class ControlledInput extends React.Component {

constructor(props) {

super(props);

this.state = { name: '' };

this.handleInput = this.handleInput.bind(this);

}

handleInput(event) {

this.setState({

name: event.target.value

});

}

render() {

return (

<input type="text" value={this.state.name} onChange={this.handleInput} />

);

}

}

We set up a kind of circular data flow: state to input value, on change event to state, and back again. This loop allows us a lot of control over the input, as we can react to changes to the value on the fly. Because of this, controlled inputs don’t suffer from the limitations of uncontrolled ones, opening up the follow possibilities:

**https://github.com/braintree/credit-card-type**

rcc > Gives code for Component

import React, { Component } from 'react'

export default class ControlledComponent extends Component {

    render() {

        return (

            <div>

            </div>

        )

    }

}

import React, { Component } from 'react'

export default class ControlledComponent extends Component {

    render() {

        return (

            <div>

                <form>

    <h2> Controlled By React </h2>

                    <input type="text" value="Practice"></input>

                </form>

            </div>

        )

    }

}

Index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json provides metadata used when your web app is installed on a

      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <h2> Not Controlled By React </h2>

    <input type="text" value="Practice"></input>

    <!--

      This HTML file is a template.

      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.

      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.

      To create a production bundle, use `npm run build` or `yarn build`.

    -->

  </body>

</html>

We can use defaultValue=”Practice”

React will control values or states

We are using state

import React, { Component } from 'react'

export default class ControlledComponent extends Component {

    state =

    {

     value:"Practice"

    }

    render() {

        return (

            <div>

                <form>

    <h2> Controlled By React </h2>

                    <input type="text" value={this.state.value}></input>

                </form>

            </div>

        )

    }

}

Added function on onChange Event

import React, { Component } from 'react'

export default class ControlledComponent extends Component {

    state =

    {

     value:"Practice"

    }

    handleChange =()  =>

    {

        console.log("Change");

    }

    render() {

        return (

            <div>

                <form>

    <h2> Controlled By React </h2>

                    <input type="text" value={this.state.value}

                   onChange={this.handleChange} ></input>

                </form>

            </div>

        )

    }

}

import React, { Component } from 'react'

export default class ControlledComponent extends Component {

    state =

    {

     value:"Practice"

    }

    handleChange =(event)  =>

    {

        console.log("Change");

        console.log(event);

        console.log(event.target.value);

    }

    render() {

        return (

            <div>

                <form>

    <h2> Controlled By React </h2>

                    <input type="text" value={this.state.value}

                   onChange={this.handleChange} ></input>

                </form>

            </div>

        )

    }

}

import React, { Component } from 'react'

export default class ControlledComponent extends Component {

    state =

    {

     value:"Practice"

    }

    handleChange = event  =>

    {

        console.log("Change");

        console.log(event);

        console.log(event.target.value);

        this.setState({value: event.target.value})

    }

    render() {

        return (

            <div>

                <form>

    <h2> Controlled By React </h2>

                    <input type="text" value={this.state.value}

                   onChange={this.handleChange} ></input>

                </form>

            </div>

        )

    }

}

this.setState({value: event.target.value.toUpperCase()})

        this.setState({value: event.target.value.toUpperCase().substr(0,10)})

Add Numbers

import React, { Component } from 'react'

export default class AddNumbers extends Component {

  constructor(props)

  {

      super(props);

      this.state=

      {

          n1 : 0,

          n2:  0,

          res: 0

      }

    }

    handle1 = event=>

    {

this.setState({n1: event.target.value})

    }

    handle2 = event=>

    {

this.setState({n2: event.target.value})

    }

 add = event => {

     event.preventDefault();

       this.setState({res :parseInt(this.state.n1) + parseInt(this.state.n2)});

