Lab: Creating Functional React Component using Hook

CN5006

Prepared by Dr.Nadeem Qazi

University of East London

Objectives:

1. To create functional components and implement state management using Hook.
    a. Develop a simple click counter Component
    b. Develop an Emoji Counter for Likes, Love, and Sad emotion

You are Encouraged to read the file HoopAPI.pdf, which is provided in Moodle, to better understand the code used in this lab.

Task1 :

Create a functional component for counting clicks. The user interface of the component would just have one button and heading. The component would record number of the clicks in the state variable of the button component, and will increment as on every click of the button.

Create a React application as you did last week.

Instructions: Setting Up a New React Project with Hooks

1. **Create a New Folder**
    - Begin by creating a folder named `HookAPI` in your desired location.
2. **Open Visual Studio Code**
    - Launch Visual Studio Code, then open the `HookAPI` folder within it.
3. **Initialize a New React Project**
    - Open the terminal in Visual Studio Code (you can do this by selecting **View > Terminal**).
    - In the terminal window, run the following command to create a new React application:

      ```
      npx create-react-app Hooksexamples
      ```

    - **Note:** You can choose any name instead of `Hooksexamples` if desired.
4. **Navigate to the Project Directory**
    - After the React app has been created, switch to the project directory by entering the following command in the terminal:

      ```
      cd Hooksexamples
      ```

Summary

- Create a folder named `HookAPI`.
- Open Visual Studio Code and navigate to this folder.
- Run the `npx create-react-app` command to initialize a new React project.
- Navigate into the new project directory using `cd`.

After completing these steps, your new React project will be ready for development.

Coding Instructions: Setting Up and Using React State with useState Hook

Step 1: Create a New Component File

1. In the src folder of your project, create a new file named **Counter.js.**
2. Right-click on the src folder, select "New File," and enter Counter.js as the file name.
3. You may see a prompt to select a language; choose **JavaScript**.
4. Refer to the video in the Week 5 lab tutorial for additional guidance on this step.

Step 2: Import React and Other Dependencies

- At the top of Counter.js, include the following code to import the required dependencies:

```
Copy code
import React from "react";
import "./App.css";
import { useState } from 'react';
```

Step 3: Create and Implement the Counter Component

1. Define a function named Hook_ControlledButtonState within Counter.js to set up a component that uses the useState hook.
2. **Initialize the State**:
   - Declare a useState Hook to create a state variable called count and a function setCount to manage it. Initialize count to 0.
   - Code example:

```
Copy code
const [count, setCount] = useState(0);
```

3. **Create a Click Event Handler**:
   - Define a function named `ClickHandle` that will increment `count` by 1 each time the button is clicked.
   - Code example:

```
Copy code
const ClickHandle = () => {
  setCount(count + 1);
};
```

4. **Add a Form with a Button**:

- Inside the component's `return` statement, create a `<form>` element with a button.
- Set the button's `onClick` attribute to call the `ClickHandle` function, and display the `count` value inside the button text.
- Code for the button:

```
<button type="button" onClick={ClickHandle}>
  Click me {count}
</button>
```

The complete code for this is given below for Counter.js

```
import React from "react"
import "./App.css"
import { useState } from 'react';

function Hook_ControlledButtonState()
{
   const ClickHandle=() =>
   {
     setCount(count + 1)
   }
  const [count, setCount] = useState(0);
  return (
   <div className="App-header">
   <form>
     <h1>Click Counts are {count}</h1>
```

```
      <button type= "button" onClick={ClickHandle}>Click me{count} <
/button>
    </form>
      </div>
      );
}
export default Hook_ControlledButtonState
```

Save the file, then open index.js and add the following code:

`import Hook_ControlledButtonState from './StatewithHookButton'`
and add following code

