Events and Interactions

What is Event?

- Event is a message sent by sender to its subscriber in order to notify the change.
- Event follows “Observer”, which is a communication design pattern.
- Event uses “Delegate mechanism” – Function Pointer.
- React JS uses all JavaScript browser events in the same way.
- Events are defined in “Camel Case”.
- Event handler uses function pointer as “JSX expression”
- Sender notified the changes.
- Subscriber comprises of actions to perform.
- **React JS can use all JavaScript Browser events, which includes**
    - Clipboard Events
    - Composition Events
    - Keyboard Events
    - Focus Events
    - Generic Events
    - Mouse Events
    - Pointer Events
    - Selection Events
    - Touch Events
    - UI Events
    - Wheel Events
    - Media Events
    - Image Events
    - Animation Events
    - Transition Events
    - Timer Events etc.
- Java Script events have 2 arguments

- this : Sends information about the object [related properties and methods]
- event : Sends information about the event [related properties and methods]

- React JS events have only “event” argument. [You can’t use this as argument].
- Object related properties can be accessed by using “event.target” object.

Ex: Event in Functional Component

App.js

```
function Product(){
  function InsertClick(){
        alert("Record Inserted");
  }
  return (
        <div>
        <button onClick={InsertClick} name="btnInsert" className="btn btn-
primary">Insert</button>
        </div>
  )
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
        )
  }
```

```
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: **Event Argument** [event]

App.js

```
function Product(){
  function InsertClick(e){
        alert(`You Clicked At X Position ${e.clientX}\n Button Class
Name=${e.target.className}`);
  }
  return (
        <div>
        <button onClick={InsertClick} name="btnInsert" className="btn btn-
primary">Insert</button>
        </div>
  )
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
```

```
        <Product />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Events in Class Component

- In Class component events actions are defined as Methods.
- Event handler points towards specific method.

Ex:

App.js

```
class Product extends React.Component
{
  InsertClick(e){
        alert(`X Position: ${e.clientX}\nName : ${e.target.name}`);
  }
  render(){
        return(
        <div>
        <button name="btnInsert" onClick={this.InsertClick} className="btn btn-
primary">Insert</button>
        </div>
        )
  }
```

```
}


class MainContent extends React.Component
{
  render(){
          return (
          <div className="container-fluid">
        <h3>Event Handling</h3>
          <Product />
          </div>
          )
  }
}


ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Index.html

```
<!DOCTYPE html>
<html>
          <head>
          <title>HTML React App</title>
          <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
          <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css" crossorigin="anonymous" >
          <link rel="stylesheet" href="../src/app.css">
```

```
        <script crossorigin
src="https://unpkg.com/react@17/umd/react.development.js"></script>

        <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-
dom.development.js"></script>

        <script src="https://cdnjs.cloudflare.com/ajax/libs/prop-types/15.7.2/prop-types.js"
integrity="sha512-
Ttnt6nUh/1oV+4T/KWkG/ORvlIEOZohMCT/GIzelxTceoIreR2A1r6zmgFnEvi7GgstoDblmpM
O/Gi9CXKTqGQ==" crossorigin="anonymous"></script>

        <script crossorigin
src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

        <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>

        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
crossorigin="anonymous"></script>

        <script type="text/jsx" src="../src/app.js"></script>

        </head>

        <body>

        <div id="root">



        </div>

        </body>

</html>
```

Event Handler

- Event handler is a function pointer.
- Multiple controls can use the same function to handle various functionalities.

Ex:

App.js

```
class Product extends React.Component

{


  DataOperations(e){

        switch(e.target.value)

        {

        case "Insert":

        alert("Record Inserted");

        break;

        case "Update":

        alert("Record Updated");

        break;

        case "Delete":

        alert("Record Deleted");

        break;

        }

  }

  render(){

        return(

        <div>

        <div className="btn-group">

        <button name="btnInsert" value="Insert" onClick={this.DataOperations}
className="btn btn-primary">Insert</button>

        <button name="btnUpdate" value="Update" onClick={this.DataOperations}
className="btn btn-success">Update</button>

        <button name="btnDelete" value="Delete" onClick={this.DataOperations}
className="btn btn-danger">Delete</button>

        </div>

        <h2 align="center"></h2>

        </div>
```

```
        )
  }
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Note: State defined for a component is initialized into component memory.

