* An object is an instance which contains a set of key value pairs. Unlike primitive data types, objects can represent multiple or complex values and can change over their life time. The values can be scalar values or functions or even array of other objects.

* The contents of an object are called **properties** (or members), and properties consist of a **name** (or key) and **value**. Property names must be strings or symbols, and values can be any type (including other objects).

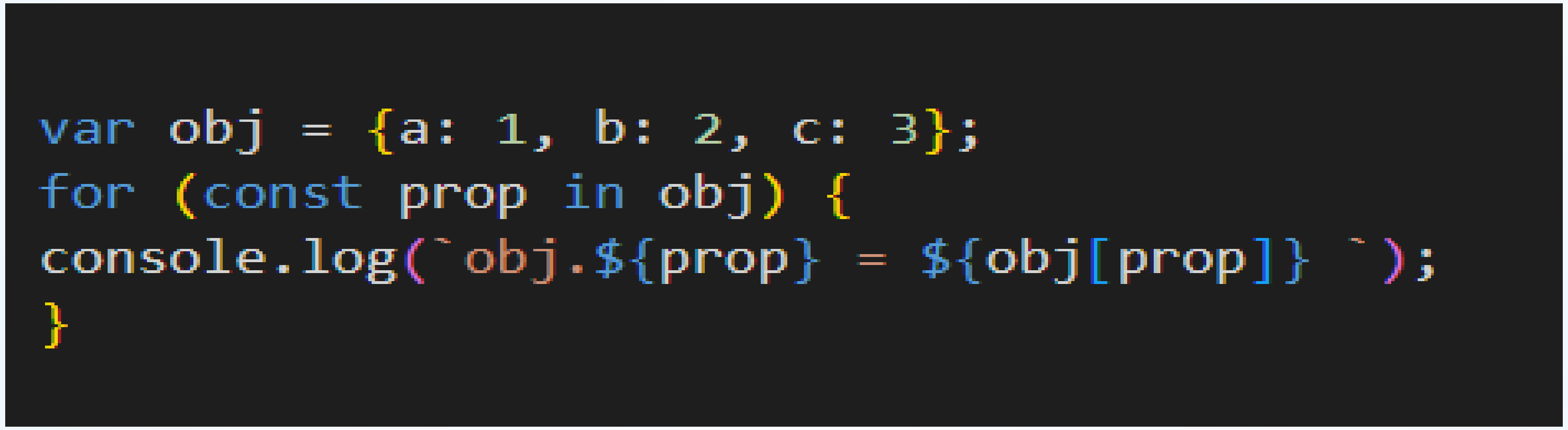
* Unassigned properties of an object are undefined (and not null).

Objects

* for...in loops This method traverses all enumerable properties of an object and its prototype chain
* Object.keys(o) This method returns an array with all the own (not in the prototype chain) enumerable properties' names ("keys") of an object o.
* Object.getOwnPropertyNames(o) This method returns an array containing all own properties' names (enumerable or not) of an object o.

*Enumerate the properties of an object

* The for...in statement iterates a specified variable over all the enumerable properties of an object. For each distinct property, JavaScript executes the specified statements.

