# Objects in JavaScript

### What are Objects in JavaScript

An object in JavaScript is a collection of key-value or name-value pairs. Each key-value pair is called a property.

A property’s value can be a function, an array, an object itself or any primitive data type i.e. integer, string, etc.

Let’s consider a simple object:

var person = {

firstName: "Bablu",

lastName: "Ahmed",

age: 30,

fullName: function(){

return this.firstName + " " + this.lastName

}

}

Here firstName, lastName, and fullName with value are properties of the same object i.e. person

For more information:

firstName: "Bablu" is a Property and firstName is called P**roperty name** & “Bablu” is called P**roperty value**.

Every person will have these properties but their values may be different i.e. firstName, lastName keys may have different values for different person.

console.log(person);

#### Properties of the object can be accessed using the following two notations:

1. Dot notation

2. Square bracket notation

### Dot notation

person.firstName; //Output: Bablu

person.fullName(); //Output: Bablu Ahmed

New properties can be added using the dot notation as shown below:

person.age = 27

person.getAge = function(){

return this.age;

}

Now console person and we will find new properties age and getAge() added to the person object as shown below:

Console.log(person);

### Square bracket notation

person["firstName"]; //Output: Bablu

person ["fullName"](); //Output: Bablu Ahmed

New properties can be added using the square bracket notation as shown below:

person ["weight"] = 65

person.getWeight = function(){

return this.weight;

}

Properties can also be accessed using a variable just assign the property name to the variable as shown below:

var firstNameProperty = "firstName";

console.log(person[firstNameProperty]) // Output: Bablu

> \*\*Note\*\*: Above method of using variable to access property names cannot be used to access properties of the object using dot notation

```javascript

Console.log(human.firstNameProperty) //Output: undefined

```

>\*\*Note\*\*: If we try to access the methods without using the \*\*()\*\*, output will be method definition as shown below:

```javascript

console.log(human.fullName); //Output: function definition

```

#### Console output

![](https://github.com/rupeshmi/CodeSprint/blob/dev/JavaScript/Part1/CodeSnippets/accessMethodWithoutBrakcets.png)

>An object property name can be any valid JavaScript string, or anything that can be converted to a string, including the empty string. However, any property name that is not a valid Javascript identifier (for example, a property name that has a space or a hyphen, or that starts with a number) can only be accessed and added to the object property using the square bracket notation

```javascript

human["date of birth"] = "Nov 28";

human[12] = 12;

human.12 = 12; //gives error

console.log(human);

```

#### Console output

![alt text](https://github.com/rupeshmi/CodeSprint/blob/dev/JavaScript/Part1/CodeSnippets/propertyNameOtherThanString.png)

```javascript

console.log(human.12); //Gives error

console.log(human[12]); //Output: 12

```

## Different ways of creating a JavaScript object:

### Using object literal

> \*\*human\*\* object created above is an example of creating an object using object literal

```javascript

var human = {

firstName: "Virat",

lastName: "Kohli",

age: 30,

fullName: function(){

return this.firstName + " " + this.lastName

}

}

```

### Using new Object()

```javascript

var human = new Object()

console.log(human);// Creates an empty object

```

> We can added as many properties as we want usign either the dot notation or the square bracket notation

```javascript

human.firstName = "Virat";

human.lastName = "Kohli";

human.age = 30;

human.fullName = function(){

return this.firstName + " " + this.lastName;

}

console.log(human)

```

#### Console output

![alt text](https://github.com/rupeshmi/CodeSprint/blob/dev/JavaScript/Part1/CodeSnippets/BasicObjectExample.png)

>Method 1 and 2 do exactly the same thing. There is no need to use new Object().

>For simplicity, readability and execution speed, use the first one that is object literal.

### Object constructor

> Objects can be created using the constructor function using the following two steps:

> 1. Define the object type by writing the constructor function.

By convention, name of the constructor function starts with a capital letter

> 2. Create an instance of the object with new

>To define an object type, create a function for the object type that specifies its name, properties, and methods.

>Lets create constructor function for the human type object

```javascript

function Human(firstName, lastName){

this.firstName = firstName,

this.lastName = lastName,

this.fullName = function(){

return this.firstName + " " + this.lastName;

}

}

```

>Now you can create as many objects as you want using this constructor function:

```javascript

var viratKohli = new Human("Virat", "Kohli");

console.log(viratKohli);

```

#### Console output

![alt text](https://github.com/rupeshmi/CodeSprint/blob/dev/JavaScript/Part1/CodeSnippets/ObjectUsingConstructorFn.png)

```javascript

var sachinTendulkar = new Human("Sachin", "Tendulkar");

console.log(sachinTendulkar);

```

#### Console output

![alt text](https://github.com/rupeshmi/CodeSprint/blob/dev/JavaScript/Part1/CodeSnippets/SachinConst.png)

#### Delete a property from an object

>To delete a property from an object we can use the ‘delete’ operator. You cannot delete properties that were inherited, nor can you

>delete properties with their attributes set to configurable.

>\*\*\*‘delete’\*\*\* operator returns true if the delete was successful. It also return true if the property to delete was non-existent or

>the property could not be deleted

```javascript

delete human.firstName; // return true

console.log(human);

```

> Let's see what happens if we try to call fullName method which uses both the firstName and lastName property of human object

```javascript

console.log(human.fullName());// undefined Kohli

```

>Output is \*\*undefined\*\* because we were trying to access firstName property of human object which does not exists