The methods configured with event handler must bind with the component memory.

Syntax:

{this.buttonClick.bind(this)**}**

Using State in React User Interactions

[Define State and Set State on Event Handling]

- State is required to configure in a component in order to store the data and use between interactions.
- Properties are read-only and will not allow to set value.
- Hence state is required to configure value dynamically.
- State is initialized

Syntax:

```
this.state = {
        property: “value”
    }
```

- You can modify the state by using “setState()”

Syntax:

```
this.setState(function(state){
     property: state.property = “New Value”
    })
```

- Events can’t access current state. You have to bind the current context to event.
- If event has to use any of the component related properties and methods then you have to bind with current context.

Syntax:

```
this.eventHandlerName.bind(this)
```

- You can bind in the initialization phase.

Syntax:

```
constructor(props)
{
  super();
  this.state = { };
  this.eventHandler = this.eventHandler.bind(this);
}
```

- You can bind at the event handler while configuring event for DOM element.

Syntax:

<button onClick={this.eventHandler.bind(this)}> Text </button>

Ex:

App.js

```
class Product extends React.Component
{
  constructor(props){
        super(props);
        this.state = {
        msg: "Select Database Command"
        }
        this.InsertClick = this.InsertClick.bind(this);
        this.UpdateClick = this.UpdateClick.bind(this);
        this.DeleteClick = this.DeleteClick.bind(this);
  }
  InsertClick(){
        this.setState(state=>({
        msg: state.msg = "Record Inserted"
        }));
  }
  UpdateClick(){
        this.setState(state=>({
        msg: state.msg = "Record Updated"
        }));
  }
  DeleteClick(){
        this.setState(state=>({
```

```
            msg: state.msg = "Record Deleted Successfully.."
            }));
    }
    render(){
            return(
            <>
            <div className="btn-group">
            <button onClick={this.InsertClick} className="btn btn-primary">Insert</button>
            <button onClick={this.UpdateClick} className="btn btn-success">Update</button>
            <button onClick={this.DeleteClick} className="btn btn-danger">Delete</button>
            </div>
            <h2 className="text-center">{this.state.msg}</h2>
            </>
            )
    }
}

class MainContent extends React.Component
{
    render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <Product />
            </div>
            )
    }
}
```

```
ReactDOM.render(

  <MainContent />,

  document.getElementById('root')

);
```

Index.html

```
<!DOCTYPE html>

<html>

        <head>

        <title>HTML React App</title>

        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">

        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css" crossorigin="anonymous" >

        <link rel="stylesheet" href="../src/app.css">

        <script crossorigin
src="https://unpkg.com/react@17/umd/react.development.js"></script>

        <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-
dom.development.js"></script>

        <script src="https://cdnjs.cloudflare.com/ajax/libs/prop-types/15.7.2/prop-types.js"
integrity="sha512-
Ttnt6nUh/1oV+4T/KWkG/ORvIlEOZohMCT/GIzelxTceoIreR2A1r6zmgFnEvi7GgstoDblmpM
O/Gi9CXKTqGQ==" crossorigin="anonymous"></script>

        <script crossorigin
src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

        <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>

        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
crossorigin="anonymous"></script>

        <script type="text/jsx" src="../src/app.js"></script>
```

```
        </head>
        <body>
        <div id="root">

        </div>
        </body>
</html>
```

Ex: Instead on using multiple methods you can use single method to handle all interactions.

