Event Handling

- An event is an action that could be triggered as a result of the user action or system generated event.
- For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.
- React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.
- Handling events with react have some syntactic differences from handling events on DOM. These are:
    - React events are named as camelCase instead of lowercase.
    - With JSX, a function is passed as the event handler instead of a string.

Example

```
import React from 'react'
const App = () => {
                        const buttonClick = () => {
                          alert("Hello ReactJS")
                        }
 return(
            <>
                <h1>ReactJS Events</h1>
                <button onClick={buttonClick}>Click</button>
            </>
 )
}

export default App;
```

Example

```
import React from 'react';
const App = () => {
  const buttonClick = () => {
    const shouldProceed = window.confirm("Do you really want to proceed?");
    if (shouldProceed) {
      alert("Proceeding...");
    } else {
        alert("Action canceled.");
    }
  };
  return (
    <>
      <h1>ReactJS Events</h1>
      <button onClick={buttonClick}>Click</button>
    </>
  );
};

export default App;
```

Example -- Passing Parameters

```
import React from 'react'
const App = () => {
  const buttonClick = (name) => {
                                        alert(`Hello ${name}`)
                                      }
  return(
          <>
                  <h1>ReactJS Events</h1>
                  <button onClick={() => buttonClick("ReactJS")}>Click</button>
          </>
  )
}

export default App;
```

Example -- Event Parameter

```
import React from 'react'
const App = () => {
                        const buttonClick = (e) => {
                          e.preventDefault()
                        }
  return(
          <>
                  <h1>ReactJS Events</h1>
                  <button onClick={buttonClick}>Click</button>
          </>
  )
}
export default App;
```

Note : The e.preventDefault() call prevents the default behavior of form submission when the button is clicked. This is useful when you have a button inside a form, and you want to perform some custom actions without the form being submitted to a server.

The above code defines a React functional component (App) that includes a button. When the button is clicked, it calls the buttonClick() function, which prevents the default behavior of form submission. This can be useful when you want to handle the button click event in a custom way without the form being submitted to a server.

Forms in ReactJS

- Forms are the most important part of any application. It is a way for the user to interact with the application. We have forms for implementing Login/Register functionality, for adding or updating data into the application, etc.
- Generally, inputs and form elements maintain their own state on a web page this is how you're able to type into them and they maintain the script. React needs access to all of the state on the page including form data. So the forms state and the applications state needs to be the same thing.
- In React, form data is usually handled by the components. When the data is handled by the components, all the data is stored in the component state.

Ways to Create Forms in ReactJS

- React has two types of components i.e. Controlled Components and Uncontrolled Components.
- So with the help of these types, we can create forms in two different ways :-

    a) Using Controlled Components
    b) Using Uncontrolled Components

a) Forms with Controlled Component

- A controlled component is a component where our form data is controlled by React state.
- Controlled components in React are those in which form data is handled by the component's state.
- Using a controlled component is the preferred way to build your Forms in React. It lets you store the state in the component that renders the input and each input field accepts its current value as a prop and has a callback function which is called when the state of the input changes.
- Controlled components uses "states" to access the form element data and the state we can create by using "useState()" hook.

b) Forms with Uncontrolled Component

- An uncontrolled component is where your form data is handled by the DOM, instead of inside your React component.
- Uncontrolled Components are the form of data that is handled by the DOM itself. These components are not controlled by React state.
- In the Uncontrolled Components the values of the form elements are traditionally controlled by and stored on the DOM.
- Uncontrolled components uses "ref" to access their own state / form element state.

List of ReactJS Form Components

We have following Components that are used in "React Forms" :-

- Input
- Textarea
- Select
- Checkbox
- Radio Button
- Range Slider

Handling form by Controlled Way

i) Input

Example -- Simple TextInput

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

const App = () => {
  const [inputVal, setInputVal] = useState("")
  return (
    <div style={{padding:20}}>
      <h1>ReactJS Forms</h1>
      <div>
        <label>Name:</label>
        <input type="text" value={inputVal} onChange={(e) =>
setInputVal(e.target.value)}/>
      </div>
    </div>
  )
}

export default App;
```

Example -- Multiple TextInput

```
import React,{useState} from 'react'
const App = () => {
   const [inputVal, setInputVal] = useState({
   name:"",
   email:""
 })
   const inputHandler = (e) => {
   const { name, value } = e.target;
   setInputVal({
     ...inputVal,
     [name]:value
   })
 }
 return (
   <div style={{ padding: 20 }}>
     <h1>ReactJS Forms</h1>      <div>
       <label>Name:</label>
       <input
         type="text"
         value={inputVal.name}
         name="name"
         onChange={inputHandler}
       />      </div>
     <div>
       <label>Email:</label>
       <input
         type="email"
         name="email"
         value={inputVal.email}
         onChange={inputHandler}
       />
     </div>   </div>
 );
}
export default App
```

ii) TextArea

Example -- Simple TextArea

