React IV

Form Handling

Agenda

- Event Handlers
- Forms

Learn Objectives

- You know how to create forms
- You know how to handle form events
- You know how to collect or put data in forms

Event Handling

Event Handling

Very similar to DOM, syntax is different

DOM:

```
<button onclick="activateLasers()">
Activate Lasers</button>
```

REACT:

```
<button onClick={activateLasers}>
Activate Lasers </button>
```

```
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('The link was clicked.');
  }

  return (
    <a href="#" onClick={handleClick}>
      Click me
    </a>
  );
}
```

An example

```
<button onClick={(e) => deleteRow(id, e)}>Delete Row</button>
```

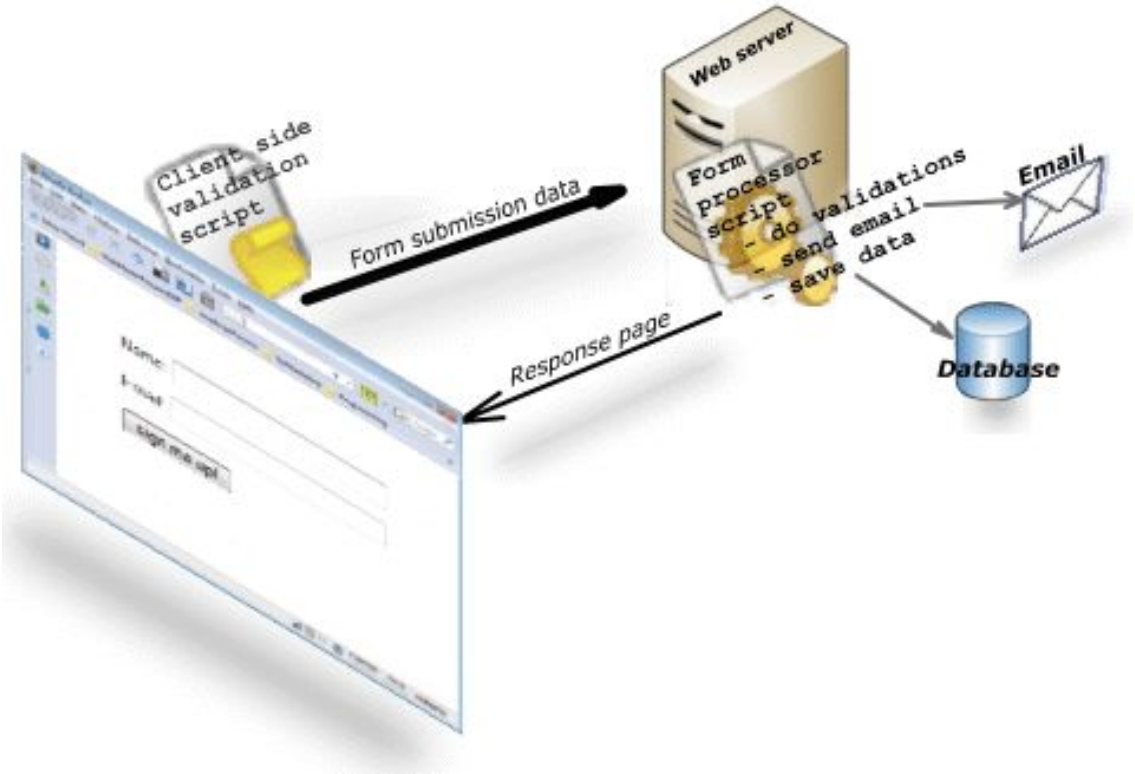
How to pass arguments into Event Handlers

Forms

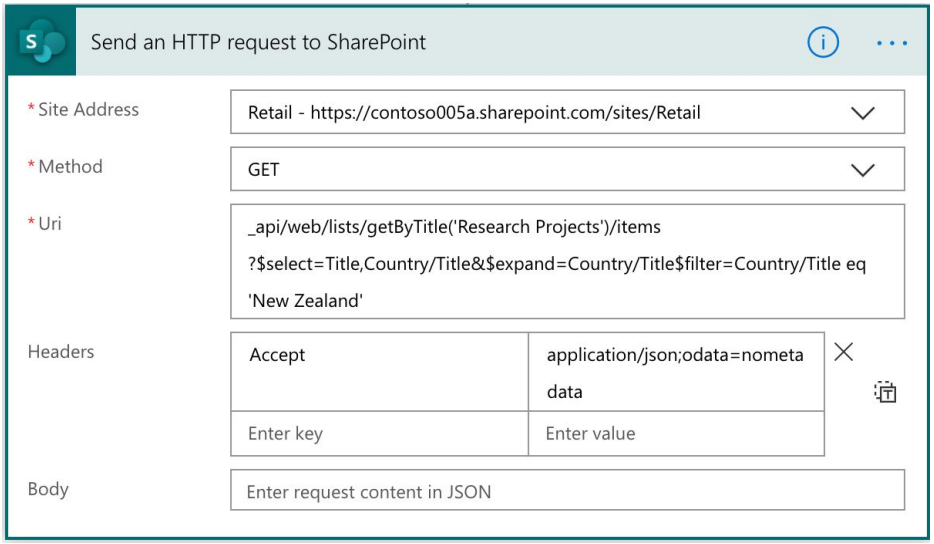
Forms

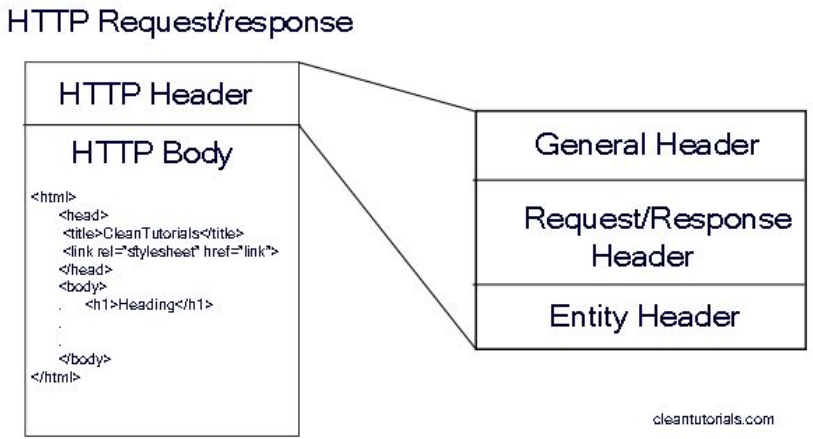
Sending data to backend

- User data is collected from a form
- In normal HTML, a form submit action sends the data to server
- In React data is sent through ajax request (fetch api)



Form sends user data to backend





Request (header & Body)


```jsx
import React from 'react';
import './App.css';

function App() {
  const handleSubmit = event => {
   event.preventDefault();
   alert('You have submitted the form.')
 }

  return(
    <div className="wrapper">
      <h1>How About Them Apples</h1>
      <form onSubmit={handleSubmit}>
        <fieldset>
          <label>
            <p>Name</p>
            <input name="name" />
          </label>
        </fieldset>
        <button type="submit">Submit</button>
      </form>
    </div>
  )
}

export default App;
```

A form example

Data collection

- In react, the form data is sent via fetch api
- It should be first collected
- The programmer has to collect each form field individually and put it them in an object/model (state)
- We will use a hook to make it easy (reducer hook), but a state object can be used directly as well
- There are other 3rd party hooks for handling (https://www.npmjs.com/package/react-hook-form)
- Checkboxes have a special handling

Collect data by using state

```
const UserForm = (props) => {
  const [user, setUser] = useState(props.user)

  const submit = e => {
    e.preventDefault()
    fetch('/api', {
      method: 'POST',
      body: JSON.stringify({ user }),
      headers: { 'Content-Type': 'application/json' },
    })
      .then(res => res.json())
      .then(json => setUser(json.user))
  }

  return (
    <form onSubmit={submit}>
      <input
        type="text"
        name="user[name]"
        value={user.name}
        onChange={e => setUser({ ...user, name: e.target.value })}
      />
      {user.errors.name && <p>{user.errors.name}</p>}

      <input
        type="email"
        name="user[email]"
        value={user.email}
        onChange={e => setUser({ ...user, email: e.target.value })}
      />
      {user.errors.name && <p>{user.errors.name}</p>}

      <input type="submit" name="Sign Up" />
    </form>
  )
}
```

Collect data using reducer hook

```
import React, { useReducer, useState } from 'react';
import './App.css';

const formReducer = (state, event) => {
 return {
   ...state,
   [event.name]: event.value
 }
}

function App() {
  const [formData, setFormData] = useReducer(formReducer, {});

  const handleSubmit = event => {
    event.preventDefault();
    // send to backend use fetch API
  }

  const handleChange = event => {
    setFormData({
      name: event.target.name,
      value: event.target.value,
    });
  }

  return(
    <div className="wrapper">
      <h1>How About Them Apples</h1>

      <form onSubmit={handleSubmit}>
        <fieldset>
          <label>
            <p>Name</p>
            <input name="name" onChange={handleChange}/>
          </label>
        </fieldset>
        <button type="submit">Submit</button>
      </form>
    </div>
  )
}

export default App;
```

```
const UserForm = props => {
  const [user, setUser] = useState(props.user)
  const form = useRef(null)

  const submit = e => {
    e.preventDefault()
    const data = new FormData(form.current)
    fetch('/api', { method: 'POST', body: data })
      .then(res => res.json())
      .then(json => setUser(json.user))
  }

  return (
    <form ref={form} onSubmit={submit}>
      <input type="text" name="user[name]" defaultValue={user.name} />
      {user.errors.name && <p>{user.errors.name}</p>}

      <input type="email" name="user[email]" defaultValue={user.email} />
      {user.errors.email && <p>{user.errors.email}</p>}

      <input type="submit" name="Sign Up" />
    </form>
  )
}
```

Collect data using formdata and ref

```
const [formData, setFormData] = useReducer(formReducer, {
  count: 100,
});
```

```
<form onSubmit={handleSubmit}>
  <fieldset>
    <label>
      <p>Name</p>
      <input name="name" onChange={handleChange} value={formData.name || ''}/>
    </label>
  </fieldset>
  <fieldset>
    <label>
      <p>Apples</p>
      <select name="apple" onChange={handleChange} value={formData.apple || ''}>
          <option value="">--Please choose an option--</option>
          <option value="fuji">Fuji</option>
          <option value="jonathan">Jonathan</option>
          <option value="honey-crisp">Honey Crisp</option>
      </select>
    </label>
```

How to put data in form

Sending a form to backend

- Basically we are just using FETCH API
- There are different actions to be used in browser to send the values
- GET, POST, PUT, DELETE
- GET sends the values as query parameters
- POST is using an internal buffer

```
const submit = e => {
  e.preventDefault()

  const data = new FormData(form.current)
  fetch('/api', {
    method: 'POST',
    headers: new Headers({
      'AuthHeader': '123',
      // STOP! Do not add the following header!
      // 'Content-Type': 'multipart/form-data'
    }),
    body: data,
  })
    .then(res => res.json())
    .then(json => setUser(json.user))
}
```

Using fetch with body and header

Validation

- Do not trust user input
- Always validate
- Use validations in forms
- Or use validations libraries

Questions