

strategy | consulting | digital | **technology** | operations

REACTJS – MODULE 8

INTRODUCTION TO FORMS



AGENDA

- Introduction to react forms
- How to create forms in react
- Controlled component
- Uncontrolled component
- Demo
- Activity



Introduction to Forms

In react, forms work in a different way compared to HTML forms

HTML form elements will maintain their own state.
And these will allow a user to perform update whenever it is needed based on the user input

In react forms elements such as textbox, textarea etc. does not allow a user to update state directly, instead it has to be modified only using `setState()`

This means react components are also responsible to control/manage interactions between form elements and user inputs. Hence react components are also known as controlled components

Introduction to Forms – Test

Create Formtest component(Formtest.js)

```
import React, {Component} from 'react';
class Formtest extends Component{
  render() {
    return (
      <div>
        <h2>Forms Example</h2>
        <input type="text" value="some text"/>
        <input type="submit" value="Submit" />
      </div>
    )
  }
}
export default Formtest;
```

Output

Forms Example



The screenshot shows a web page with a single form element. It consists of a text input field with the placeholder "some text" and a submit button labeled "Submit". The input field is highlighted with a blue border.

Introduction to Forms – Test

Create Formtest component(Formtest.js)

```
import React, {Component} from 'react';
class Formtest extends Component{
  render() {
    return (
      <div>
        <h2>Forms Example</h2>
        <input type="text" value="some text"/>
        <input type="submit" value="Submit" />
      </div>
    )
  }
  export default Formtest;
```

Output

Forms Example



Try to modify the textbox
value and you will observe you
cannot modify the value

Introduction to Forms (Cont...)

By the previous example it is very clear that by default react will not allow a user to modify forms elements directly

To modify the state of form elements react will use `setState()`

It is also suggested that in order to modify the state of an individual element an associated handler function needs to be created

Demo – React Form

Inside App.js

```
import React, { Component } from 'react';
class App extends Component {
  constructor() {
    super();
    this.state = {
      customerName: ''
    };

    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({customerName: event.target.value});
  }

  handleSubmit(event) {
    alert('Customer Name is: ' + this.state.customerName);
    event.preventDefault();
  }
}
```

Demo – React Form(Cont...)

Inside App.js

```
render() {
  return (
    <div>
      <br/>
      <br/>
      <form onSubmit={this.handleSubmit}>
        Enter Customer Name:
        <input type="text" value={this.state.customerName}
               onChange={this.handleChange} />
        <input type="submit" value="Submit" />
      </form>
    </div>
  );
}
export default App;
```

Demo – React Form - Explanation

Inside App.js

```
import React, { Component } from 'react';
class App extends Component {
  constructor() {
    super();
    this.state = {
      customerName: ''
    };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({customerName: event.target.value});
  }
  handleSubmit(event) {
    alert('Customer Name is: ' + this.state.customerName);
    event.preventDefault();
  }
}
```

“customerName” is a state of a component and initially it is empty

Event handler are bound inside the constructor

Demo – React Form –Explanation(Cont..)

Inside App.js

```
render() {  
  return (  
    <div>  
      <br/>  
      <br/>  
      <form onSubmit={this.handleSubmit}>  
        Enter Customer Name:  
        <input type="text" value={this.state.customerName}  
          onChange={this.handleChange} />  
        <input type="submit" value="Submit" />  
      </form>  
    </div>  
  );  
}  
  
export default App;
```

The state “customerName” is captured here but initially it is empty. If the user wants he will modify it by entering some data inside the textbox

The entered data is captured using onChange() event and associated handler function i.e “handleChange” is invoked

Demo – React Form - Explanation (Cont..)

Inside App.js

```
import React, { Component } from 'react';
class App extends Component {
  constructor() {
    super();
    this.state = {
      customerName: ''
    };

    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({customerName: event.target.value});
  }

  handleSubmit(event) {
    alert('Customer Name is: ' + this.state.customerName);
    event.preventDefault();
  }
}
```

Using the handler function the state “customerName” is modified inside setState()

Demo – React Form –Explanation(Cont..)

