Web, Mobile and Security

Lab: Higher Order Array Functions

1. Creating the project

- This lab does not require Laravel. **Therefore, it is not necessary to launch Docker or WSL.**
- Create a new repository for this exercise in your dedicated Gitlab group.
- Create a new folder on your hard drive where you will build your web application.
- In the root of that exercise folder, create a new yml file with the name `.gitlab-ci.yml` and the following contents:

  ```yaml
  image: monitor:5000/howest-html-validator:20.6.30
  stages:
    - QA

  validateHTML:
    stage: QA
    script:
      - vnu --Werror --skip-non-html ./
  ```

  Committing to your repo with this file will trigger the HTML validation, as you are used to from the previous semester.

- Now for the actual work.

2. Introduction

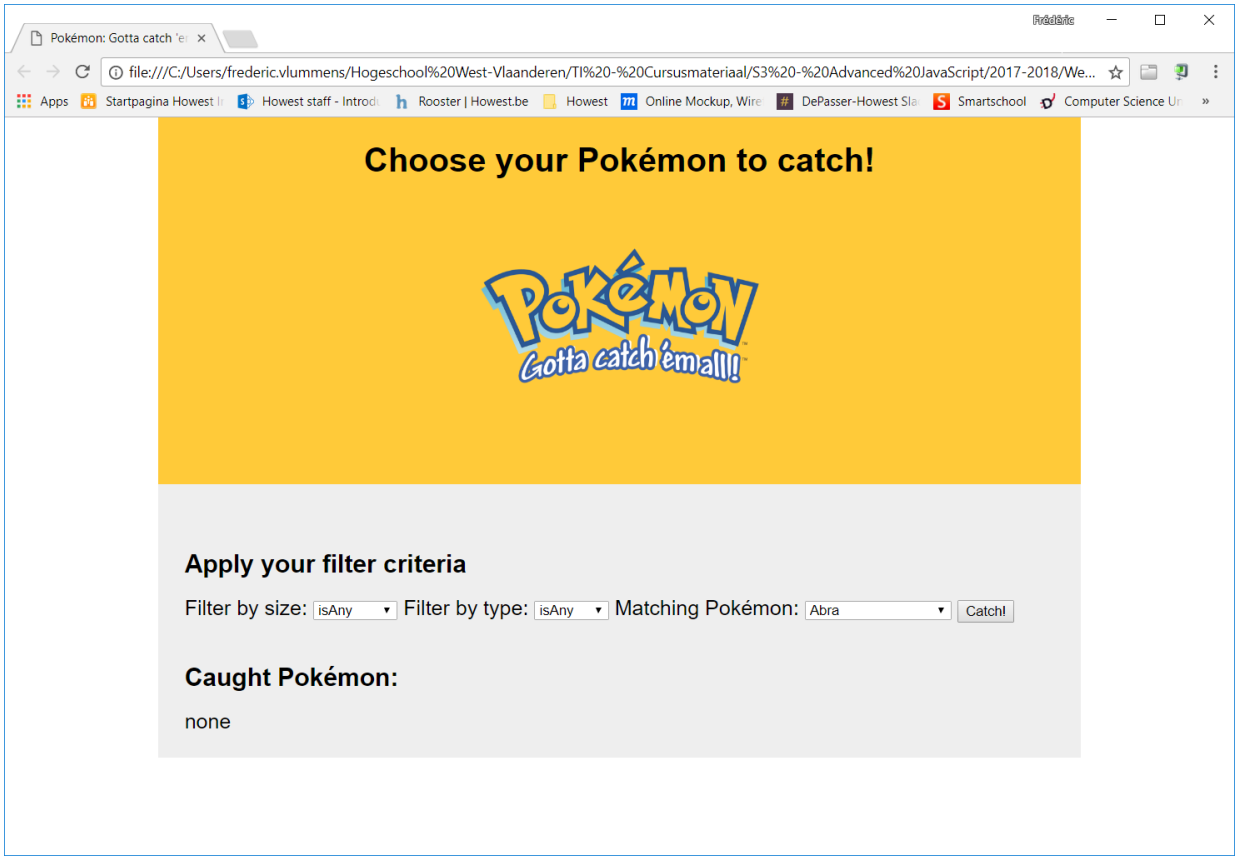
Develop a client-side JavaScript web application that allows a user to filter Pokémon based on their properties (size + type) and allows them to be "caught" (added to a list).

