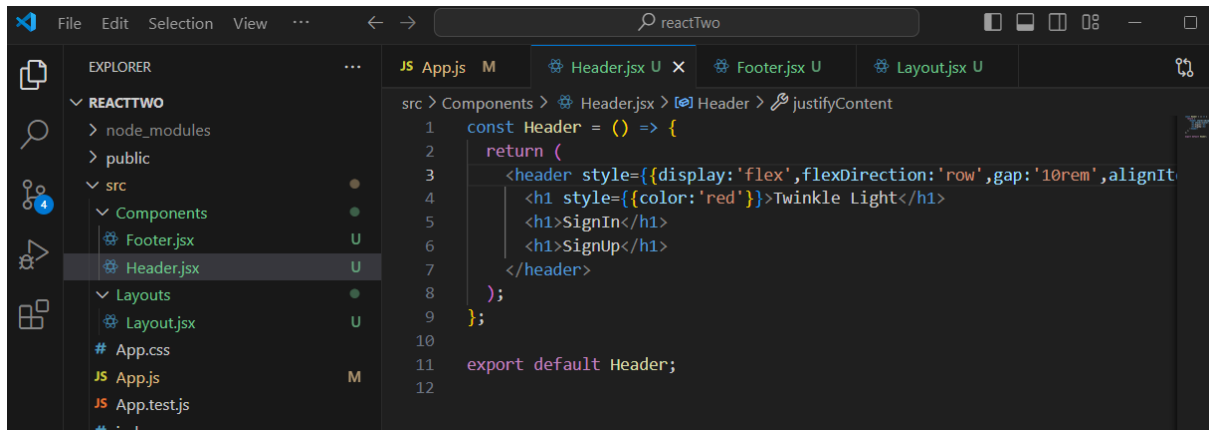


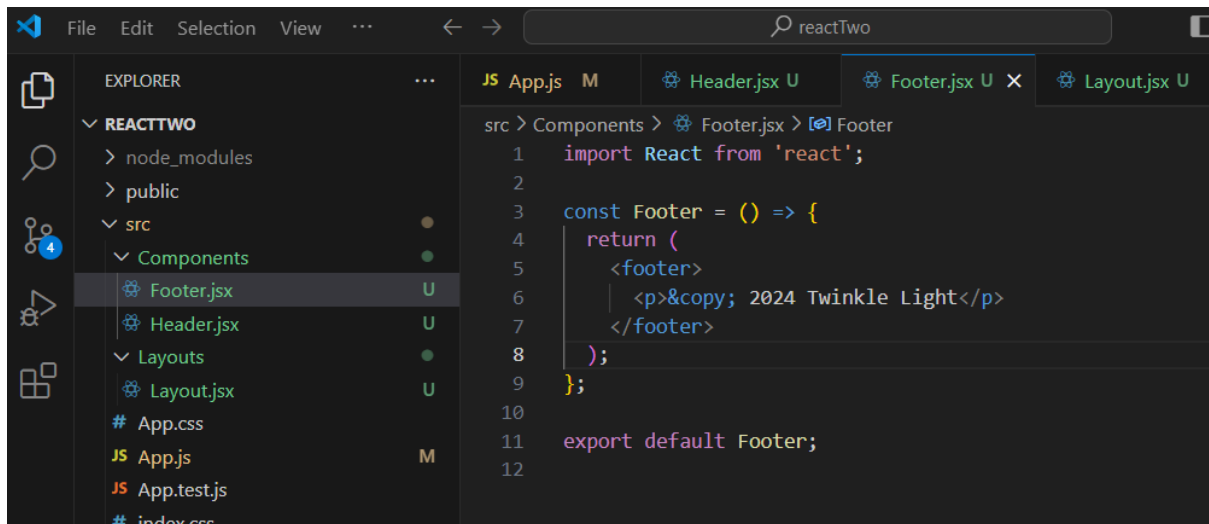
React - 2

Q1. Create two components, `Header` and `Footer`. Use them inside a `Layout` component to build a basic page layout.



