

## Lesson 03 Demo 04

### Creating a Component that Re-renders

**Objective:** To create a component in React that re-renders with a prop input

**Tools Required:** Node terminal, React app, and Visual Studio Code

**Prerequisites:** Knowledge of creating a React app and an understanding of the folder structure

Steps to be followed:

1. Create a new React app
2. Configure the src/App.js file
3. Create a new file called Greetings.js
4. Run the app and verify the functionality

#### Step 1: Create a new React app

- 1.1. Open your terminal and run the following command to create a new **React** app:

**npx create-react-app greetings-app**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
shreemayeebhatt@ip-172-31-22-250:~$ npx create-react-app greetings-app
```

- 1.2. Navigate into the newly created project directory by running the following command: **cd greetings-app**

## Step 2: Configure the file src/App.js

2.1 Open the React project in the Visual Studio code and navigate through the project directory of **greetings-app** to open the **src/App.js** file and replace the existing code in **App.js** with the following code:

```
import React, { useState } from 'react';
import Greetings from './Greetings';
```

```
function App() {
  const [name, setName] = useState('John');

  return (
    <div>
      <Greetings name={name} />
      <button onClick={() => setName('Jane')}>Change Name</button>
    </div>
  );
}
```

```
export default App;
```



```
src > Js App.js > ...
1  import React, { useState } from 'react';
2  import Greetings from './Greetings';
3
4  function App() {
5    const [name, setName] = useState('John');
6
7    return (
8      <div>
9        <Greetings name={name} />
10       <button onClick={() => setName('Jane')}>Change Name</button>
11     </div>
12   );
13 }
14
15 export default App;
16
```

The above code defines the **App** component that renders the **Greetings** component and a button to change the name prop.

### Step 3: Create a new file called Greetings.js

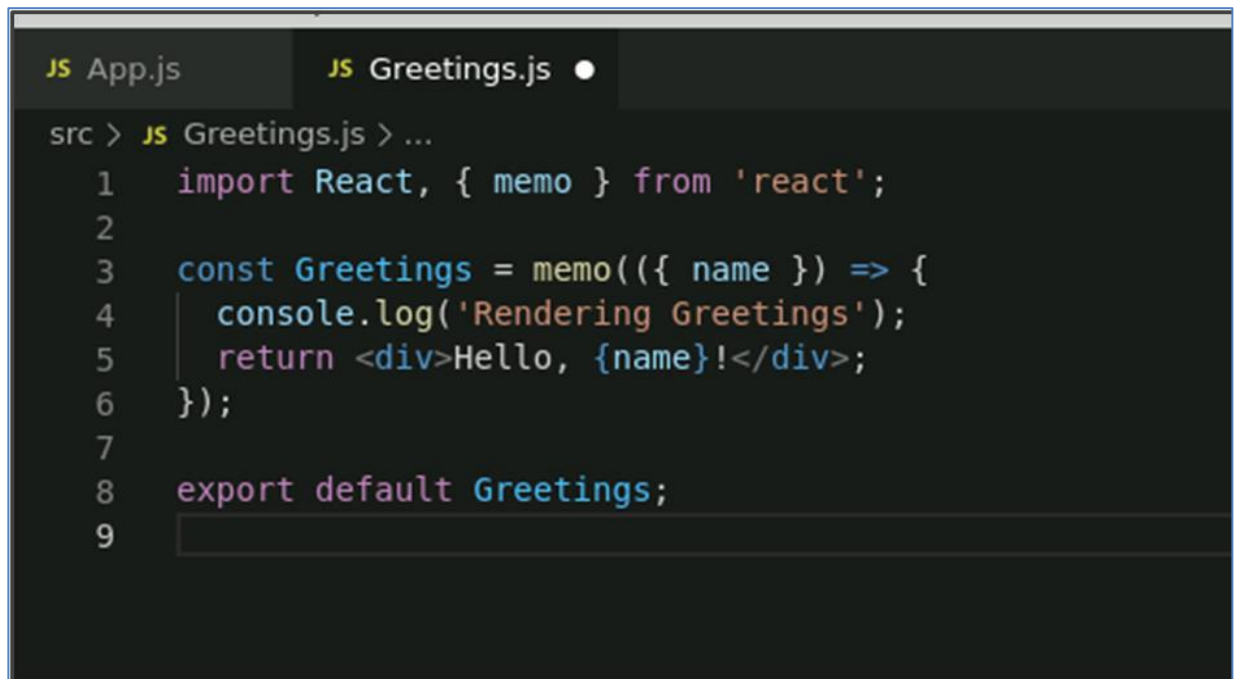
3.1 In your code editor, create a new file called **Greetings.js** inside the **src** directory

3.2 Copy and paste the provided code into the **Greetings.js** file:

```
import React, { memo } from 'react';

const Greetings = memo(({ name }) => {
  console.log('Rendering Greetings');
  return <div>Hello, {name}!</div>;
});

export default Greetings;
```

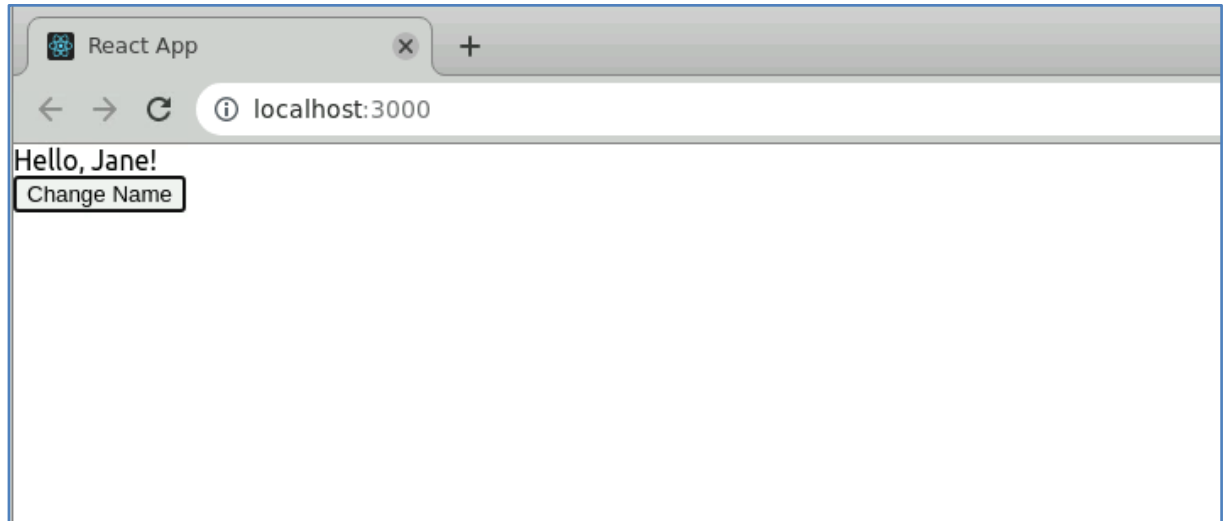
A screenshot of a code editor with a dark theme. At the top, there are two tabs: 'JS App.js' and 'JS Greetings.js', with the latter being the active tab. Below the tabs, the code for Greetings.js is displayed with line numbers from 1 to 9. The code is: 1 import React, { memo } from 'react'; 2 3 const Greetings = memo(({ name }) => { 4 console.log('Rendering Greetings'); 5 return <div>Hello, {name}!</div>; 6 }); 7 8 export default Greetings; 9  

```
src > JS Greetings.js > ...
1  import React, { memo } from 'react';
2
3  const Greetings = memo(({ name }) => {
4    console.log('Rendering Greetings');
5    return <div>Hello, {name}!</div>;
6  });
7
8  export default Greetings;
9
```

### Step 4: Run the app and verify the functionality

4.1 Go to the terminal and execute the **npm start** command within the project directory **greetings-app** to run the app

4.2 Once the server starts successfully, open **http://localhost:3000** in your browser to view the app



In the browser, you will see the default greeting message **Hello, John!** rendered by the **Greetings** component. Click the **Change Name** button to update the **name** state to **Jane** and trigger a re-render of the **Greetings** component with the new name prop value.

With this, you have successfully created a React program to re-render with a prop input.