React JS, Function

LinkedIn: Shiza Muneer
Whatsapp :+92 326 9331695

- **Types of Functions"**

1. Event Handling Functions

React allows you to handle events like clicks, form submissions, or changes in input fields through functions. These functions are called event handlers.

Example: Button Click Event Handler

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

function App() {
  const [counter, setCounter] = useState(0);

  // Event handler function
  const handleClick = () => {
    setCounter(counter + 1);
  };

  return (
    <div>
      <h1>Counter: {counter}</h1>
      <button onClick={handleClick}>Increment</button>
    </div>
  );
}

export default App;
```

Explanation: The handle Click function is an event handler that increments the counter when the button is clicked. The onClick event handler is bound to the button element.

2. State Update Functions (Using useState)

React components often need to maintain and update the state. You can use functions in combination with the useState hook to achieve this.

Example: Updating State with Functions

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

function ToggleButton() {
  const [isToggled, setIsToggled] = useState(false);

  // Toggle function
  const toggle = () => {
    setIsToggled(prevState => !prevState); // Toggling the state
  };

  return (
    <div>
      <button onClick={toggle}>
        {isToggled ? 'Turn Off' : 'Turn On'}
      </button>
    </div>
  );
}

export default ToggleButton;
```

Explanation: The toggle function changes the state of toggled from true to false (or vice versa). When the button is clicked, it calls the toggle function and updates the button text accordingly.

3. Rendering Functions (Functional Components)

In React, components are often written as functions, and these functions return JSX to render the UI.

Example: Functional Component

```
import React from 'react';

function WelcomeMessage({ name }) {
  return <h1>Welcome, {name}!</h1>;
}

export default WelcomeMessage;
```

Explanation: Welcome-message is a functional component that accepts name as a prop and returns a JSX element displaying a welcome message.

4. Effect Functions (Using useEffect)

The useEffect hook lets you perform side effects in your function components, such as fetching data or updating the DOM. This hook accepts a function as the first argument.

Example: Fetching Data with useEffect

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

function UserList() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    // Fetching data inside useEffect function
    fetch('https://jsonplaceholder.typicode.com/users')
      .then(response => response.json())
```

```
      .then(data => setUsers(data));
  }, []); // Empty dependency array ensures this effect runs only
once

  return (
    <div>
      <h2>User List</h2>
      <ul>
        {users.map(user => (
          <li key={user.id}>{user.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default UserList;
```

- **Explanation**: The useEffect function is used to fetch user data from an API. This function runs after the component is mounted and updates the users state.

5. Custom Functions (Utility Functions)

In React, you can create your own utility functions to perform operations and organize your code.

Example: Custom Function to Format a Date

```
import React from 'react';

function FormatDate({ date }) {
  const formatDate = (dateStr) => {
    const dateObj = new Date(dateStr);
    return dateObj.toLocaleDateString(); // Formatting date as a
string
  };
```

```
  return <p>{formatDate(date)}</p>;
}

export default FormatDate;
```

- **Explanation**: formatDate is a custom utility function that formats a date string into a more readable format. It's used within the component to display the formatted date.

6. Conditional Rendering with Functions

You can use functions to conditionally render different parts of your UI based on certain conditions.

Example: Conditional Rendering Function

```
import React from 'react';

function Greeting({ isLoggedIn }) {
  const renderMessage = () => {
    if (isLoggedIn) {
      return <h1>Welcome back!</h1>;
    } else {
      return <h1>Please log in</h1>;
    }
  };

  return <div>{renderMessage()}</div>;
}

export default Greeting;
```

Explanation: The renderMessage function returns different messages depending on whether the user is logged in or not.
