**Node.js Beginner For ITVMH**

**Tutorial 1:- Introduction to Node js**

**What is Node.js?**

It is a js runtime built on chromes v8 javascript engine.

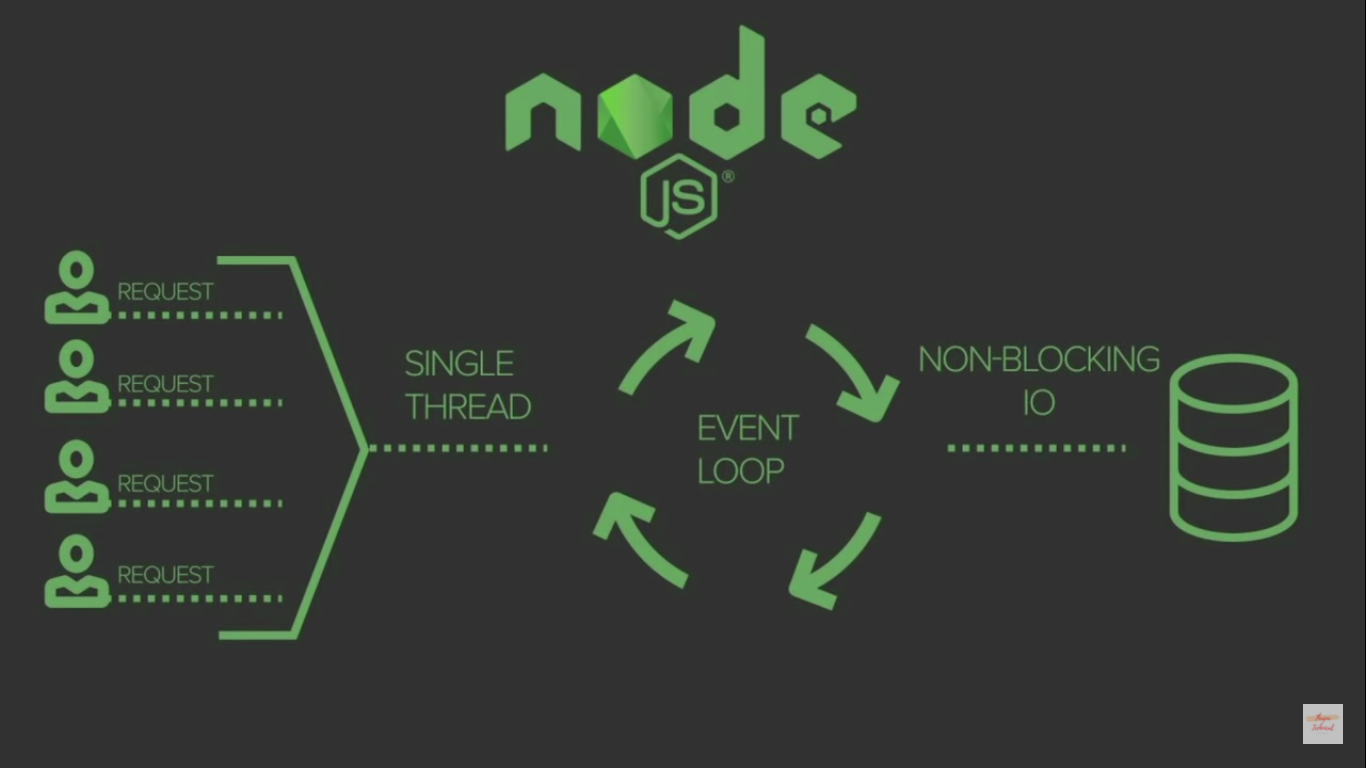
It is a javascript running on the server.

**When to use Node.js?**

i/o bound

data streaming application eg Netflix page note end

real time chat application



Node js only not used when we create cpu instance application

**TUTORIAL 2**

**Installing of nodejs IDE and NPM**

Node Package Manager provides two main functionalities:

* It provides online repositories for node.js packages/modules which are searchable on search.nodejs.org
* It also provides command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

The npm comes bundled with Node.js installables in versions after that v0.6.3. You can check the version by opening Node.js command prompt and typing the following command:

**Tutorial 3- prerequisties**

Nodejs is nothing but the javascript file working on server

1. Basic javascript
2. ES5 & ES6 | ECMA Script 6(2015)
3. Client server model

**Tutorial -4 complete REPL in Node.js**

The REPL features of node is very useful in experimenting with node.js codes and to debug javascript codes.

Read- reads user input parses the input into java script data-structure

Eval- takes and evaluates the data structure

Print- print the result

Loop**-** loops the above command until the user presses

In short it is cmd for windows and similarly node js for nodejs cmd provide by node js.

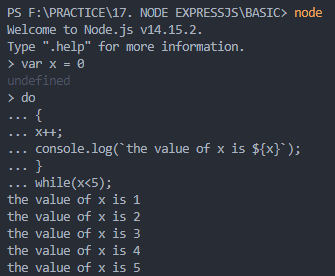
To insert into node js repl you jst need to write node in cmd then cmd is open for node

To create node file you just need to write type nul > index.js

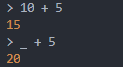
You have got error but your file will be create successfully.

What we can do with repl?