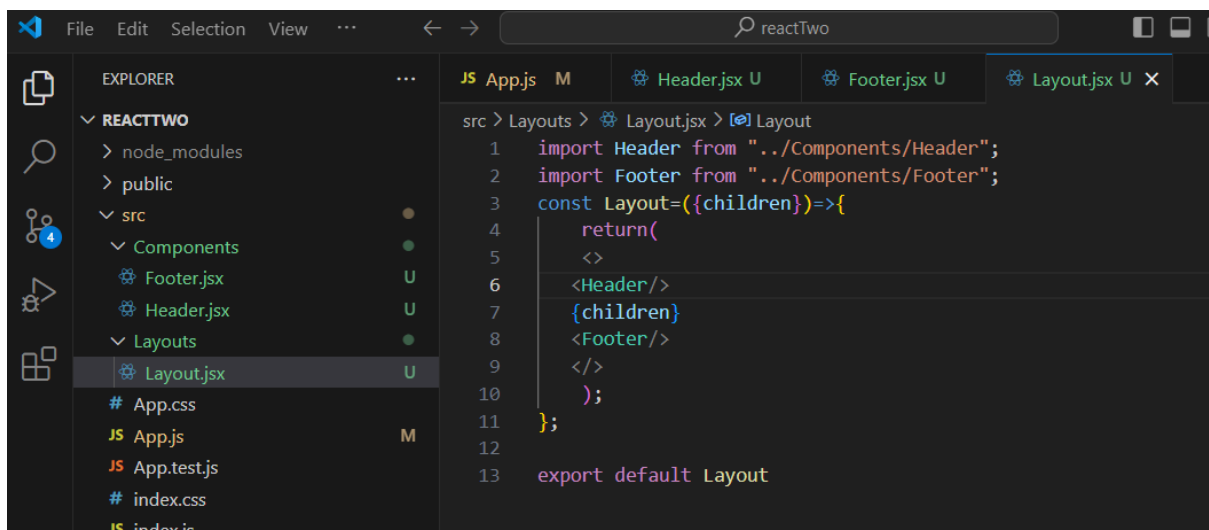
This screenshot shows the VS Code editor with the `Header.js` file open. The Explorer sidebar on the left shows the project structure: `REACTTWO` with subfolders `node_modules`, `public`, and `src`. Inside `src`, there are folders `Components` and `Layouts`, and files `App.css`, `App.js`, `App.test.js`, and `index.css`. The `Header.js` file is selected in the Explorer. The main editor area shows the following code:

```
src > Components > Header.js > Header > justifyContent
1  const Header = () => {
2      return (
3          <header style={{display:'flex',flexDirection:'row',gap:'10rem',alignIt
4              <h1 style={{color:'red'}}>Twinkle Light</h1>
5              <h1>SignIn</h1>
6              <h1>SignUp</h1>
7          </header>
8      );
9  };
10
11  export default Header;
12
```



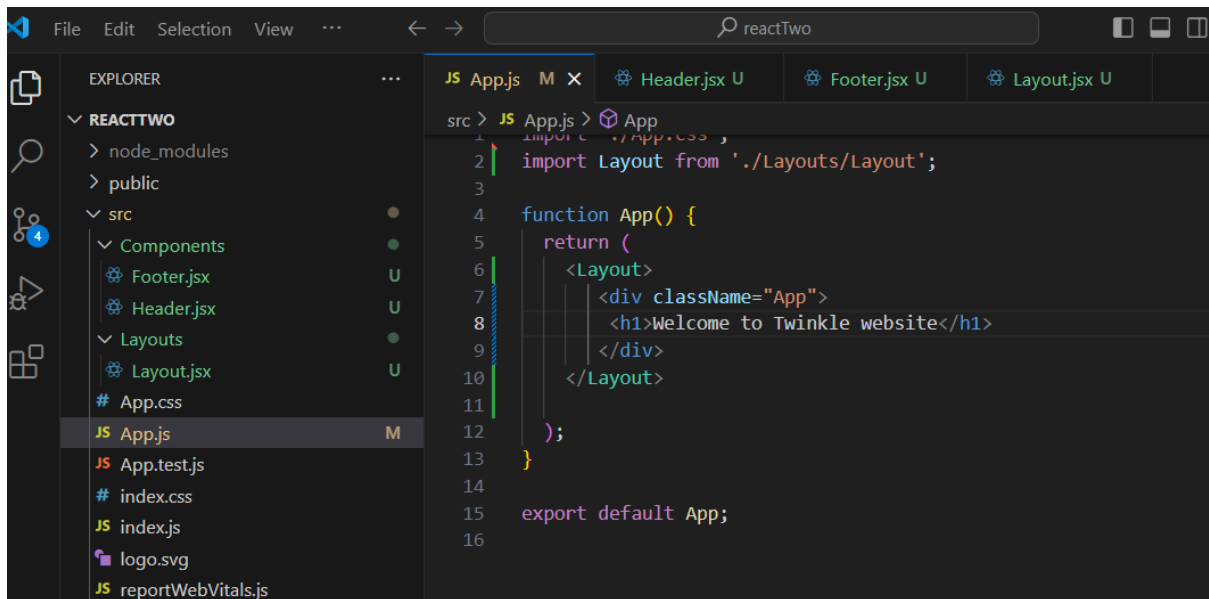
This screenshot shows the VS Code editor with the `Footer.js` file open. The Explorer sidebar on the left shows the project structure, with `Footer.js` selected in the `Components` folder. The main editor area shows the following code:

```
src > Components > Footer.js > Footer
1  import React from 'react';
2
3  const Footer = () => {
4      return (
5          <footer>
6              <p>&copy; 2024 Twinkle Light</p>
7          </footer>
8      );
9  };
10
11  export default Footer;
12
```



