CODEPATH*ORG

LEARN. BUILD. CHANGE THE WORLD.

Consuming External APIs

Week 1 Day 3 - June 8, 2022 - Doug Case

Week 1 Day 3 Goals

- Handle form submissions in JavaScript
- Write asynchronous code in JavaScript
- Manage asynchronous code with async / await
- Call external API to get Pokemon information

Forms in HTML

- *<form>* elements in HTML are used to bundle up data and send it somewhere (like a server)
- Inside a form, *<input>* tags are one way to collect information
- When the form is submitted, the form makes a request to the URL provided in the *action* attribute of the form

Forms and JavaScript

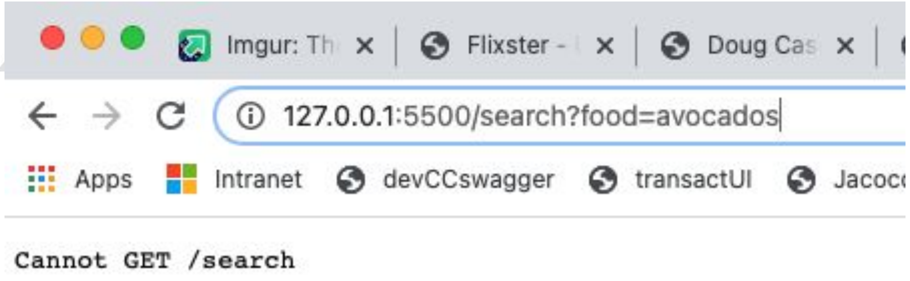
- Sometimes we want to process the form using JavaScript
- We can use JavaScript to make a request instead of the form - this technology is called AJAX
- In order to process form data in JS, we need to:
  - Listen for the “submit” event on a form
  - Prevent the default behavior of the form (sending you to the action page)
  - Write JS code to manage the form submit!

Forms and JavaScript

Click button in form - query string appended to URL
The form extracts the data and makes request for me
Tries to find page based on action in form ("search")

HTML form assembles data,
and sends request to a page

Forms and JavaScript



```
Cannot GET /search
```

Correct error message since do not have a "search" page.
(try going to http://www.google.com or fish.html)

http://127.0.0.1:5500/search?food=avocados

5500	is port
search	is path
food=avocados	is query string

So far - default behavior for form - kinda boring

- But we want to change that!
- We change that with JS, AJAX, APIs
- What's an API? What is AJAX?
- Call this API: https://pokeapi.co/api/v2/pokemon/ditto
- What is the format of the output?

Does this output have familiar format?

```
{
	"abilities": [{
		"ability": {
			"name": "limber",
			"url": "https://pokeapi.co/api/v2/ability/7/"
		},
		"is_hidden": false,
		"slot": 1
	}, {
		"ability": {
			"name": "imposter",
			"url": "https://pokeapi.co/api/v2/ability/150/"
		},
		"is_hidden": true,
		"slot": 3
	}],
	"base_experience": 101,
```

Does it look like this from yesterday? What is this?

```
let obj = {
   key: "valuuuuue",
   second: 2,
   third: "THREE",
   four: {         // nested object
      five: 5,
      six: "SIXX"
   },
   eight:          // array
     [
       {
         color: "red"
       },
       {
         color: "blue"
       }
     ]
}
```

CODEPATH*ORG

Looks like a JS Object

- Called **JSON**: JavaScript Object Notation
- JSON is syntactically correct JS object - perfect to convert to JS
- There are packages that easily convert JSON to Java, C, Python, etc
- Call this API: https://pokeapi.co/api/v2/pokemon/ditto
- What is the format of the output?

FYI - online JSON formatter is jsonlint.com

- https://jsonlint.com
- Paste JSON there - it'll format it nicely and tell you if it is valid JSON

JSON and REST APIs go together

JSON is based on a subset of the JavaScript Programming Language. Representative State Transfer (REST) is a client-server architectural style that uses the HTTP protocol in a simple and effective way. Systems that adhere to REST practices are often referred to as RESTful interfaces.

day3.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Day 3 Demo</title>
</head>
<body>
   <h1>My Day3 Heading</h1>
   <form action="fish.html">
   <!--
   <form action="search">
   <form>
   <form action="https://www.google.com">
   <form action="fish.html">
   <form action="https://pokeapi.co/api/v2/pokemon/ditto">
   -->
      <label>What Pokemon character do you want?</label>
      <input type="text" name="pokemon" />
      <button id='clicker'>Search for character</button>
   </form>
   <div id=”pokemon-area”></div>
   <!-- put script at bottom so loaded last -->
   <script src="day3.js"></script>
</body>
</html>
```

day3.js

```
let pokemonForm = document.querySelector("form");
let pokemonArea = document.querySelector("#pokemon-area");

// addEventListener method has 2 parameters:
// one is event to listen for, second is callback function
// (1) control the form behavior upon submit
async pokemonForm.addEventListener("submit", (evt) => {
    // this prevents the page from re-loading
    evt.preventDefault();
     console.log("evt.target.pokemon.value = ", evt.target.pokemon.value);
    let apiUrl = "https://pokeapi.co/api/v2/pokemon/" + evt.target.pokemon.value;
    // all these console.log entries are kinda ugly - but facilitate debugging
    console.log(apiUrl);
    // (2) upon form submit, call the Pokemon API
    //let response = fetch(apiUrl);
    //console.log("response is: ", response);   // a Promise is placeholder for future data
    // async and await go together
    // the next two lines with await ARE THE DEAL!!!
    let response = await fetch(apiUrl);
    console.log("response is: ", response);   // now call is made, but data still not arrived
    let responseData = await response.json();
    console.log("responseData is: ", responseData);   // now have actual data
    // (3) grab what want want from results of the pokemon API call, and put it on page
    generateHTML(responseData);
}
```

day3.js continued

```
function generateHTML(pokeData) {
    pokemonArea.innerHTML = `
        <h1>${pokeData.name}</h1>
        <p>Height: ${pokeData.height}</p>
        <p>Weight: ${pokeData.weight}</p>
        <img src="${pokeData.sprites.front_default}" alt="Pokemon image" />
    `;
}
```

Does this work with Upper case argument?

- Call with Pikachu instead of pikachu
- 404!
- The API we call only recognizes lower case arguments - argh!

Enhance code to accept upper case

```
let pokemonForm = document.querySelector("form");
let pokemonArea = document.querySelector("#pokemon-area");

// (1) control the form behavior upon submit
async pokemonForm.addEventListener("submit", (evt) => {
   // this prevents the page from re-loading
   evt.preventDefault();
  let apiUrl = "https://pokeapi.co/api/v2/pokemon/" + evt.target.pokemon.value.toLowerCase();
   // (2) upon form submit, call the Pokemon API
   let response = await fetch(apiUrl);
   let responseData = await response.json();
   // (3) grab what want want from results of the pokemon API call, and put it on page
   generateHTML(responseData);
}
```

Making API Calls with *fetch*

- *fetch* is built into the browser, and allows us to try to get data from other applications - asynchronously
- API calls with fetch are asynchronous - JS won't wait for the response to come back before moving on!
- We can use **async** / **await** to manage calls with fetch
- Need to **await** twice: once for the response, and again for the json body

Happy Coding - short break first!