Theory: Introduction to Objects

9 minutes 0 / 5 problems solved

You are already familiar with strings and numbers. These data types are considered *primitive* because they can only store one value. This can be inconvenient when you have to deal with sets of related information. To work comfortably with collections of values, there are **objects**. Let's find out what objects are and how they work!

§1. Not everything in JS is an object

An **object** is a *non-primitive* data type that represents an unordered collection of *properties*. A **property** is a part of the object that imitates a variable. It consists of a **key** and a **value** separated by a colon. The key can only be a string or `Symbol` (data type, which we will get acquainted with later), but the value may be of any data type. You can create empty objects without a single property:

```
let book = {};
```

There is an alternative way to create an empty object, but it is rarely used in practice:

```
let book = new Object();
```

You can check that both of these methods create an object using the `typeof` operator that you already know.

```
let book1 = {};
let book2 = new Object();

console.log(typeof book1); // object
console.log(typeof book2); // object
```

If you want to create an object with several properties, all but the last one must be followed by a comma. To understand the syntax better, take a look at the following example:

```
let country = {
  name: "Netherlands",
  population: 17.18
};
```

As in many cases in JavaScript, object creation often starts with defining and initializing a variable. In the example above, we've assigned the object the name `country`. Then two properties were specified in curly brackets: the key `name` with the value `"Netherlands"`, and the key `population` with `17.18`. As you can see, objects are very useful for grouping data.

> The syntax with curly brackets used to create objects has its own name: **literal notation**. It's not the only way to do it, but probably the most common.

§2. Properties

Objects are one of the pillars of the JavaScript language, so it's important to know how to work with their properties.

- There is an opportunity to refer to the properties. To access the properties, we use a record with the object name and a dot. Let's use the previous code sample and try to access the `name` property:

```
console.log(country.name); // Netherlands
```

Current topic:

Introduction to Objects

Topic depends on:

- ✕ Variables
- ✕ Strings and numbers

Topic is required for:

- Null and Undefined
- Window Object
- DOM methods
- Audio Object

Table of contents:

- Properties can also be added using the dot symbol and the `=` assignment symbol. Let's add to our object a property with the key `capital` and the value `"Amsterdam"` :

```
country.capital = "Amsterdam";
```

- To delete a property, we can use the `delete` operator and a dot. This is how removing the `population` property for an object named `country` will look like:

```
delete country.population;
```

§3. Conclusion

Working with grouped data is one of the most common tasks in programming. You now have an important tool that will help you with that: objects and their properties. Way to go!

Report a typo

129 users liked this theory. **3** didn't like it. **What about you?**

Start practicing

Comments (5) Hints (0) Useful links (0) Show discussion