Objects in JavaScript. 23rd **July** 2020

Introduction to objects.

Objects, in JavaScript, is it's most important data–type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript's primitive data–types(Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data–types all store a single value each (depending on their types).

- Objects are more complex and each object may contain any combination of these primitive data–types as well as reference data–types.
- An object, is a reference data type. Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don't actually store the value.
- Loosely speaking, objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of "key: value" pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

Remember: In JavaScript, objects are king. If you understand objects, you understand JavaScript.

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

To understand this rather abstract definition, let us look at an example of a JavaScript Object :

```
let User = {

    name : "Liam",
    password: 12345;
    age: 23,
    country: “Ghana”
}
```

In the above example we created an object called **User**. In our user object name, password, age and countries are all keys while Liam, 12345, 23 and Ghana are values of these respectively.

So in short, Object is a non-primitive data type in JavaScript. It is like any other variable, the only difference is that an object holds multiple values in terms of properties and methods. Properties can hold values of primitive data types and methods are functions.

Creating Objects.

There are several ways or syntax’s to create objects. One of which, known as the Object literal syntax, we have already used. Besides the object literal syntax, objects in JavaScript may also be created using the constructors, Object Constructor or the prototype pattern.

Object Literal.

The object literal is a simple way of creating an object using { } brackets.

You can include key-value pair in { }, where key would be property or method name and value will be value of property of any data type or a function. Use comma (,) to separate multiple key-value pairs.

The following example creates an object using object literal syntax.

```
var emptyObject = {}; // object with no properties or methods

let person = { firstName: "liam" }; // object with single property
// object with single method
let message = {
             showMessage: function (val) {
                       alert(val);
             }};
// object with properties & method
let person = {
             firstName: "John",
             lastName: "Liam",
             age: 15,
             getFullName: function () {
                   return this.firstName + ' ' + this.lastName
             }};
```

You must specify key-value pair in object for properties or methods. Only property or method name without value is not valid.
Example:

```
let person = { firstName }; // this syntax is invalid
```

JavaScript Object Properties & Methods.

You can get or set values of an object′s properties using dot notation or bracket. However, you can call an object′s method only using dot notation.

```
let person = {
             firstName: "John",
             lastName: "Liam",
             age: 25,
             getFullName: function () {
                return this.firstName + ' ' + this.lastName
             }
          };

person.firstName; // returns John
person.lastName; // returns Liam

person["firstName"];// returns John
person["lastName"];// returns Liam

person.getFullName();
```

Object Constructor

The second way to create an object is with Object Constructor using **new** keyword. You can attach properties and methods using dot notation. Optionally, you can also create properties using [ ] brackets and specifying property name as string.

Example:

```
let person = new Object();
// Attach properties and methods to person object

person.firstName = "John";

person["lastName"] = "Liam";

person.age = 24;

person.getFullName = function () {

        return this.firstName + ' ' + this.lastName;

    };

// access properties & methods

person.firstName; // John

person.lastName; // Liam

person.getFullName(); // John Liam
```

Undefined Property or Method

JavaScript will return 'undefined' if you try to access properties or call methods that do not exist.

If you are not sure whether an object has a particular property or not, then use hasOwnProperty() method before accessing properties.

```
let person = new Object();
person.firstName; // returns undefined

if(person.hasOwnProperty("firstName")){

        person.firstName;

    }
```

Prototypes.

Another way to create objects involves using prototypes. Every JavaScript function has a prototype object property by default(it is empty by default). Methods or properties may be attached to this property. A detailed description of prototypes is beyond the scope of this introduction to objects.

However you may familiarize yourself with the basic syntax used as below:

```
let obj = Object.create(prototype_object, propertiesObject)
       // the second propertiesObject argument is optional
```

Example:

```
let footballers = {
    position: "Striker"
}
let footballer1 = Object.create(footballers);
    // Output : Striker
console.log(footballer1.position);
```

In the above example footballers served as a prototype for creating the object “footballer1”.

All objects created in this way inherits all properties and methods from its prototype objects. Prototypes can have prototypes and those can have prototypes and so on. This is referred to as prototype chaining in JavaScript. This chain terminates with the Object.prototype which is the default prototype fallback for all objects. Javascript objects, by default, inherit properties and methods from Object.prototype but these may easily be overridden. It is also interesting to note that the default prototype is not always Object.prototype.For example Strings and Arrays have their own

default prototypes – String.prototype and Array.prototype respectively.

Access Object Keys.

Use for..in loop to get the list of all properties and methods of an object.

```
let person = new Object();
person.firstName = "John";

person["lastName"] = "Liam";

person.age = 23;

person.getFullName = function () {

      return this.firstName + ' ' + this.lastName;

   };

for(let key in person){

      console.log(key);

   };
```

Nested JavaScript Object.

You can assign another object as a property of an object.

```
let person = {
   firstName: "John",
   lastName: "Liam",
   age: 23,
   address: {
      id: 10101243,
      country:"Ghana"
   }};
person.address.country; // returns "Ghana"
```

Always Remember:

JavaScript object is a standalone entity that holds multiple values in terms of properties and methods. Object property stores a literal value and method represents function.

– An object can be created using object literal or object constructor syntax.

object literal

```
let person  = {  firstName: "John",

                  lastName: "Liam",

                  age: 23,

                 getFullName: function () {

              return this.firstName + '  ' + this.lastName; }  };
```

Object constructor:

```
let person = new Object();

person.firstName = "John";

person["lastName"] = "Liam";

person.age = 23;

person.getFullName = function () {

        return this.firstName + '  ' + this.lastName;

};
```

– Object properties and methods can be accessed using dot notation or [ ] bracket.

– An object is passed by reference from one function to another.

–An object can include another object as a property.

The best performance improvement is the transition from the nonworking state to the working state.

Best wishes.
Lux Tech Academy