Intro to JavaScript Objects Lab

Exercise

Introduction

This lab provides an opportunity to practice defining, accessing, and manipulating objects.

A quick note before you dive in

Don't hesitate to collaborate with your classmates when working through labs.

If you get stuck during the lab, we recommend revisiting the lesson materials first. They're designed to provide you with the information and examples that will help you complete the exercises.

Lab exercises

Before we get started, copy and paste the following `game` object into `app.js`:

```
Copy
```

```
const game = {
  party: [],
  gyms: [
    { location: "Pewter City", completed: false, difficulty: 1 },
    { location: "Cerulean City", completed: false, difficulty: 2 },
    { location: "Vermilion City", completed: false, difficulty: 3 },
    { location: "Celadon City", completed: false, difficulty: 4 },
    { location: "Fuchsia City", completed: false, difficulty: 5 },
    { location: "Saffron City", completed: false, difficulty: 6 },
    { location: "Cinnabar Island", completed: false, difficulty: 7 },
    { location: "Viridian City", completed: false, difficulty: 8 },
  ],
  items: [
    { name: "potion", quantity: 4 },
    { name: "pokeball", quantity: 8 },
    { name: "rare candy", quantity: 99 },
  ],
}
```

As you work through the following lab, copy and paste the following exercises into your `app.js` file. To check your work, open the file in a web browser and inspect the console output, or run the file in your terminal using `node app.js`.

Exercise 1

Spend some time inspecting the `pokemon` array by running the following command:

```
Copy
```

```
console.dir(pokemon, { maxArrayLength: null })
```

In this lab, we're using a method you might be unfamiliar with - `console.dir()`. We're using it here so that you can see all of the Pokemon in the console. Normally, both `console.dir()` and `console.log()` show only the first 100 items in an array. However, by using `{ maxArrayLength: null }` with `console.dir()`, we display every item in the array. This specific option isn't available with `console.log()`, making `console.dir()` the necessary choice for the full visibility needed in this lab.

Take note of the shape of the data here. Each Pokemon object in the array has the following properties:

- Number: A number between 1 and 151.
- Name: A string representing the Pokémon's name.
- Type: A string indicating the Pokémon's type.
- HP (Hit Points): A numerical value representing the Pokémon's health.
- Starter: A boolean indicating whether the Pokémon is a starter.

There's some game-specific terminology here, if you're new to Pokémon, here's an explanation of the terms used in the game:

- **Number:** Each Pokémon has a unique number as its identifier.
- **Type:** This refers to a Pokémon's primary abilities. While Pokémon can have multiple types, we're focusing on their primary type here for simplicity.
- **Hit Points (HP):** This is a measure of a Pokémon's health, expressed in a numerical value.
- **Starter Pokémon:** At the beginning of the game, players choose a starting Pokémon. In our data, starter Pokémon are marked with a starter property set to true.

The starter Pokémon options are:

- Pokémon 1: Bulbasaur
- Pokémon 4: Charmander
- Pokémon 7: Squirtle
- Pokémon 25: Pikachu

When you've completed your inspection of the data, you can comment out the `console.dir()` method and use `console.log()` to log JUST the name of the Pokemon with the number 59 using the index of the Pokemon in the array. Feel free to uncomment the `console.dir()` as needed to help you complete the rest of the lab.

Exercise 2

Next, add the following `console.log`.

Run the following:

```
Copy
```

```
console.log(game)
```

Take a moment to familiarize yourself with the `game` object being logged. It has four properties: `party`, `gyms`, and `items`. All of these hold or will hold an array of objects.

As you move through the exercises:

- `game.party` will hold an array of Pokemon objects that will be retrieved from the `pokemon` array we reviewed in Exercise 1. These are Pokemon that you have caught.
- `game.gyms` holds an array of objects, each representing a gym (checkpoints placed throughout the game). Note the `completed` boolean property on each. This will be important later.
- `game.items` holds an array of objects, each representing an item in a player's inventory.

When you've completed your inspection of the data, you can comment out the console.log and move on to Exercise 3. Feel free to uncomment the console.log as needed to help you complete the rest of the lab.

