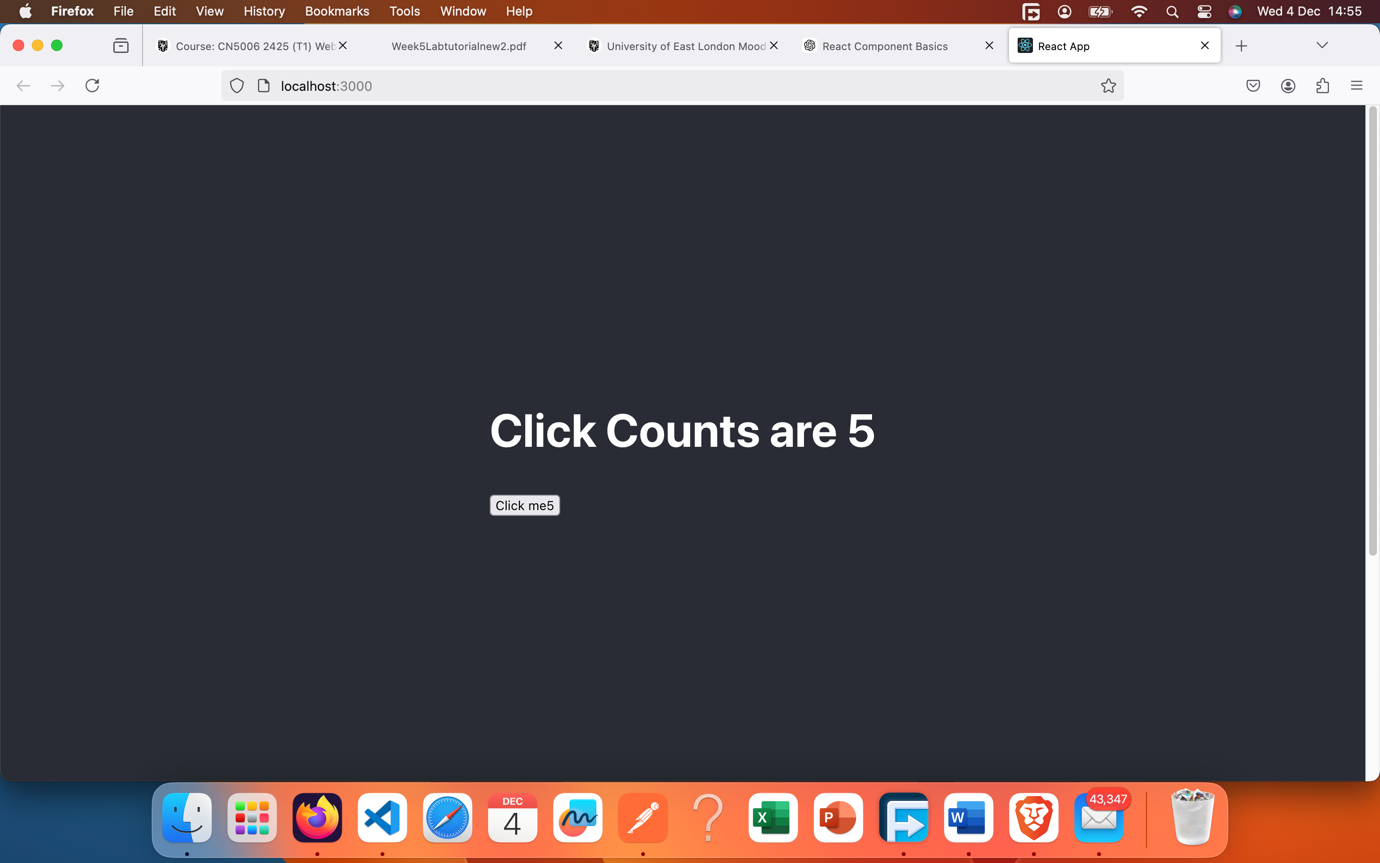
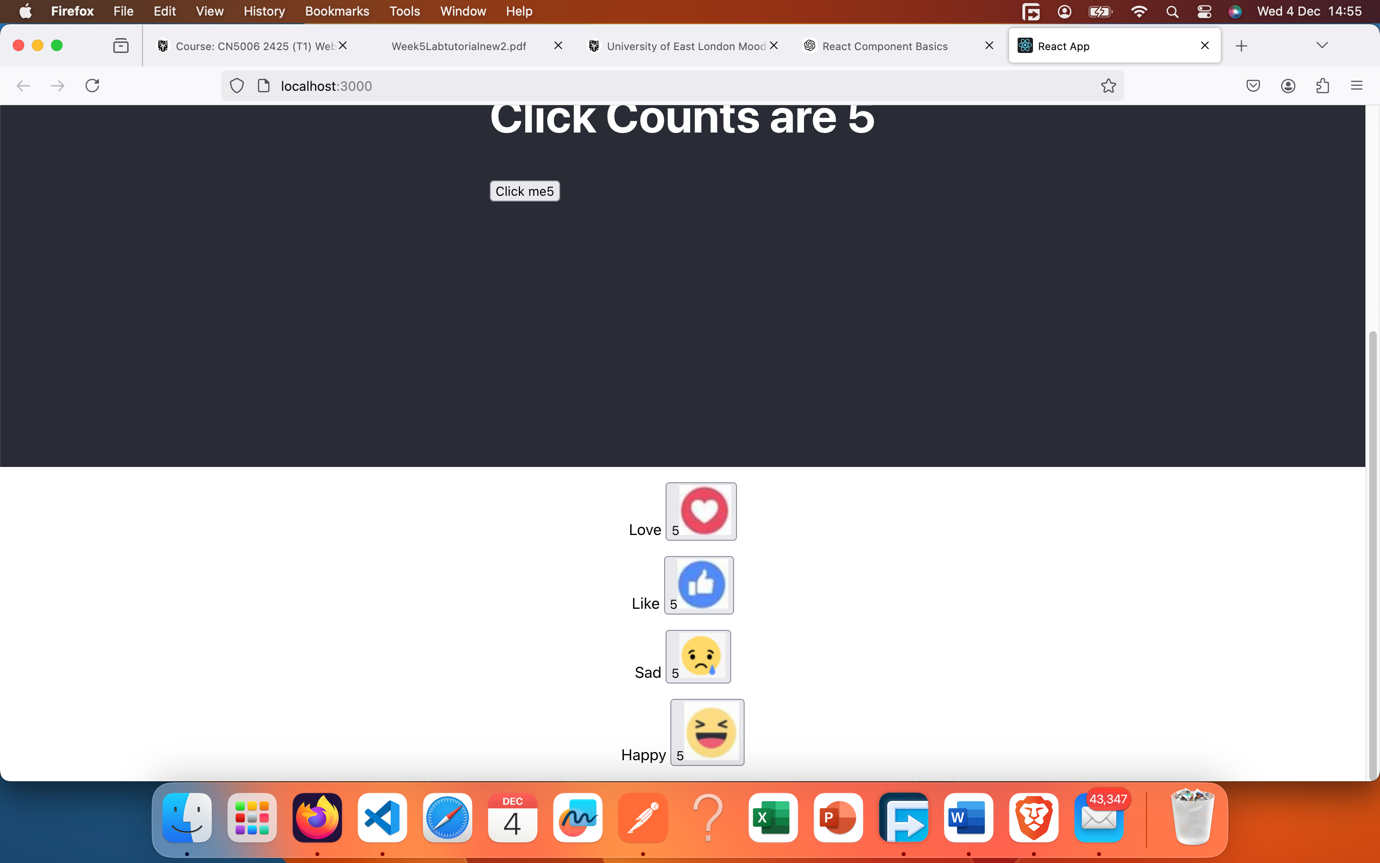
Week 7