This screenshot shows the VS Code editor with the `Layout.js` file open. The Explorer sidebar on the left shows the project structure, with `Layout.js` selected in the `Layouts` folder. The main editor area shows the following code:

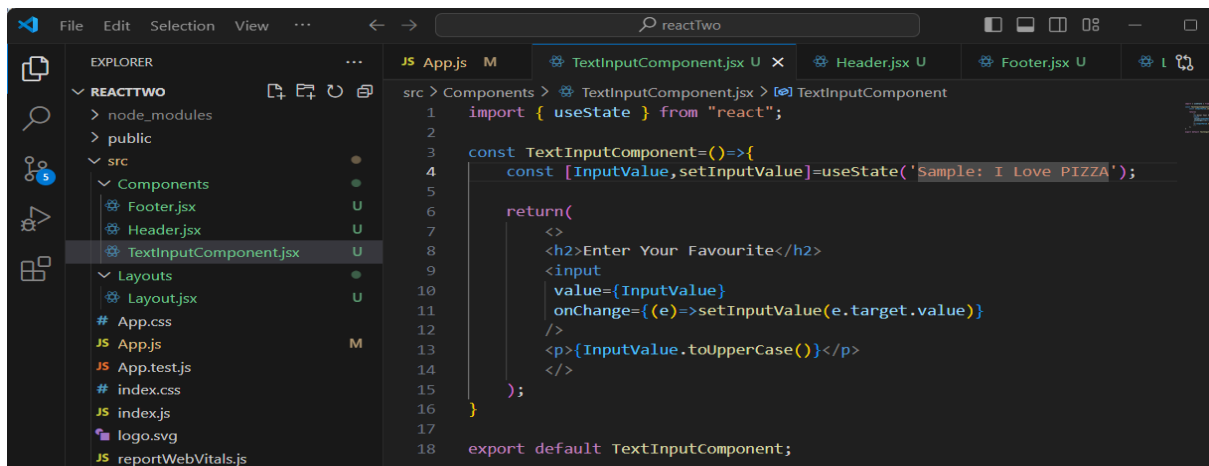
```
src > Layouts > Layout.js > Layout
1  import Header from "../Components/Header";
2  import Footer from "../Components/Footer";
3  const Layout=({children})=>{
4      return(
5          <>
6              <Header/>
7              {children}
8              <Footer/>
9          </>
10      );
11  };
12
13  export default Layout
```



The screenshot shows the VS Code editor with the Explorer sidebar on the left. The project is named 'reactTwo'. The Explorer shows a file tree with 'src' containing 'Components' (Footer.jsx, Header.jsx), 'Layouts' (Layout.jsx), and 'App.css'. The main editor displays 'App.js' with the following code:

```
1 import './App.css';
2 import Layout from './Layouts/Layout';
3
4 function App() {
5   return (
6     <Layout>
7       <div className="App">
8         <h1>Welcome to Twinkle website</h1>
9       </div>
10    </Layout>
11  );
12 }
13
14 export default App;
```

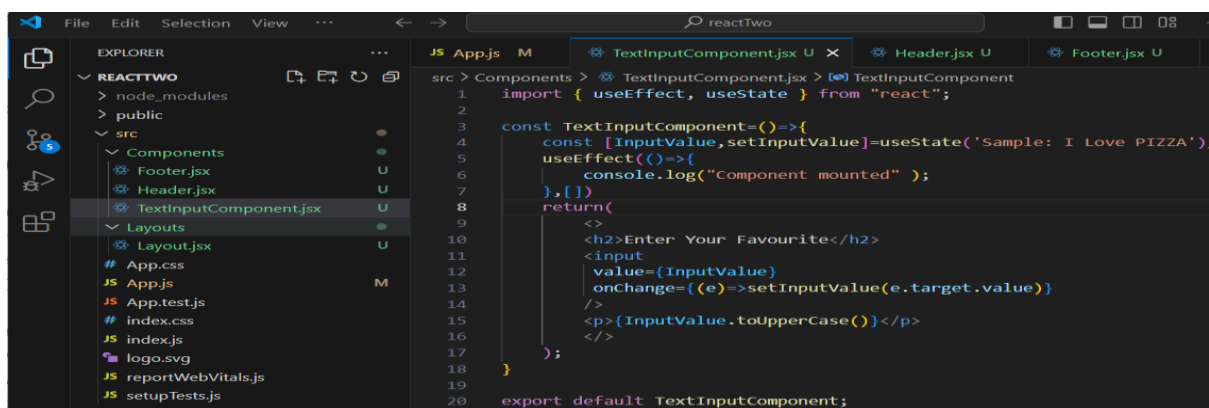
Q2. Create a component with a text input field that updates the displayed text below the input field as the user types.