App.js

```
class Product extends React.Component
{
  constructor(props){
        super(props);
        this.state = {
        msg: "Select Database Command"
        }
        this.DatabaseOperations = this.DatabaseOperations.bind(this);
 }
   DatabaseOperations(e)
  {
        switch(e.target.value)
        {
        case "Insert":
        this.setState(state=>({
        msg : state.msg = "Record Inserted"
        }))
        break;
```

```
        case "Update":
        this.setState(state=>({
        msg : state.msg = "Record Updated"
        }))
        break;
        case "Delete":
        this.setState(state=>({
        msg : state.msg = "Record Deleted"
        }))
        break;
        }
  }
  render(){
        return(
        <>
        <div className="btn-group">
        <button value="Insert" onClick={this.DatabaseOperations} className="btn btn-
primary">Insert</button>
        <button value="Update" onClick={this.DatabaseOperations} className="btn btn-
success">Update</button>
        <button value="Delete" onClick={this.DatabaseOperations} className="btn btn-
danger">Delete</button>
        </div>
        <h2 className="text-center">{this.state.msg}</h2>
        </>
        )
  }
}

class MainContent extends React.Component
```

```
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

FAQ: What are the various event binding techniques used for accessing the current component members?

- *You can bind the event in initialization phase of component "constructor()"*
- *You can bind the event in "event handler" of DOM element.*
- *You can bind the event without using "bind()" method, you have to uses a call back technique to return the event and bind to element.*
- *You can use the event return value and bind to event by using anonymous technique.*

Ex: Constructor

App.js

```
class Product extends React.Component
{
  constructor(props){
        super(props);
```

```
        this.state = {
        msg : 'Select Database Operation'
        }
        this.InsertClick = this.InsertClick.bind(this);
  }

  InsertClick(){
        this.setState({
        msg: 'Record Inserted'
        })
  }
  render(){
        return(
        <>
        <div className="btn-group">
        <button onClick={this.InsertClick} className="btn btn-primary">Insert</button>
        </div>
        <h2 className="text-center">{this.state.msg}</h2>
        </>
        )
  }
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
```

```
            <h3>Event Handling</h3>
            <Product />
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: DOM Event Handler

App.js

```
class Product extends React.Component
{
  constructor(props){
            super(props);
            this.state = {
            msg : 'Select Database Operation'
            }

  }

  InsertClick(){
            this.setState({
            msg: 'Record Inserted'
            })
  }
```

```
  render(){
        return(
        <>
        <div className="btn-group">
        <button onClick={this.InsertClick.bind(this)} className="btn btn-
primary">Insert</button>
        </div>
        <h2 className="text-center">{this.state.msg}</h2>
        </>
        )
  }
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: Without bind() method

```
class Product extends React.Component
{
  constructor(props){
        super(props);
        this.state = {
        msg : 'Select Database Operation'
        }

  }

  InsertClick(){
        this.setState({
        msg: 'Record Inserted'
        })
  }
  render(){
        return(
        <>
        <div className="btn-group">
        <button onClick={()=> this.InsertClick()} className="btn btn-
primary">Insert</button>
        </div>
        <h2 className="text-center">{this.state.msg}</h2>
        </>
        )
  }
}
```

```
class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: **Anonymous**

App.js

```
class Product extends React.Component
{
  constructor(props){
        super(props);
        this.state = {
        msg : 'Select Database Operation'
        }

  }
```

```
  InsertClick = ()=> {
        this.setState({
        msg: 'Record Inserted'
        })
  }

  render(){
        return(
        <>
        <div className="btn-group">
        <button onClick={this.InsertClick} className="btn btn-primary">Insert</button>
        </div>
        <h2 className="text-center">{this.state.msg}</h2>
        </>
        )
  }
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
        )
```

```
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

FAQ: What are controlled and uncontrolled components?

- **Uncontrolled component:** It handles the events by using basic DOM manipulations, It doesn't require any event binding.
- **Controller Component:** It handles the events by using its internal state. It requires event binding.

Ex: Uncontrolled Component [No-State defined]

App.js

```
class Product extends React.Component
{

  InsertClick(){
        document.write("Record Inserted");
  }

  render(){
        return(
        <>
        <div className="btn-group">
        <button onClick={this.InsertClick} className="btn btn-primary">Insert</button>
        </div>
```

```
        <h2 className="text-center"></h2>
        </>
        )
  }
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Note: React can use all browser events for elements, which require event binding based on the component type. [Controlled Components require – Event Binding]

Ex: Uncontrolled Component using “onChange”