**What is the Name of the Component you have created in EmojeeCounters.js?**

* The component is named **EmojiCounter**.

**Identify the line of code that uses the EmojiCounter in index.js.**

Typically, the usage of EmojiCounter in index.js would look something like:

<EmojiCounter pic="Love" />

This line renders the EmojiCounter component, passing pic as a prop with the value "Love".

**Declares the states of each of the HTML elements defined in the EmojiCounter.js:**

* State declarations:

const [pic, setPic] = useState(Love);

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

* + pic: This state is used to store the current emoji image (e.g., Love, Sad, etc.). Its initial value is the Love emoji image.
  + count: This state tracks the number of clicks or interactions. It starts at 0.

**Lines of code that are used to associate the event handler:**

* The event handler association happens here:

<button onClick={ClickHandle}>{count}

<img src={pic} alt="" />

</button>

* The onClick event is associated with the ClickHandle function.

**Explain the line: <EmojiCounter pic="Love"/>. What does pic='Love' mean in this line?**

* pic="Love" is a **prop** passed to the EmojiCounter component.
  + It determines the emoji to display. In this case, the emoji associated with "Love" is shown.
  + Inside the component, props.pic is used to check the value and dynamically set the displayed emoji.

**What is useEffect and why do you think we have used it in the Component?**

* **useEffect** is a React Hook that lets you perform side effects in a function component.
* **Usage in this 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);

else if (props.pic === "Happy") setPic(Happy);

}, [props.pic]);

* + useEffect is used to update the emoji image (pic) whenever the props.pic value changes.
  + It ensures that the displayed image reflects the correct emoji based on the pic prop.

**Explain these lines of code in the functional component EmojiCounter.js:**

<div className="App">

<p>

{props.pic} <span></span>

<button onClick={ClickHandle}>

{count}

<img src={pic} alt="" />

</button>

</p>

</div>

**Explanation:**

* props.pic: Displays the name of the emoji type (e.g., "Love").
* <button onClick={ClickHandle}>: Adds an event listener to the button. When clicked, ClickHandle increments the count.
* {count}: Shows the current count of button clicks.
* <img src={pic} alt="" />: Displays the emoji image based on the current pic state.

### **Q3:Code for Component with Textbox and Label**

#### ****Code:****

#### import React, { useState } from "react";

#### import HappyImg from "./happy.png";

#### import LikeImg from "./like.png";

#### import SadImg from "./sad.png";

#### function EmojiDisplay() {

#### const [input, setInput] = useState("");

#### const [image, setImage] = useState("");

#### const handleChange = (event) => {

#### setInput(event.target.value);

#### if (event.target.value === "Happy") setImage(HappyImg);

#### else if (event.target.value === "Like") setImage(LikeImg);

#### else if (event.target.value === "Sad") setImage(SadImg);

#### else setImage("");

#### };

#### return (

#### <div>

#### <input type="text" onChange={handleChange} placeholder="Type emoji name..." />

#### <label>

#### {image && <img src={image} alt="emoji" />}

#### </label>

#### </div>

#### );

#### }

#### export default EmojiDisplay;

 Created a useState hook to manage the text input and image states.

 Added an onChange event to the input field to track the user's input.

 Used conditional logic to map the input value (e.g., "Happy", "Sad") to a corresponding emoji image.

 Dynamically displayed the image in the label using JSX.

 Tested the component by typing various emoji names and ensured the correct image was displayed.