The screenshot shows the VS Code editor with the Explorer sidebar. The Explorer shows the file tree with 'src' containing 'Components' (Footer.jsx, Header.jsx, TextInputComponent.jsx), 'Layouts' (Layout.jsx), and 'App.css'. The main editor displays 'TextInputComponent.jsx' with the following code:

```
1 import { useState } from "react";
2
3 const TextInputComponent=()=>{
4   const [InputValue,setInputValue]=useState('Sample: I Love PIZZA');
5
6   return(
7     <>
8       <h2>Enter Your Favourite</h2>
9       <input
10         value={InputValue}
11         onChange={(e)=>setInputValue(e.target.value)}
12       />
13       <p>{InputValue.toUpperCase()}</p>
14     </>
15   );
16 }
17
18 export default TextInputComponent;
```

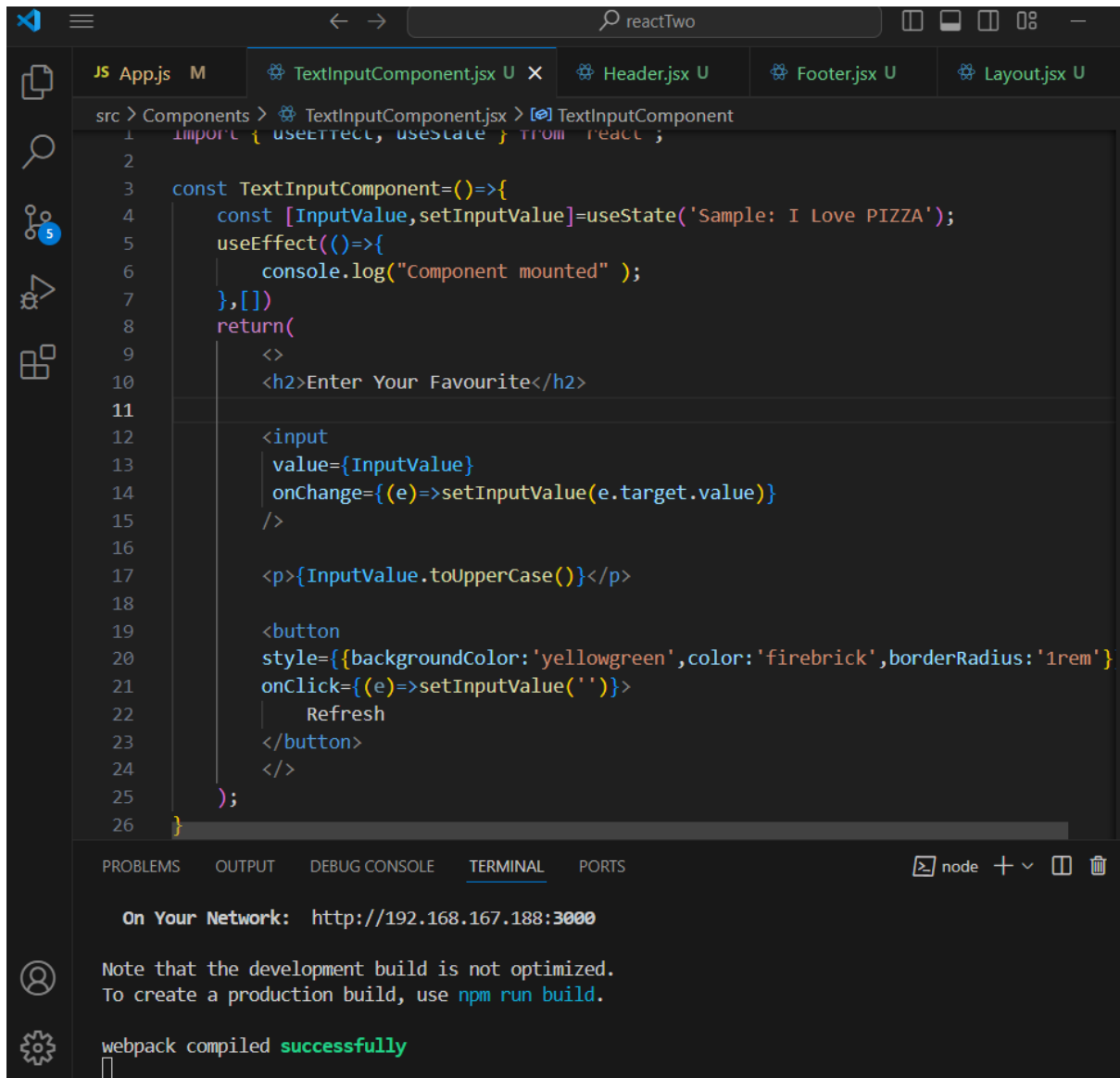
Q3. Write a component that uses `useEffect` to log "Component mounted" to the console when the component is first rendered.