Inside App.js

```
render() {
  return (
    <div>
      <br/>
      <br/>
      <form onSubmit={this.handleSubmit}>
        Enter Customer Name:
        <input type="text" value={this.state.customerName}
               onChange={this.handleChange} />
        <input type="submit" value="Submit" />
      </form>
    </div>
  );
}

export default App;
```

After textbox is modified by the user he will click on submit button, which will submit the form. onSubmit of the form the handler function i.e “handleSubmit” is invoked

Demo – React Form - Explanation (Cont..)

Inside App.js

```
import React, { Component } from 'react';
class App extends Component {
  constructor() {
    super();
    this.state = {
      customerName: ''
    };

    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({customerName: event.target.value});
  }

  handleSubmit(event) {
    alert('Customer Name is: ' + this.state.customerName);
    event.preventDefault();
  }
}
```

This will display alert with
customerName entered inside
the textbox

Demo – React Form

Initial Output

Enter Customer Name: Submit

User enters data in the textbox

After clicking on submit button

Enter Customer Name: Submit

localhost:3000 says:

Customer Name is: Mark Samuel

OK

Activity

1. Create a new component with dropdown list which will display list of country names
2. Give a choice to user to select the country
3. Display the selected country using alert

Activity - Expected Output

Initial Output

Select your country: India ▾ Submit

After user selects a country

Select your country: Dubai ▾ Submit

After user clicks on submit button

localhost:3000 says:

Selected country is : Dubai

OK

React Forms – Uncontrolled Components

In the previous example where the form data is completely controlled by react component by writing different event handlers

This may be drawback as program will become lengthy due to many event handlers

The alternate to this approach is uncontrolled component

In an uncontrolled component the form data is controlled/managed by DOM, in this case we can use “refs” to retrieve form values from the DOM

Demo – Uncontrolled Component

Create new component Empdata(Empdata.js)

```
import React, { Component } from 'react';
class Empdata extends Component {
  handleSubmit(event) {
    alert('Customer Name is: ' + this.refs.cname.value+ ' and ' +
          'Customer Email is: '+this.refs.cemail.value);
    event.preventDefault();
  }
  render() {
    return (
      <div>
        <form onSubmit={()=>this.handleSubmit()}>
          Enter Customer Name:
          <input type="text" ref="cname"/><br/><br/>
          Enter Customer Email:
          <input type="email" ref="cemail"/><br/>
          <input type="submit" value="Submit" />
        </form>
      </div>
    );
  }
}
export default Empdata;
```

Demo – Uncontrolled Component - Explanation

Create new component Empdata(Empdata.js)

```
import React, { Component } from 'react';
class Empdata extends Component {
  handleSubmit(event) {
    alert('Customer Name is: ' + this.refs.cname.value + ' and ' +
      'Customer Email is: ' + this.refs.cemail.value);
    event.preventDefault();
  }
  render() {
    return (
      <div>
        <form onSubmit={()=>this.handleSubmit()}>
          Enter Customer Name:  

          <input type="text" ref="cname"/><br/><br/>
          Enter Customer Email:  

          <input type="email" ref="cemail"/><br/>
          <input type="submit" value="Submit" />
        </form>
      </div>
    );
  }
  export default Empdata;
```

Ref is used here using which
the textbox can be referred
from the DOM

Demo – Uncontrolled Component - Explanation

Create new component Empdata(Empdata.js)

```
import React, { Component } from 'react';
class Empdata extends Component {
  handleSubmit(event) {
    alert('Customer Name is: ' + this.refs.cname.value + ' and ' +
      'Customer Email is: ' + this.refs.cemail.value);
    event.preventDefault();
  }
  render() {
    return (
      <div>
        <form onSubmit={()=>this.handleSubmit()}>
          Enter Customer Name:
          <input type="text" ref="cname"/><br/><br/>
          Enter Customer Email:
          <input type="email" ref="cemail"/><br/>
          <input type="submit" value="Submit" />
        </form>
      </div>
    );
  }
  export default Empdata;
```

Refs is used here to capture modified value of “cname” and “cemail” from the DOM

Demo – Uncontrolled Component

Initial Output

Enter Customer Name:

Enter Customer Email:

Submit

After user enters name and email

Enter Customer Name:

Enter Customer Email:

Submit

After user clicks on submit button

localhost:3000 says:

Customer Name is: Mark Samuel and Customer Email is: Mark.s@gmail.com

OK

MODULE SUMMARY

- Understand how forms works in react
- Controlled components
- Uncontrolled components



THANK YOU