```
import React,{useState} from 'react'
const App = () => {
   const [inputVal, setInputVal] = useState("")
  return (
    <div style={{ padding: 20 }}>
      <h1>ReactJS Forms</h1>
      <div>
        <label>Message :</label>
        <textarea placeholder='Enter message ... ' value={inputVal} onChange={(e)
=> setInputVal(e.target.value)}></textarea>
      </div>
    </div>
  );
}
export default App
```

iii) Select

Example

```
import React,{useState} from 'react'
const App = () => {
   const [inputVal, setInputVal] = useState("")
  return (
    <div style={{ padding: 20 }}>
      <h1>ReactJS Forms</h1>
      <div>
        <label>Select Education :</label>
        <select value={inputVal} onChange={(e) => setInputVal(e.target.value)}>
          <option value="highschool">High School</option>
          <option value="intermediate">Intermediate</option>
          <option value="graduation">Graduation</option>
          <option value="postgraduation">Post Graduation</option>
        </select>
      </div>
    </div>
  );
}
export default App
```

iv) Checkbox

Example

```
import React,{useState} from 'react'
const App = () => {
   const [inputVal, setInputVal] = useState([])
  const handleCheckbox = (e) => {
   const { value, checked } = e.target;
   if(checked){
    setInputVal([...inputVal,value])
   }else{
    setInputVal(inputVal.filter((item) => item !== value))
   }
  }
  console.log("inputVal",inputVal)
  return (
   <div style={{ padding: 20 }}>
    <h1>ReactJS Forms</h1>
    <div>
     <label>Select Favourite Game :</label>
     <div>
      <label>Football</label>
      <input type="checkbox" value="football" onChange={handleCheckbox}/>
     </div>
     <div>
      <label>Hockey</label>
      <input type="checkbox" value="hockey" onChange={handleCheckbox}/>
     </div>
     <div>
      <label>Cricket</label>
      <input type="checkbox" value="cricket" onChange={handleCheckbox}/>
     </div>
     <div>
      <label>Badminton</label>
      <input type="checkbox" value="badminton"
onChange={handleCheckbox}/>
     </div>
```

```
      </div>
    </div>
  );
}
export default App
```

v) Radio Button

Example

```
import React,{useState} from 'react'
const App = () => {
   const [inputVal, setInputVal] = useState("")
  const handleRadio = (e) => {
   const { value, checked } = e.target;
   if(checked){
    setInputVal(value)
   }else{
    setInputVal()
   }
  }
  return (
   <div style={{ padding: 20 }}>
    <h1>ReactJS Forms</h1>
    <div>
     <label>Select Favourite Game : {inputVal}</label>
     <div>
      <label>Football</label>
      <input type="radio" name='game' value="football"
onChange={handleRadio}/>
     </div>
     <div>
      <label>Hockey</label>
      <input type="radio"  name='game' value="hockey"
onChange={handleRadio}/>
     </div>
     <div>
      <label>Cricket</label>
      <input type="radio"  name='game' value="cricket"
onChange={handleRadio}/>
     </div>
     <div>
      <label>Badminton</label>
```

```
          <input type="radio"  name='game' value="badminton"
onChange={handleRadio}/>
        </div>
      </div>
    </div>
  );
}

export default App
```

Example

```
import React from 'react';
const App = () => {
  const [gender, setGender] = React.useState('male');
  const handleChange = (event) => {
    setGender(event.target.value)
  }
  return (
    <form>
      <p>Gender</p>
      <div>
        <input
          type="radio"
          value="male"
          checked={gender === 'male'}
          onChange={handleChange}
        /> Male
      </div>
      <div>
        <input
          type="radio"
          value="female"
          checked={gender === 'female'}
          onChange={handleChange}
        /> Female
      </div>
      <div>
        <input
          type="radio"
          value="transgender"
          checked={gender === 'transgender'}
          onChange={handleChange}
        /> Transgender
      </div>
    </form>
  )}
export default App
```

vi) Range Slider

Example

```
import React,{useState} from 'react'
const App = () => {
   const [inputVal, setInputVal] = useState(50)
  const handleSlider = (e) => {
   setInputVal(e.target.value)
  }
  return (
   <div style={{ padding: 20 }}>
    <h1>ReactJS Forms</h1>
    <div>
     <label>Select Range : {inputVal}</label>
     <div>
      <span>0</span>
      <input type="range" value={inputVal} min={0} max={100}
onChange={handleSlider}/>
       <span>100</span>
     </div>
    </div>
   </div>
  );
}
export default App;
```

vii) Form Validation in ReactJS (homework)

The form must contains username, email, phone number, gender (two radio buttons). while clicking on submit button all the data should be displayed at console in the form of an array or object.

b) Handling form by Uncontrolled Way

i) Input

Example