The screenshot shows the VS Code editor with the Explorer sidebar. The Explorer shows the file tree with 'src' containing 'Components' (Footer.jsx, Header.jsx, TextInputComponent.jsx), 'Layouts' (Layout.jsx), and 'App.css'. The main editor displays 'TextInputComponent.jsx' with the following code:

```
1 import { useEffect, useState } from "react";
2
3 const TextInputComponent=()=>{
4   const [InputValue,setInputValue]=useState('Sample: I Love PIZZA');
5   useEffect(()=>{
6     console.log("Component mounted" );
7   },[])
8   return(
9     <>
10       <h2>Enter Your Favourite</h2>
11       <input
12         value={InputValue}
13         onChange={(e)=>setInputValue(e.target.value)}
14       />
15       <p>{InputValue.toUpperCase()}</p>
16     </>
17   );
18 }
19
20 export default TextInputComponent;
```

Q4. Create a component with a button that has inline styles applied to change its background color and text color.



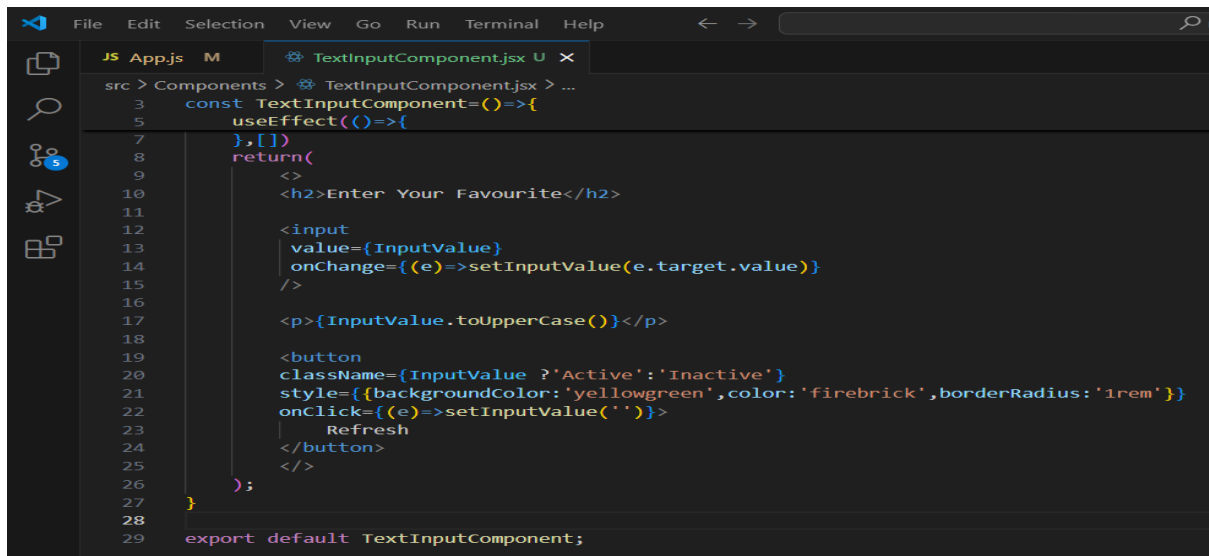
The screenshot shows a VS Code editor with a project named 'reactTwo'. The file explorer on the left shows a directory structure with 'Components' containing 'TextInputComponent.jsx'. The editor is open to 'TextInputComponent.jsx', which contains the following code:

```
1 import { useEffect, useState } from 'react';
2
3 const TextInputComponent=()=>{
4   const [InputValue,setInputValue]=useState('Sample: I Love PIZZA');
5   useEffect(()=>{
6     console.log("Component mounted" );
7   },[])
8   return(
9     <>
10      <h2>Enter Your Favourite</h2>
11
12      <input
13        value={InputValue}
14        onChange={(e)=>setInputValue(e.target.value)}
15      />
16
17      <p>{InputValue.toUpperCase()}</p>
18
19      <button
20        style={{backgroundColor:'yellowgreen',color:'firebrick',borderRadius:'1rem'}}
21        onClick={(e)=>setInputValue('')}>
22        Refresh
23      </button>
24    </>
25  );
26 }
```

