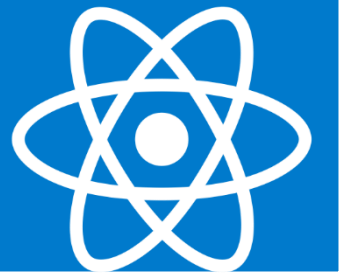
Controlled vs Uncontrolled Components in React

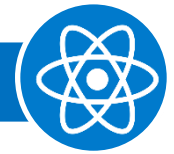
#React Notes

Controlled vs Uncontrolled Components in React



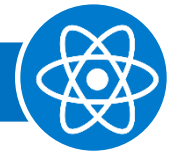
controlled component

- In a controlled component, the form data is handled by the state *within the component*.
- The state within the component serves as *"the single source of truth"* for the input elements that are rendered by the component.
- Controlled components have functions that govern the data passing into them on every onChange event occurs.
- This data is then saved to state and updated with setState() method.
- It makes component have better control over the form elements and data.



Controlled vs Uncontrolled

Controlled	Uncontrolled
It does not maintain its internal state.	It maintains its internal states.
Here, data is controlled by the parent component.	Here, data is controlled by the DOM itself.
It accepts its current value as a **prop**.	It uses a **ref** for their current values.
It allows validation control.	It does not allow validation control.
It has better control over the form elements and data.	It has limited control over the form elements and data.



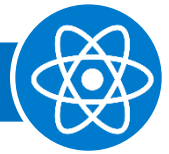
App.js M

src > App.js > default

```
1  import React, { Component } from 'react';
2  class App extends Component {
3      state = {
4          message: ''
5      }
6      updateMessage = (newText) => {
7          console.log(newText);
8          this.setState(() => ({
9              message: newText
10         }));
11     }
12     render() {
13         return (
14             <div className="App">
15                 <div className="container">
16                     <input type="text"
17                         placeholder="Your message here.."
18                         value={this.state.message}
19                         onChange={(event) => this.updateMessage(event.target.value)}
20                     />
21                     <p>the message is: {this.state.message}</p>
22                 </div>
23             </div>
24         );
25     }
26 }
27 export default App;
```

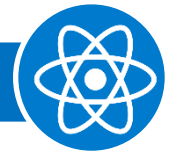
the message is:

```
import React, { Component } from 'react';
class App extends Component {
   state = {
      message: ''
   }
   updateMessage = (newText) => {
      console.log(newText);
      this.setState(() => ({
         message: newText
      }));
   }
   render() {
      return (
         <div className="App">
            <div className="container">
               <input type="text"
                  placeholder="Your message here.."
                  value={this.state.message}
                  onChange={(event) => this.updateMessage(event.target.value)}
               />
               <p>the message is: {this.state.message}</p>
            </div>
         </div>
      );
   }
}
export default App;
```

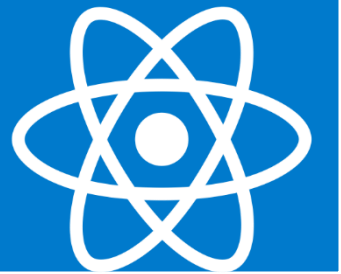


In Previous Example

- The textbox has a value attribute bound to the message property in the state.
- We have an onChange. event handler declared
- When changes are made to any of the input elements that have an *event handler*, the handler is fired.
- The handler calls setState() as you can see in line 9 above. This updates the state within the component.
- When a state update occurs via setState(), it causes the component to re-render and the newly entered value is displayed in the element.
- The data flow is uni-directional from the component state to the input element.

