→ Suppose we have a input box and we wish to see that what value is in there.

Ex →

```
function handleInputClick(event)
{
    console.log("Someone is typing", event.target.value);
}

function App()
{
    return (
    <>
    <h1> — — </h1>
    <input onChange={handleInputClick} />
    </>
    );
}
```

→ There are various properties inside the 'event' which is been passed as a parameter in any function.

To learn more, we can simply google out 'Synthetic event - React'.

How can we use event handlers in Class Components →

Ex →

```
import { Component } from 'react';
class App extends Component {
 handleInputClick (event)
{
 Console.log ("Someone is typing", event.target.
                                    value);
}

handleFocus ()
{
Console.log (" Input in focus");
}

render ()
{
 return (
   <>
   <Heading />

   <input onChange = {this.handleInputClick}
          onFocus = {this.handleFocus} />

   </>
);
}

export default APP;
```

* States in Class Components →

→ States are very important part in react.
→ All the re-rendering we see on the page is due to the change in states.

→ Suppose, for an example →

If we want to have an add button on the screen and on the click of that button I want the output to increase by 1 and to be rendered on the screen.

Ex

```
Class App extends Component {

  state = {
      Counter: 1 ;
  };

  handleClick = () => {

    let localCounter = this.state.Counter;
    this.setState ({
        Counter: localCounter + 1
    })
  }

    render () {
      return (<><h1> Welcome </h1>
      <button onClick = { this.handleClick }> ADD </button>
      <p> { this.state.Counter } </p>
      </>); } }
```

1st Way

→ So, if we use { () ⇒ } function then automatically the 'this' keyword inside gets defined and it ~~refers~~ basically corresponds to the 'App' class, so whatever will be the instance of the 'App class' will get initialized with that instance.

2nd Way We can use 'bind' method.

In the example we saw before, what else can be done is instead of using 'arrow function' we can use the normal function as well.

we can do →

```
handleClick() {
    let localCounter = this.state.counter;
}
render()
  return (
    <>
    ———
    ———
    <button onClick={this.handleClick.bind(this)}>
      Add </button>
    <———————>
    </>
);
```

~~so, in bind the this of bind will~~ [instance of this will bind with the 'this' above]

3rd Way →

Simply we can use 'event' and pass that as a parameter in arrow function.

→ handleClick(event) {

}

render()
{
return
(
<>
<button onClick={(event) ⇒ this.handleClick(event)} ⇒ Add </button>
</>
);
}

4th Way → We can do the same by a 4th way in which we use constructors, basically we use constructors for the purpose of initialization.

Which is most preferably not advised to use as the same code is been distributed at 3-4 different places.

→ If I want my heading to be dynamic and to show the text inserted inside an input box below it in an <p> element then what can be done is →

Ex

```
class App extends Component
{
    state = { programCounter: "1"
      heading: "Welcome to React",
      inputText: " ".
    };

handleClick = () => {
    let counter = this.state.programCounter;
    this.setState ({
      programCounter: counter+1,
      heading: "You clicked" , }) }

handleInput = (event) => {
        this.setState ({
              inputText: event.target.value }) }

render ()
{
  return ( <>
        <h1> {this.state.heading} </h1>
    <button onClick={this.handleClick}> ADD </button>
    <p> {this.state.programCounter} </p>
   <input onChange= { this.handleInput } />
  <p> {this.state.inputText} </p>
    </> ); }}}
export default App;
```