The terminal at the bottom shows the following output:

```
On Your Network: http://192.168.167.188:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
```

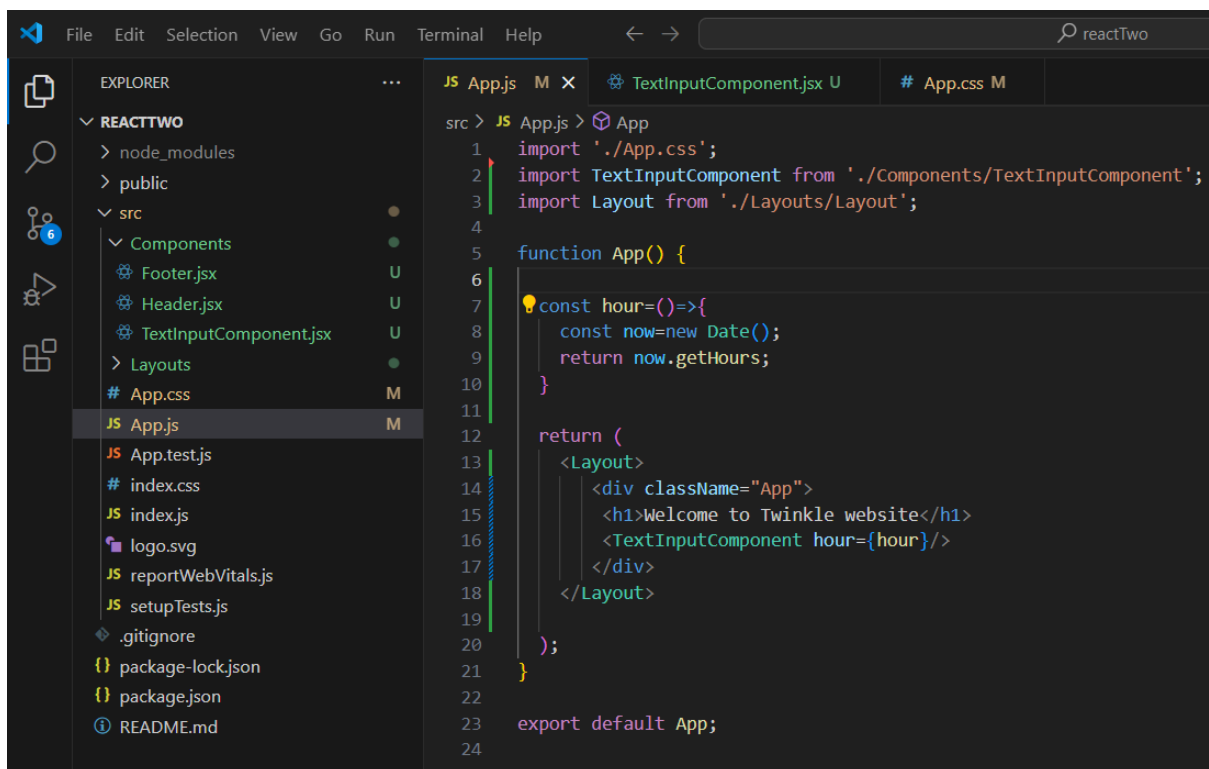
Q5. Create a component that applies a class name based on a boolean state. For example, apply the class "active" if the state is true, and "inactive" if false.



The screenshot shows the VS Code editor with the file `TextInputComponent.jsx` open. The code defines a functional component that uses `useState` and `useEffect` to manage an input field and a refresh button.

```
src > Components > TextInputComponent.jsx > ...
3  const TextInputComponent=()=>{
5    useEffect(()=>{
7      },[])
8    },
9    return(
10     <>
11       <h2>Enter Your Favourite</h2>
12
13       <input
14         value={InputValue}
15         onChange={(e)=>setInputValue(e.target.value)}
16       />
17
18       <p>{InputValue.toUpperCase()}</p>
19
20       <button
21         className={InputValue ?'Active':'Inactive'}
22         style={{backgroundColor:'yellowgreen',color:'firebrick',borderRadius:'1rem'}}
23         onClick={(e)=>setInputValue('')}>
24         Refresh
25       </button>
26     </>
27   );
28 }
29 export default TextInputComponent;
```

Q6. Create a parent component that passes a function as a prop to a child component. The child component should call this function when a button is clicked.



