React Forms

45th /100 **Day**

Just like in HTML, React uses forms to allow users to interact with the web page. You add a form with React like any other element

Example: A for that allows users to enter their names.

```
class MyForm extends React.Component {
  render() {
    return (
      <form>
        <h1>Hello</h1>
        <label>Enter your name:</label>
         <br />
        <input type="text" required />
      </form>
    );
  }
}
ReactDOM.render(<MyForm />, document.getElementById('root'));
```

Handling Forms.

Handling forms is about how you handle the data when it changes value or gets submitted. In HTML, form data is usually handled by the DOM whileIn React, form data is usually handled by the components. When the data is handled by the components, all the data is stored in the component state.

You can control changes by adding event handlers in the onChange attribute.

Example: Add an event handler in the onChange attribute, and let the event handler update the state object:

```
class MyForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = { username: '' };
  }
  myChangeHandler = (event) => {
    this.setState({username: event.target.value});
  }
  render() {
    return (
      <form>
      <h1> Hello {this.state.username} </h1>
      <label> Enter your name: </label>
     <br />
      <input type='text' onChange={this.myChangeHandler} />
      </form>
    );
  }
}

ReactDOM.render(<MyForm />, document.getElementById('root'));
```

Two thinks to note: You must initialize the state in the constructor method before you can use it. You get access to the field value by using the event.target.value syntax.

Conditional Rendering

If you do not want to display the h1 element until the user has done any input, you can add an if statement.

Look at the example below and note the following:

1. We create an empty variable, in this example we call it header.

2. We add an if statement to insert content to the header variable if the user has done any input.

3. We insert the header variable in the output, using curly brackets.

Example:

```
class MyForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = { username: '' };
}
  myChangeHandler = (event) => {
    this.setState({username: event.target.value});
  }
  render() {
    let header = '';
    if (this.state.username) {
      header = <h1>Hello {this.state.username}</h1>;
    } else {
      header = '';
 }
    return (
      <form>
      {header}
      <label>Enter your name: </label> <br />
      <input type='text' onChange={this.myChangeHandler} />
      </form>
    );
}
}
ReactDOM.render(<MyForm />, document.getElementById('root'));
```

Submitting Forms.

You can control the submit action by adding an event handler in the onSubmit attribute:

Example: Add a submit button and an event handler in the onSubmit attribute:

```
class MyForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = { username: '' };
  }
  mySubmitHandler = (event) => {
    event.preventDefault();
    alert("You are submitting " + this.state.username);
  }
  myChangeHandler = (event) => {
    this.setState({username: event.target.value});
  }
  render() {
    return (
      <form onSubmit={this.mySubmitHandler}>
      <h1>Hello {this.state.username}</h1>
      <label>Enter your name, and submit: </label> <br />
      <input type='text' onChange={this.myChangeHandler} />
      <input
        type='submit'
      />
      </form>
    );
  }}
ReactDOM.render(<MyForm />, document.getElementById('root'));
```

Note that we use event.preventDefault() to prevent the form from actually being submitted.

Multiple Input Fields

You can control the values of more than one input field by adding a name attribute to each element. When you initialize the state in the constructor, use the field names. To access the fields in the event handler use the event.target.name and event.target.value syntax. To update the state in the this.setState method, use square brackets [bracket notation] around the property name.

Example: Write a form with two input fields:

```
class MyForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      username: '',
      age: null,
    }; }
  myChangeHandler = (event) => {
    let nam = event.target.name;
    let val = event.target.value;
    this.setState({[nam]: val});
  }
  render() {
    return (
      <form>
      <h1>Hello {this.state.username} {this.state.age}</h1>
      <p>Enter your name:</p>
      <input type='text' name='username'
             onChange={this.myChangeHandler} />
      <p>Enter your age:</p>
      <input type='text' name='age'
        onChange={this.myChangeHandler}/>
      </form>
    ); }}
ReactDOM.render(<MyForm />, document.getElementById('root'));
```

Note that in the above example we use the same event handler function for both input fields, we could write one event handler for each, but this gives us much cleaner code and is the preferred way in React.

Textarea

The textarea element in React is slightly different from ordinary HTML.

In HTML the value of a textarea was the text between the start tag <textarea> and the end tag </textarea>, in React the value of a textarea is placed in a value attribute:

Example: A simple textarea with some content initialized in the constructor:

```
class MyForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      description: 'The content of a textarea goes in the value
attribute'
    };
  }
  render() {
    return (
      <form>
      <textarea value={this.state.description} />
      </form>
    );
  }
}

ReactDOM.render(<MyForm />, document.getElementById('root'));
```

Select.

A drop down list, or a select box, in React is also a bit different from HTML. In HTML, the selected value in the drop down list was defined with the selected attribute

Example: HTML5

```
<select>
  <option value="Ford">Ford</option>
  <option value="Volvo" selected>Volvo</option>
  <option value="Fiat">Fiat</option>
</select>
```

Example: Reactjs

```
class MyForm extends React.Component {

constructor(props) {
    super(props);
    this.state = {
      mycar: 'Volvo'
    };
  }
  render() {
    return (
      <form>
      <select value={this.state.mycar}>
        <option value="Ford">Ford</option>
        <option value="Volvo">Volvo</option>
        <option value="Fiat">Fiat</option>
      </select>
      </form>
    );
  } }
ReactDOM.render(<MyForm />, document.getElementById('root'));
```