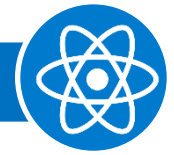


Uncontrolled Components



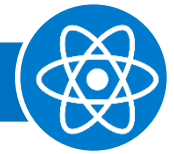
Uncontrolled Components

- Uncontrolled components act more like traditional HTML form elements.
- The **data for each input element is stored in the DOM**, not in the component.
- Instead of writing an event handler for all of your state updates, you use a ***ref* to retrieve values from the DOM**.
- *Refs provide a way to access DOM nodes or React elements created in the render method.*
- https://reactjs.org/docs/refs-and-the-dom.html



Syntax

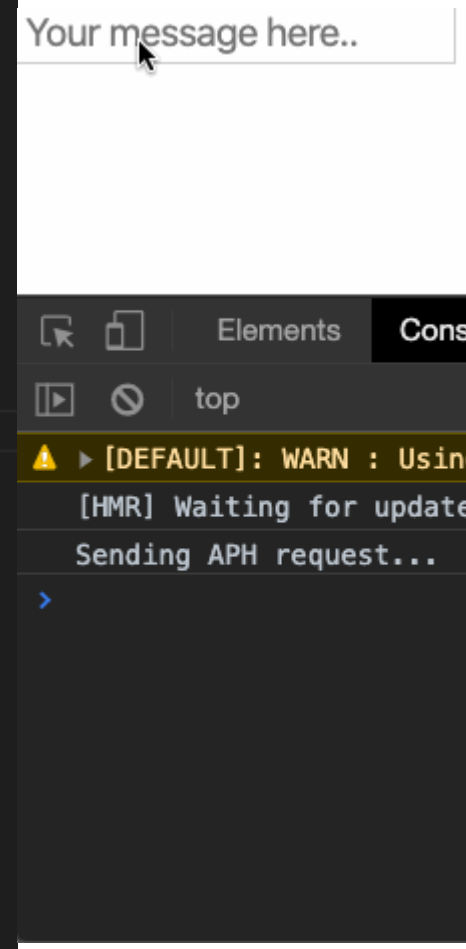
- Create Reference
  - this.input = React.createRef();
- Get Value
  - ref={this.input}



App.js M

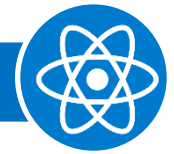
src > App.js > App > render

```
1  import React, { Component } from 'react';
2  class App extends Component {
3      constructor(props){
4          super(props);
5          this.handleChange = this.handleChange.bind(this);
6          this.input = React.createRef();
7      }
8
9      handleChange = (newText) => {
10         console.log(newText);
11     }
12     render() {
13         return (
14             <div className="App2">
15                 <div className="container">
16                     <input type="text"
17                         placeholder="Your message here.."
18                         ref={this.input}
19                         onChange={(event) => this.handleChange(event.target.value)}
20                     />
21                 </div>
22             </div>
23
24         );
25     }
26 }
27 export default App;
```



Ref Details

- Refs are created using React.createRef().
- Refs are attached to input elements using the *ref* attribute on the element in question.
- Refs are often used as instance properties on a component. The ref is set in the constructor (as shown above) and the value is available throughout the component.
- You cannot use the ***ref* attribute *on* functional components because an instance is not created.**



Controller Vs UnController Component

App.js > default

```
import React, { Component } from 'react';
class App extends Component {
    state = {
        message: ''
    }
    updateMessage = (newText) => {
        console.log(newText);
        this.setState(() => ({
            message: newText
        }));
    }
    render() {
        return (
            <div className="App">
                <div className="container">
                    <input type="text"
                        placeholder="Your message here.."
                        value={this.state.message}
                        onChange={(event) => this.updateMessage(event.target.value)}
                    />
                    <p>the message is: {this.state.message}</p>
                </div>
            </div>
        );
    }
}
export default App;
```

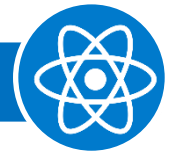
App.js M

src > App.js > App > render

```
1  import React, { Component } from 'react';
2  class App extends Component {
3      constructor(props){
4          super(props);
5          this.handleChange = this.handleChange.bind(this);
6          this.input = React.createRef();
7      }
8
9      handleChange = (newText) => {
10         console.log(newText);
11     }
12     render() {
13         return (
14             <div className="App2">
15                 <div className="container">
16                     <input type="text"
17                         placeholder="Your message here.."
18                         ref={this.input}
19                         onChange={(event) => this.handleChange(event.target.value)}
20                     />
21                 </div>
22             </div>
23
24         );
25     }
26 }
27 export default App;
```

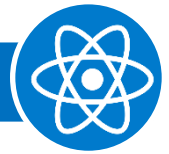
Conclusion

- Use controlled components whenever possible.
- Controlled components do not require a form element in order to be considered a controlled component.
- If a component has an input element that has a value attribute bound to state and an event handler to update said state, it is a controlled component.
- For pages that have a large number of input elements, it can be cumbersome (manageable) to work with controlled components.
- All state changes within a controlled component should be made via the setState function.
- Uncontrolled components store their data in the DOM like a traditional HTML input element.