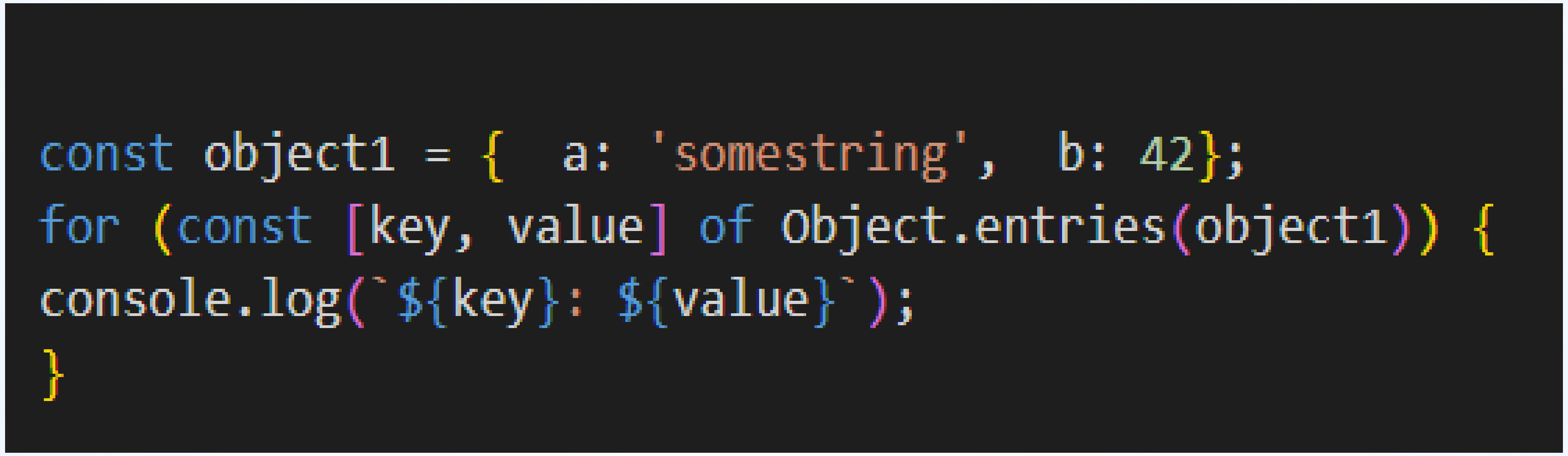


```
var obj = {a: 1, b: 2, c: 3};
for (const prop in obj) {
console.log(`obj.${prop} = ${obj[prop]} `);
}
```

* for...in statement

*Object.entries

*The Object.entries() method returns an array of a given object's own enumerable string-keyed property [key, value] pairs.



```
const object1 = {  a: 'somestring',  b: 42};
for (const [key, value] of Object.entries(object1)) {
console.log(`${key}: ${value}`);
}
```

Object() Constructor

* JavaScript provides a special constructor function called **Object()** to build the object. The new operator is used to create an instance of an object. To create an object, the new operator is followed by the constructor method.

* The **Object.assign()** method is used to copy the values of all enumerable own properties from one or more source objects to a target object. It will return the target object.
* Unlike copying objects, when objects are merged, the larger object doesn’t maintain a new copy of the properties. Rather it holds the reference to the properties contained in the original objects.

The Object.assign() Function

* you can remove a property by using the delete operator.

*Deleting Properties

* In JavaScript, objects are a reference type. Two distinct objects are never equal, even if they have the same properties. This is because, they point to a completely different memory address. Only those objects that share a common reference yields true on comparison.

*Comparing Objects

- array is a collection of values of the same data type. It is a user-defined type.
- An array can also be created using the Array object. The Array constructor can be passed as −
  - A numeric value that represents the size of the array or.
  - A list of comma separated values.

Arrays

- pop() Removes the last element from an array and returns that element.
- push() Adds one or more elements to the end of an array and returns the new length of the array.
- shift() Removes the first element from an array and returns that element slice.
- unshift() Adds one or more elements to the front of an array and returns the new length of the array.

Array Methods

*Array Methods

- slice(start_index, upto_index) extracts a section of an array and returns a new
- splice(index, count_to_remove, addElement1, addElement2, ...) removes elements from an array and (optionally) replaces them. It returns the items which were removed from the array.
- sort() sorts the elements of an array in place, and returns a reference to the array.
- indexOf(searchElement[, fromIndex]) searches the array for searchElement and returns the index of the first match.
- lastIndexOf(searchElement[, fromIndex]) works like indexOf, but starts at the end and searches backwards.

*Array Methods

- *concat() Returns a new array comprised of this array joined with other array(s) and/or value(s)
- *filter() Creates a new array with all of the elements of this array for which the provided filtering function returns true.
- *forEach() Calls a function for each element in the array.
- *reverse() Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.

* The debugger statement invokes any available debugging functionality, such as setting a breakpoint. If no debugging functionality is available, this statement has no effect.

* debugger