Implementing Forms in React

Creating Forms Using Vanilla React

Nitin Singh
Full-stack Developer

Overview

- Creating controlled forms using vanilla React
- Creating forms using Formik
- Implementing data validation
- Creating reusable custom form elements
- Implementing uncontrolled forms using vanilla React
- Using React Hook Form to create uncontrolled forms

Managing Form State in React

HTML Forms

A Form is used to collect user input

A top level <form> tag with multiple child form-elements

A submit button that collects user input and usually sends it to a server for processing

Sample HTML Form

```
<form>
  <input type="email" name="email" />

  <input type="password" name="password" />

  <label for="rememberme"> Remember me</label>
  <input type="checkbox" id="rememberme" name="rememberme" value="true">

  <input type="submit" value="Sign In" />
</form>
```

State Management in HTML Forms

1

Form elements have internal state

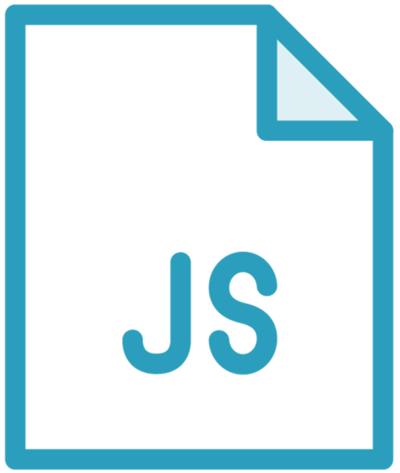
Different types of form elements keep various types of form state

2

DOM handles the state data

The user input is stored on the DOM itself

State Management in React Forms



Controlled forms

React component maintains state

Uncontrolled forms

DOM maintains state

Controlled Forms

React component maintains the state of the form

The component sets the state of the form-element during rendering using its own state

User input is sent back to the component using callbacks on form-elements

The component updates itself on these callbacks and re-renders the form to update the UI

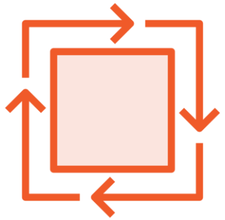
Sample Controlled Form

```
class EmailForm extends React.Component {
  constructor(props) {
    super(props);

    this.state = {value: ''};
    this.handleChange = this.handleChange.bind(this);
  }

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

  render() {
    return (
      <form>
        <input type="email" value={this.state.value} onChange={this.handleChange} />
      </form>
    );
  }}
```

Controlling <input> Form-element

State is set using the value/checked attribute on the tag

onChange callback allows listening for changes in user inputs



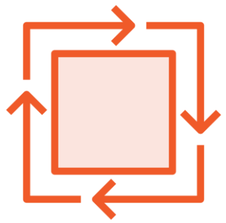
Input type radio and checkbox use the checked attribute and all other input tags use the value attribute

Setting the value attribute on an input tag prevents user from changing the input. It can now only be done via the onChange callback.

Controlling <textarea> Form-element

State is set using the value attribute like input tag

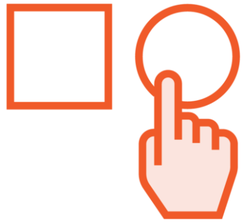
textarea tag in React need not have any children unlike in HTML



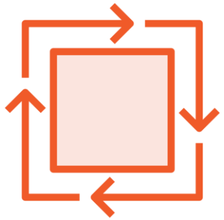
onChange callback received on every user input

Controlling <select> Form-element

No selected attribute needed in React



Selection is determined by value property on the select tag



onChange callback received when user changes selection

```
<select value={this.state.value}>
    <option value="zebra">Zebra</option>
    <option value=”hen">Hen</option>
    <option value="lion">Lion</option>
    <option value="tiger">Tiger</option>
</select>
```

- The option whose value attribute matches the value set on the select tag will be selected

