Core | Deadline: Sunday of Week 1 | Difficulty Level: Intermediate | Est. Time: 00:00-03:00

Object Master (Core)

Use callbacks and functional programming to select exactly the data we need!

Learning Objectives

- Demonstrate proficiency in manipulating immutable data using callbacks
- Apply the structure of existing data to perform operations

Welcome to another **Core assignment**! Some students like to explore the assignments before they're finished reading through the lessons, and that's okay! It can be good for your brain to have a preview of what your future challenges might be. However, before you begin this assignment, it's important that you've first:

- Completed the preceding lesson modules
- Taken the knowledge checks to confirm your understanding
- Viewed lecture material related to the assignment topics
- Completed and submitted your practice assignments

Object Master

Some data we encounter will be immutable, meaning we can not modify it in place. Using .map(), .filter(), and .forEach(), create new arrays with the below requirements met.

```
const pokemon = Object.freeze([
    { "id": 1,   "name": "Bulbasaur",  "types": ["poison", "grass"] },
```

```
    { "id": 5,   "name": "Charmeleon", "types": ["fire"] },
    { "id": 9,   "name": "Blastoise",  "types": ["water"] },
    { "id": 12,  "name": "Butterfree", "types": ["bug", "flying"] },
    { "id": 16,  "name": "Pidgey",     "types": ["normal", "flying"] },
    { "id": 23,  "name": "Ekans",      "types": ["poison"] },
    { "id": 24,  "name": "Arbok",      "types": ["poison"] },
    { "id": 25,  "name": "Pikachu",    "types": ["electric"] },
    { "id": 35,  "name": "Clefairy",   "types": ["normal"] },
    { "id": 37,  "name": "Vulpix",     "types": ["fire"] },
    { "id": 52,  "name": "Meowth",     "types": ["normal"] },
    { "id": 63,  "name": "Abra",       "types": ["psychic"] },
    { "id": 67,  "name": "Machamp",    "types": ["fighting"] },
    { "id": 72,  "name": "Tentacool",  "types": ["water", "poison"] },
    { "id": 74,  "name": "Geodude",    "types": ["rock", "ground"] },
    { "id": 87,  "name": "Dewgong",    "types": ["water", "ice"] },
    { "id": 98,  "name": "Krabby",     "types": ["water"] },
    { "id": 115, "name": "Kangaskhan", "types": ["normal"] },
    { "id": 122, "name": "Mr. Mime",   "types": ["psychic"] },
    { "id": 133, "name": "Eevee",      "types": ["normal"] },
    { "id": 144, "name": "Articuno",   "types": ["ice", "flying"] },
    { "id": 145, "name": "Zapdos",     "types": ["electric", "flying"] },
    { "id": 146, "name": "Moltres",    "types": ["fire", "flying"] },
    { "id": 148, "name": "Dragonair",  "types": ["dragon"] }
];
```

For example, we could create a list of Pokemon that have names that start with the letter "B" by using the following code.

```
const bListPkmn = pokemon.filter( p => p.name[0] === "B" );
```

Or if we wanted to return an array of just the ids, we could use something like this.

```
const pkmnIds = pokemon.map(p => p.id);
```

Hint: Click here to read more about how to use `.map()` to update arrays

---

Using the above Pokemon array, find the following:

- [ ] an array of pokémon objects where the id is evenly divisible by 3

- [ ] an array of pokémon objects that are "fire" type
- [ ] an array of pokémon objects that have more than one type
- [ ] an array with just the names of the pokémon
- [ ] an array with just the names of pokémon with an id greater than 99
- [ ] an array with just the names of the pokémon whose only type is poison
- [ ] an array containing just the first type of all the pokémon whose second type is "flying"
- [ ] a count of the number of pokémon that are "normal" type
- [ ] an array with all pokemon except the pokemon with the id of 148
- [ ] an array with all pokemon and for pokemon id: 35 replacing "normal" with "fairy"