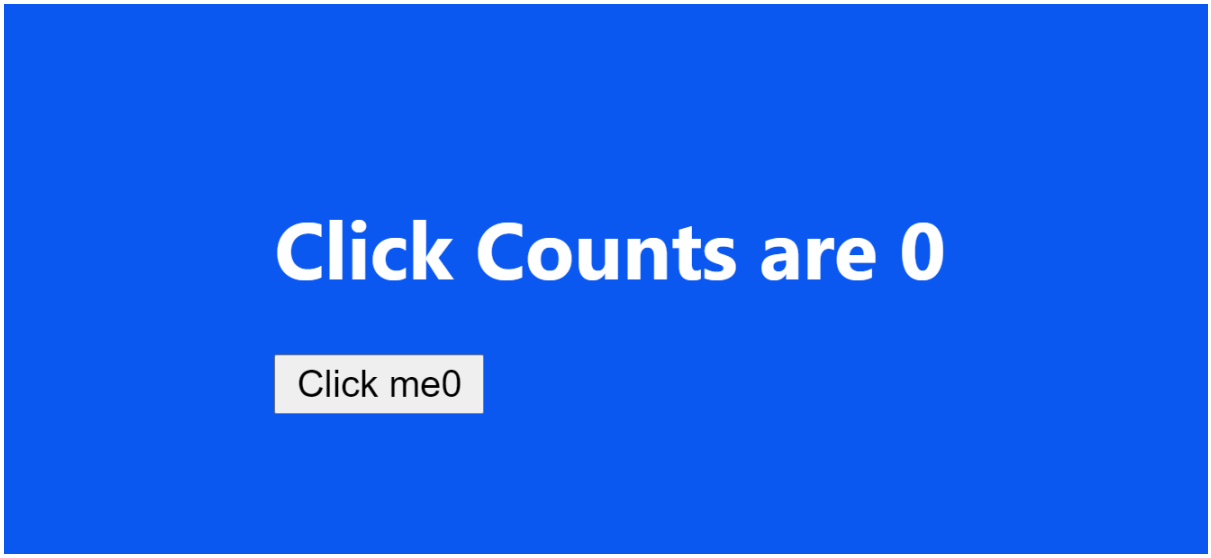
```
<React.Fragment>
   <Hook_ControlledButtonState/>
<React.element/>
```

Your code in the index,js file should look like this.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import * as serviceWorker from './serviceWorker';
import Hook_ControlledButtonState from './Counter'
ReactDOM.render(
  <React.Fragment>
   <Hook_ControlledButtonState/>
   </React.Fragment>
   ,
   document.getElementById('root')
   );
// If you want your app to work offline and load faster, you can change
// Learn more about service workers: https://bit.ly/CRA-PWA
```

```
serviceWorker.unregister();
```

Now run this in visual studio code . using npm start
You will see something like this



Click button and you will see counter working

Task2 : Emoji Counter

This task is similar to the task 2 however in this task we will add an image for **three emojis Love,Like,sad emotions.** In doing so we will be using <img> tag in the our functional component. The use of IMG tag is slightly different in React.

When you want to add a picture to a website you would write a simple line such as this
<img src="Love.gif" alt="Love">

The name of the tag is "img", the "src" attribute is the physical location of the file relative to where our current file is and "alt" attribute is a short description of the picture that is with screen readers. That is the basic structure of an image tag. With React, the same structure applies, there are just two changes between how you would add images with html and how you add pictures with React.

You need to import the picture into the React component. If the image is in the same folder as React component, your import would look something like this.

import Love from "./love.png";

This will import the file logo.png and reference in the file as "Love. Now when you reference the img tag in project you want it to now look like this

Copy the files Love.png,sad.png, Like.png into the src folder of your project. Make sure you can see the names of these file in the src folder after you copied these files.

In this task we will demonstrate the use of props and state in the functional component.

1. The functional component is named as EmojeeCounter, with following syntax definition :
   Function EmojeeCounter(props)
   {
   //code here
   }
2. it has a property object props which has a property named as pic which renders the given image on the button. This pic property will be passed to this functional component when we declare this component in the index.js as : for example creating three components each having different images.
   <EmojeeCounter pic='Love'/>
   <EmojeeCounter pic='sad'/>

a. <EmojeeCounter pic='Like'/>

3 Create the state to save the name of the picture in the state variable using the following lines:

const [pic, setPic]=useState(Love)

the state variable pic is initialized with the default pic Love. The state variable pic will be updated through the state function setPic
Another state count to record the number of the button click

const [count,setCount]=useState(0)

4. Next we define the userEffect, the purpose here is to choose the correct pic according the given property of the function. This function will be executed every time when the property is changed and when the component is first used. So it acts as a initializer for our functional component.

```
useEffect(()=>{
console.log ("function called",props.pic)
if (props.pic==="Love")
 setPic(Love)
else if   (props.pic==="Like")
 setPic(Like)
else if  (props.pic==="sad")
 setPic(Sad)
})
```

Then we create an arrow function called as ClickHandle, it will be assigned to the click event of the button . This function will increment the number of the click each time the button is clicked in the component.