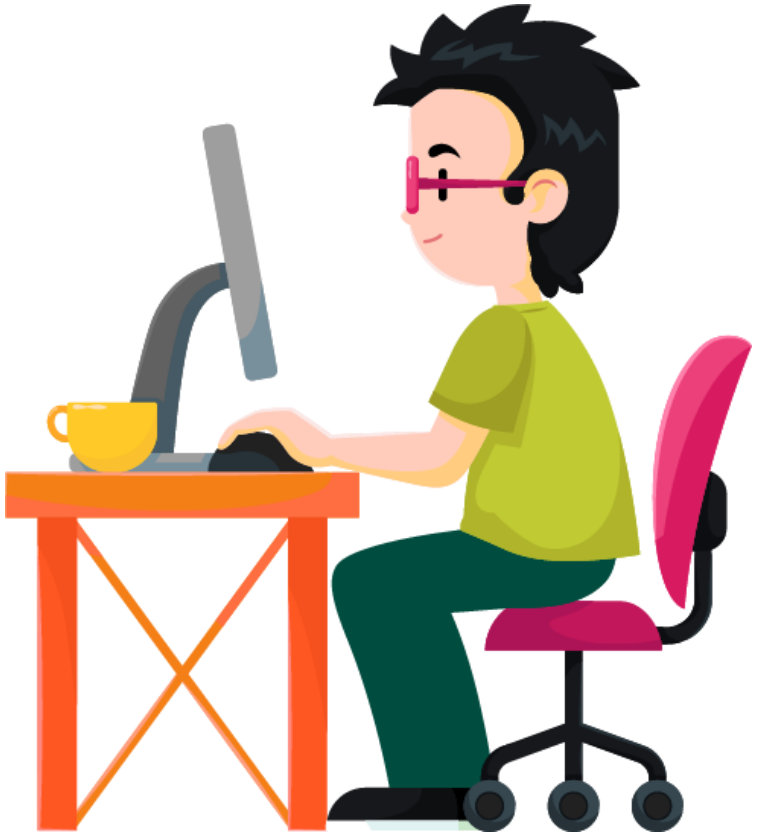


Conti…

- React.createRef() is used to create instance variables within uncontrolled component constructors. These variables are then associated with input elements via the *ref* attribute.
- **Refs cannot be used *on* functional components** as there is no instance.
- Refs can be used ***inside*** class components.



Get Exclusive Video Tutorials



www.aptutorials.com

https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

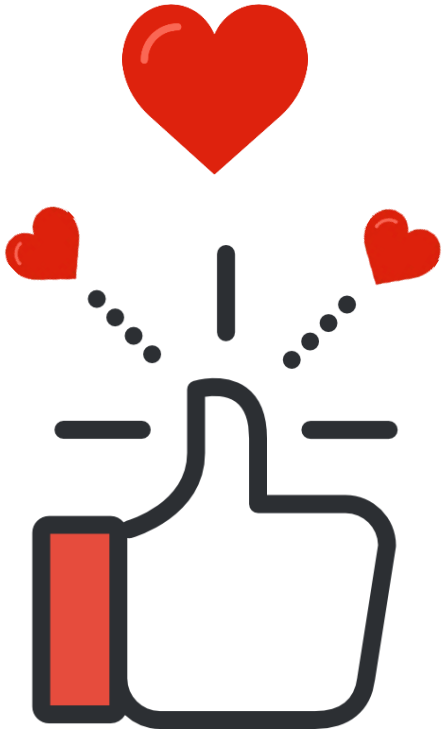
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials