FORMS 201

More where that came from

TRAJECTORY

- How to get data
  - Controlled components
  - Uncontrolled components
- Managing form state

HOW TO GET DATA

HOW TO GET DATA

- **Collect from the change event while users are typing**
  - Place the data on state
  - Able to calculate validations with each change
- **Collect from the submit event**
  - Does not keep any form data on state
  - Unable to perform any inline validation

First bullet: example, those password forms that yell at you while you're typing

Second bullet: example, posting a comment on facebook. Can get key-value pairs for our input as long as they have name properties (event.target.username.value)

CONTROLLED/UNCONTROLLED

- ***Controlled* form components have their values directly controlled by state**
  - Changes to the value on state update what you see in the form
  - Changes to what you see in the form update the value on state
- ***Uncontrolled* form components do not**
  - Changes to what you see in the form update the value on state
  - …But NOT the other way around

UNCONTROLLED COMPONENT

```
render () {
  <form onSubmit={this.handleSubmit}>
    <input name='username' onChange={this.handleChange} />
    <button type='submit'>Submit</button>
  </form>
}
```


Here is an uncontrolled component. We’re not controlling the value of this input because it doesn’t have a value prop.

CONTROLLED COMPONENT

```
render () {
  <form onSubmit={this.handleSubmit}>
    <input name='username' onChange={this.handleChange} value={this.state.username} />
    <button type='submit'>Submit</button>
  </form>
}
```


And here’s a controlled component. That’s the only difference - that this input now has its value being controlled by a value somewhere on our state. And it doesn’t need to be state on the component doing the rendering - as long as it’s being controlled by some component in the hierarchy, it’s controlled

CONTROLLED/UNCONTROLLED

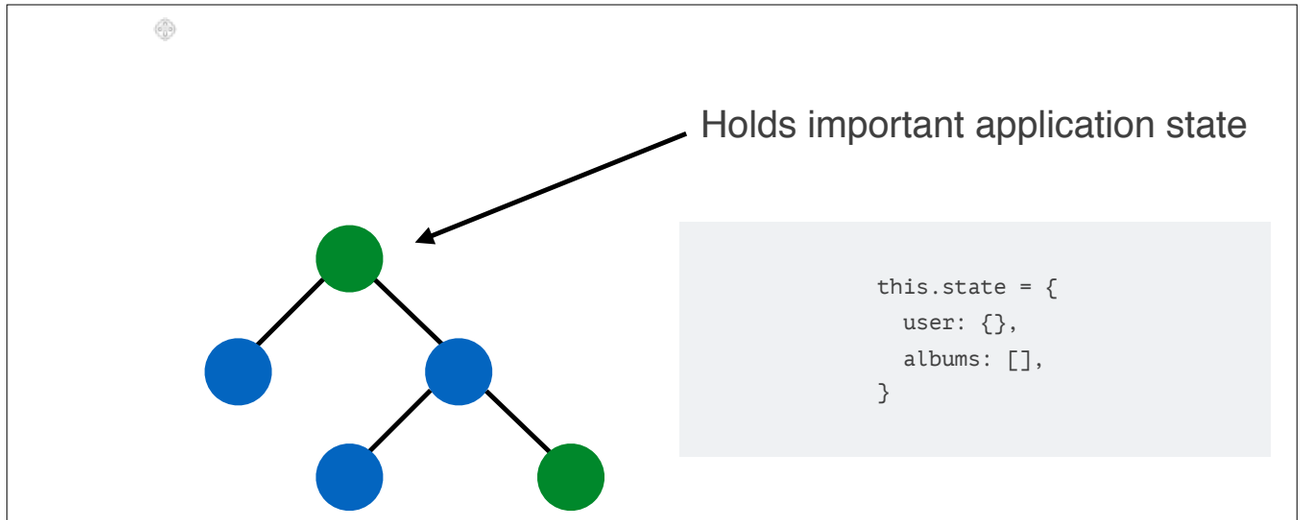
- **Controlled components**
  - MUCH easier to programmatically handle your form's data
  - A bit tedious to set up initially
- **Uncontrolled components**
  - No need to use `this.state` to manage form data
  - Harder to deal with

WHERE DOES FORM STATE BELONG?

