React Forms

Helping Code

https://1drv.ms/f/s!AtGKdbMmNBGdg_JXAVkOWxwYhZRPlg

As usual

Start and End

Forms Are Different in React

Form elements naturally keep some internal state

In React, mutable state is typically kept in the state property of components

E.g.

Submit button will submit the form naturally

But in React you don’t want to do that

Solutions for Making Powerful Forms

React Final Forms

Informed (Preferred)

Continue to Master Yourself

VC Hint

Select a tag

CTRL+SHIFT+P //command Pallet

>Wrap then press enter

You can use live templates here to wrap a tag around another tag quickly

Bootstrap Forms in React

```
<div className="form-group">
<label htmlFor="exampleInputEmail1">Email address</label>
<input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email" />
<small id="emailHelp" className="form-text text-muted">We'll never share your email with anyone else.</small>
</div>
```

Form Subission

```
handleSubmit = (e) => {
 e.preventDefault();
 console.log("Event Stoppped");
}

//never do stuff like below
//var email = document.getElementById("email").value;
```

React Refs: Access DOM in React. If you REALLY want to

```
username = React.createRef();// a class variable
//Access like this in render
var email = this.username.current.value;
Make Fields like This
<input ref={this.username} />
//Use Refs at Last Resort
```

Controlled Inputs

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

//handle the event like this

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

Form Submission

```
handleSubmit=(event)=> {
    alert('A name was submitted: ' + this.state.value);
    //Send Ajax Request
    event.preventDefault();
  }
```

Controlled Inputs

STATE

```
state = {
    account: {
        username: '',
        password: ''
    }
}
```

INPUT

```
<input
value={this.state.account.username}
onChange={this.handleChange}

/>
```

Handling Change in Controlled Input

```
handleChange = (e) => {
  let account = { ...this.state.account };
  account.username = e.target.value;
  this.setState({ account });
  console.log(this.state.account);
}
```

Handling Multiple Inputs

Set Name properties of Inputs and attach same onChange Handler

```
handleChange = (e) => {
  let account = { ...this.state.account };
  account[e.target.name] = e.target.value;
  this.setState({ account });
  console.log(this.state.account);
}
```

You are Good to Go

But go ahead to become a pro

Common Errors

If you pass null or undefined in state then the controlled input would not work,

Reusable Input Component

```
import React from "react";
const Input = ({ name, label, error, ...rest }) => {
return (
  <div className="form-group">
   <label htmlFor={name}>{label}</label>
   <input {...rest} name={name} id={name} className="form-control" />
   {error && <div className="alert alert-danger">{error}</div>}
  </div>
 );
};
export default Input;
```

Get Components

Input Component

Select Component

Form

Validation

Modify State

```
 state = {
  data: { username: "", password: "" },
  errors: {} //should be empty for valid
//form
 };
```

Validation Submission Logic

```
handleSubmit = e => {
 e.preventDefault(); //Dont Submit Form
 const errors = this.validate(); // Check for Errors
 this.setState({ errors: errors || {} });
 if (errors) return; // Return Immediately if errors
 this.doSubmit();
};
```

joi-browser

npm i joi-browser

import Joi from "joi-browser";

Define Schema in loginForm as

```
schema = {
  username: Joi.string().required().label("Username"),
  password: Joi.string().required().label("Password")
};
```

Validate With JOI

```
validate = () => {
  const options = { abortEarly: false };
  const { error } = Joi.validate(this.state.data, this.schema,
options);
  if (!error) return null;
 const errors = {};
  for (let item of error.details)
   errors[item.path[0]] = item.message;
return errors;
};
```

Validate with JOI Single OnChange

```
validateProperty = ({ name, value }) => {
 const obj = { [name]: value };
 const schema = { [name]: this.schema[name] };
 const { error } = Joi.validate(obj, schema);
 return error ? error.details[0].message : null;
};
```