```
import React,{useRef} from 'react'
const App = () => {
  const inputRef = useRef(null);
  const handleInputValue = () => {
    console.log(inputRef.current.value)
  }
  return (
    <div style={{padding:20}}>
      <h1>ReactJS Forms</h1>
      <div>
        <label>Name:</label>
        <input ref={inputRef} type="text"/>
        <button onClick={handleInputValue}>Get Value</button>
      </div>
    </div>
  )
}
export default App;
```

ii) TextArea

Example

```
import {useRef} from 'react';
const App = () => {
  const ref = useRef(null);
  const handleClick = event => {
    console.log(ref.current.value);
  };
  return (
    <div>
      <label htmlFor="message">My Textarea</label>
      <textarea ref={ref} id="message" name="message" />
      <button onClick={handleClick}>Click</button>
    </div>
  );
};
export default App;
```

iii) Select

Example

```
import {useRef} from 'react';
const App = () => {
  const inputRef = useRef(null);
    const handleClick = () => {
    console.log(inputRef.current.value);
  };
  return (
    <div>
      <select ref={inputRef}>
        <option value="html">HTML</option>
        <option value="css">CSS</option>
        <option value="js">JavaScript</option>
      </select>
      <button onClick={handleClick}>Click</button>
    </div>
  );
};

export default App;
```

iv) Checkbox

Example

```
import {useRef} from 'react';
const App = () => {
  const checkboxRef = useRef(null);
  const handleClick = () => {
    console.log(checkboxRef.current.checked);
    console.log(checkboxRef.current.value);
  };
  return (
    <div>
      <input ref={checkboxRef} type="checkbox" value="cricket" id="subscribe"
name="subscribe" />
      <button onClick={handleClick}>Click</button>
    </div>
  );
}
export default App
```

v) Radio Button

In the below example, we are using two ref cause if we use same ref for both the element, "radioButtonRef2" overrides the first one and it always points to the second radio button.So, in this way we have to create 2 different refs for both inputs.

Example

```
import {useRef} from 'react';
const App = () => {
  const radioButtonRef1 = useRef(null);
  const radioButtonRef2 = useRef(null);
  const handleClick = () => {
    const selectedRadio = radioButtonRef1.current.checked ?
radioButtonRef1.current.value : radioButtonRef2.current.value
    console.log(selectedRadio);
  };
  return (
    <div>
      <div>
        <label>Male</label>
        <input ref={radioButtonRef1} type="radio" id="subscribe1"
name="subscribe" value="male"/>
      </div>
      <div>
        <label>Female</label>
        <input ref={radioButtonRef2} type="radio" id="subscribe2"
name="subscribe" value="female"/>
      </div>
      <button onClick={handleClick}>Click</button>
    </div>
  );
}
export default App
```

'children' Prop

- The React docs say that you can use props.children on components that represent 'generic boxes' and that don't know their children ahead of time.
- The "children" prop is used to display whatever you include between the opening and closing tags when invoking a component.
- <User>Hello Student</User>, so in order to access "Hello Student" we should write {props.children}.

Example

App.js

```
import React, { Component } from 'react'
import Info from './info'
export default class App extends Component {
 render() {
  return (
   <div>
    <Info>Rohit</Info>
    <Info>Shubham</Info>
    <Info>Rishabh</Info>
   </div>
  )
 }}
```

info.js