3. Source files

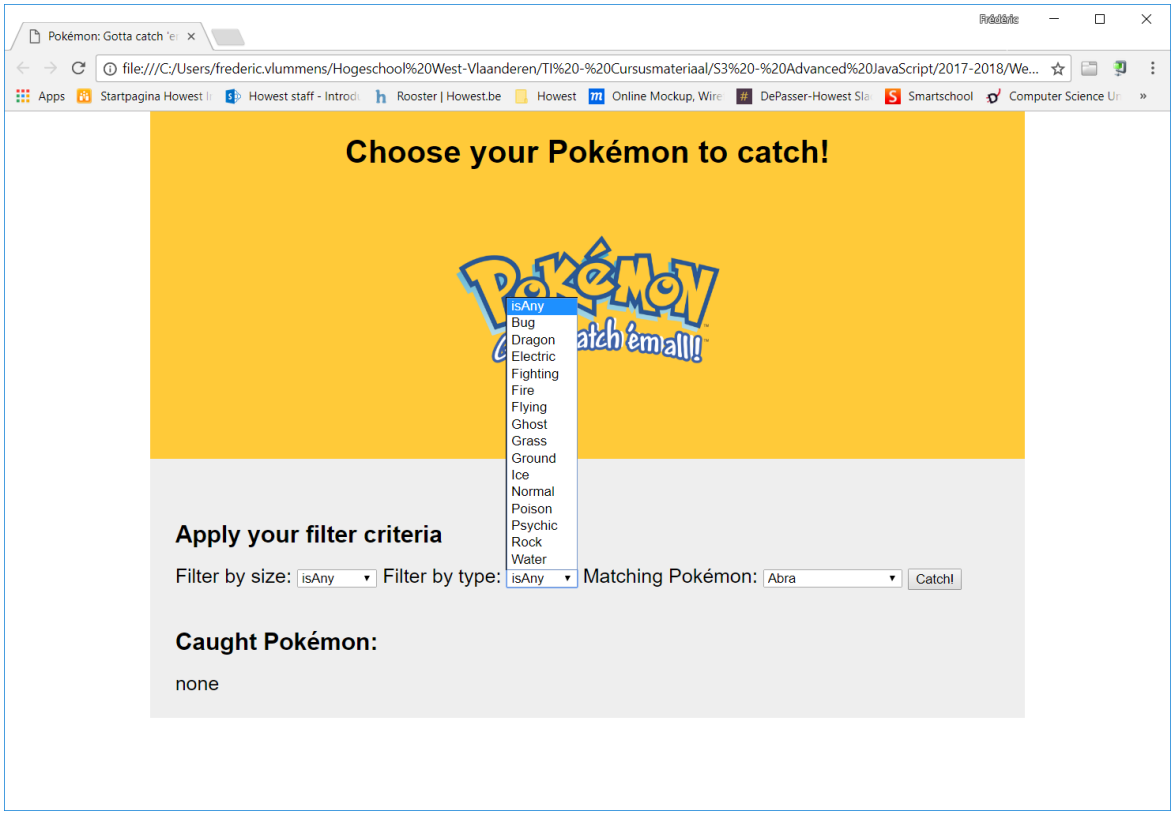
Given is a JSON-file `pokedex.js` as well as the various images (one per Pokémon + the Pokémon logo).