The screenshot shows the VS Code editor with the file `App.jsx` open. The Explorer sidebar on the left shows the project structure. The code in `App.jsx` imports `App.css`, `TextInputComponent`, and `Layout`, and defines the `App` component.

```
src > JS App.js > App
1  import './App.css';
2  import TextInputComponent from './Components/TextInputComponent';
3  import Layout from './Layouts/Layout';
4
5  function App() {
6
7    const hour=()=>{
8      const now=new Date();
9      return now.getHours();
10   }
11
12   return (
13     <Layout>
14       <div className="App">
15         <h1>Welcome to Twinkle website</h1>
16         <TextInputComponent hour={hour}/>
17       </div>
18     </Layout>
19   );
20 }
21
22 export default App;
```

Explorer sidebar:

- REACTTWO
 - node_modules
 - public
 - src
 - Components
 - Footer.jsx
 - Header.jsx
 - TextInputComponent.jsx
 - Layouts
 - App.css
 - App.js
 - App.test.js
 - index.css
 - index.js
 - logo.svg
 - reportWebVitals.js
 - setupTests.js
 - .gitignore
 - package-lock.json
 - package.json
 - README.md

```
1 import { useEffect, useState } from "react";
2
3 const TextInputComponent=({hour})=>{
4   const [InputValue,setInputValue]=useState('Sample: I Love PIZZA');
5   useEffect(()=>{
6     console.log("Component mounted" );
7   },[])
8   return(
9     <>
10      {hour<12? <h1>Good Morning</h1>:<h1>Good Evening</h1> }
11      <h2>Enter Your Favourite</h2>
12
13      <input
14        value={InputValue}
15        onChange={(e)=>setInputValue(e.target.value)}
16      />
17
18      <p>{InputValue.toUpperCase()}</p>
19
20      <button
21        className={InputValue ?'Active':'Inactive'}
22        style={{backgroundColor: 'yellowgreen',color: 'firebrick',borderRadius:'1rem'}}
23        onClick={(e)=>setInputValue('')}>
24        Refresh
25      </button>
26    </>
27  );
28 }
```

Q7. Create a component that accepts a `greeting` prop and uses "Hello" as the default value if no `greeting` is provided.

```
1 import './App.css';
2 import TextInputComponent from './Components/TextInputComponent';
3 import Layout from './Layouts/Layout';
4
5 function App() {
6
7   const hour=()=>{
8     const now=new Date();
9     return now.getHours();
10  }
11
12  return (
13    <Layout>
14      <div className="App">
15        <h1>Welcome to Twinkle website</h1>
16        <TextInputComponent hour={hour} greeting='Hi' />
17      </div>
18    </Layout>
19  );
20 }
21
22 export default App;
```

```
import { useEffect, useState } from "react";

const TextInputComponent=({hour,greeting='Hellow'})=>{
  const [InputValue,setInputValue]=useState('Sample: I Love PIZZA');
  useEffect(()=>{
    console.log("Component mounted" );
  },[])
  return(
    <>
      <h1>{greeting}</h1>
      {hour<12? <h1>Good Morning</h1>:<h1>Good Evening</h1> }
      <h2>Enter Your Favourite Plz</h2>
    </>
  )
}
```

Twinkle Light

SignIn

SignUp

Welcome to Twinkle website

Hi

Good Evening

Enter Your Favourite Plz

Sample: I Love PIZZA

SAMPLE: I LOVE PIZZA

Refresh

Q8. Create a class-based component that logs a message when the component is mounted using `componentDidMount`.

```
src > Components > ClassComponent.jsx > MyComponent > render
1  import React, { Component } from 'react';
2
3  class MyComponent extends Component {
4    componentDidMount() {
5      console.log('Component has mounted.');