```
const ClickHandle=() =>
{
setCount(count+1)
}
```

The arrow function is assigned to the button in the functional component using the command

```
<button onClick={ClickHandle}>{count }
   <img src={pic} alt=""/>
```

Note the use of , the {pic} is the state variable and holds the value of the image to be drawn on button. The value the pic is set using the useEffect based on the property of the component set in the index,js

Task: Emoji Counter

In this task, you'll create an **Emoji Counter** component similar to Task 2, but now with images representing three emojis: Love, Like, and Sad. We'll use the <img> tag in our React functional component to display these emojis.

In React, using the <img> tag has a few differences compared to HTML. Here's how it works:

1. Normally, in HTML, you might include an image like this:

```
html
<img src="Love.gif" alt="Love">
```

   - `<img>` is the tag.
   - `src` specifies the file location.
   - `alt` provides a description for accessibility.
2. In React, you'll need to import the image into the component file before using it.

Steps to Set Up the Emoji Counter Component

1. **Prepare the Images**:
   - Copy the image files `Love.png`, `Sad.png`, and `Like.png` into the `src` folder of your project.
   - Make sure you can see these files listed in the `src` folder.
2. **Create a Functional Component**:
   - Name the component `EmojiCounter`. Here's the basic syntax:

```
javascript
function EmojiCounter(props) {
  // Code here
}
```

3. **Import Images**:
   - At the top of the component file, import the images:

```
javascript
import Love from "./Love.png";
import Sad from "./Sad.png";
import Like from "./Like.png";
```

4. **Define Props for Dynamic Image Selection**:
   - The component will receive a `pic` prop to specify which image to display. For example:

```
javascript

<EmojiCounter pic="Love" />
<EmojiCounter pic="Sad" />
<EmojiCounter pic="Like" />
```

5. **Initialize State Variables**:

- Define two state variables: `pic` for the image, initialized to `Love`, and `count` for the button clicks, initialized to `0`:

```
javascript
const [pic, setPic] = useState(Love);
const [count, setCount] = useState(0);
```

6. **Use `useEffect` to Set the Image Dynamically**:
   - Add a `useEffect` hook to update the `pic` state based on the `pic` prop. This function will run whenever the `pic` prop changes:

```
javascript
useEffect(() => {
  if (props.pic === "Love") {
    setPic(Love);
  } else if (props.pic === "Like") {
    setPic(Like);
  } else if (props.pic === "Sad") {
    setPic(Sad);
  }
}, [props.pic]);
```

7. **Define the Click Handler**:
   - Create an arrow function called `ClickHandle` that increments `count` each time the button is clicked:

```
javascript
Copy code
const ClickHandle = () => {
  setCount(count + 1);
};
```

8. **Render the Button with Image and Click Counter**:
   - In the component's return statement, add a button with an `onClick` event that triggers `ClickHandle`:

```
javascript
Copy code
return (
  <button onClick={ClickHandle}>
    {count}
    <img src={pic} alt="" />
  </button>
);
```

   - Here, `{pic}` is the state variable holding the image source, and `useEffect` updates it based on the `pic` prop.

Complete Code for `EmojiCounter` Component

Now, you can put all the above code into a separate file to create the functional component. Follow this steps

Step 1 : create a file **EmojeeCounters.js ,**write the code in this file which is given below:

```
import React, { useState ,useEffect} from "react";
import Love from './Love.png'
import Sad from './sad.png'
import Like from './like.png'
function EmojeeCounter(props){
console.log("pic is ",props.pic)

const [pic, setPic]=useState(Love)
const [count,setCount]=useState(0)
useEffect(()=>{
console.log ("function called",props.pic)
if (props.pic==="Love")
  setPic(Love)
else if   (props.pic==="Like")
  setPic(Like)
else if  (props.pic==="sad")
  setPic(Sad)
})
const ClickHandle=() =>
   {
    setCount(count+1)
   }
return (
<div className="App">
    <p>{props.pic} <span></span>
    <button onClick={ClickHandle}>{count }
    <img src={pic} alt=""/>
```

