1. Arrays

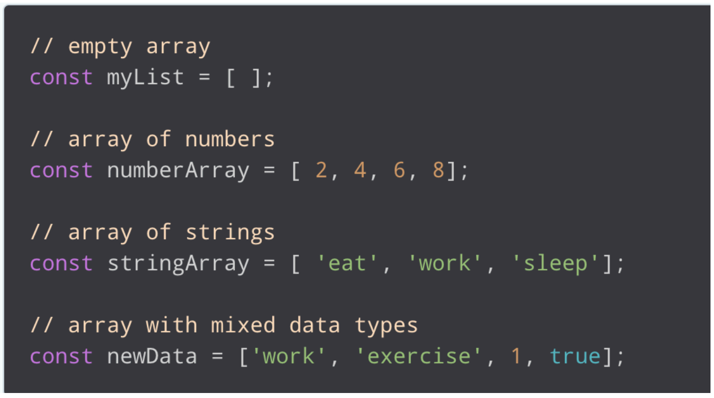
An array is a special variable, which can hold more than one value. If we want to store the multiple values in a single variable we can use array. Arrays are represented by **square brackets** ([ ]).

More Formally we can say that, Array is the ordered collection of multiple values which may have same of different data types

How to create army

var array_name = [item1, item2, ...];

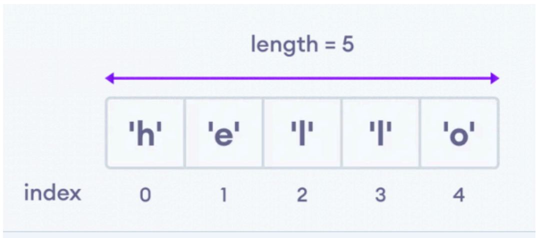
Example



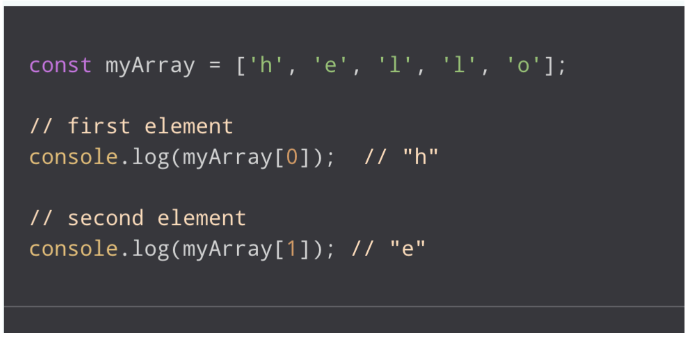
```
// empty array
const myList = [ ];

// array of numbers
const numberArray = [ 2, 4, 6, 8];

// array of strings
const stringArray = [ 'eat', 'work', 'sleep'];

// array with mixed data types
const newData = ['work', 'exercise', 1, true];
```

Indexing

Each element of the array has some index, and this index will uniquely identify that element. When we want to fetch a specific element from array we have to give it's index in square brackets as shown in following example. Index of array always starts from 0.



Array indexing in JavaScript



```
const myArray = ['h', 'e', 'l', 'l', 'o'];

// first element
console.log(myArray[0]);  // "h"

// second element
console.log(myArray[1]); // "e"
```

