EVENT HANDLING:

Just like HTML DOM events, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

ADDING EVENTS

React events are written in camelCase syntax:

onClick instead of onclick.

React event handlers are written inside curly braces:

onClick={shoot} instead of onClick="shoot()".

EXAMPLES

<button onClick={shoot}>CLICK ME</button>

APP.JS

```
import React ,{Component} from 'react';
import './App.css';

import FunctionClick from './Components/FunctionClick';
import ClassClick from './Components/ClassClick';
class App extends Component{
render() {
return(
<div className="App">
<FunctionClick></FunctionClick>
<ClassClick></ClassClick>
</div>
)
}
}
export default App;
```

FunctonClick.js

```
import React from 'react';

function FunctionClick(){

function clickHandler() {
console.log('Button clicked')
}
return (
```

```
<div>
<button onClick ={clickHandler}>Click</button>
</div>
)
}

export default FunctionClick;
```

ClassClick.js

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

class ClassClick extends Component {
clickHandler(){
console.log('Clicked the button')
}
render() {
return(
<div>
<button onClick={this.clickHandler}>Click me</button>
</div>
)
}
}
export default ClassClick;
```

BINDING EVENT HANDLERS

App.js

```
import React ,{Component} from 'react';
import './App.css';

import FunctionClick from './Components/FunctionClick';
import ClassClick from './Components/ClassClick';
import EventBind from './Components/EventBind';

class App extends Component{
render() {
return(
<div className="App">
<EventBind />
</div>
)
}
}
export default App;
```

EventBind.js

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

class EventBind extends Component {

constructor(props) {
super(props)

this.state={
message: 'Hello'
}
this.clickHandler=this.clickHandler.bind(this)
}
clickHandler(){
this.setState({
message:'Goodbye!'
})
console.log(this)
}
render() {
return(
<div>
<div>{this.state.message}</div>
<button onClick={this.clickHandler}>Click</button>
</div>
)
}
}
export default EventBind;
```

Life cycle methods:

1.Mounting: When an instance of a component is being created and inserted into the DOM

constructor,static getDerivedStateFromProps,render and componentDidMount

2.Updating : When a componnet is being re-rendered as a result of changes to either its props or state

static getDerivedStateFromProps,shouldComponentUpdate,render,getSnapshotBeforeUpdate and componentDicUpdate

3.Unmounting: When a component is being removed from the DOM

componentWillUnmount

4.Error Handling : When there is a error during rendering,in a life cycle method or in the constructor of any child component
static getDerivedStateFromError and componentDidCatch