ReactJS - Component Life Cycle Using React Hooks

React Hooks provides a special Hook, *useEffect()* to execute certain functionality during the life cycle of the component. *useEffect()* combines *componentDidMount, componentDidUpdate,* and *componentWillUnmount* life cycle into a single api.

The signature of the *useEffect()* api is as follows −

```
useEffect(
<executeFn>,
<values>
);
```

Here, • ***executeFn*** − Function to execute when an effect occurs with an optional return function. The return function will be execute when a clean up is required (similar to *componentWillUnmount*).

- ***values*** − array of values the effect depends on. React Hooks execute the *executeFn* only when the values are changed. This will reduce unnecessary calling of the *executeFn*.

Let us add *useEffect()* Hooks in our react-clock-hook-app application.

Open *react-clock-hook-app* in your favorite editor.

Next, *open src/components/Clock.js* file and start editing.

Next, import *useEffect api*.

```
import React, { useState, useEffect } from 'react';
```

Next, call *useEffect* with function to set date and time every second using *setInterval* and return a function to stop updating the date and time using *clearInterval*.

```
useEffect(    () => {       let setTime
= () => {          console.log("setTime
is called");
setCurrentDateTime(new Date());
      }       let interval =
setInterval(setTime, 1000);       return ()
=> {          clearInterval(interval);
      }
   },
   []
);
```

Here,

- Created a function, *setTime* to set the current time into the state of the component.
- Called the *setInterval* JavaScript api to execute *setTime* every second and stored the reference of the *setInterval* in the *interval* variable.
- Created a return function, which calls the *clearInterval* api to stop executing *setTime* every second by passing the *interval reference*.

Now, we have updated the Clock component and the complete source code of the component is as follows −

```
import React, { useState, useEffect } from 'react';

function Clock() {    const [currentDateTime, setCurrentDateTime]
= useState(new Date());    useEffect(       () => {          let
setTime = () => {              console.log("setTime is called");
setCurrentDateTime(new Date());
         }          let interval =
setInterval(setTime, 1000);          return ()
=> {             clearInterval(interval);
         }
      },
      []
);
return (
<div>
         <p>The current time is {currentDateTime.toString()}</p>
      </div>
   );
}
export default Clock;
```

Next, open *index.js* and use *setTimeout* to remove the clock from the DOM after 5 seconds.

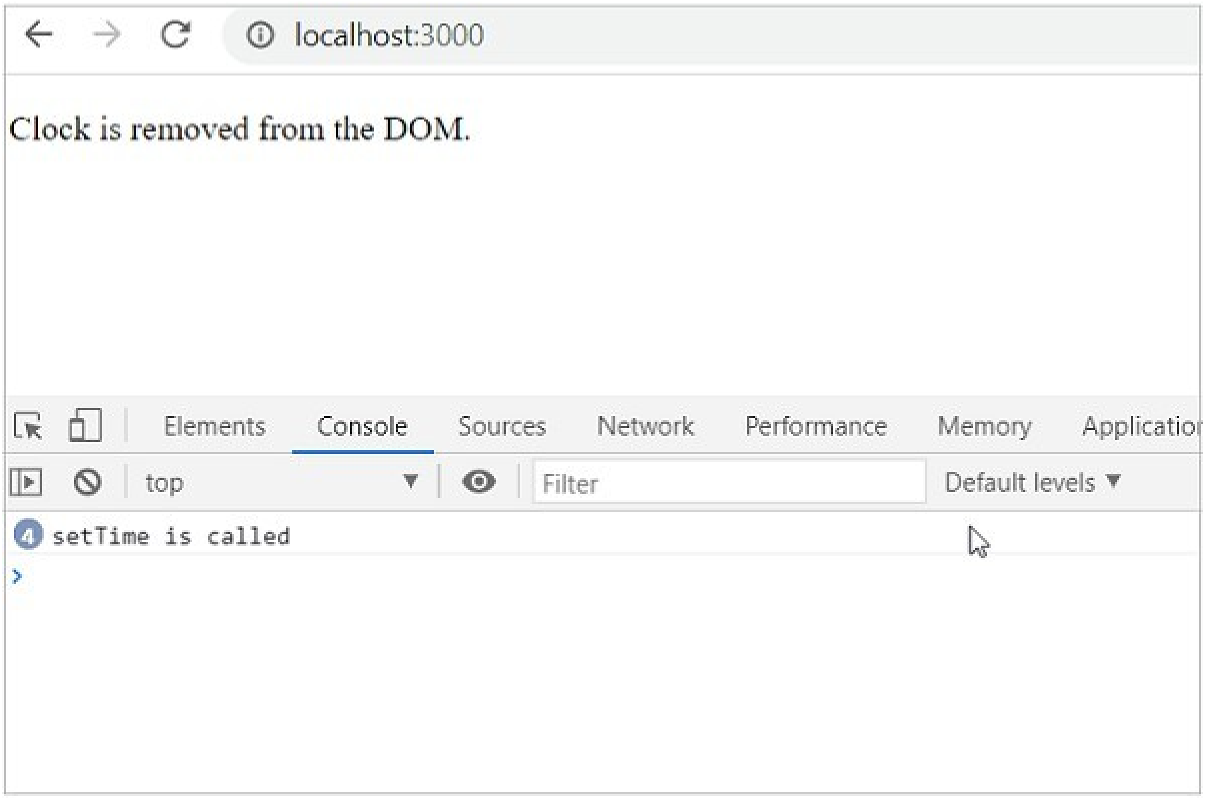
```
import React from 'react'; import
ReactDOM from 'react-dom'; import
Clock from './components/Clock';

ReactDOM.render(
   <React.StrictMode>
      <Clock />
</React.StrictMode>,
document.getElementById('root')
);
setTimeout(() => {
   ReactDOM.render(
      <React.StrictMode>
         <div><p>Clock is removed from the DOM.</p></div>
      </React.StrictMode>,
document.getElementById('root')
   );
}, 5000);
```

Next, serve the application using npm command.

```
npm start
```

Next, open the browser and enter *http://localhost:3000* in the address bar and press enter.

The clock will be shown for 5 seconds and then, it will be removed from the DOM. By checking the console log, we can found that the cleanup code is properly executed.



Clock is removed from the DOM.

```
4 setTime is called
>
```