Array Methods

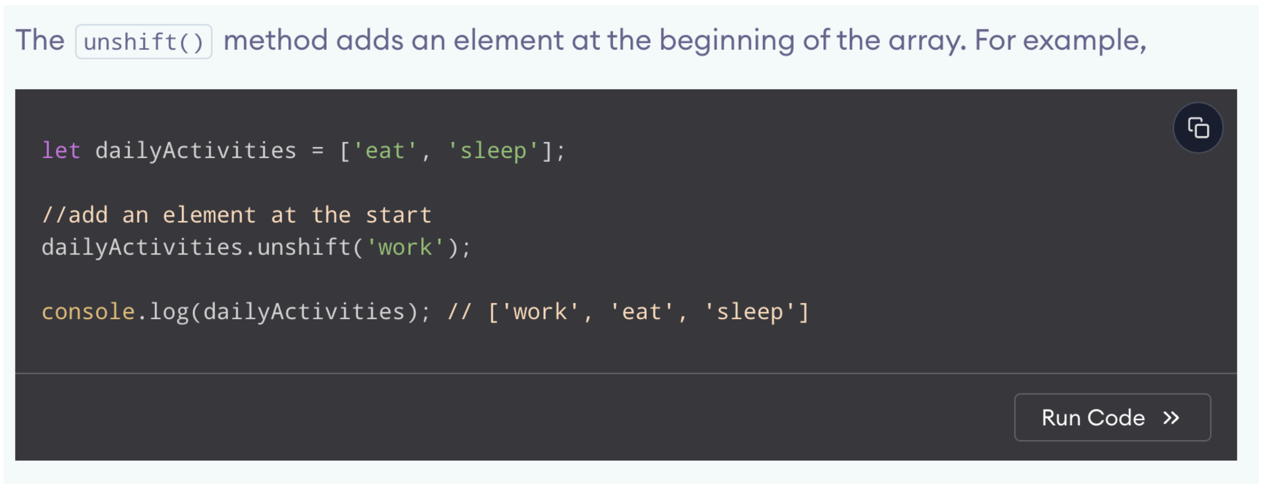
Array being a special datatype have lot of methods and some of them are

1. Add an Element to an Array

You can use the built-in method push() and unshift() to add elements to an array.

The push() method adds an element at the end of the array. For example,

```
let dailyActivities = ['eat', 'sleep'];

// add an element at the end
dailyActivities.push('exercise');

console.log(dailyActivities); //  ['eat', 'sleep', 'exercise']
```



The `unshift()` method adds an element at the beginning of the array. For example,

```
let dailyActivities = ['eat', 'sleep'];

//add an element at the start
dailyActivities.unshift('work');

console.log(dailyActivities); // ['work', 'eat', 'sleep']
```

2. **You can also add elements or change the elements by accessing the index value.**

```
let dailyActivities = [ 'eat', 'sleep'];

// this will add the new element 'exercise' at the 2 index
dailyActivities[2] = 'exercise';

console.log(dailyActivities); // ['eat', 'sleep', 'exercise']
```

3. Remove an Element from an Array

You can use the pop() method to remove the last element from an array.

```
let dailyActivities = ['work', 'eat', 'sleep', 'exercise'];

// remove the last element
dailyActivities.pop();
console.log(dailyActivities); // ['work', 'eat', 'sleep']
```

The pop() method also returns the returned value. For example,

4. Array length

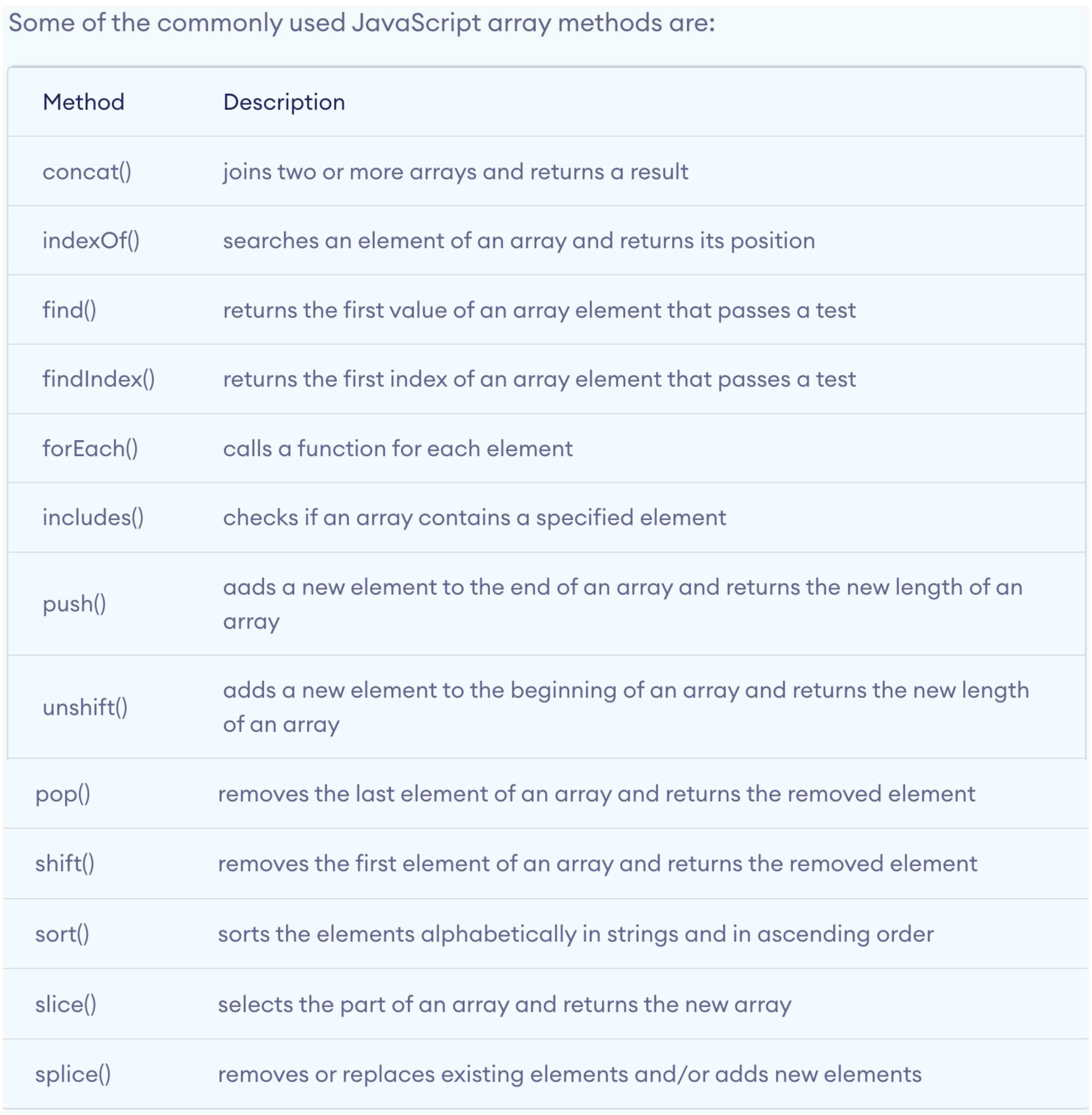
You can find the length of an element (the number of elements in an array) using the length property. For example,

```
const dailyActivities = [ 'eat', 'sleep'];

// this gives the total number of elements in an array
console.log(dailyActivities.length); // 2
```

Most Commonly used array Methods

Go on the following link to study all the array methods in detail with good examples

Detailed Explanation With Code: *https://www.w3schools.com/js/js_array_methods.asp*



Some of the commonly used JavaScript array methods are:

Method	Description
concat()	joins two or more arrays and returns a result
indexOf()	searches an element of an array and returns its position
find()	returns the first value of an array element that passes a test
findIndex()	returns the first index of an array element that passes a test
forEach()	calls a function for each element
includes()	checks if an array contains a specified element
push()	aads a new element to the end of an array and returns the new length of an array
unshift()	adds a new element to the beginning of an array and returns the new length of an array
pop()	removes the last element of an array and returns the removed element
shift()	removes the first element of an array and returns the removed element
sort()	sorts the elements alphabetically in strings and in ascending order
slice()	selects the part of an array and returns the new array
splice()	removes or replaces existing elements and/or adds new elements

2. Strings

JavaScript strings are for storing and manipulating text. Strings are **immutable** in the nature. It means when the string is created we cannot change it. Unlike array all the strings methods will not change the original string , they will return you new string with all the modifications and original one will remain same.

A JavaScript string is zero or more characters written inside quotes.

