

**Date :** 23 - 07 - 2020

**Morning Session :** 9 am – 11.00 PM

**By ~** Sundeep Charan Ramkumar Today

## Topics: React Refs

### What are React Refs?

Refs provide a way to access and manipulate DOM nodes and React elements , directly.

**Link:** <https://reactjs.org/docs/refs-and-the-dom.html>

```
index.html > html > body > div#root
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8" />
5    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6    <title>Document</title>
7  </head>
8  <body>
9    <div id="root"></div>
10   <script
11     crossorigin
12     src="https://unpkg.com/react@16/umd/react.development.js"
13   ></script>
14   <script
15     crossorigin
16     src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
17   ></script>
18   <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
19   <script type="text/babel" src="./app.js"></script>
20 </body>
21 </html>
22
```

```

JS app.js > Something
1 // Custom code.
2 class Something extends React.Component {
3   constructor(props) {
4     super(props);
5     this.inputRef = React.createRef();
6   }
7   (method) Something.componentDidMount(): void
8   componentDidMount() {
9     console.log(this.inputRef);
10  }
11
12  componentDidUpdate() {
13    console.log("Updating");
14  }
15
16  handleClick = event => {
17    const inputElement = this.inputRef.current;
18    inputElement.focus();
19  };
20
21  render() {
22    return (
23      <div>
24        Hi I am using Refs.
25        <input type="text" name="name" ref={this.inputRef} />
26        <button onClick={this.handleClick}>Click me</button>
27      </div>
28    );

```

React Ref is like an anti-pattern and follows an imperative Approach instead of the React's Declarative Approach

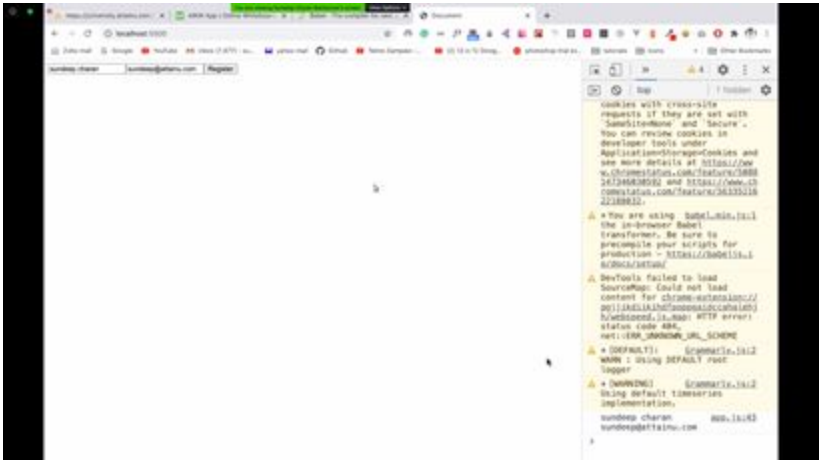
We can avoid Render in spite of changing the State with Refs.

```
1 // Custom code
2 class Something extends React.Component {
3   constructor(props) {
4     super(props);
5     this.inputRef = React.createRef();
6   }
7
8   componentDidMount() {
9     console.log(this.inputRef);
10  }
11
12  // (method) Something.componentDidUpdate(): void
13  componentDidUpdate() {
14    console.log("Updating");
15  }
16
17  handleClick = event => {
18    const inputElement = this.inputRef.current;
19    inputElement.focus();
20  };
21
22  render() {
23    return (
24      <div>
25        Hi I am using Refs.
26        <input type="text" name="name" ref={this.inputRef} />
27        <button onClick={this.handleClick}>Click me</button>
28      </div>
29    );
30  }
31 }
```

Refs usage in Forms → Bad habit

```
32 class RegisterForm extends React.Component {
33   constructor(props) {
34     super(props);
35     this.nameRef = React.createRef();
36     this.emailRef = React.createRef();
37   }
38
39   handleSubmit = event => {
40     event.preventDefault();
41     const inputElement = this.nameRef.current;
42     const emailElement = this.emailRef.current;
43     console.log(inputElement.value, emailElement.value);
44   };
45
46   render() {
47     return (
48       <form onSubmit={this.handleSubmit}>
49         <input type="text" name="name" ref={this.nameRef} />
50         <input type="email" name="email" ref={this.emailRef} />
51         <input type="submit" value="Register" />
52       </form>
53     );
54   }
55 }
56
57 // App component
58 const App = () => {
```

