Event Handling

React also handles events similar to the DOM elements. But there are a few syntactical differences:

- React uses camelCase to name events.
- React refers to a function as an event handler rather than a string.
- In React we have to call preventDefault() method explicitly to prevent the default form behavior of submitting.

Example:

```
function EvenOdd(){
  const [value, setValue] = useState('');

  const changeHandler = (e) => {
    setValue(e.target.value);
  };

  const checkEvenOdd = () => {

    if (value % 2 === 0){
      alert('Even');
    }
    else{
     alert('Odd');
    }
  };
  return (
    <div>
      <h2>Enter a number and click on the button to check even or odd</h2>
      <input onChange={changeHandler}></input>
      <button onClick={checkEvenOdd}>Check</button>
    </div>
```

```
 )
}
```

In the above code we observe the following points:

- We trigger the events onChange and onClick by referring the respective event handlers.
- The changeHandler function sets the current input value of the input field.
- The checkEvenOdd checks whether the number entered is even or odd and sends an alert message.

Conditional Rendering

Conditional Rendering is used to render components of React based on a certain condition. React's conditional rendering functions in the same way that JavaScript's conditions do. Create items that describe the current state using JavaScript operators like if or the conditional operator, and let React adapt the UI to match them.

For example:

```
function CourseCompleted() {
  return <h1>COMPLETED</h1>;
}

function CourseInProgress() {
  return <h1>IN PROGRESS</h1>;
}
function CourseStatus(props) {
  const isComplete = props.isComplete;
  if (isComplete) {
    return <CourseCompleted />;
  }
  else {
    return <CourseInProgress />;
  }

}

function App() {
  return(
    <CourseStatus isComplete = {true} />
  );
}
```

In the above example, the component rendered by React depends upon the value of isComplete prop. The if statement is used to conditionally render UI component.

Next, we can see an example of conditional rendering using Class.

```
class ChildComponent extends React.Component {
    render() {
        return (
            <div>
                {this.props.isVisible ? <p>This is visible</p>: null}
            </div>
        );
    }
}
```

In this example, the child component uses the isVisible prop to determine whether or not to render a paragraph element.