Event Flow for a Controlled Form

Component initialization
Initial form state provided in the state property

Component rendering
Value for the form-elements is picked from the component state and set using the value property

User input
onChange callback on form-element triggered with new value

Change callback
Update the component state with new value using setState

Component re-renders setState with new value triggers component re-render and form updated with new value

Event Flow for a Controlled Form

Component initialization
Initial form state provided in the state property

User input
onChange callback on form-element triggered with new value

Component re-renders setState with new value triggers component re-render and form updated with new value

Component rendering
Value for the form-element is picked from the component state and set using the value property

Change callback
Update the component state with new value using setState

Event Flow for a Controlled Form

Component initialization
Initial form state provided in the state property

Component rendering
Value for the form-element is picked from the component state and set using the value property

User input
onChange callback on form-element triggered with new value

Change callback
Update the component state with new value using setState

Component re-renders setState with new value triggers component re-render and form updated with new value

Event Flow for a Controlled Form

Component initialization
Initial form state provided in the state property

Component rendering
Value for the form-element is picked from the component state and set using the value property

User input
onChange callback on form-element triggered with new value

Change callback
Update the component state with new value using setState

Component re-renders setState with new value triggers component re-render and form updated with new value

Event Flow for a Controlled Form

Component initialization
Initial form state provided in the state property

Component rendering
Value for the form-element is picked from the component state and set using the value property

User input
onChange callback on form-element triggered with new value

Change callback
Update the component state with new value using setState

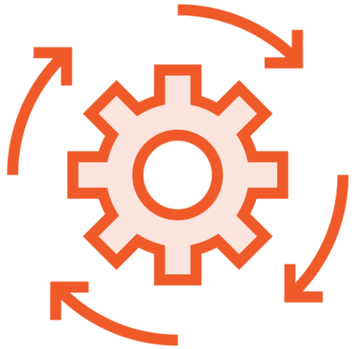
Component re-renders
setState with new value triggers component re-render and form updated with new value

Advantages of Controlled Forms

The recommended way of handling forms in React is to create Controlled forms.

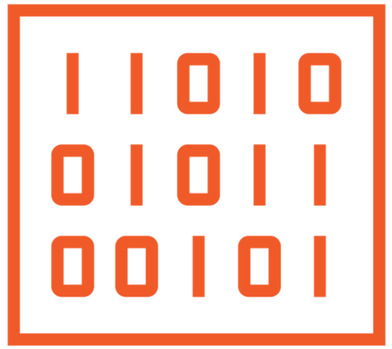
Instant feedback

Show in-place errors to users as they type



Disable UI conditionally

Disable portions on UI based on current user input



Format user input

Show user input in specific formats e.g. credit cards and phone number

Demo

Invoice tracker application

Setup app using create-react-app

Add Sign-in form as a controlled form

- Email, Password and Remember me input fields
- Handle user input and update UI

Adding Data Validation in a Controlled Form

Validation Triggers

1

On user input

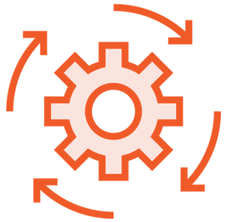
This provides in-place feedback as user interacts with the form

2

On form submission

This provides feedback when the user hits the Submit button on the form

In-place Data Validation



The validation logic is executed in the onChange callbacks

Set an error object for the input in the component's state using setState

Conditionally render the error message based on the state

On Submit Data Validation

The validation logic is executed in the onSubmit callback for the form

Set an error object for all invalid inputs in the component's state

Conditionally render the error message based on the state and prevent form submission

Demo

Invoice tracker application

Add on submit data validations for Email and Password fields

Show errors when user input is invalid and prevent form submission

Summary

- How state management works in a HTML form
- How state management works in React forms
- How to create controlled forms
- Handling different types of form elements in controlled forms
- Advantages of using controlled forms
- Implementing data validation in controlled forms

Up Next:
Creating Forms Using Formik