What Is Node?

Node.js is an open-source, cross-platform JavaScript run-time environment i.e. V-8 enginethat executes JavaScript code server-side.

V-8 engine: - Open Source written in C++ that takes JS code and compiles is to machine code.V8 engine written in C++.

Browser has **window** global variable

Node has **global** as global variable

Browser has **document** as DOM element variable

Node has **process** as which process node has

Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Lightweight as single thread

Argv- Argument Vector

process.argv - to read command line argument

**lodash**-module to work with array, string or other things

[**yargs**](https://github.com/yargs/yargs)Yargs helps you build interactive command line tools, by parsing arguments and generating an elegant user interface.

JSON.stringify(JSONobjectname)- convert json object into string

JSON.parse(JSONString)- convert json string into json object

Debugging Node: - node inspect filename (work on v8or more) on CMD

List(no of line)-show no of line code that we want to debug

n- move next line

c- continue till program find debugger in program or run till end of program.

repl-The repl module provides a Read-Eval-Print-Loop (REPL) implementation that is available both as a standalone program or includible in other applications

On Google Browser

Node –inspect-brk filename

Then in chorme go to chorme://inspect

Yargs.command(commandname,commanddescipbe,{

Argumentname:{

Describe:’’,

Demand:’’,

Alias:false (by default) if true then argument is compalrory

}

}).

Help(). (for help)

argv

Arrow Function:- we can not bound this keyword in arrow function.

Also we can not have arguments array in arrow function.

setTimeout(function(){},time) Execute after time given

execution happen in node as follows

1. Call back stack execute line by line
2. Node API – API call in node api from Call back stack
3. Call back queue – wait till call back stack did not clear

JSON.stringify(JSONObject,Undefined,indentsize)

encodeURIComponent(String) :- encode string by adding %20

decodeURIComponent(encoded string) :%20 will replace by space

why promise?

We don’t have to write much if else statement

Resolve or reject only once

Asynchronous programming is a design pattern which ensures the non-blocking code execution.

synchronous code executes without having any dependency and no order. This improves the system efficiency and throughput.

What is callback hell?

**How to avoid callback hell**

To avoid callback hell, follow one or combination of the following :

* Modularise your code.
* Use generators.
* Use promises
* Use event-driven programming.
* Use Async.js

The callback function is a closure and can only be accessed inside the function. However you can create separate function by providing some name and pass that function as callback.

Like this.

|  |  |
| --- | --- |
|  | var fs = require("fs"); |
|  |  |
|  | fs.readFile('async.js','utf8',fileContent); |
|  |  |
|  | function fileContent(err,data) { |
|  | if(!err) { |
|  | console.log(data); |
|  | } |
|  | } |

Request module does not support promise directly

Axios support promise

\_\_dirname :- directory of your current file

View engine in express :- ejs, jade, handlebar, angularjs, pug etc

Nodemon filename –e fileextension,, :- -e extenstion to which we want to watch files

HBS- Handlebars

hbs.registerPartials(\_\_dirname+'/views/partials')

app.set('view engine','hbs');

hbs.registerHelper(helpername,function);

in express module app.use() works as middle ware

app.use work in order

Unit Testing Mocha:- build on behavior driven development

To run test suite continues use nodemon as follows:

Nodemon --exec “npm test” ie. Nodemon execute the command in continues manner

Assertion module except