IMY 220
Assignment 1

JS & ES6

General Instructions

- This assignment must be completed and submitted by the due date which is available on ClickUP.
- This assignment is a take-home assignment and may take one or two days to complete.
- This assignment should be completed on your own, but you may come to the tutorial session or email the assistant lecturer if you need further assistance.
- No late / email submissions will be accepted.

This assignment focuses on **Functional JS, JS OOP, JS Arrays, and ES6 syntax**. You will be required to create a JS class called `PetHandler` that defines several functions to display and organise a list of pets for an adoption agency.

Note: *Where appropriate, you should **always use ES6 syntax**. You may **not** write any loops (e.g.,* `for` *loops /* `while` *loops) or use the* `.forEach()` *function in this assignment. All of the functionality must be implemented with the appropriate JS Array functions as discussed in the slides and resources linked from the slides. Marks are awarded for following these instructions.*

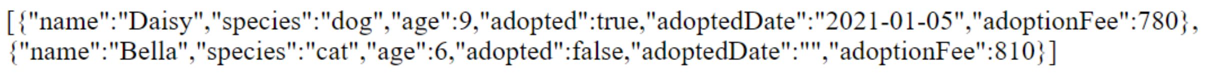
Download *index.html* and *script.js* from ClickUP. These contain the basic HTML for a valid HTML5 document and a populated JS array respectively. Include *script.js* inside *index.html*.

Part 1 - PetHandler Object

Create a constructor for a `PetHandler` object in JavaScript which takes one argument: an array of objects. You may assume this array will always be an array of objects like the `pets` variable in *script.js*. This array must be saved as a member variable of the object.

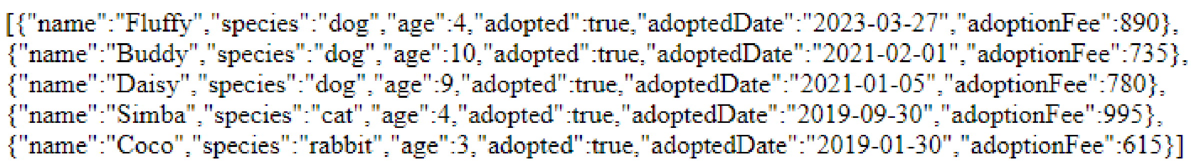
The `PetHandler` class should have the following member functions:

1. `findPetsInAgeRange(minAge, maxAge)` - This function should take in two arguments, a minimum and maximum age, and return all of the pets from the member variable whose ages **fall between** these two integers specified. The range should be **inclusive** of the two integers (meaning that both numbers specified should be included in the range). For example, with the given array, `findPetsInAgeRange(5,9)` should return:

   ```
   [{"name":"Daisy","species":"dog","age":9,"adopted":true,"adoptedDate":"2021-01-05","adoptionFee":780},
   {"name":"Bella","species":"cat","age":6,"adopted":false,"adoptedDate":"","adoptionFee":810}]
   ```

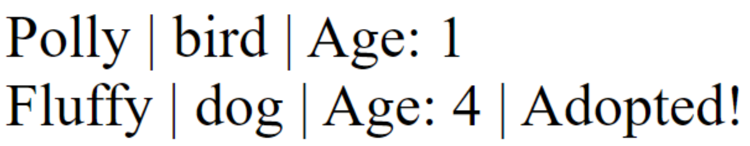
2. `listAdoptedPetsByDate()` - This function should use the member variable to find all of the **adopted** pets and then **sort** them by the date they were adopted in **reverse chronological order**. This means the most recently adopted pet should be shown first. For example, with the given array, `ListAdopedPetsByDate()` should return:

   ```
   [{"name":"Fluffy","species":"dog","age":4,"adopted":true,"adoptedDate":"2023-03-27","adoptionFee":890},
   {"name":"Buddy","species":"dog","age":10,"adopted":true,"adoptedDate":"2021-02-01","adoptionFee":735},
   {"name":"Daisy","species":"dog","age":9,"adopted":true,"adoptedDate":"2021-01-05","adoptionFee":780},
   {"name":"Simba","species":"cat","age":4,"adopted":true,"adoptedDate":"2019-09-30","adoptionFee":995},
   {"name":"Coco","species":"rabbit","age":3,"adopted":true,"adoptedDate":"2019-01-30","adoptionFee":615}]
   ```

3. `listPets()` - This function should take in one or more **optional** argument(s), which can either be a single array of objects or any number of objects. If **no argument(s)** are provided, the function should use the member variable.

   This function should return a list of pets formatted as **strings** like the screenshot below, where each pet's name, age and species is shown as follows:

   ```
   Polly | bird | Age: 1
   Fluffy | dog | Age: 4 | Adopted!
   ```

   If the pet has been **adopted**, then this should also be shown as above.

   To create this function, you must **define another function** (named `createPetItem()`) **inside** `ListPets()` that handles creating the strings. You should then use an array function to apply `createPetItem()` to each pet object.

4. `calculateUniqueAdoptionFee(petNames)` - This function should take one or more pet names (strings) as its argument(s). You can always assume **at least one pet name will be provided**.

   This function should search through the member variable and return a list of all of the pet objects specified by those names. If **more than one identical pet name is provided as an argument, there should only be one pet object with that name returned** (i.e., there should be no duplicates). It should then calculate the total **adoption fee** when adopting all those specified pets together (You don't need to check if the pet has already been adopted or not).

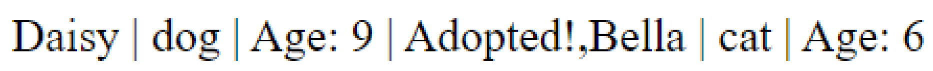
   For example, with the given array, `calculateUniqueAdoptionFee('Milo', 'Coco', 'Milo')` should return: `1355`

Part 2 - Extending Array Functionality

After defining your `PetHandler` object, you should be able to instantiate an instance of that object and use its member functions, for example, if `handler` is instantiated as an instance of the object with the given array variable, `handler.findPetsInAgeRange(5,9)` should return an array of all the pet objects whose ages are from 5-9 (inclusive).

However, since some of the functions return arrays, the functions can't be chained, for example, the following will not work: `handler.findPetsInAgeRange(5,9).listPets()`, since `findPetsInAgeRange` returns an array and `listPets` is not defined as an Array function.

So, to allow for this chaining of functions, extend the `Array` class using the `prototype` property and add all of PetHandler's member functions.
If done correctly, the output to `handler.findPetsInAgeRange(5,9).listPets()` would be:



```
Daisy | dog | Age: 9 | Adopted!,Bella | cat | Age: 6
```

Additional Information

- **Note about testing**: this assignment does not have a DOM-aspect, thus you are expected to do your own testing using `console.log`. However, make

    sure to take out all `console.log` statements before submitting as only your class definition and Array prototype extensions will be marked.
- Refer to the slides and online resources for help

Submission Instructions

Place these in a folder named `A1_u12345678` where `12345678` is your student number.

- script.js

Zip the **folder** and upload this to ClickUP in the relevant submission slot before the deadline.