Exercise 3

```
Copy
```

```
/*
Exercise 3
1. Add a new property to the `game` object. Let's call it "difficulty".
2. Choose a value for "difficulty" that you think fits the game. Ex: "Easy", "Med" or "Hard". How would you assign

Solve Exercise 3 here:
*/
```

Exercise 4

```
Copy
```

```
/*
Exercise 4
1. Select a starter Pokémon from the `pokemon` array. Remember, a starter Pokémon's `starter` property is true.
2. Add this Pokémon to the `game.party` array. Which array method will you use to add them?

Solve Exercise 4 here:
*/
```

Exercise 5

```
Copy
```

```
/*
Exercise 5
1. Choose three more Pokémon from the `pokemon` array and add them to your party.
2. Consider different attributes like `type` or `HP` for your selection. Which array method will you use to add th

Solve Exercise 5 here:
*/
```

Exercise 6

```
Copy
```

```
/*
Exercise 6
1. Set the `completed` property to true for gyms with a `difficulty` below 3.
2. Think about how you'd loop through the `game` array to check and update the `completed` property.

Solve Exercise 6 here:
*/
```

Exercise 7

```
Copy
```

```
/*
Exercise 7
1. Evolve the starter Pokémon you added to your party earlier. Each starter Pokémon evolves into a specific one.
2. How would you replace the current starter Pokémon in your party with its evolved form?

Hint:
  - Pokemon 1: Bulbasaur evolves into Pokemon 2: Ivysaur.
  - Pokemon 4: Charmander evolves into Pokemon 5: Charmeleon
  - Pokemon 7: Squirtle evolves into Pokemon 8: Wartortle
  - Pokemon 25: Pikachu evolves into Pokemon 26: Raichu

More Hints: The existing starter Pokémon will be *replaced* in your party with the Pokémon it evolved into. When w


Solve Exercise 7 here:
*/
```

Visit the MDN docs to learn more about the `splice()` method.

Exercise 8

```
Copy
```

```
/*
Exercise 8
1. Print the name of each Pokémon in your party.
2. Consider using a loop or an array method to access each Pokémon's name.

Solve Exercise 8 here:
*/
```

Exercise 9

```
Copy
```

```
/*
Exercise 9
1. Can you print out all the starter Pokémon from the `pokemon` array?
2. Think about how you can identify a starter Pokémon and then log their names.


Solve Exercise 9 here:
*/
```

Exercise 10

```
Copy
```

```
/*
Exercise 10
Create a method called `catchPokemon` and add it to the `game` object. You should not need to edit the original ga
  - Accept an object as a parameter called `pokemonObj`
  - Add the `pokemonObj` to the `game.party` array.
  - not return anything

After writing this method, call it and pass in a Pokemon object of your choice from the `pokemon` data to catch it

Solve Exercise 10 here:
*/
```

Exercise 11

```
Copy
```

```
/*
Exercise 11
1. Copy the `catchPokemon` method that you just wrote above, and paste it below. Modify it so that it also decreas
2. How will you find and update the quantity of pokeballs in the `game.items` array?

Tips:
For this exercise, it's okay to have a negative number of pokeballs.
After updating the method, call it and pass in a Pokemon object of your choice from the `pokemon` data to catch it
Also, log the `game.items` array to confirm that the pokeball quantity is being decremented.

Solve Exercise 11 here:
*/
```

Exercise 12

```
Copy
```

```
/*
Exercise 12
1. Similar to Exercise 6, now complete gyms with a difficulty below 4. How will you approach this?
(Change the value of `complete` in the qualifying objects from false to true).

Solve Exercise 12 here:
*/
```

Exercise 13

```
Copy
```

```
/*
Exercise 13
1. Create a `gymTotals` method in `game` to tally completed and incompleted gyms.
2. How will you iterate through the `gyms` array and update the tally? Remember to log the final tally.

This method should:
  - Not accept any arguments.
  - Initially create a constant `gymTally`, which is an object that has two
    properties: `completed` and `incomplete`, both of which are initially set to 0.
  - Iterate over each gym in the `game.gyms` array and update the
    properties on `gymTally` as follows:
    - `completed` should count how many gyms in the array have a value of `true`
      for their `completed` property.
    - `incomplete` should count how many gyms in the array have a value of
      `false` for their `completed` property.
  - Log the value of `gymTally`.
  - The method should not return anything.

For example, if five gym objects have a value of `true` on their `completed` property and three gym objects have a

Solve Exercise 13 here:
*/
```

Exercise 14

```
Copy
```

```
/*
Exercise 14
1. Add a `partyCount` method to `game` that counts the number of Pokémon in your party.

This method should:
  - Not accept any arguments.
  - Count the number of Pokémon in the party.
  - return the found number of Pokémon in the party.

Solve Exercise 14 here:
*/
```

Exercise 15

```
Copy
```

```
/*
Exercise 15
1. Now, complete gyms with a difficulty below 6. Reflect on how this is similar to or different from the previous
Change the value of `complete` in the qualifying objects from false to true).

Solve Exercise 15 here:
*/
```

Exercise 16

```
Copy
```

```
/*
Exercise 16
1. Log the entire `game` object to the console. Take a moment to review the changes you've made throughout the exe


Solve Exercise 16 here:
*/
```

Congrats, you're done! If you'd like to challenge yourself, you can continue on to the exercises in the Level Up.