```

Q9. Create a component with two input fields and display the values entered in both fields below the inputs.

```
components > TwoInput.jsx > TwoInput
import { useState } from "react";

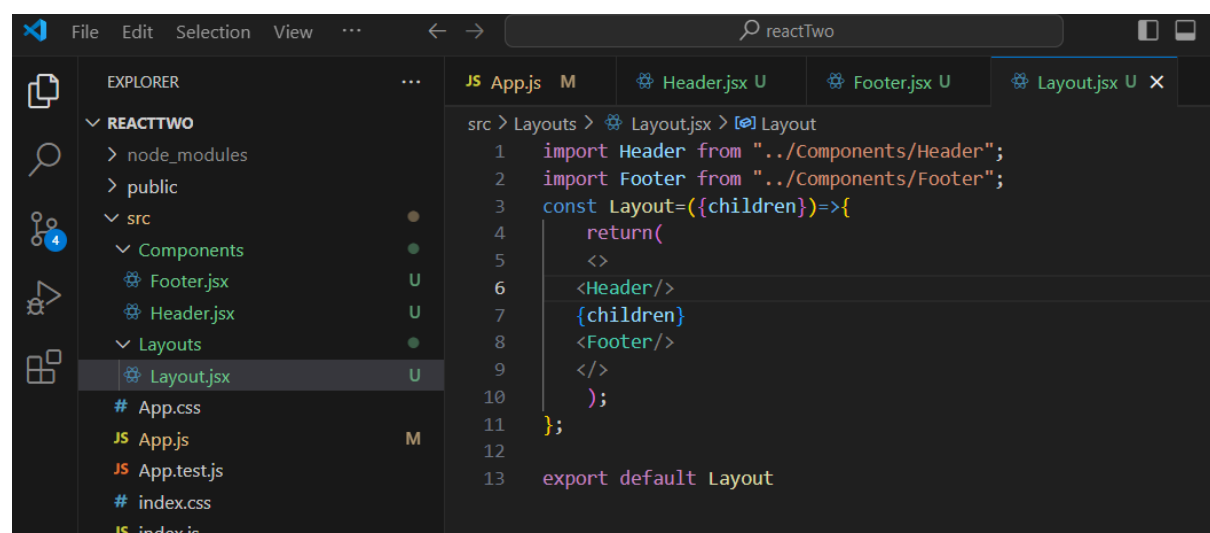
const TwoInput=()=>{

  const [input1, setInput1] = useState('');
  const [input2, setInput2] = useState('');

  return (
    <div>
      <input value={input1} onChange={(e) => setInput1(e.target.value)} />
      <p>{input1}</p>
      <input value={input2} onChange={(e) => setInput2(e.target.value)} />
      <p>{input2}</p>
    </div>
  );
}

export default TwoInput;
```

Q10. Create a component that renders multiple elements without adding extra nodes to the DOM, using `React.Fragment` or the shorthand `<></>`.



The screenshot shows a Visual Studio Code editor with a React project named 'reactTwo'. The Explorer sidebar on the left shows the project structure: 'REACTTWO' with subfolders 'node_modules', 'public', and 'src'. Inside 'src', there are folders 'Components' and 'Layouts'. 'Components' contains 'Footer.jsx' and 'Header.jsx'. 'Layouts' contains 'Layout.jsx'. The main editor area shows the 'Layout.jsx' file with the following code:

```
src > Layouts > Layout.jsx > Layout
1  import Header from "../Components/Header";
2  import Footer from "../Components/Footer";
3  const Layout=({children})=>{
4    return(
5      <>
6        <Header/>
7        {children}
8        <Footer/>
9      </>
10     );
11  };
12
13  export default Layout
```

Q11. Render a list of items and use the `key` prop to uniquely identify each item in the list.

```

1  import React, { Component } from 'react';
2
3  class MyComponent extends Component {
4
5      componentDidMount() {
6          console.log('Component has mounted.');
7      }
8
9      render() {
10         const items = ['Apple', 'Banana', 'Orange'];
11         return (
12             <ol>
13                 {items.map((item,index)=>{
14                     return <li key={index}>{item}</li>
15                 })}
16             </ol>
17         );
18     }
19 }
20
21 export default MyComponent;
22

```

Q12. Create a reusable button component that accepts text and a click handler as props and renders a button with the provided text.

```

src / Components / Button.jsx / default
1  const Button=({text,onClick})=>{
2      return <button onClick={onClick}>{text}</button>
3  }
4  export default Button;

```

```

import Layout from './Layouts/Layout';
import Button from './Components/Button';
function App() {
    const hour=()=>{
        const now=new Date();
        return now.getHours();
    }
    const handleClick=()=>{
        return window.alert('Button Clicked');
    }
    return (
        <Layout>
            <div className="App">
                <h1>Welcome to Twinkle website</h1>
                <TextInputComponent hour={hour} greeting='Hi' />
                <MyComponent />
                <TwoInput />
                <Button text='click Here' onClick={handleclick} />
            </div>
        </Layout>
    );
}
export default App;

```


Twinkle Light

SignIn

SignUp

Welcome to Twinkle website

Hi

Good Evening

Enter Your Favourite Plz

Sample: I Love PIZZA

SAMPLE: I LOVE PIZZA

Refresh

Apple

Banana

Orange

click Here

1.

2.

3.

© 2024 Twinkle Light