App.js

```
class Product extends React.Component
```

```
{
  CityChanged(e){
        document.write(`Selected City : ${e.target.value}`);
  }
  render(){
        return(
        <>
        <div className="btn-group">
        <select onChange={this.CityChanged} className="form-control">
        <option>Delhi</option>
        <option>Hyd</option>
        <option>Chennai</option>
        </select>
        </div>
        <h2 className="text-center"></h2>
        </>
        )
  }
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Product />
        </div>
```

```
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

FAQ: How you can manage cross-browser compatibility while working with events in "React.js"?

A: React.js manages the cross-browser compatibility by using "**SyntheticEvent".**

SyntheticEvent

- SyntheticEvent is a part of React.js Event Handling.
- It is responsible for making the react event compatible with "cross-browser".
- Events can work identically across all browsers.
- The event argument "event" is managed by SyntheticEvent object of React.js
- It is responsible for making the event native for specific browser.

Ex: Get all properties of SyntheticEvent

App.js

```
class Register extends React.Component
{
  RegisterClick(SyntheticEvent){
        for(var property in SyntheticEvent)
        {
        document.write(property + "<br>");
        }
  }
```

```
  render(){
        return(
        <>
        <button value="Register" onClick={this.RegisterClick}>Register</button>
        </>
        )
  }
}

class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <Register />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

- **Every SyntheticEvent object comprises of following attribute**

bubbles	Boolean
cancelable	Boolean

isTrusted	Boolean
target	DOMEventTarget
type	String
preventDefault()	Void
stopPropagation()	Void
persist()	Void
nativeEvent	DOMEvent

- You can access the browser native event related details by using “nativeEvent”

Keyboard Events

React uses Keyboard events to handle various interactions with regard to key code, and key actions.

- onKeyDown : Indicates the actions to perform on key down
- onKeyUp : Indicates actions to perform on key up.
- onKeyPress : Indicates actions to perform when user finish a key and uses another key.

Properties:

- keyCode
- charCode
- shiftKey
- ctrlKey
- which
- key [Return the key used]

Ex: Identifying the Key used

App.js

```
class Register extends React.Component
{
  constructor(props){
        super(props);
        this.state = {
        msg: ''
        }
        this.VerifyPassword = this.VerifyPassword.bind(this);
  }
  VerifyPassword(e){
        if(e.key == "CapsLock"){
        this.setState({
        msg: 'Caps Lock ON'
        })
        } else {
        this.setState({
        msg: ''
        })
        }
  }
   render(){
        return(
        <>
        <div>
        <lable>Password</lable>
        <div>
```

```
            <input onKeyUp={this.VerifyPassword} type="password" className="form-control"
/>
            <span>{this.state.msg}</span>
            </div>
            </div>
            </>
            )
   }
}



class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <Register />
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: **Actions on specific key handling**

```
class Register extends React.Component
{
   VerifyChar(e){
            if(e.key=="y") {
          location.href="http://www.amazon.in";
            } else {
            document.write("Action Canceled..");
            }
            }
   render(){
            return(
            <>
                <input onKeyUp={this.VerifyChar} type="text" placeholder="Y to Continue and N
to Stop" />
            </>
            )
   }
}



class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <Register />
            </div>
            )
```

```
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: **Verify Value entered in input element on Key Event**

App.js

```
class Register extends React.Component
{
   constructor(props){
          super(props);
          this.state = {
          users: [
          {UserName: 'john'},
          {UserName: 'john1'},
          {UserName: 'david'},
          {UserName: 'david_nit'}
          ],
          msg: ''
          };
          this.VerifyUser = this.VerifyUser.bind(this);
   }
   VerifyUser(e){
          for(var user of this.state.users)
          {
```

```
            if(user.UserName==e.target.value)
            {
            this.setState({
            msg: 'User Name Taken - Try Another'
            })
            break;
            } else {
            this.setState({
            msg: 'User Name Available'
            })
            }
            }
    }
    render(){
            return(
            <>
            <div>
            <label>User Name</label>
            <div>
            <input onKeyUp={this.VerifyUser} type="text" className="form-control" />
             <span>{this.state.msg}</span>
            </div>
            </div>
            </>
            )
    }
}
```

```
class MainContent extends React.Component
{
  render(){
          return (
          <div className="container-fluid">
          <h3>Event Handling</h3>
          <Register />
          </div>
          )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: **Verify User Name and Password**

App.js

```
class Register extends React.Component
{
   constructor(props){
          super(props);
          this.state = {
          users: [
          {UserName: 'john'},
          {UserName: 'john1'},
```