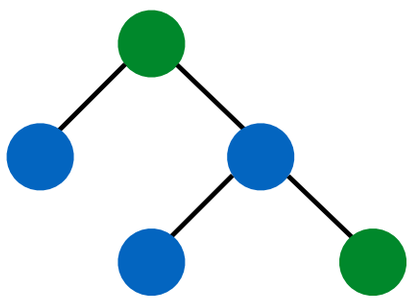
APP STATE V LOCAL STATE

- **Many components in your app might need access to who the user is - this is known as application state**
  - Should be managed high up on your state tree
- **Usually, only the form itself needs to know about the data inside it**
  - Can be managed lower down

app(application) state vs local state.

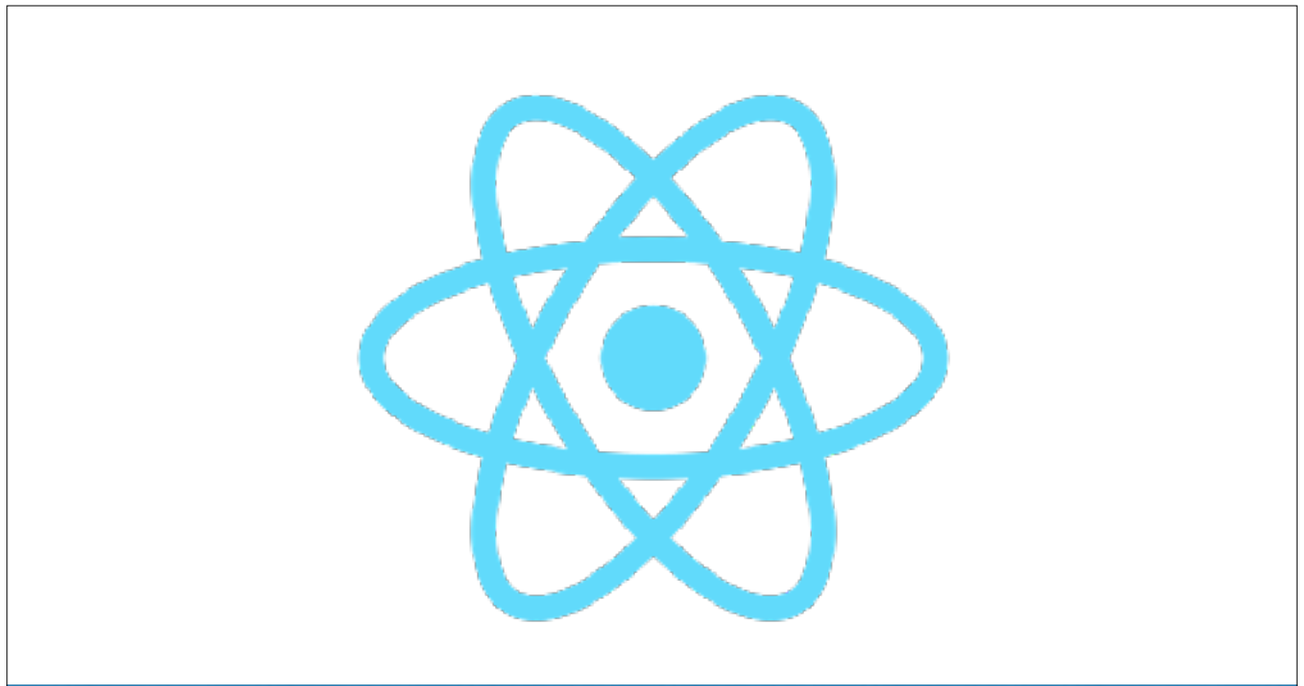


Green - stateful component, blue - stateless component



```
this.state = {
  username: ‘’,
  address: ‘’
}
```

Holds form data



Go workshop!