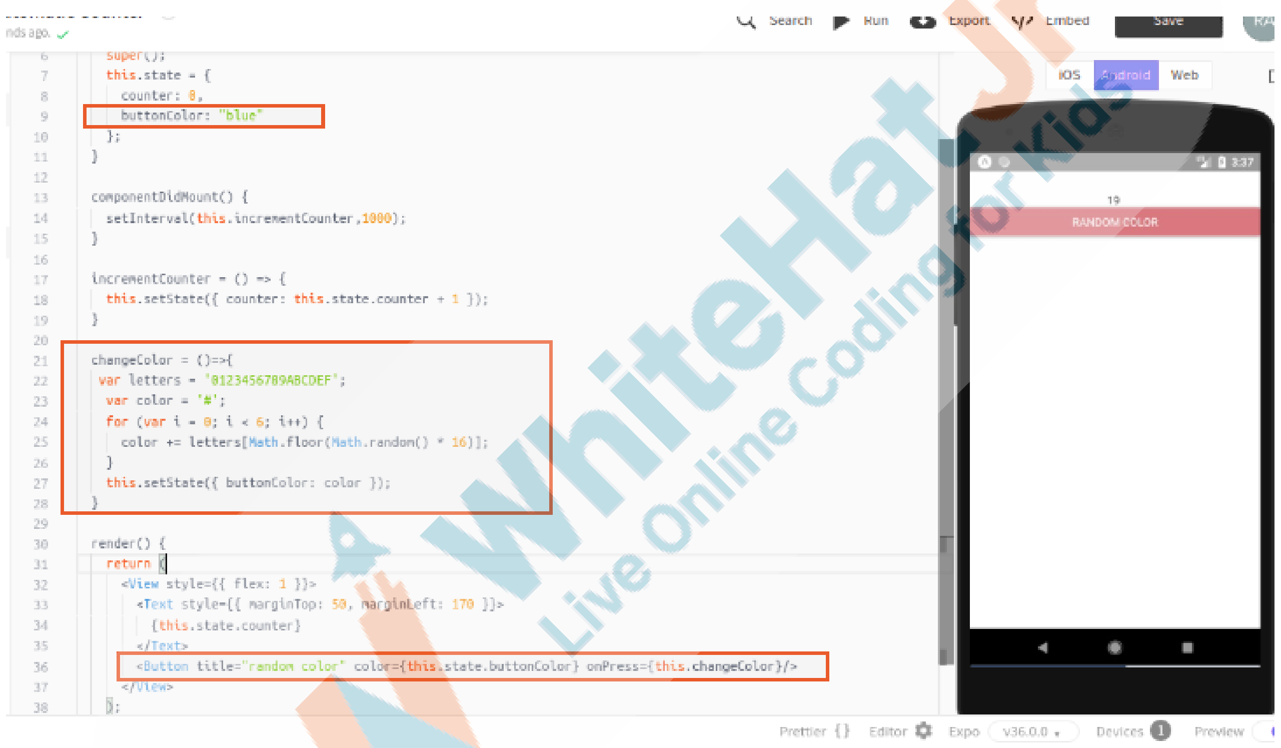
	Also check if the student understands how ‘setInterval’ function works.	



```
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! Can you change the color of a button with a random color, every time the button is clicked. 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. The hexadecimal way of representing color is a 6 digit number starting with "#". Each digit contains any number from 0 to F.	The student recalls the hexadecimal color representation and writes code to change the color of the button every time it is pressed.



```
  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

WRAP-UP SESSION - 5 Mins

Teacher starts slideshow from slide 13 to slide 23	
Activity details	**Solution/Guidelines**
Run the presentation from slide 13 to slide 23 **Following are the wrap-up session deliverables:** ● **Explain the facts and trivias** ● **Next class challenge** ● **Project for the day** ● **Additional Activity**	Guide the student to develop the project and share with us.
Quiz time - Click on in-class quiz	
Question	**Answer**
State of a component can be updated by only using the function ____________. A. componentDidUnmount() B. componentDidMount() C. componentDidUpdate() D. setState()	**D**
componentDidMount() function gets called automatically in which stage of component lifecycle? A. Updating B. Mounting C. Unmounting D. Unmasking	**C**
If we want to give instructions to do something as soon as the app gets unmounted, where can we write the code? A. componentDidUnmount() B. componentDidMount() C. componentDidUpdate()	**A**

D. onPress() of the button		
End the quiz panel		
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.**		
	You get a “hats off”. 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: Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
Project Pointers and Cues (5 min)	***This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.** NEWSLETTER APP **Goal of the Project:** In class 59, you learned about various “Lifecycle and States of Components” and developed a counter app. In this project, you will apply what you have learned in class and update the ratings in the Newsletter App. **Story:**	**Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.**

	In a poll that you ran, ninety percent of your friends said that they would really benefit from a Newsletter type of app! You have created different buttons for the user to quickly navigate to different screens. Now you have to code to keep a track of ratings of likes and dislikes. I am very excited to see your project solution and I know you both will do really well. Bye Bye!	
	Teacher Clicks ✖ End Class	
	Teacher ends slideshow	
Additional Activities	Encourage the student to write reflection notes in their reflection journal using markdown. Use these as guiding questions: ● What happened today? - Describe what happened - Code I wrote ● How did I feel after the class? ● 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
Project Solution	Newsletter App	https://snack.expo.dev/@whitehatjr/45dc75e3751541423a19c885868005f6
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C59_withcues.html

Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/d52df860-f6a4-470c-aa62-1f94d0b66bb0.pdf