Exercise: Working with Objects

Objects - Step 1

Introduction to Objects

Along with the primitive types (strings, numbers, Booleans) we have objects. Objects allow you to map keys to values.

For example, the key `firstName` could map to the value "Judith". Or the key `isLawyer` could map to the Boolean value `true`. These are examples of key-value mappings.

Values of object properties can either contain primitive data types (such as a string, number, or Boolean) or other objects.

This is an empty object:

```
let empty_object = {}
```

In this example we are declaring three properties in an object: `firstName`, `lastName`, and `isLawyer` as you can see below.

```
let firstObj = {
    firstName: "Judith",
    lastName: "Scott",
    isLawyer: true
};
```

In this example, `firstname` is a key, and `Judith` is a value. This is called a key-value pair. An object can contain a number of key- value pairs.

After creating an object, it is helpful to access the values using the object's keys.

Accessing Object Values

To access the values in an object, you could use the dot notation:

```
firstObj.firstName;       // returns "Judith"
firstObj.lastName;        // returns "Scott"
firstObj.isLawyer;    // returns true
firstObj.keyDoesntExist;  // returns undefined
```

Or you could use the bracket notation:

```
firstObj["firstName"];       // returns "Judith"
firstObj["lastName"];        // returns "Scott"
firstObj["isLawyer"];    // returns true
firstObj["keyDoesntExist"];  // returns undefined
```

You will learn the difference between these two later on.

Now look at another object:

```
let Judith = {
    name: "Judith",
    houseOwner: true,
    officeOwner: true
};
```

In the object above, the keys are `name`, `houseOwner`, `officeOwner` and the values are `Judith`, `true` and `true`.

Bracket Notation vs. Dot Notation

If you want to access values in the object you can either use brackets ([]) or dot notation. Through the examples below, you'll learn how to use dot notation versus bracket notation.

Dot notation

```
let person = {
    firstName: "Lara",
    lastName: "Scott",
    favoriteColor: "purple",
    job: "instructor",
    isDeveloper: true
};

person.firstName; // Lara
```

Using dot notation allows you to access an object's value using a key. The syntax is as follows: `object.key`. In the example above, you're accessing the value of the key `firstname` using `person.firstname`. The result is "Lara".

Using brackets ([])

```
let person = {
    firstName: "Lara",
    lastName: "Scott",
    favoriteColor: "purple",
    job: "instructor",
    isDeveloper: true
};

person["lastName"]; // Scott
person[favoriteFood]; // This gives an error, because there is no variable called "favoriteFood"!
```

Using brackets allows you to access an object's value using a key. The syntax is as follows: `object['key']`. In the example above, you're accessing the value of the key `lastName` using `person["lastName"]`. The result is "Scott". Note that the quotation marks are important in this syntax. Using `person[lastName]` will result in an error since the quotations around `lastname` are missing.

Task Instructions

- Open the objects-01 folder.
- Given the function getProduct, return an object with the following key: value properties
- id:productId
- serialNumber: 'WD579000'
- manufacturer: 'Apple'
- price: 1500

Hint: Don't forget to put a comma in between properties.

Task

- Given the getObj function, return an object with three properties.

Objects - Step 2

Adding to Objects

To add properties to objects, it is best to do so using the . operator.

Note: Properties are `key: value` pair entries in objects. For example, `name: 'Isaac Asimov'`

```
let person = {
    name: 'Isaac Asimov',
    occupation: 'science fiction writer',
}

person.number_of_books = 500
```

This code will result in adding the key "number_of_books" to the object `person`, and assign a value of "500" to that key. The new object will look like this:

```
person = {
    name: 'Isaac Asimov',
    occupation: 'science fiction writer',
    number_of_books: 500
}
```

Removing Keys from Objects

We can remove a key from an object by using the delete keyword.

```
let person = {
    name: 'Isaac Asimov',
    occupation: 'science novels writer',
    number_of_books: 500,
    university: 'University of Toronto'
}
delete person.university; // returns true
```

This will remove the key:value pair "university: University of Toronto" from the `person` object.

Task Instructions

Given the bike object, open the `objects-02` folder, in `main.js`, change the speed value to 12, and then return the new value.

When you're done, check the task as completed.

Task

- Given the bike object, change the speed value to 12 and then return the new value.