50 drops of JavaScript

50 useful, powerful, joyful JS functions

ROBERTO BUTTI

Table of Contents

Welcome to 50 drops of JavaScript **3**

The reason why 4

Requirements 4

Continuous release 4

Where to find this book 4

Thanks to... 5

License 5

System **6**

Info from CPU: os.cpus() 7

Info from Operating System: os.version() 8

Info for the current user: os.userInfo() 9

Get the amount of free memory in bytes (as integer): os.freemem() 10

Access to environment variables: process.env 11

Get the amount of total memory available in bytes (as integer): os.totalmem() 12

The load average of CPU: os.loadavg() 13

Welcome to 50 drops of JavaScript

The reason why

This book collects 50 useful, unknown, underrated JavaScript functions or stuff discovered, used, and learned during JavaScript daily use.

Using JavaScript frameworks/libraries daily (like React, Vue, Angular), sometimes the perception of the power of the language and the basic functionalities provided by the JavaScript core could be lost. I see that usually, I used to look at the framework documentation or look for a package in Npm for the system, array, and string functions instead of using core functionalities provided by the language.

While I wrote this book, I also wrote some scripts to better understand the behavior of the functions. You can find these examples here: https://github.com/roberto-butti/50-drops-of-javascript in the *examples* directory.

Requirements

The code used in this book is tested with **NodeJs version 18 (LTS)**. Node.js is an open-source, cross-platform JavaScript runtime environment, and you can obtain Node.js on the official https://nodejs.org/ website.

Continuous release

I was thinking to print this book, but I think that is not so eco-friendly and a book about development could be improved daily in terms of spellchecking and the content and examples. So, I expect to adopt the same approach in the software with the CI/CD, with a continuous release of the book.

Releases:

- (WIP) release 1.0.0 (WIP not yet released): writing... ;

So, if you have any feedback, or you want to suggest some corrections, feel free to open an issue here: https://github.com/roberto-butti/50-drops-of-javascript/issues

Where to find this book

This book is available for download for free here: https://github.com/roberto-butti/50-drops-of-javascript/

Thanks to...

Thanks to all the Open Source community.

License

System

The chapter will cover functions about "system" functionalities (for example how to retrieve the operating system version).

Info from CPU: `os.cpus()`

The `os` module allows you to retrieve some relevant information about the environment used for running your *Node.js* script or application.

```
import { cpus } from 'os'
const c = cpus()
```

The `os` module provides you with some methods. One of them is the `cpus()` that returns the list of CPUs available in your environment. For each element of the list (each element is one CPU), you have the `model` attribute that has the CPU model name (for example "Apple M1 Pro") and the `speed` attribute, an integer for showing the speed of the CPU. Then you have also `times` attribute for the times (in milliseconds) that the CPU has spent in these modes: `user`, `nice` (valid only for Posix systems, in Windows this value is always 0), `sys`, `idle`, `irq`. So, you can walk through the list of CPUs:

```
import { cpus } from 'os'
cpus().forEach(cpu => {
  console.info(cpu.model + ' speed: ' + cpu.speed)
  console.table(cpu.times)
})
```

Info from Operating System: os.version()

The `os` module has the `version`()` method to return the version of the Operating System in use.

```
import { version } from 'os'
const version = version()
console.log(v)
```

The `version()` method returns a string with the full name of the Operating System version, like this:

```
Darwin Kernel Version 21.6.0: Wed Aug 10 14:28:23 PDT 2022;
root:xnu-8020.141.5~2/RELEASE_ARM64_T6000
```

Info for the current user: os.userInfo()

The `os` module has the `userInfo()` method to return information on the current system user.

```
import { userInfo } from 'os'
const u = userInfo()
```

The `userInfo()` method returns an object with attributes:

- `uid`: the user identifier (integer)
- `gid`: the group identifier (integer)
- `username`: the username (string)
- `homedir`: the user's home directory (string)
- `shell`: the user's shell (string)

Get the amount of free memory in bytes (as integer): os.freemem()

The `os` module has the `freemem()` method to return information on the free memory available in the system. The `freemem()` function returns an integer and it represents the bytes. If you need megabytes or kilobytes you have to convert it.

```
import { freemem } from 'os'
const mem = freemem()
console.log('The amount of free memory is %d bytes', mem)
```

Access to environment variables: process.env

The `process` module has the `env` attribute to return information on the environment variable. The `process.env` attribute contains an object and it represents the list of environment variables. Each attribute is an environment variable.

For retrieving the environment variables object:

```
console.log(process.env)
```

If you want to access a specific environment variable via the name (for example to the `PATH` environment variable):

```
console.log(process.env.PATH)
```

If you have to access dynamically to an environment variable, you can use the square brackets:

```
const envVarName = 'PATH'
if (envVarName in process.env) {
  console.log(process.env[envVarName])
} else {
  console.log('no %s defined', envVarName)
}
```

If you want to walk through all the environment variables you can iterate on the object attributes via `Object.keys()` method:

```
Object.keys(process.env).forEach(function (key, index) {
  console.log(key, index, process.env[key])
})
```

Get the amount of total memory available in bytes (as integer): os.totalmem()

The `os` module has the `totalmem()` method to return information on the total memory available in the system. The `totalmem()` function returns an integer and it represents the bytes. If you need megabytes or kilobytes you have to convert it.

```
import { totalmem } from 'os'
const mem = totalmem()
console.log(
  'Hi, the total memory is %d gigabytes',
  mem / 1024 / 1024 / 1024
)
```

The load average of CPU: `os.loadavg()`

The `os` module has the `loadavg()` method to return the "load" measurement information about the CPU usage of the system. The "load" measurement is calculated by the number of processes that are being executed by the CPU or in a 'waiting' state.

The `loadavg()` function returns an array with 3 float numbers. These three numbers represents the average system load calculated over a given period of 1, 5 and 15 minutes

```
import { loadavg } from 'os'
const la = loadavg()
console.log(
  la[0], // last minute
  la[1], // last 5 minutes
  la[2]  // last 15 minutes
)
// it returns: 1.59619140625 2.42822265625 2.60400390625
```

> *On Windows machine this functionality is not available, the method returns an array with 0 values, like `[0, 0, 0]`*