```
        {UserName: 'david'},
        {UserName: 'david_nit'}
        ],
        msg: '',
        regExp: /(?=.*[A-Z])\w{4,15}/,
        pwdMsg: ''
        };
        this.VerifyUser = this.VerifyUser.bind(this);
        this.VerifyPassword = this.VerifyPassword.bind(this);
}
VerifyUser(e){
        for(var user of this.state.users)
        {
          if(user.UserName==e.target.value)
        {
        this.setState({
        msg: 'User Name Taken - Try Another'
        })
        break;
        } else {
        this.setState({
        msg: 'User Name Available'
        })
        }
        }
}
VerifyPassword(e){
   if(e.target.value.match(this.state.regExp)){
```

```
        this.setState({
        pwdMsg: 'Strong Password'
        })
        } else {
        if(e.target.value.length<4){
        this.setState({
        pwdMsg: 'Poor Password'
        })
        } else {
        this.setState({
        pwdMsg: 'Weak Password'
        })
        }
        }
}
render(){
        return(
        <>
        <div className="form-group">
        <label>User Name</label>
        <div>
        <input onKeyUp={this.VerifyUser} type="text" className="form-control" />
         <span>{this.state.msg}</span>
        </div>
        </div>
        <div className="form-group">
        <label>Password</label>
        <div>
```

```
            <input onKeyUp={this.VerifyPassword} type="password" className="form-control"
/>
             <span>{this.state.pwdMsg}</span>
            </div>
            </div>
            </>
            )
   }
}



class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <Register />
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Mouse Events

Mouse events specifies actions to perform with regard to mouse interactions on web page or app.

- onClick
- onContextMenu
- onDoubleClick
- onDrag
- onDragEnd
- onDragEnter
- onDargExit
- onMouseDown
- onMouseEnter
- onMouseMove
- onMouseLeave
- onMouseOver etc.

Properties

- altKey
- button
- clientX
- clientY etc..

Ex: onMouseOver, onMouseLeave (onmouseout)

App.js

```
class MouseDemo extends React.Component
{
   constructor(props){
        super(props);
        this.state = {
        styleObj: {
        color: 'black'
        }
```

```
        }
        this.GetColor = this.GetColor.bind(this);
        this.SetDefaultColor = this.SetDefaultColor.bind(this);
}
GetColor(e){
        switch(e.target.id){
        case "red":
        this.setState({
        styleObj: {
        color: 'red'
        }
        })
        break;
        case "green":
        this.setState({
        styleObj: {
        color: 'green'
        }
        })
        break;
        case "blue":
        this.setState({
        styleObj: {
       color: 'blue'
        }
        })
        break;
        }
```

```
  }
  SetDefaultColor(){
        this.setState({
        styleObj: {
        color:'black'
        }
        })
  }
  render(){
        return(
        <>
        <div onMouseOver={this.GetColor} onMouseLeave={this.SetDefaultColor}
className="row mr-5">
        <div className="col" id="red" className="bg-danger">
        Red color
        </div>
        <div className="col" id="green" className="bg-success" >
          Green color
        </div>
        <div className="col" id="blue" className="bg-info">
        Blue color
        </div>
        </div>
        <h1 style={this.state.styleObj}>Sample Text</h1>
        </>
        )
  }
}
```

```
class MainContent extends React.Component
{
  render(){
          return (
          <div className="container-fluid">
          <h3>Event Handling</h3>
          <MouseDemo />
          </div>
          )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: Accessing the mouse pointier position on mousemove

App.js

```
class MouseDemo extends React.Component
{
   constructor(props){
          super(props);
          this.state = {
          styleObj : {
          position: '',
          left: '',
          top: ''
```

```
        }
        }
        this.GetMousePosition = this.GetMousePosition.bind(this);
    }
    GetMousePosition(e){
        this.setState({
        styleObj: {
        position: 'fixed',
        left: e.clientX + 'px',
        top: e.clientY + 'px'
        }
        })
    }
    render(){
        return(
        <>
        <div onMouseMove={this.GetMousePosition}>
        <div style={{height:'1000px'}}>

