# Activity 7: Front-end Web Development II

|  |  |
| --- | --- |
| Group # |  |

Participating group members

|  |  |
| --- | --- |
| Student ID | Name |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Part 2: Building Your Own Pokédex

## Overall Objective

In this part you will build an interactive Pokédex web app that uses the PokéAPI. You will:

1. **Learn basic object-oriented programming in JavaScript** by creating a custom class (PokemonData) that encapsulates a Pokémon’s data and renders a detail card.
2. **Build the Pokédex UI:**
   * A scrollable **sidebar** listing the first 100 Pokémon (showing a small image and name).
   * A **main panel** to display a detailed Pokémon data card when a Pokémon is clicked.
3. **Integrate your class:** Use the PokemonData class to process fetched data and update the UI.

## Task 2.1: Working with the PokemonData Class

**Objective:**

* **Encapsulate Data:** Bundle key Pokémon attributes (name, image, types) and related methods into a single object.
* **Use a Constructor:** Initialize these properties when a new object is created.
* **Call object methods:** Call a method that returns an HTML snippet to display the Pokémon’s details.

**Background Concepts:**

* **Data Encapsulation:**  
  This is the process of bundling the data (attributes) and the functions (methods) that operate on that data into one unit (object). It makes your code more organized, maintainable, and helps protect data integrity.
* **Constructor & Instantiation:**  
  The constructor is a special method used for initializing new objects. When you create an object from a class (called instantiation), the constructor sets the initial properties of that object.

You are provided with pokemonData\_starter.js in which there are:

* The (working but incomplete) class definition of ***PokemonData***, which, through out this activity, we will use the class to instantiate ***PokemonData*** objects encapsulating the data of each Pokémon obtained from the PokéAPI.
* A function called ***testPokemonDataClass()***.

**TO DO:**

* Create by yourself an HTML file (and name it part2\_1.html) with a <div> whose id is “output”.
* Rename pokemonData\_starter.js to pokemonData.js and use <script src= “…”> to load pokemonData.js into the HTML page.
* Observe the content of pokemonData.js. Can you see a constructor? How many methods (functions), apart from the constructor, are provided in the class definition of PokemonData?

3

* Execute ***testPokemonDataClass()***from the console. Describe what you see.

1 : bulbasaur (grass, poison)

* Study code in ***testPokemonDataClass()***and briefly describe how mockData and mokePoke are related.

MokePoke is constructor using mockData as parameter

* Modify ***testPokemonDataClass()***so that, instead of displaying the simple data from ***toString()***, the code display the HTML elements rendered from ***renderInfoTable()*** in the <div> whose id is “output”.
* Capture the screen and paste it in the box below.

A white background with black text

AI-generated content may be incorrect.

Hopefully, by now, you should be familiar with

* how a custom class definition is used in instantiating an object of that class using the construction function
* and how a function (actually a function of an object is usually called “method”) of an object can be called.

## Task 2.2: Instatiate a PokemonData object from PokéAPI real data

Next, we willl try instantiating a PokemonData object from the real data obtained from the PokéAPI instead of a mock-up one. To ease the process, you are given with:

* part2\_2\_starter.html : A web page to host the code.
* style\_2\_2.css : An accompanied style sheet to be linked with part2\_2.html.
* script.js : A JavaScript file containing a utility function to call the endpoint of the PokéAPI.

TO DO:

* Rename part2\_2\_starter.html to part2\_2.html.
* In part2\_2.html, complete the ***doSomethingWithData()*** function so that:
  + A new PokemonData object is instantiated from the data obtained from the pokemon endpoint of the PokéAPI.
  + Use ***renderInfoTable()*** to render the content of <div id=“output”>.
* You should notice the the src of <img class="pokemon-image"> in the info table is invalid. Therefore, the image seems to be broken. Modify the constructor of PokemonData so that the image of the Pokémon is the default front-facing sprite obtained from the the pokemon endpoint of the PokéAPI.
* Describe how you modify the code.

class PokemonData {

    constructor(data) {

        // Store the Pokémon's ID.

        this.id = data.id;

        // Store the Pokémon's name.

        this.name = data.name;

        // Store the Pokémon's types as an array of type names.

        this.types = data.types.map(typeInfo => typeInfo.type.name);

        this.image = data.sprites.front\_default;

        // Store the full API data for the Pokémon.

        this.apiData = data;

    }

}

## Task 2.3: Complete a simple Pokédex

In this part, we will create a working web page functioning as a simple Pokédex in which:

* A list of the first 100 Pokémons is shown in the sidebar.
* When a list item is clicked, a Pokémon card associated with the clicked item is shown in the main panel.

You are given with:

* part2\_3\_starter.html : The HTML for our Pokédex.
* style\_2\_3.css : An accompanied style sheet to be linked the HTML file.
* script\_2\_3\_starter.js : A JavaScript file you will complete so that our Pokédex works.
* Please be noted that the HTML file still use the JavaScript from pokemonData.js as well as from script.js.

TO DO:

* Rename part2\_3\_starter.html to part2\_3.html
* Rename script\_2\_3\_starter.js to script\_2\_3.js
* Modify the HTML so to make sure that the ***fetchPokemonListAndAddToSideBar()*** function is executed once the DOM is loaded. (Just call the function at the end of <body>!)
* Look in the console. With script\_2\_3.js provided as it is, describe you saw when the HTML has been loaded.

A white background with black text

AI-generated content may be incorrect.

* Modify script\_2\_3.js
  + Remove console.log().
  + Complete the code by tackling all the parts commented with “TO DO”.

## Task 2.4: Part 2 checkpoint

* Call a TA to demo your program.
* Submit this worksheet in myCourseVille