4. Your task

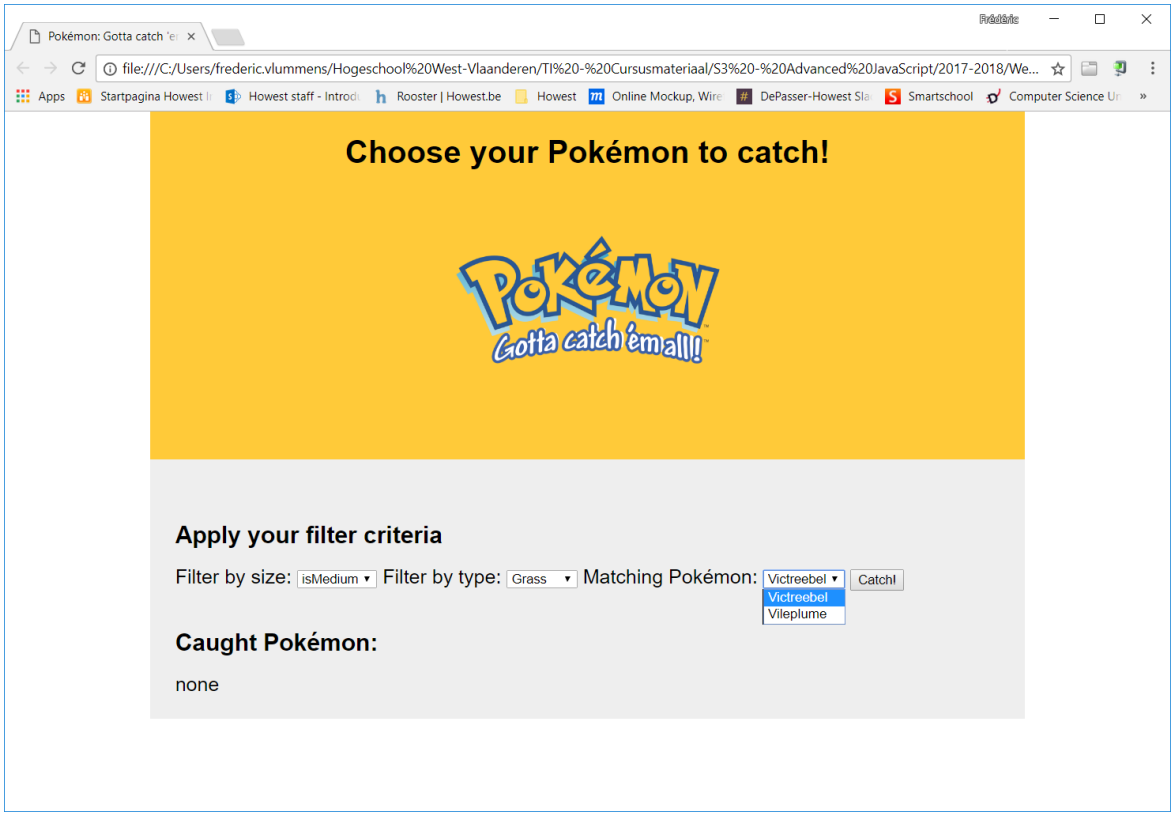
Build the following web UI, using which the Pokedex-DB can be queried and matching Pokémon can be added to the list.



