Introduction to Programming: Project 2

Part One: Index.js

- Listed below are the contents of the code that I have submitted via Github. The following information will contain the syntax functionality of every function used, and the purpose of the written code. Listed below will be the source for this project and its inspiration.

```
async function fetchData(){

    try{

        const pokemonName = document.getElementById("pokemonName").value.toLowerCase();
        const response = await fetch(`https://pokeapi.co/api/v2/pokemon/${pokemonName}`);

        if(!response.ok){
            throw new Error("Could not fetch resource");
        }

        const data = await response.json();
        const pokemonSprite = data.sprites.front_default;
        const imgElement = document.getElementById("pokemonSprite");

        imgElement.src = pokemonSprite;
        imgElement.style.display = "block";
    }
    catch(error){
        console.error(error);
    }
}
```

- For the first example, I will be going through the functionality of the *index.js* file itself.

- **Line 1: *"Async function fetchData(){":***

The term "***async"*** is used to define a function as asynchronous. This means that the function will return a ***promise***. A ***promise*** being a placeholder for a value that will be available in the future, but is not immediately available. The next term is the term simply named as ***"function"***. The ***"function"*** keyword defines a new function. Within this scenario, the term ***"fetchData"***. In short, the functionality of the eternity of *line one* is to define a function that works asynchronously and allows you to use await for handling asynchronous tasks. As for the use of the ***"()"*** and ***"{}"*** characters, the brackets ***"()"*** are used to define the parameters of the function. In this case, the brackets are empty, meaning the fetchData function does not take any parameters. The ***Curly Braces ({ })*** are used to define the body of that function. Everything within the ***({ })*** is part of that function's logic, which will be provoked when the function is run.

- **Line 3: *"try{":***

This character is used to execute code that might fail. If any errors occur while running the code, the resulting control will be given to the ***"catch"*** function to handle the error. As you will see, there will be a ***"catch"*** later.

- **Line 5: *"const pokemonName = document.getElementById("pokemonName").value.tolowerCase();"***

Here, the ***"const"*** function is used to permanently define a function. That function is ***"pokemonName"***. Next, we can see ***"document.getElementById()"***. This is used to retrieve an HTML (*hyper text markup language*) element with the id attribute being set to ***"pokemonName"***. It will then return the element that belongs to that id.
In ***".value"***, the function will *fetch* the value that is returned from ***"getElementById"***. Within this example, it will return with the value entered in an input field with the ***"id="pokemonName"***. ***.toLowerCase()*** is a way to ensure that any value that is entered within a propted input, will be automatically converted to lowercase. This is for the API's sake, as in this case, all data is entered in lowercase.

- **Line 6: "const response = await fetch('https://pokeapi.co/api/v2/pokemon/${pokemonName}');"**

Within this line of code, the ***"response"*** function is being defined by the ***"const"*** variable, as before. However, in this line of code, ***"await"*** is a function used in asynchronous situations. It will pause the execution of a function until a promise is returned by the ***"fetch"*** function that is used directly after. ***"Fetch"*** is a Javascript function that is used to perform a network request. After the request is made, there will be a response. This response is known as a *promise.* This will ensure that there will be a resolution with the response. The ***backticks (')*** are known as *template strings.* These will allow you to instil expressions within a string. Within these ***backticks*** is a link, ***"***https://pokeapi.co/api/v2/pokemon/${pokemonName***".*** This is the API that will be used to *fetch* data about a certain type of pokemon. Next, we may find the string ***"${pokemonName}***. First, we have to address the expression ***"${}"***. This is used to instill the value of pokemonName. This will ensure that the value of ***"pokemonName"***. Finally, we have a ***semicolon (";")***. This is an indicator that the code is now finalized, as this is Javascript's way of marking the end of a line of code.

- **Line 8/9/10: *"if(!response.ok){***
  Throw new Error("Could not fetch resource");
  }:"

  Line 8: ***"if(!response.ok){***

At the start of this line of code, we are greeted by the function ***"if"***. This is simply a condition parameter that is used to check if the condition within the parameters is *true*. Within this example, it will check if the string ***"!response.ok"*** is true or false. ***"response.ok"*** is a part of a *Response* function, which is used in conjunction with the *fetch* function. This indicates that the request that was sent was successful, and has a status code value 200-299. (*see http response codes*). The ***exclamation mark " ! "*** used before the value "***response.ok"*** is there to prevent the code from executing if the response code is anything other than ***"response.ok"***. Think, "if ***"response.ok"*** is ***false***, then ***"!response.ok"*** must be ***true***.

