Objects And Its Internal Representation In JavaScript

Task for JavaScript - Day -1 : Introduction to Browser & web

Submitted By

Yeswanth R

Objects overview

Objects in JavaScript, just as in many other programming languages, can be compared to objects in real life. The concept of objects in JavaScript can be understood with real life, tangible objects.

In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a color, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

As we know from the chapter Data types, there are eight data types in JavaScript. Seven of them are called "primitive", because their values contain only a single thing (be it a string or a number or whatever).

In contrast, objects are used to store keyed collections of various data and more complex entities. In JavaScript, objects penetrate almost every aspect of the language. So we must understand them first before going in-depth anywhere else.

An object can be created with figure brackets {…} with an optional list of properties. A property is a "key: value" pair, where key is a string (also called a "property name"), and value can be anything.

We can imagine an object as a cabinet with signed files. Every piece of data is stored in its file by the key. It's easy to find a file by its name or add/remove a file.

Working with objects

JavaScript is designed on a simple object-based paradigm. An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method. In addition to objects that are predefined in the browser, you can define your own objects. This chapter describes how to use objects, properties, functions, and methods, and how to create your own objects.

Accessing properties

To access a property of an object, you use one of two notations: the dot notation and array-like notation.

1) The dot notation (.)

The following illustrates how to use the dot notation to access a property of an object:

```
objectName.propertyName
```

Code language: CSS (css)

For example, to access the firstName property of the person object, you use the following expression:

```
person.firstName
```

Code language: CSS (css)

This example creates a person object and shows the first name and last name to the console:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

console.log(person.firstName);
console.log(person.lastName);
```

Code language: JavaScript (javascript)

2) Array-like notation ([])

The following illustrates how to access the value of an object's property via the array-like notation:

```
objectName['propertyName']
```

Code language: CSS (css)

For example:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

console.log(person['firstName']);
console.log(person['lastName']);
```

Code language: JavaScript (javascript)

When a property name contains spaces, you need to place it inside quotes.

For example, the following address object has the 'building no' as a property:

```
let address = {
    'building no': 3960,
    street: 'North 1st street',
    state: 'CA',
    country: 'USA'
};
```

Code language: JavaScript (javascript)

To access the 'building no' property, you need to use the array-like notation:

```
address['building no'];
```

Code language: CSS (css)

If you use the dot notation, you'll get an error:

```
address.'building no';
```

Code language: JavaScript (javascript)

Error:

```
SyntaxError: Unexpected string
```

Code language: JavaScript (javascript)

Note that it is **not** a good practice to use spaces in the property names of an object.

Reading from a property that does **not** exist will result in an undefined. For example:

```
console.log(address.district);
```

Code language: CSS (css)

Output:

```
undefined
```

Code language: JavaScript (javascript)

Modifying the value of a property

To change the value of a property, you use the assignment **operator (=)**. For example:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

person.firstName = 'Jane';

console.log(person);
```

Code language: JavaScript (javascript)

Output:

```
{ firstName: 'Jane', lastName: 'Doe' }
```

Code language: CSS (css)

In this example, we changed the value of the firstName property of the person object from 'John' to 'Jane'.

Adding a new property to an object

Unlike objects in other programming languages such as Java and C#, you can add a property to an object after object creation.

The following statement adds the age property to the person object and assigns 25 to it:

```
person.age = 25;
```

Deleting a property of an object

To delete a property of an object, you use the delete operator:

```
delete objectName.propertyName;
```

Code language: JavaScript (javascript)

The following example removes the age property from the person object:

```
delete person.age;
```

Code language: JavaScript (javascript)

If you attempt to reaccess the age property, you'll get an undefined value.

Checking if a property exists

To check if a property exists in an object, you use the in operator:

```
propertyName in objectName
```

The in operator returns true if the propertyName exists in the objectName.

The following example creates an employee object and uses the in operator to check if the ssn and employeeId properties exist in the object:

```
let employee = {
    firstName: 'Peter',
    lastName: 'Doe',
    employeeId: 1
};

console.log('ssn' in employee);
console.log('employeeId' in employee);
```

Code language: JavaScript (javascript)

Output:

```
false
true
```

Code language: JavaScript (javascript)

Summary

An object is a collection of key-value pairs.

Use the dot notation (.) or array-like notation ([]) to access a property of an object.

The delete operator removes a property from an object.

The in operator check if a property exists in an object.