COMP 2537 Web Development 2

Lab 3

Introduction:

(https://upload.wikimedia.org/wikipedia/commons/thumb/9/98/International_Pok%C3%A9mon_logo.svg/1024px-International_Pok%C3%A9mon_logo.svg.png
https://img.rankedboost.com/wp-content/uploads/2018/10/Pokemon-Lets-Go-Shiny-Pokemon-1.png)

Pokémon: There are a lot of them (“Gotta catch ‘em all”)!
The image above shows some of them, but there are many more.

There is a Pokémon API (https://pokeapi.co/about) maintained by an online community which lists most (if not all) of the Pokémon.

In this lab we are going to create a page to display Pokémon on an infinitely scrolling page.

Pokemon API:

The Pokémon API available at Pokiapi.co has an extensive collection of Pokémon and their stats.

To get a list of all the Pokémon available, you can go to this API endpoint:

Pokemon List API Endpoint:
https://pokeapi.co/api/v2/pokemon?limit=1500

In the above URL, I have provided a limit of 1500 so that we can see all the Pokémon. By default, the API will limit the result to just 10. As right now, there are just over 1300 Pokémon.

This endpoint returns a JSON object in the following format:

```
{
    "count": 1302,
    "next": null,
    "previous": null,
    "results": [
        {
          "name": "bulbasaur",
          "url": "https://pokeapi.co/api/v2/pokemon/1/"
        },
        {
          "name": "ivysaur",
          "url": "https://pokeapi.co/api/v2/pokemon/2/"
        },
        ...
    ]
}
```

It will give you the total number of Pokémon in the `count` property and will list all of the Pokémon in the `results` property. The `results` will be an array of objects, one for each Pokémon. Each Pokémon has a name and another URL to get more detailed information about that Pokémon.

Note that not all the Pokemon IDs are sequential. There are some gaps in the Pokemon IDs once you get beyond ID 1025.

To get more detail about and to get an image for a Pokemon you can go to this endpoint:

Pokemon Detail API Endpoint:
https://pokeapi.co/api/v2/pokemon/**<ID>**
(where **<ID>** is the ID of the Pokemon you are trying to get more details about)

ex: https://pokeapi.co/api/v2/pokemon/1 is for Bulbasaur 

and: https://pokeapi.co/api/v2/pokemon/25/ is for Pikachu 

This endpoint returns a JSON object in the following format (although some of the fields are optional and some Pokémon won't have all these fields):

```
{
```

```
  "abilities": [],
  "base_experience": 64,
  "cries": { },
  "forms": [],
  "game_indices": [],
  "height": 7,
  "held_items": [],
  "id": 1,
  "is_default": true,
  "location_area_encounters":
    "https://pokeapi.co/api/v2/pokemon/1/encounters",
  "moves": [  ],
  "name": "bulbasaur",
  "order": 1,
  "past_abilities": [],
  "past_types": [],
  "species": { },
  "sprites": {
    ...
    "other": {
      ...
      "official-artwork": {
        "front_default":
"https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon
/other/official-artwork/1.png",
        "front_shiny":
    "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon
    /other/official-artwork/shiny/1.png"
      }
    }
    ...
  },
  "stats": [],
  "types": [],
  "weight": 69
}
```


(I have omitted some of the field values - ex: `abilities` and `moves` for brevity)

Most Pokémon (with IDs 1025 and below) will have a `sprites` field with an `official-artwork` field under `other` which will contain a `front_default`. This probably the best place to get an image for each Pokémon. There might be some other images that you come across in the `sprites` field, however, some of them are very low resolution (< 96 x 96 pixels) or the images may not always be in the same place. Some Pokémon will have a `generation-i` field with an image, but not all will.

Sample Code:

The sample code provided has the following files:

1. `app.js`
   This is the main application file that runs the server (you'll need Node.js to run it).
   There is nothing really complicated here. There is a single route for the home page `'/'`.
   This route renders the `index.ejs` template.
2. `/views/index.ejs`
   This home page displays a hard-coded bootstrap themed card with the first Pokémon: Bulbasaur.

   Note this page includes the bootstrap CSS file to for styling.
   There is also a `main.js` for client-side scripting.
3. `/public/js/main.js`
   This file has some sample code to get you started.
   It has a simple function that will detect when the user has scrolled to the bottom of the page:

```
document.addEventListener("scroll", function () {
    let scrollTop = document.documentElement.scrollTop || document.body.scrollTop;
    let scrollHeight = document.documentElement.scrollHeight || document.body.scrollHeight;
    let clientHeight = document.documentElement.clientHeight || document.body.clientHeight;
    if (scrollTop + clientHeight >= scrollHeight) {
        console.log("End of page reached");
    }
});
```

   This code figures out where you are scrolling on the page and if you scroll beyond the bottom of the page, it will do a `console.log()`. Instead of `console.log()` you should load the next set of 10 Pokémon.

Also in `main.js` is a function that runs on page load which uses the Pokémon API to load Bulbasaur: the first Pokémon:

```
async function loadPokemon() {
    let response = await fetch(`https://pokeapi.co/api/v2/pokemon?offset=0&limit=1`);
    let jsonObj = await response.json();
    console.log(jsonObj);
    let pokemon = jsonObj.results[0];

    console.log(pokemon.name);
    let response2 = await fetch(`https://pokeapi.co/api/v2/pokemon/${pokemon.name}`);
    let jsonObj2 = await response2.json();
    console.log(jsonObj2);
    console.log(jsonObj2.sprites.other['official-artwork'].front_default);
}
loadPokemon();
```

Instructions:

Create a simple page that will initially display the first 10 Pokémon. When you get to the bottom of the page, use the scroll event to detect that you are at the end of the page and load the next 10 Pokémon, giving the page the effect of the infinite scroll.

In the file `main.js`, you have sample code to pull information about a single Pokémon. You can change this to retrieve 10 Pokémon instead of 1 by changing the `limit` parameter. To get the next set of 10, you can use the `offset` parameter.

Example 1:

```
await fetch(`https://pokeapi.co/api/v2/pokemon?offset=0&limit=1`);
```

This will get the first Pokémon (Pokémon #1).

Example 2:

```
await fetch(`https://pokeapi.co/api/v2/pokemon?offset=0&limit=10`);
```

This will get the first set of 10 Pokémon (Pokémon 1 through 10).

Example 3:

```
await fetch(`https://pokeapi.co/api/v2/pokemon?offset=10&limit=10`);
```

This will get the 2nd set of 10 Pokémon (Pokémon 21 through 30).

Example 4:

```
await fetch(`https://pokeapi.co/api/v2/pokemon?offset=20&limit=10`);
```

This will get the 3rd set of 10 Pokémon (Pokémon 31 through 40).

Use the scroll event listener to load the next set of 10 Pokemon when the user scrolls to the end of the page.

```
document.addEventListener("scroll", function () {
```

Use the Bootstrap template provided in `index.ejs` to display each of the dynamically loaded Pokemon. In other words, make sure the page looks pretty and is well formatted.

Example:

Marking Guide:

Criteria	Marks
Demo the page loading and showing the first 10 Pokémon. Make sure that the Pokémon are shown using the bootstrap template provided. Scroll to the bottom of the page. Show that the next 10 Pokémon are loaded. Keep scrolling…. The Pokémon should keep coming!	10 marks
Total:	**10 marks**

Submission Requirements:

Submission:
Show your lab instructor during lab time. No submission needed (unless you can't make it to class).