```
    </button>
  </p>
</div>
)
}
export default EmojeeCounter;
```

Step 2: call this functional component in index.js using the two lines

i) Import the functional component using the following code:

import EmojeeCounter from './EmojeeCounters'

ii.) Add following code in ReactDOm.render:

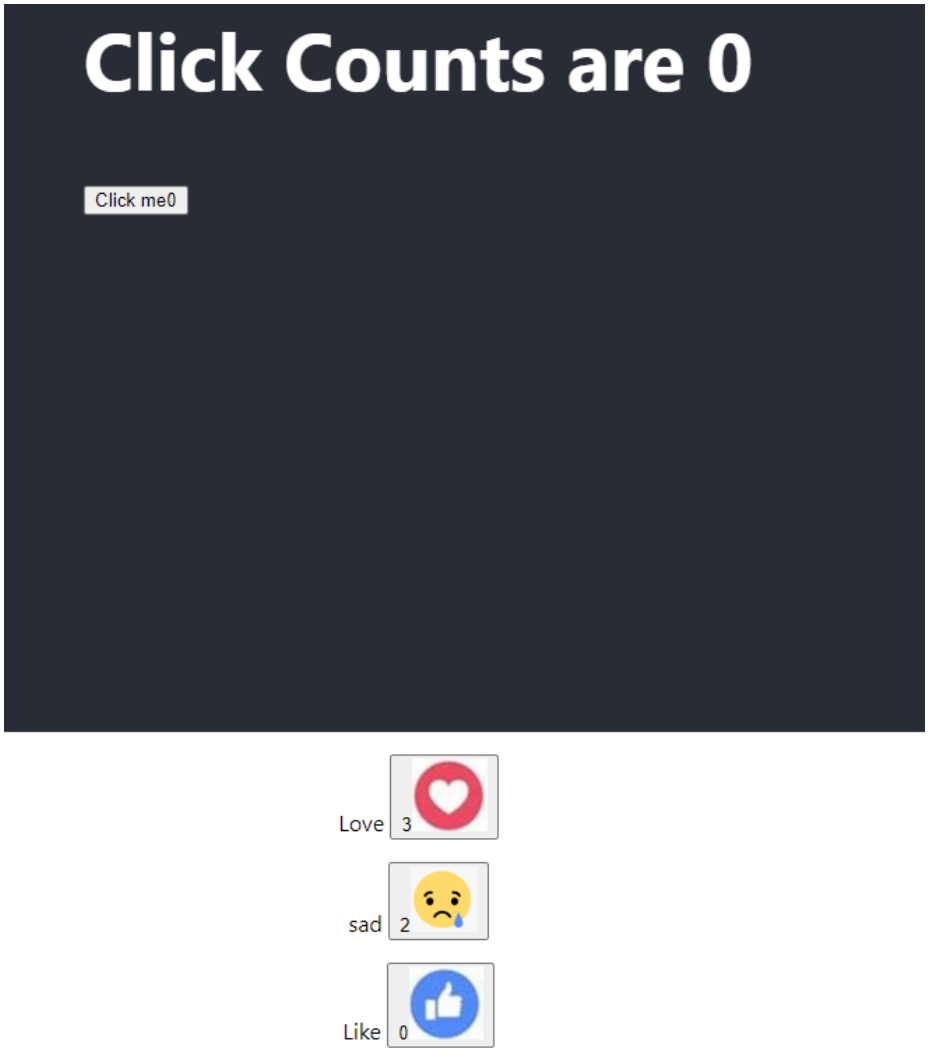
ReactDOM.render(
<React.StrictMode>
<Hook_ControlledButtonState/>
<EmojeeCounter pic='Love'/>
<EmojeeCounter pic='sad'/>
<EmojeeCounter pic='Like'/>
</React.StrictMode>,.....

Note now in your index.js file you have used two functional components Hook_ControlledButtonState which you created in the task 1 and three separate components of Emjoee Counter each having different image. The full listing of the index.js is given below : index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

import EmojeeCounter from './EmojeeCounters'
import Hook_ControlledButtonState from ‘./StatewithHookButton’
ReactDOM.render(
  <React.StrictMode>
   <Hook_ControlledButtonState/>
```

```
    <EmojeeCounter pic='Love'/>
    <EmojeeCounter pic='sad'/>
    <EmojeeCounter pic='Like'/>

    </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
```

you will see the following in the browser window



What you need to write for your Portfolio of this Week

Q1 Write one page reflective what did you learn about React Hook API during this week

Q2. Study the code in **EmojeeCounters.js,** Please note, You Do not need to submit the full code rather you need to answer the following questions for your this week portfolio

- What is Name of the Component you have created in **EmojeeCounters.js**
- **Identify the line of code that uses the EmojeeCounter in index.js**
- Declares the states of each of the html elements defined in the **EmojeeCounters.js** (identify these lines and explain only those lines)
- Lines of codes that are used to associate the event handler used.
- Explain the line : <EmojeeCounter pic='Love'/> , what is pic=’Love’ means in this line.
- What is useEffect and why you think we have used it in the Component.
- Explain these line of the codes in functional component EmojeeCounter.js:

  return (

```
<div className="App">
  <p>{props.pic} <span></span>
  <button onClick={ClickHandle}>{count }
  <img src={pic} alt=""/>
  </button>
 </p>
</div>
)
}
```

Q3

Create a code for a Component that takes two HTML one text box and one label. Label will be used to display the images. So it should be like this

If I write “Happy” in the text box the label should show happy face (You can use any image)

If I write “Like” in the text box the label should show Like icon

If I write “sad “ the label should show sad emoji.

Run this component take the screen shot of your newly run component and write a paragraph how did you develop this component