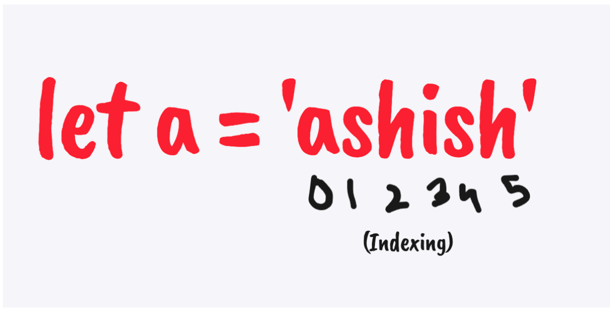
Example

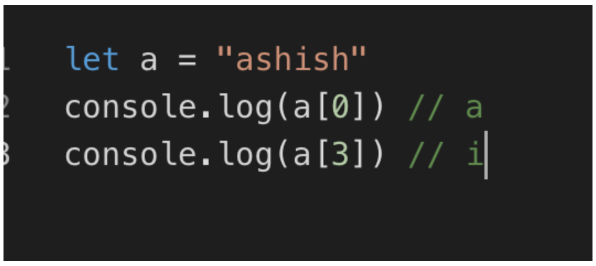
let carName1 = "Volvo XC60"; // Double quotes

let carName2 = 'Volvo XC60'; // Single quotes

Indexing in the string:

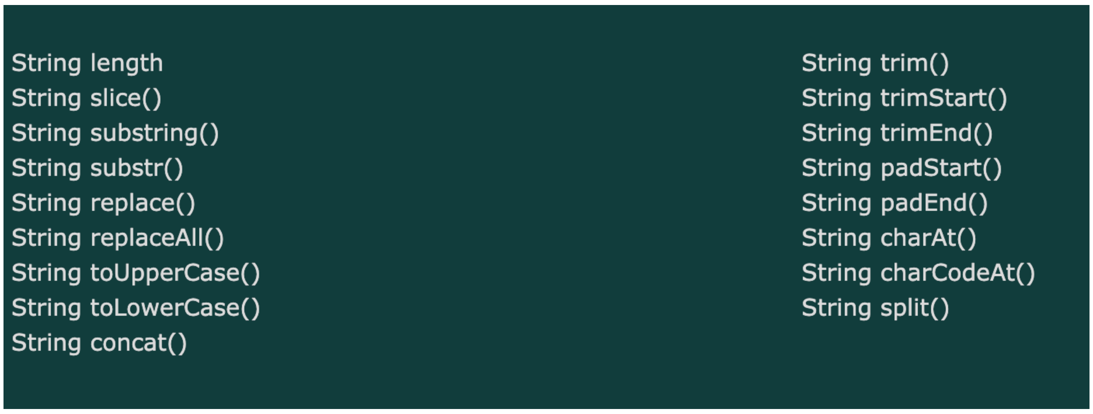
Strings also have indexes. Indexing in the strings are same as of array. Each character of string has some index and index will always start from 0. If you want to access speicif character of string you have to provide its index in square bracket as showing in below example:





```
let a = "ashish"
console.log(a[0]) // a
console.log(a[3]) // i
```

String Methods (Open following link to study all the string methods)
Detailed Explanation With Code: https://www.w3schools.com/js/js_string_methods.asp



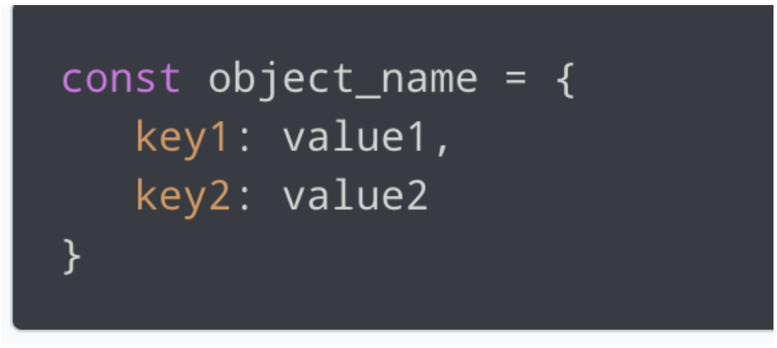
String length
String slice()
String substring()
String substr()
String replace()
String replaceAll()
String toUpperCase()
String toLowerCase()
String concat()
String trim()
String trimStart()
String trimEnd()
String padStart()
String padEnd()
String charAt()
String charCodeAt()
String split()

3. Javascript Objects

Objects are the collection of different values but here, each value will be having a separate key that will uniquely identify that value. Whenever you want to access any value we must know the key. With the help of key wan can access the value belonging to that key. Like array and strings there is no indexing in object.