        </div>
        <img style={this.state.styleObj} src="flag.gif" height="50" width="50"/>
      </div>
        </>
        )
    }
}
```

```
class MainContent extends React.Component
{
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <MouseDemo />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: Controlling Animations on Mouse Event

App.js

```
class MouseDemo extends React.Component
{
   StartScrolling(e){
        e.target.start();
   }
   StopScrolling(e){
        e.target.stop();
   }
```

```
   render(){
            return(
            <>
            <div>
            <marquee onMouseLeave={this.StartScrolling} onMouseOver={this.StopScrolling}
scrollamount="10">
            <img src="speaker.jpg" width="100" height="100" />
            <img src="shirt.jpg" width="100" height="100" />
            <img src="earpods.jpg" width="100" height="100" />
            <img src="shoe.jpg" width="100" height="100" />
            <img src="jeans.jpg" width="100" height="100" />
               </marquee>
            </div>
            </>
            )
   }
}



class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <MouseDemo />
            </div>
            )
  }
```

```
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: onChange Event

App.js

```
class EventsDemo extends React.Component
{
        constructor(props){
        super(props);
        this.state = {
        msg: 'Select a City'
        }
        this.CityChanged = this.CityChanged.bind(this);
        }
        CityChanged(e){
        this.setState({
        msg: e.target.value
        })
        }
        render(){
        return(
        <>
        <div>
        <label>Select City</label>
```

```
            <div>
            <select onChange={this.CityChanged}>
            <option>Select Your City</option>
              <option>Delhi</option>
            <option>Hyd</option>
              <option>Chennai</option>
            </select>
           </div>
            </div>
            <h3 align="center">City Name : {this.state.msg}</h3>
            </>
            )
            }
}
class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <EventsDemo />
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
```

```
  document.getElementById('root')
);
```

onBlur, onFocus, onSelect, onCut, onCopy, onPaste, onDoubleClick, onContextMenu

Ex: Double Click Native is “ondblclick” – SyntenticEvent “onDoubleClick”

```
class EventsDemo extends React.Component
{
        constructor(props){
        super(props);
        this.state = {
        msg: 'Select a City'
        }
        this.CityChanged = this.CityChanged.bind(this);
        }
        CityChanged(e){
        this.setState({
        msg: e.target.value
        })
        }
        OpenImage(){
        window.open('shoe.jpg','Nike','Width=400 height=500');
        }
        render(){
        return(
        <>
        <div>
        <img onDoubleClick={this.OpenImage} src="shoe.jpg" width="40" height="40"/>
```

```
            <p>Double Click to View Large</p>
            </div>
            <div>
            <label>Select City</label>
            <div>
            <select onChange={this.CityChanged}>
            <option>Select Your City</option>
              <option>Delhi</option>
            <option>Hyd</option>
              <option>Chennai</option>
            </select>
            </div>
            </div>
            <h3 align="center">City Name : {this.state.msg}</h3>
            </>
            )
            }
}


class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <EventsDemo />
            </div>
```

```
        )
    }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

onSubmit

- It is the event configured for “form” element.
- It fires up the functionality when form is submitted.

Ex:

App.js

```
class EventsDemo extends React.Component
{
   SubmitClick(e){
      alert(`Name=${e.target[0].value}\nMobile=${e.target[1].value}`)
   }
   render(){
        return(
        <>
        <form  onSubmit={this.SubmitClick}>
        <dl>
        <dt>User Name</dt>
        <dd><input type="text" name="username" /></dd>
        <dt>Mobile</dt>
        <dd>
        <input type="text" name="mobile" />
```

```
            </dd>
            </dl>
            <button>Submit</button>
            </form>
            </>
            )
    }
}


class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <EventsDemo />
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex: Reading all element values

```
class EventsDemo extends React.Component
{
  SubmitClick(e){
        for(var element of e.target) {
        document.write(element.value + "<br>");
        }
  }
  render(){
        return(
        <>
        <form  onSubmit={this.SubmitClick}>
        <dl>
        <dt>User Name</dt>
        <dd><input type="text" name="username" /></dd>
        <dt>Mobile</dt>
        <dd>
        <input type="text" name="mobile" />
        </dd>
        </dl>
       <button>Submit</button>
        </form>
        </>
        )
  }
}
class MainContent extends React.Component
{
```

```
  render(){
        return (
        <div className="container-fluid">
        <h3>Event Handling</h3>
        <EventsDemo />
        </div>
        )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Touch Events

Specifies interactions with devices that support "Touch". User can touch, swipe using pointing devices or even mouse.