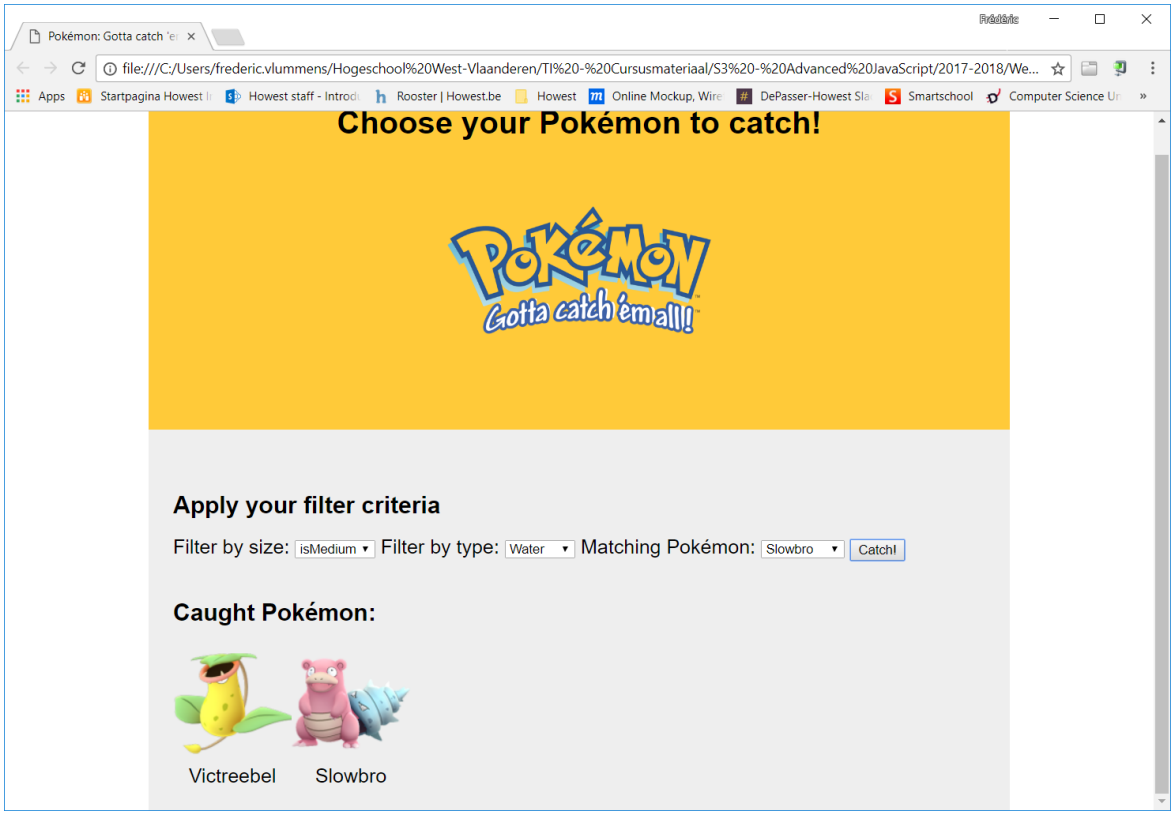
- Upon application launch, the first field displays the following possible sizes:
    - isAny
    - isSmall
    - isMedium
    - isLarge
- The second field displays the catch-all condition **isAny** but also all the various types of Pokémon that exist in the file `pokedex.js`. These should be retrieved dynamically:



- When the user selects a size and condition filter, the third field automatically displays the corresponding Pokémon:



- Once the user clicks the `Catch!` button, the selected Pokémon is added to the list and displayed at the bottom of the page:



5. Some attention points

- Use as many built-in array functions as possible (when appropriate).
- Populate the second field with all types of Pokémon, based on the information found in the file `pokedex.js`. The types should be sorted alphabetically and there shouldn't be any doubles. (HINT: first decide upon a *battle-plan* before you start coding: i.e., which array functions do I need and in which order)
- Before you can enable the filters, you'll need predicate functions that can check both the size and type of a Pokémon. A predicate is *just* a function that returns a `boolean` (No magic, "predicate" is just a name).
    - Write the predicate `isAny`, which can be applied for both the sizes and types. This predicate will always return `true`.

      JavaScript
      ```
      function isAny(pokemon) {
          return true;
      }
      ```
    - Write the predicates `isSmall`, `isMedium` and `isLarge` to check whether a Pokémon is less than a meter tall (small), taller than 2 meters (large) or somewhere in between (medium).
    - Now, also write a function `makeTypePredicate` which **returns** a predicate that

determines whether a certain Pokémon belongs to a given type. The following pseudo code might be of assistance:

```JavaScript
let isGrass = makeTypePredicate("grass");

if (isGrass(pokemon)) {
    console.log(`${pokemon.name} is a grass Pokémon.`);
}
else {
    console.log(`${pokemon.name} isn't a grass Pokémon.`);
}
```

→ Also, take a look at the video pokemon.mp4 to give you an idea how your application should behave.