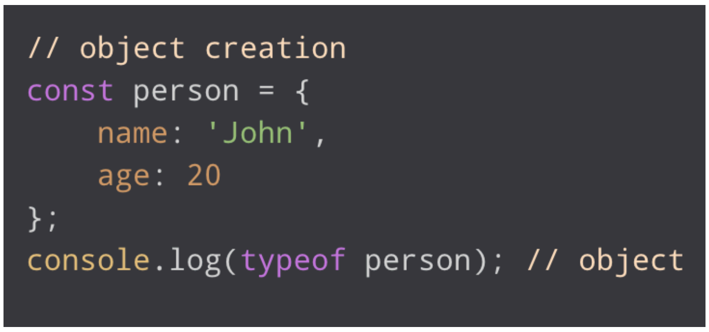
Values in Object can be any datatype. The values can be number, String, Boolean, Array, Function or Object itself.

The syntax to declare an object is we use **curly brackets { }** as shown:

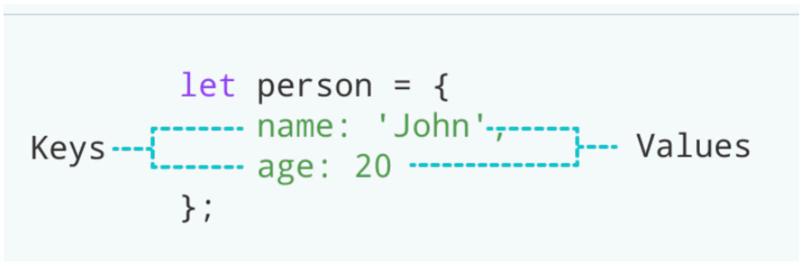


```
const object_name = {
    key1: value1,
    key2: value2
}
```

Example:



```
// object creation
const person = {
    name: 'John',
    age: 20
};
console.log(typeof person); // object
```



```
let person = {
    name: 'John',
    age: 20
};
```

In the above example, name and age are keys, and John and 20 are values respectively.

Accessing Object Properties

You can access the value of a property by using its key. There are 2 ways to access the value of object using it’s key.

1. Using dot Notation

Here's the syntax of the dot notation.

Syntax: *objectName.keyname*

Example:

```javascript
const person = {
    name: 'John',
    age: 20,
};

// accessing property
console.log(person.name); // John
```

2. Using bracket Notation

Here is the syntax of the bracket notation. Key name must be written in quotes.

Syntax: *objectName[“keyname”]*

```javascript
const person = {
    name: 'John',
    age: 20,
};

// accessing property
console.log(person["name"]); // John
```

