Node.js Globals

Despite being modeled after one standard, Node.js and browser JavaScript differ when it comes to globals. As you might know, in browser JavaScript we have a `window` object. However, in Node.js, it is absent (obviously we don't deal with a browser window), but developers are provided with new objects/keywords:

- `process`
- `global`
- `module.exports` and `exports`

So, let's take a look at the main differences between Node.js and JavaScript.

global

There is a variable named `global` which is accessible by any Node script or program. It refers to the global object. This object has properties. For example `global.process` or `global.require` or `global.console`.

Any first level property of the `global` object is accessible without the `global` prefix. For example, `global.process` and just `process` will be the same.

The `GLOBAL` alias for `global` can be seen in older project but is deprecated. Use `global` instead of `GLOBAL`.

Main Globals

These are the main properties of the `global` object and are known as globals:

- `process`
- `require()`
- `module` and `module.exports`
- `console` and console.log()`
- `setTimeout()` and `setInterval()`
- `__dirname` and `__filename`

`console.log()` and `setTimeout()` work similarly to the browser methods. We will cover `process`, `require()` and `module.exports` in the following lessons.

`__dirname`, `__filename` and `process.cwd`

`__dirname` is an absolute path to the file in which the global variable is called, whereas `process.cwd` is an absolute path to the process that runs the script. The latter might not be the same as the former if we started the program from a different folder, such as `node ./code/program.js`.

`__filename` is similar to `__dirname` but only with the file name attached to the absolute path of the currently running file/script.