  }

    render() {

        return (

            <div>

                <form onSubmit={this.add}>

                   No1:   <input type="number" value={this.state.n1} onChange={this.handle1}/>

                    <br/>

              No2:      <input type="number"  value={this.state.n2} onChange={this.handle2}/>

              <br/>  Result : <input type="number" value={this.state.res}  />

              <button type="submit">Add</button>

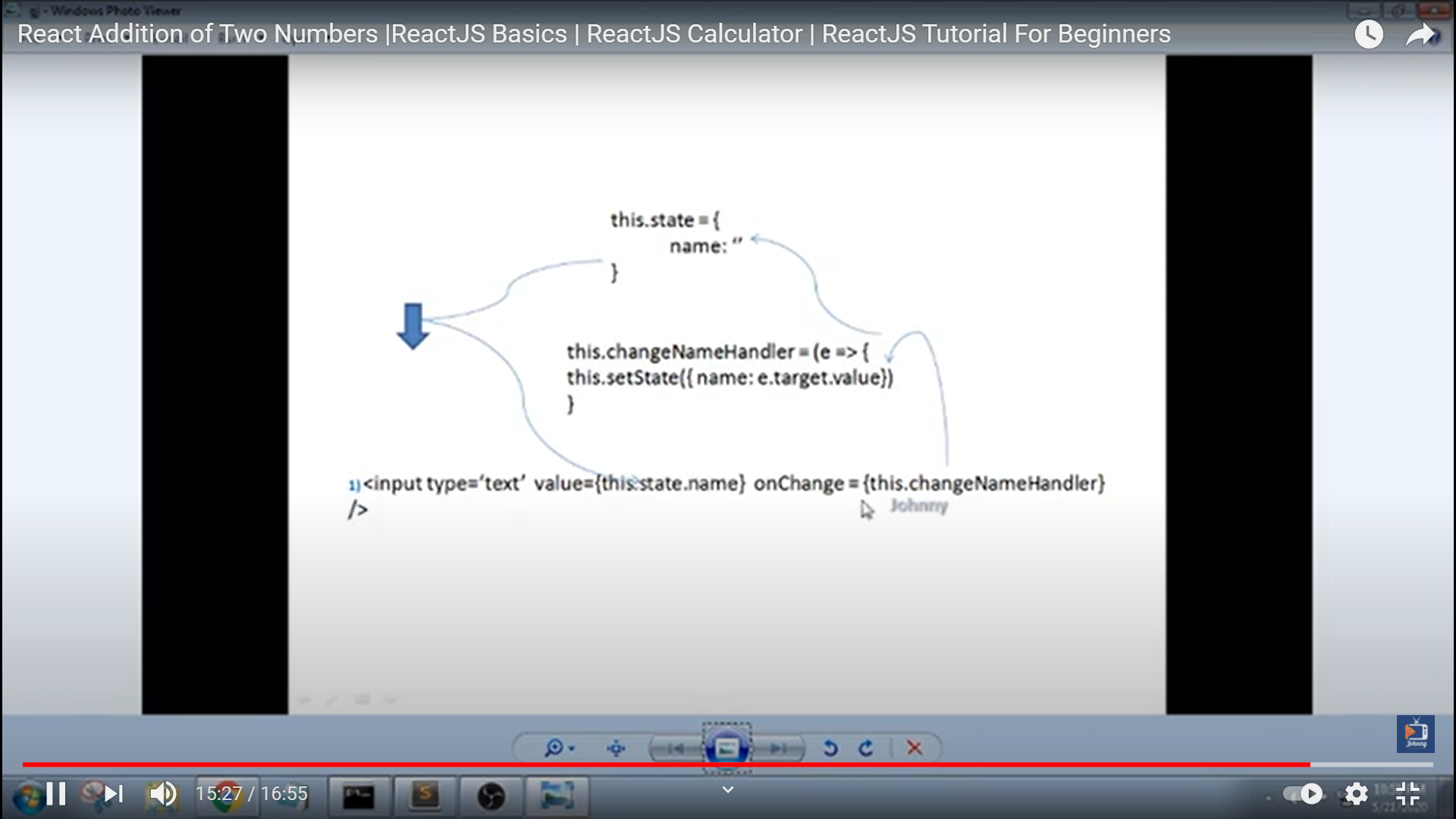
                </form>

            </div>

        )

    }

}



Using one event for multiple input elements

import React, { Component } from 'react'

export default class AddNumbersWithEvent extends Component {

  constructor(props)

  {

      super(props);

      this.state=

      {

          n1 : 0,

          n2:  0,

          res: 0

      }

    }

    handle1 = event=>

    {

        console.log(event.target.name);

        console.log(event.target.value);

        this.setState({[event.target.name]:event.target.value})

    }

    add = event => {

     event.preventDefault();

     this.setState({res :parseInt(this.state.n1) + parseInt(this.state.n2)});

  }

    render() {

        return (

            <div>

                <form onSubmit={this.add}>

                   No1:   <input type="number" value={this.state.n1} name="n1" onChange={this.handle1}/>

                    <br/>

              No2:      <input type="number"  value={this.state.n2} name ="n2" onChange={this.handle1}/>

              <br/>  Result : <input type="number" value={this.state.res}  />

              <button type="submit">Add</button>

                </form>

            </div>

        )

    }

}

import React , {Component} from 'react';

export default class MyForm extends React.Component {

    constructor(props) {

      super(props);

      this.state = { username: '' };

    }

    myChangeHandler = (event) => {

        this.setState({username: event.target.value});

        event.preventDefault();

    }

    render() {

      let header = '';

      if (this.state.username) {

        header = <h1>Hello {this.state.username}</h1>;

      } else {

        header = <h1>No Name </h1>;

      }

      return (

        <form>

        {header}

        <p>Enter your name:</p>

        <input

          type='text'

          onChange={this.myChangeHandler}

        />

        </form>

      );

    }

  }