Line 9/10: ***Throw new Error("Could not fetch resource");***
};

"throw" is used to make an exception within Javascript. When the ***"throw"*** function is used in Javascript, it will stop the normal flow of the program and provoke a *catch* block that may be part of a *try - catch* statement.

"New Error" simply creates a new error *object*. This error can include any message that you desire, but as in this case, the error is ***"Could not fetch resource"***. This is a conditional statement that is used in the event that ***"response.ok"*** is marked as ***false***.

- **Line 12: "*const data = await response.json():"***

Once again, we are defining a variable via the "***const"*** function. In this example, ***"data"*** is being set to the ***"await"*** variable. This is an async function that is used to pause the execution of the code until a *promise* is resolved or rejected. The *promise* mentioned is the ***"response.json()"*** function. The ***"response.json"*** can be broken into two parts for definition's sake. First of all, ***.json (****javascript object notation) is a method available on the response object (from the fetch API) found within Javascript*. "***Response"*** is simply the object used within the fetch API used within Javascript.

- **Line 13. "Const pokemonSprite = data.sprites.front_default;"**

"pokemonSprite" is assigned to the ***"data.sprites.front_default;"***. The ***data*** portion of this string is simply an object that contains information about the desired pokemon. Within the ***data*** portion of this segment, we may find the ***sprites*** property. Then, not so surprisingly, ***front_default*** is a specific property found within the ***sprites*** *object.*

- **Line 14. *"Const imgelement = document.getElementById("pokemonSprite");"***

“imgElement” is defined as ***“document.getElementById(“pokemonSprite”);***

getElementById() Is provided by the document object. It returns a reference to the HTML element with the specified id attribute. In this case, this function is being utilized to find the ***“pokemonSprite”*** value. This will all be stored within the permanent value ***“imgElement”***.

- **Line 16. *“imgElement = document.getElementById(‘image-id’);”***

imgElement is our previously defined model that we had made previously. This line of code is assigning this *DOM (document object model)* to the value that is stored within the ***pokemonSprite*** model.

- **Line 17. *“imgElement.style.display = “block”;”***

Within this line, we are continuing the process of defining our variable ***“imgElement”***. However, this time we are accessing the styles of the HTML element via the ***“.style”*** variable. After this, we may now access the ***“.display”*** property that is found within the ***“style”*** property. After this, the value that is being assigned to ***“display”*** is another property known as ***“block”***. This value makes the element behave as a block level element, meaning it will take up the full width that is available and start on a new line, pushing any subsequent elements downward.

- **Line 18/19/20/21: }**

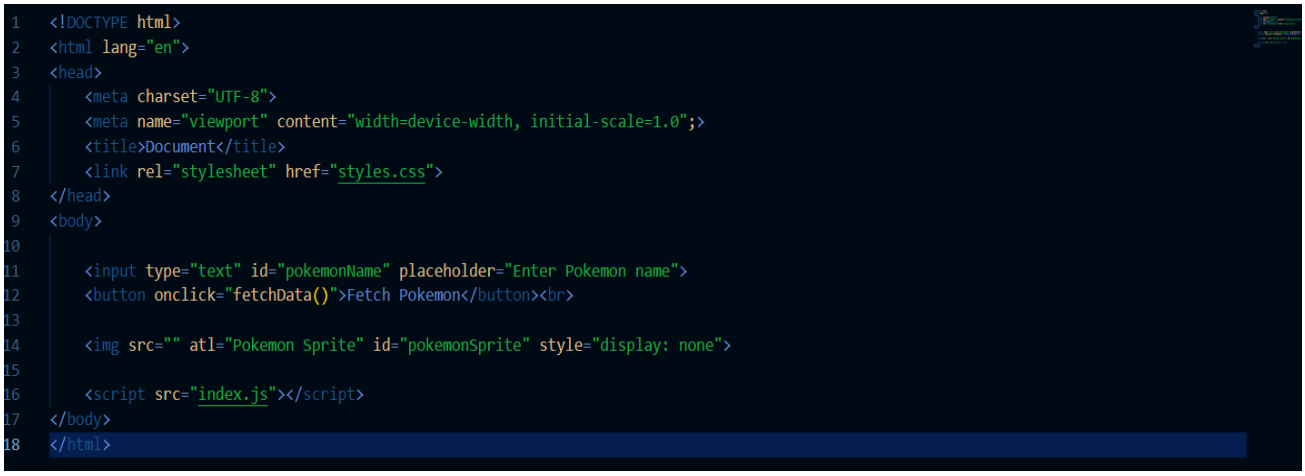
```
“catch(error){
    console.error(error);
}
}”
```