```
import React from 'react'
const Info = props => {
   return(
      <h1>Hello {props.children}</h1>
   )
}
export default Info;
```

Component Reusability

Reusable components are those React components that can be used multiple times in your application.

Examples

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

const TextInput = ({
  inputType,
  inputPlaceholder,
  inputHandler,
  inputValue,
  inputName,
  ...props
}) => {
  return (
    <input
      type={inputType}
      placeholder={inputPlaceholder}
      onChange={inputHandler}
      value={inputValue}
      name={inputName}
      {...props}
    />
  );
};

const Button = ({ children, handleClick }) => {
  return <button onClick={handleClick}>{children}</button>;
};

const TextArea = ({
  textareaValue,
  textareaPlaceholder,
  textareaRows,
```

```
  textareaCols,
  textareaname,
  textareaHandler,
  textareaResizable,
}) => {
  return (
    <textarea
      value={textareaValue}
      placeholder={textareaPlaceholder}
      rows={textareaRows ? textareaRows : 5}
      cols={textareaCols ? textareaCols : 50}
      name={textareaname}
      onChange={textareaHandler}
      style={{ resize: textareaResizable ? "both" : "none" }}
    ></textarea>
  );
};

const Dropdown = ({
  dropdownValue,
  dropdownData,
  dropdownInfo,
  dropdownHandler,
}) => {
  return (
    <select value={dropdownValue} onChange={dropdownHandler}>
      <option>{dropdownInfo}</option>
      {dropdownData && dropdownData.length !== 0 ? (
        dropdownData.map((item) => (
          <option key={item.id} value={item.value}>
            {item.label}
          </option>
        ))
      ) : (
        <option>No Data Found</option>
      )}
    </select>
```

```
  );
};

const Checkbox = ({
  checkboxValue,
  checkboxHandler,
  checkboxSelectedValue,
  checkboxLable,
}) => {
  return (
    <>
      <input
        type="checkbox"
        checked={checkboxSelectedValue.includes(checkboxValue)}
        value={checkboxValue}
        onChange={checkboxHandler}
      />
      <label>{checkboxLable}</label>
    </>
  );
};

const RadioButton = ({
  radioButtonLabel,
  radioButtonHandler,
  radioButtonName,
  radioButtonChecked,
  radioButtonValue,
}) => {
  return (
    <>
      <input
        type="radio"
        value={radioButtonValue}
        checked={radioButtonChecked === radioButtonValue}
        onChange={radioButtonHandler}
        name={radioButtonName}
```

```
      />
      <label>{radioButtonLabel}</label>
    </>
  );
};

const App = () => {
  const [inputVal, setInputVal] = useState({
    email: "",
    password: "",
    message: "",
  });

  const [dropdownState, setDropdownState] = useState("");

  const dropdownData = [
    { id: 1, value: "HTML", label: "Hypertext Markup Language" },
    { id: 2, value: "CSS", label: "Cascading StyleSheet" },
    { id: 3, value: "JS", label: "Javascript" },
    { id: 4, value: "React", label: "ReactJS" },
  ];

  const [checkboxState, setCheckboxState] = useState([]);

  const [radioButtonState, setRadioButtonState] = useState(false);

  const handleInput = (e) => {
    const { name, value } = e.target;
    setInputVal({
      ...inputVal,
      [name]: value,
    });
  };

  const buttonClick = (e) => {
    console.log("Hello World");
  };
```

```
const handleDropdown = (e) => {
  setDropdownState(e.target.value);
};

const handleCheckbox = (e) => {
  const { value, checked } = e.target;
  if (checked) {
    setCheckboxState([...checkboxState, value]);
  } else {
    setCheckboxState(checkboxState.filter((item) => item !== value));
  }
};

const handleRadioButton = (e) => {
  setRadioButtonState(e.target.value);
};

return (
  <>
    <div>
      <h3>Reusable TextInput</h3>
      <TextInput
        inputValue={inputVal.email}
        inputHandler={handleInput}
        inputPlaceholder="Enter email"
        inputType="email"
        inputName="email"
      />
      <TextInput
        inputValue={inputVal.password}
        inputHandler={handleInput}
        inputPlaceholder="Enter password"
        inputType="password"
        inputName="password"
      />
    </div>
```

```
<div>
  <h3>Reusable Button</h3>
  <Button handleClick={buttonClick}>Click Me</Button>
</div>
<div>
  <h3>Reusable TextArea</h3>
  <TextArea
    textareaPlaceholder="Enter message"
    textareaResizable={false}
    textareaname="message"
    textareaValue={inputVal.message}
    textareaHandler={handleInput}
  />
</div>
<div>
  <h3>Reusable Dropdown</h3>
  <Dropdown
    dropdownData={dropdownData}
    dropdownValue={dropdownState}
    dropdownHandler={handleDropdown}
    dropdownInfo="Please select a language"
  />
</div>
<div>
  <h3>Reusable Checkbox</h3>
  <Checkbox
    checkboxValue="HTML"
    che
    checkboxHandler={handleCheckbox}
    checkboxSelectedValue={checkboxState}
    checkboxLable="Hypertext Markup Language"
  />
  <Checkbox
    checkboxValue="CSS"
    checkboxHandler={handleCheckbox}
    checkboxSelectedValue={checkboxState}
    checkboxLable="Cascading Stylesheet"
```

```
        />
        <Checkbox
          checkboxValue="JS"
          checkboxHandler={handleCheckbox}
          checkboxSelectedValue={checkboxState}
          checkboxLable="JavaScript"
        />
      </div>
      <div>
        <h3>Reusable RadioButton</h3>
        <RadioButton
          radioButtonName="genderinput"
          radioButtonHandler={handleRadioButton}
          radioButtonChecked={radioButtonState}
          radioButtonValue="male"
          radioButtonLabel="Male"
        />
        <RadioButton
          radioButtonName="genderinput"
          radioButtonHandler={handleRadioButton}
          radioButtonChecked={radioButtonState}
          radioButtonValue="female"
          radioButtonLabel="Female"
        />
      </div>
    </>
  );
};

export default App;
```