- onTouchStart : When user starts swipe
- onTouchMove : While swiping on screen
- onTouchEnd : When user ends swipe.

Every Touch event comprises of 3 major properties related to "Touch"

- touches
- targetTouches
- changedTouches

Ex: Touch Event

App.js

```
class TouchEvent extends React.Component
{
  constructor(props) {
          super(props);
          this.state = {
          shoeName: ''
          }
          this.TouchStart = this.TouchStart.bind(this);

  }

  TouchStart(e){
          if(e.touches[0].clientX >= 24 && e.touches[0].clientX<=203) {
      this.setState({
          shoeName: 'Nike Casuals'
          });
          }
          if(e.touches[0].clientX >= 230 && e.touches[0].clientX<=401) {
          this.setState({
          shoeName: 'Lee Cooper Boot'
          });
          }

  }

  render(){
          return (
          <>
```

```
            <div onTouchStart={this.TouchStart}>
            <img src="shoe.jpg" width="200" height="200" />
            <img src="shoe1.jpg" width="200" height="200" />
            </div>
            <h2>You Touched : {this.state.shoeName} </h2>
            </>
    )
  }
}

class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <h3>Event Handling</h3>
            <TouchEvent/>
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

Ex:

App.js

```
class TouchEvent extends React.Component
{
  constructor(props) {
          super(props);
          this.state = {
          styleObj : {
          position: 'fixed',
          top: '',
          left: ''
          }
          }
          this.TouchStart = this.TouchStart.bind(this);
  }
  TouchStart(e){
          this.setState({
          styleObj: {
          position: 'fixed',
          left: e.touches[0].clientX + 'px',
          top: e.touches[0].clientY + 'px'
          }
          })
  }
  render(){
    return (
          <>
          <div onTouchStart={this.TouchStart}>
```

```
            <img style={this.state.styleObj} src="ball.png" height="50" width="50"/>
            </div>
            </>
            )
  }
}

class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <TouchEvent/>
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```

onTouchStart : It fires up when user starts the touch.

onTouchMove : It fires up while dragging on touch.

onTouchEnd : It fires up when touch stopped.

How to handle swipe on application?

- Swipe uses the 3 touch events
    - TouchStart
    - TouchMove
    - TouchEnd
- You can access the touch property called “onSwiped” and “onSwipping”

Ex:

App.js

```
class TouchEvent extends React.Component
{
  constructor(props) {
        super(props);
        this.state = {
        styleObj : {
        position: 'fixed',
        top: '',
        left: ''
        },
        ads : {
        position : 'fixed',
        top: '',
        left: ''
        }
        }
        this.TouchStart = this.TouchStart.bind(this);
        this.MoveAd = this.MoveAd.bind(this);
  }
  TouchStart(e){
        this.setState({
```

```
        styleObj: {
        position: 'fixed',
        left: e.touches[0].clientX + 'px',
        top: e.touches[0].clientY + 'px'
        }
        })
}
MoveAd(e){
        this.setState({
        ads: {
        position: 'fixed',
        left: e.touches[0].clientX + 'px',
        top: e.touches[0].clientX + 'px'
        }
        })
}
render(){
        return (
        <>
        <div onTouchMove={this.MoveAd} style={this.state.ads} className="card">
        <div className="card-header">
        <h3>Ads - Nike</h3>
        </div>
        <div className="card-body">
        <img src="shoe.jpg" width="50" height="50" />
        </div>
        </div>
        <div onTouchMove={this.TouchStart}>
```

```
            <img style={this.state.styleObj} src="ball.png" height="50" width="50"/>
            </div>
            </>
            )
  }
}

class MainContent extends React.Component
{
  render(){
            return (
            <div className="container-fluid">
            <TouchEvent/>
            </div>
            )
  }
}

ReactDOM.render(
  <MainContent />,
  document.getElementById('root')
);
```