[Rupesh Mishra](https://hackernoon.com/@happymishra66?source=post_header_lockup)Follow

Full Stack Developer

May 4, 2017

**Objects in JavaScript**

You can also read the [article on Github](https://github.com/happymishra/JavaScriptTutorials/blob/master/Part1/Object.md)

An object in JavaScript is a collection of key-value pairs. Each key-value pair is called as a property. A property can be a function, an array, an object itself or any primitive data type i.e. integer, string, etc. Functions in object are called as methods.

Example:

Here firstName, lastName, and fullName are properties of the same object i.e. *human*. *firstName* is the key and *Virat* is the value of the property.

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

**Dot notation:**

human.firstName; //Output: Virat  
  
human.fullName(); //Output: Virat Kohli

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

human.age = 27  
human.getAge = function(){  
 return this.age;  
}

**Square bracket notation:**

human["firstName"]; //Output: Virat  
  
human["fullName"](); //Output: Virat Kohli

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

human["weight"] = 65  
human.getWeight = function(){  
 return this.weight;  
}

Properties can also be accessed using a variable having value equal to the property name as shown below:

var firstNameProperty = "firstName";  
  
console.log(human[firstNameProperty]) // Output: Virat

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

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

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

**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.

delete human.firstName; // return true

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

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

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

In the [next part](https://medium.com/@happymishra66/create-objects-in-javascript-10924cfa9fc7), we will see various ways of creating objects in JavaScript

1. [Virtual DOM in ReactJS](https://medium.com/@happymishra66/virtual-dom-in-reactjs-43a3fdb1d130)
2. [Execution Context in JavaScript](https://medium.com/@happymishra66/execution-context-in-javascript-319dd72e8e2c)
3. [Prototypes in JavaScript](https://medium.com/@happymishra66/prototypes-in-javascript-5bba2990e04b)
4. [Inheritance in JavaScript](https://medium.com/@happymishra66/inheritance-in-javascript-21d2b82ffa6f)
5. [Create objects in JavaScript](https://medium.com/@happymishra66/create-objects-in-javascript-10924cfa9fc7)
6. [this in JavaScript](https://hackernoon.com/this-in-javascript-8e8d4cd3930)