JavaScript Nested Objects

An object can also contain another object. For example,

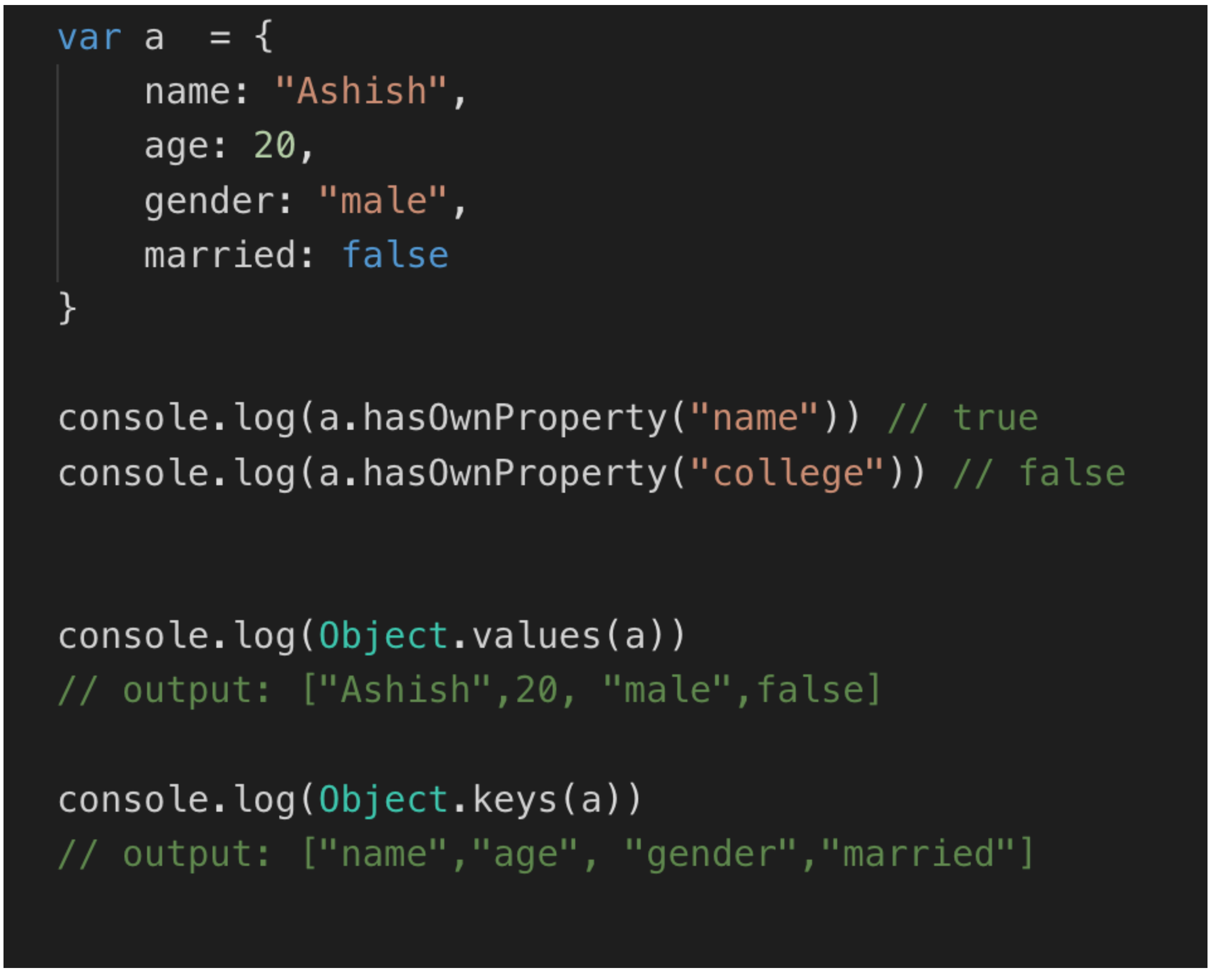
```
// nested object
const student = {
    name: 'John',
    age: 20,
    marks: {
        science: 70,
        math: 75
    }
}

// accessing property of student object
console.log(student.marks); // {science: 70, math: 75}

// accessing property of marks object
console.log(student.marks.science); // 70
```

Objects Methods:

Objects in javascript has some special methods like array and strings the most 2 majority used method of the objects are as follow

1. **hasOwnProperty(key_name):** It will take a key name and return true if the key exists in the object else return false
2. **Object.keys(object_name):** it will take an object and returns an array containing all the keys of the object
3. **Object.values(object_name):** it will take and object and returns an array containing all the values of the object

Example:



```
var a  = {
    name: "Ashish",
    age: 20,
    gender: "male",
    married: false
}

console.log(a.hasOwnProperty("name")) // true
console.log(a.hasOwnProperty("college")) // false


console.log(Object.values(a))
// output: ["Ashish",20, "male",false]

console.log(Object.keys(a))
// output: ["name","age", "gender","married"]
```