Node.js Global Objects

Node.js global objects are global in nature and available in all modules. You don't need to include these objects in your application; rather they can be used directly. These objects are modules, functions, strings and object etc. Some of these objects aren't actually in the global scope but in the module scope.

A list of Node.js global objects are given below:

* \_\_dirname
* \_\_filename
* Console
* Process
* Buffer
* setImmediate(callback[, arg][, ...])
* setInterval(callback, delay[, arg][, ...])
* setTimeout(callback, delay[, arg][, ...])
* clearImmediate(immediateObject)
* clearInterval(intervalObject)
* clearTimeout(timeoutObject)

Node.js \_\_dirname

It is a string. It specifies the name of the directory that currently contains the code.

*File: global-example1.js*

console.log(\_\_dirname);

Node.js \_\_filename

It specifies the filename of the code being executed. This is the resolved absolute path of this code file. The value inside a module is the path to that module file.

*File: global-example2.js*

1. console.log(\_\_filename);

## setTimeout(cb, ms)

The **setTimeout(cb, ms)** global function is used to run callback cb after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer.

### **Example**

Create a js file named main.js with the following code −

function printHello() {

console.log( "Hello, World!");

}

// Now call above function after 2 seconds

setTimeout(printHello, 2000);

Now run the main.js to see the result −

$ node main.js

Verify the output is printed after a little delay.

Hello, World!

## clearTimeout(t)

The **clearTimeout(t)** global function is used to stop a timer that was previously created with setTimeout(). Here **t** is the timer returned by the setTimeout() function.

### **Example**

Create a js file named main.js with the following code –

function printHello() {

console.log( "Hello, World!");

}

// Now call above function after 2 seconds

var t = setTimeout(printHello, 2000);

// Now clear the timer

clearTimeout(t);

Now run the main.js to see the result –>$ node main.js

## setInterval(cb, ms)

The **setInterval(cb, ms)** global function is used to run callback cb repeatedly after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer using the function **clearInterval(t)**.

### **Example**

Create a js file named main.js with the following code –

function printHello() {

console.log( "Hello, World!");

}

// Now call above function after 2 seconds

setInterval(printHello, 2000);

Now run the main.js to see the result −

>>>$ node main.js

Other modules are:

|  |  |
| --- | --- |
| 1 | [OS Module](https://www.tutorialspoint.com/nodejs/nodejs_os_module.htm)  Provides basic operating-system related utility functions. |
| 2 | [Path Module](https://www.tutorialspoint.com/nodejs/nodejs_path_module.htm)  Provides utilities for handling and transforming file paths. |
| 3 | [Net Module](https://www.tutorialspoint.com/nodejs/nodejs_net_module.htm)  Provides both servers and clients as streams. Acts as a network wrapper. |
| 4 | [DNS Module](https://www.tutorialspoint.com/nodejs/nodejs_dns_module.htm)  Provides functions to do actual DNS lookup as well as to use underlying operating system name resolution functionalities. |
| 5 | [Domain Module](https://www.tutorialspoint.com/nodejs/nodejs_domain_module.htm)  Provides ways to handle multiple different I/O operations as a single group. |