React 6: Forms

IN608: Intermediate Application Development Concepts

Last Session’s Content

- Lists & keys

Today’s Content

- Forms
  - Controlled components
  - Uncontrolled components

Controlled Components

Controlled Components

- In HTML, form elements, i.e., `input`, `textarea`, `select` maintain their own state & update based on user input
- In React, mutable state is stored in the state property of a component & updated with setState or the Hooks equivalent
- A component that renders a form also controls what happens on subsequent user input

```
import React, { useState } from 'react'

const NameForm = () => {
  const [value, setValue] = useState('')

  const handleChange = (e) => setValue(e.target.value) // Property of the Event interface is a reference
                                                       // to the obj onto which the event was dispatched

  const handleSubmit = (e) => {
    e.preventDefault() // Tells the user agent that if the event does not get explicitly handled, its
                       // default action shouldn't be taken as it normally would be
    alert(`A name was submitted: ${value}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input type="text" value={value} required onChange={handleChange} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  )
}

export default NameForm
```

Controlled Components

- The value attribute is set on the form element
- The displayed value will always be `this.state.value` or `value`
- `handleChange` is called on every keystroke meaning `value` will update as the user types

```
import React, { useState } from 'react'

const NameForm = () => {
  const [value, setValue] = useState('')

  const handleChange = (e) => setValue(e.target.value) // Property of the Event interface is a reference
                                                       // to the obj onto which the event was dispatched

  const handleSubmit = (e) => {
    e.preventDefault() // Tells the user agent that if the event does not get explicitly handled, its
                       // default action shouldn't be taken as it normally would be
    alert(`A name was submitted: ${value}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input type="text" value={value} required onChange={handleChange} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  )
}

export default NameForm
```

Controlled Components

- `textarea` example

```
import React, { useState } from 'react'

const EssayForm = () => {
  const [value, setValue] = useState(
    'Please write an essay about your favorite DOM element.'
  )

  const handleChange = (e) => setValue(e.target.value)

  const handleSubmit = (e) => {
    e.preventDefault()
    alert(`An essay was submitted: ${value}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Essay:
        <textarea value={value} required onChange={handleChange} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  )
}

export default EssayForm
```

Controlled Components

- `select` example
- Note: `lime` is the initial state of `value`

```
import React, { useState } from 'react'

const FlavorForm = () => {
  const [value, setValue] = useState('lime')

  const handleChange = (e) => setValue(e.target.value)

  const handleSubmit = (e) => {
    e.preventDefault()
    alert(`A name was submitted: ${value}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Pick your favorite flavor:
        <select value={value} required onChange={handleChange}>
          <option value="grapefruit">Grapefruit</option>
          <option value="lime">Lime</option>
          <option value="coconut">Coconut</option>
          <option value="mango">Mango</option>
        </select>
      </label>
      <input type="submit" value="Submit" />
    </form>
  )
}

export default FlavorForm
```

Controlled Components

- Handling multiple inputs

```
import React, { useState } from 'react'

const Reservation = () => {
  const [state, setState] = useState({ name: '', numOfGuests: 0 })

  const handleInputChange = (e) => setState({ ...state, [e.target.name]: e.target.value }) // ... rest operator - represent an
                                                                                           // indefinite number of arguments as an array

  const handleSubmit = (e) => {
    e.preventDefault()
    alert(`${state.name} + ${state.numOfGuests} have made a reservation`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input name="name" type="text" required value={state.name} onChange={handleInputChange} />
      </label>
      <br />
      <label>
        Number of guests:
        <input name="numOfGuests" type="number" min="0" max="5" required value={state.numOfGuests} onChange={handleInputChange} />
      </label>
      <br />
      <input type="submit" value="Submit" />
    </form>
  )
}

export default Reservation
```

Uncontrolled Components

Uncontrolled Components

- In most cases, using controlled components to implement forms is recommended
- In a controlled component, form data is handled by the component
- In an uncontrolled component, form data is handled by the DOM
- How do I write an uncontrolled component? Use a `ref` to get form values from the DOM
- What is a `ref`? Provides a way to access DOM nodes or React elements created in the render method
- When should I use a `ref`?
  - When parent components need to interact with children components, we use `props`
  - In some cases, we might need to modify a child component without re-rendering it with new `props`
- Resource: https://reactjs.org/docs/uncontrolled-components.html

Uncontrolled Components

- This example accepts a single report name in an uncontrolled component
- You often want React to specify an initial value, but leave subsequent updates uncontrolled
- To handle this case, you can specify a `defaultValue` attribute instead of a `value` attribute

```
import React, { useRef } from 'react'

const ReportForm = () => {
  const input = useRef()

  const handleSubmit = (e) => {
    e.preventDefault()
    alert(`A report was submitted: ${input.current.value}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Report Name:
        <input type="text" ref={input} required />
      </label>
      <input type="submit" value="Submit" />
    </form>
  )
}

export default ReportForm
```

Uncontrolled Components

- File input
- In React, an `<input type="file" />` is an uncontrolled component
- Why? Its value can only be set by a user & not programatically

```
import React, { useRef } from 'react'

const FileInput = () => {
  const fileInput = useRef()

  const handleSubmit = (e) => {
    e.preventDefault()
    alert(`Selected file - ${fileInput.current.files[0].name}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Upload file:
        <input type="file" ref={fileInput} />
      </label>
      <br />
      <input type="submit">Submit</input>
    </form>
  )
}

export default FileInput
```

Programming Activity

- Checkout to master - `git checkout master`
- Create a new branch called 21-practical - `git checkout -b 21-practical`
- Open the file `21-practical.pdf` and work on the tasks described