Within the ***“catch(error)”*** variable, we may look back to our previously written lines of code. *(See Line 3).*

This is the conclusion of a *try-catch* statement, which is used for exception handling within Javascript. Within this statement, we were able to define a block of code that was located within the *try* block. Finally, ***“console.error()”*** is used to output an error message ***(error)*** to a web browser’s console, or a Javascript console. In my case, it would have been the ladder, as this code was being written and ran within *Visual Studio Code*.

The function of this code is to push and pull data from the pokeAPI website. As you will see with the second part of this document, it is an interactive web-based application that pulls data from any Pokemon name entered, and displays the sprite image of said Pokemon.

Part 2: Index.html



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0";>
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <input type="text" id="pokemonName" placeholder="Enter Pokemon name">
    <button onclick="fetchData()">Fetch Pokemon</button><br>

    <img src="" atl="Pokemon Sprite" id="pokemonSprite" style="display: none">

    <script src="index.js"></script>
</body>
</html>
```

- **Line 1: “<!DOCTYPE html>”**

This is a header that will indicate the document type. In this case, we are utilizing an html document for an interface.

- **Line 2: “Html lang=”en”>”**

This line of code specifies that the language that is used in this document is English. (lang=”en”)

- **Line 3: “<head>:”**

This section will contain data and links that are related to the document being written.

- **Line 4: “<meta charset= “UTF-8”>”**

Specifies the character encoding for the document. In this case, UTF-8, which includes most characters from all known languages.

- **Line 5: “<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>”**

Ensures that the page will scale properly on different devices, such as mobile devices.

- **Line 6. “<title>Document</title>”**

Sets the title of the webpage, which will appear in the browser’s title bar.

- **Line 7. “<link rel=stylesheet” href=”styles.css”>”**

This tag will link an external file, which in this case, is a *CSS* file. The ***“rel=”stylesheet”*** attribute specifies that the file is a *CSS* file. The ***“href=’styles.css”*** attribute provides the path to the *CSS* file.

- **Line 8. “<body>”**

This is where the visible content of the webpage will go. It starts the ***“body”*** section of the HTML document, where you’ll typically find text, images, or buttons.

- **Line 9. “<head>”**

This element will contain the data and links to external resources. This will not display content directly on the page, rather, it will configure how the page is rendered and behaves.

- **Line 11. “<input type=”text” id=”pokemonName” placeholder=”Enter Pokemon name”>”**

This is an input field in which the user can enter the name of any Pokemon. As demonstrated, it has an **id** of ***pokemonName***, which can be accessed to return a value that is entered by the user.

- **Line 12. “<button onclick=”fetchData()”>”**

This will create a button on the webpage. The ***“onclick”*** function will trigger the function ***“fetchData()”*** when the button is clicked.

- **Line 14. “<img src=”” alt=”Pokemon Sprite” id=”pokemonSprite” style=display: none”>”**

This is an image tag that is used to display the Pokemon sprite image that is being retrieved from the pokeAPI website. Initially, it has no source and is hidden using ***“style=”display: none”***.

- **Line 16. “<script src=”index.js”></script>**

Since the code references our external ***index.js*** file, the implementation of the ***“fetchData()”*** function will be inside this file.

Summary:

As for a brief summation, there is an HTTP request that is made to provide the URL. Then, the following response will be captured and converted into the *JSON* format. Afterward, it will then process the resulting data. In this example, it updates the ***"src"*** of the Pokemon image with the sprite URL from the API response. Finally, the ***"pokemonSprite.style.display = "block"*** line of code will make the image visible on the web page by changing the *CSS* display property to *block (none)*.

To clarify, this idea was taken from a YouTube channel that goes by the name *Bro Code*. As this was a learning curve for me I was unable to rely on my own intuition for this project. Thus, it is only right to credit the original creator of this project. Below will be a resource link to his channel.

How to FETCH data from an API using JavaScript. YouTube, uploaded by Bro Code, 28 Dec. 2023, https://www.youtube.com/watch?v=37vxWr0WgQk.