This gives rise to uncontrolled Component:

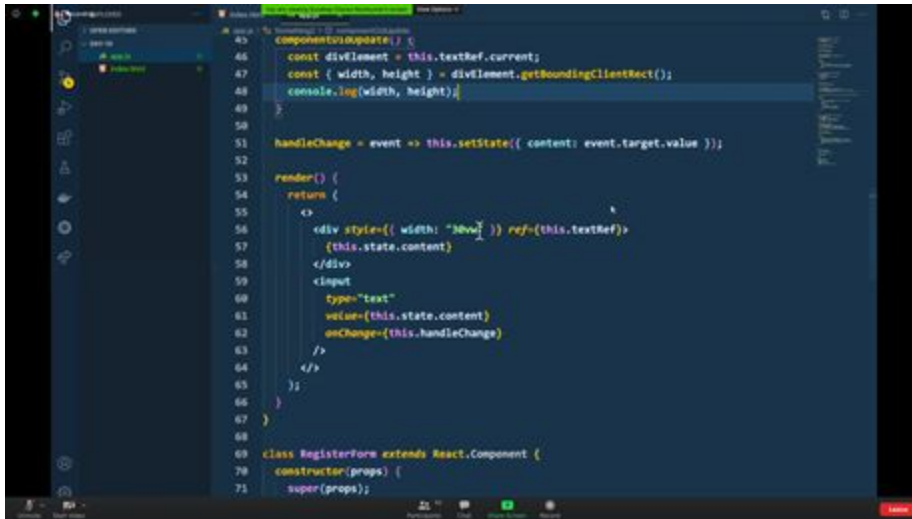


## Note :

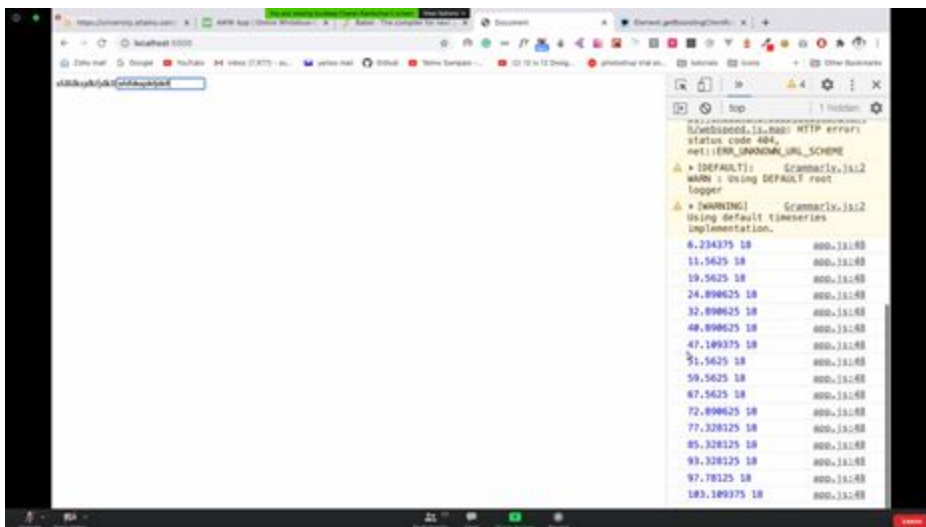
- Refs are a thing of the past and shouldn't be used unless you know what you are doing.

## Another use case:

Paragraphs(we can find the width and height and do some logic)

