# Episode-03 → Let's Write Code

- Node.js installation →
  - There are multiple ways to install Node.js -
    - The first step is to install via command line - just copy the command and install it.
    - nvm - node Version Manager
    - The other easier way is by pre built installer, it will give us a package file, by double clicking we can install it.
    - Out of 100 there is 1-2 person who faces issues while installation and those issues are very specific to their system.
  - Just google the error and get the solution for that error.
  - If we have installed it already then how would we verify that we have installed correctly or not, here are the steps -
    → go to the terminal
    → write command node -V. This command tells us the version of Node.js.
    - If Node.js is not installed in the system this command will give the error.
    - Whenever we install Node.js, there is one more thing that is automatically installed in the system, that is known as npm.
    → Now do - npm -V → version of npm
  - It's a package manager for Node.js.

Q:> How would we write code ?

A:> The very quick and easiest way to write code, that is known as Node REPL.

- REPL → Read, Evaluate, Print, Loop

Q:> How do we go to Node REPL ?

A:> There is a command, that command is "node", if we write this keyword in terminal, we enter the node REPL.

- So, basically it enters into a programme, it's kind of like C++ programme.

- It's a node given programme we have kind of run the node in REPL mode.

- Now we can run any piece of code, any JS code will work now.

- Node.js is JS Runtime Environment

→ Whenever we write any piece of JS code in REPL, it basically gave that code to V8 engine and V8 engine actually execute that code.

→ It is very similar to the console window of the Browser.

→ Browser also uses V8 Engine behind the scene and execute that code.

→ The REPL doesn't work for production, in daily life we don't use REPL, we create file and then we write code in our file.

→ Now create folder on desktop like "namaste-node".

→ Open this folder on code Editor.

→ Now, create first file "app.js", whatever nam we like we can use for naming the file.

→ We write code whatever we wish to suppose, we want to take this file and execute inside our Node.js runtime environment.

→ First of all, we will go to the terminal inside our code editor (VS code).

→ write command in terminal —
Node file name (app.js) → we will see the output

→ This is how we execute a JS File inside the JS runtime environment.

→ In console we get the access to a global object and that global object is known as window.

→ 'this' also points to window Object.

→ The global object is given to us by the browser, not by V8 engine.

→ In Node.js we have the global Object, it is known as "global".

→ console.log(global).

→ And we see there that the setInterval, setTimeout, setImmediate all these things are given to us by this global object.

→ Suppose if we have to execute a setTimeOut inside Node.js it comes via global object.

→ So, 'global' is not the part of V8 engine, it is one of the superpowers which is given to us by Node.js.

→ The global object gives us access to a lot of cool features → setTimeOut, SetInterval, etc..

→ V8 engine doesn't understand "global", it only understand when Node.js give access to "global" inside our V8 engine.

→ "global" is Node.js thing, not V8 engine thing.

→ If we do console.log(this) → it will not print global object, it print {} an empty object.

→ "this" is not equal to the "global" object which was in case of browsers.

→ In browsers if we write "self", "frames" etc, these will also print global object in console window.

→ So many types of global Object are there.

→ To standardize the global object, in 2020 JS committee they came up with the proposition that there should be a standard global object in all the runtime environment.

There should be single way to represent it.

→ The proposal came in, in that proposal somebody said let's make "global" keyword as the global for everywhere.
The browser should understand the global, the web workers also etc.

→ There were different proposals also, but the committee didn't decide on any of these keywords, because in later version of browsers global started pointing to the window object.
Suppose some websites, people are using this keyword as their variable name, there will be a conflict with the variable, there will be a lot of confusion if make these keyword as the standard.

→ The committee came up with a new keyword that was known as "globalThis".

"globalThis" was referring to global object in all across all JS runtime.

→ Now every JS Engine uses "globalThis", it is supported by all the browsers.

→ In Node.js – console.log(globalThis === global)
– This returns "true"

For more PdF, visit

gitHub → rajeshjha2000