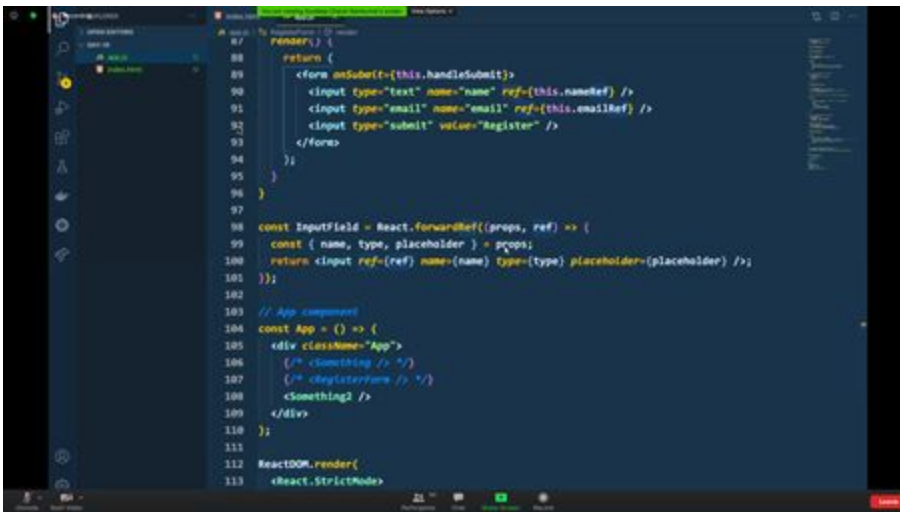


```
45 componentWillUpdate() {  
46   const divElement = this.textRef.current;  
47   const { width, height } = divElement.getBoundingClientRect();  
48   console.log(width, height);  
49 }  
50  
51 handleChange = event => this.setState({ content: event.target.value });  
52  
53 render() {  
54   return (  
55     <>  
56     <div style={{ width: "300px" }} ref={this.textRef}>  
57       {this.state.content}  
58     </div>  
59     <input  
60       type="text"  
61       value={this.state.content}  
62       onChange={this.handleChange}  
63     />  
64     </>  
65   );  
66 }  
67  
68  
69 class RegisterForm extends React.Component {  
70   constructor(props) {  
71     super(props);
```



## Ref Forwarding:

- The ability to make sure that a child gets the parent ref prop is called ref Forwarding

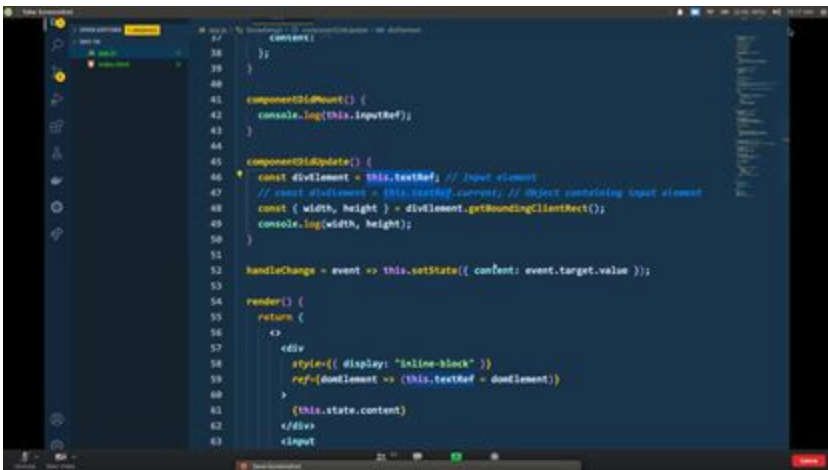


```
17 render() {
18   return (
19     <form onSubmit={this.handleSubmit}>
20       <input type="text" name="name" ref={this.nameRef} />
21       <input type="email" name="email" ref={this.emailRef} />
22       <input type="submit" value="Register" />
23     </form>
24   )
25 }
26
27 // App component
28 const App = () => {
29   <div className="App">
30     { /* <Something /> */ }
31     { /* <RegisterForm /> */ }
32     <Something2 />
33   </div>
34 };
35
36 ReactDOM.render(
37   <React.StrictMode>
```

## Understanding Ref Forwarding in React

In simple words, Refs helps to get /target any DOM element, without the need for Rendering

This is another way to add ref: directly referring Dom element as this.ref



```
38   }
39 }
40
41 componentDidMount() {
42   console.log(this.inputRef);
43 }
44
45 componentDidUpdate() {
46   // const divElement = this.textRef // input element
47   // const divElement = this.textRef.current // Object containing input element
48   const { width, height } = divElement.getBoundingClientRect();
49   console.log(width, height);
50 }
51
52 handleChange = event => this.setState({ content: event.target.value });
53
54 render() {
55   return (
56     <div
57       style={{ display: "inline-block" }}
58       ref={divElement => (this.textRef = divElement)}
59     >
60       {this.state.content}
61     </div>
62     <input
```

## Resources:

<https://www.youtube.com/watch?v=FXa9mMTKOu8>

<https://medium.com/@mrewusi/a-gentle-introduction-to-refs-in-react-f407101a5ea6>