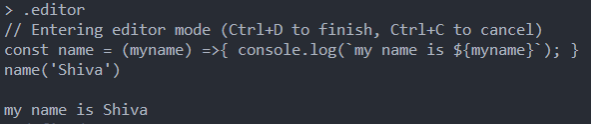
1. Js expression operator like +-\*/
2. Use variables var a =”shiva”
3. Multiline code/loops



1. Use(\_) to get the last result



1. We can use editor mode



And we can see all property of node js using press 2 times tab

**TUT 5 :- NODE JS CORE MODULES**

Consider modules to be the same as javascript library . a set of function you want to include in your application

Nodejs has a set of built in module which you can use without any future installation.

So we are understanding file system in node.js so before we are geeing hands on nodejs core module we first understand sync and async function in javascript

**Synchronous :-**  as the name suggest it means to be in a sequence.i.e every statement of codes gets executed one by one. So basically a statement has to wait for the earlier statement to get executed.

Every line of code waits for its previous one to get executed first and then it gets executed.

**Asynchronous JavaScript:-** it allows the program to be executed Immediately where the synchronous code will be block further execution of the remaining code until it finish the current one. This may not look like a big problem but when you see in it a bigger picture you realize that it may lead to delaying the user interface.

<script>

    document.write("Hi");

    document.write("<br>");

    setTimeout(() => {

        document.write("Let us see what happens");

    }, 2000);

    document.write("<br>");

    document.write("End");

    document.write("<br>");

</script>

At first, as usual, the **Hi** statement got logged in. As we use browsers to run JavaScript, there are the web APIs that handle these things for users. So, what JavaScript does is, it passes the **setTimeout** function in such web API and then we keep on running our code as usual. So it does not block the rest of the code from executing and after all the code its execution, it gets pushed to the call stack and then finally gets executed. This is what happens in asynchronous JavaScript.

**NODE JS FILE MODULE**

* In node.js file I/0 is provide by simple wrapper around standard POSIX function. Node file system (fs) module can be imported using
* Var fs = require(‘fs’;)
* Every method in fs module has synchronous and asynchronous forms.
* Asynchronous methods take a last parameter as completion function callback. Asynchronous method is preferred over synchronous method because it never blocks the program execution where as the synchronous method blocks**.**

**Open A FILE**

SYNTAX:[-

Fs.open(path,flags,[mode],callback)

Path:- path will be here

Flag ;- the behaviour of that file to be opened.

Modes:- the set the file mode but only if the file was created . It default to 0666, readable and writeable

Callback:- this is the callback func which gets two argument(err,fd)

**Flag:-**

r: open file for reading an excep occur if the file dose not exist

r+: open file for reading and writing.

Rs: open file for reading in synchronous mode.

Rs+: open file for reading and writing with synchronously

W: open file for writing. The file is created or truncated.

Wx: like w but fail if path exists.

W+: open file for reading and writing the file is created if it does not exist or truncated(if it exists)

Wx+: like w+ but fail if path exists

A: open file for appending the file is created if file is not exist.

Ax- like a but fails if path exists

A+: open file for reading and appending .

Ax+: open file for reading and appending. the file is created if it does not exist.

**File information method:-**

Fs.stat(path,callback)

1. Stats.isfile()
2. Stats.isdirectory()
3. Stats.isblockdevice()
4. Stats.ischaracterdevice()
5. Stats.issumboliclink()
6. Stats.isfifo()
7. Stats.issocket()

**Read file**

The fs.readFile() method is used to read files on your computer.

Assume we have the following html file.

**<html>  
<body>  
<h1>My Header</h1>  
<p>My paragraph.</p>  
</body>  
</html>**

**Syntax:-**

readFIle(file\_name,callable function)

**create file**

The File System module has methods for creating new files:

* fs.appendFile()
* fs.open()
* fs.writeFile()

The fs.appendFile() method appends specified content to a file. If the file does not exist, the file will be created:

**Update Files**

The File System module has methods for updating files:

* fs.appendFile()
* fs.writeFile()

The fs.appendFile() method appends the specified content at the end of the specified file:

**Delete Files**

To delete a file with the File System module,  use the  fs.unlink() method.

The fs.unlink() method deletes the specified file:

**Rename Files**

To rename a file with the File System module,  use the fs.rename() method.

The fs.rename() method renames the specified file:

**Tutorial 6 challenge 1**

**crud operation using fs module**

**complete**

**Tutorial 7**

**async**

**complete**

**Tutorial 8**

**Sync vs async**

**complete**

**Tutorial 9 challenge 2**

**Crud operation using async**

**complete**

**Tutorial 10**

**Os modules**

|  |  |  |
| --- | --- | --- |
| Index | Method | Description |
| 1. | os.arch() | This method is used to fetch the operating system CPU architecture. |
| 2. | os.cpus() | This method is used to fetch an array of objects containing information about each cpu/core installed: model, speed (in MHz), and times (an object containing the number of milliseconds the cpu/core spent in: user, nice, sys, idle, and irq). |
| 3. | os.endianness() | This method returns the endianness of the cpu. Its possible values are 'BE' for big endian or 'LE' for little endian. |
| 4. | os.freemem() | This methods returns the amount of free system memory in bytes. |
| 5. | os.homedir() | This method returns the home directory of the current user. |
| 6. | os.hostname() | This method is used to returns the hostname of the operating system. |
| 7. | os.loadavg() | This method returns an array containing the 1, 5, and 15 minute load averages. The load average is a time fraction taken by system activity, calculated by the operating system and expressed as a fractional number. |
| 8. | os.networkinterfaces() | This method returns a list of network interfaces. |
| 9. | os.platform() | This method returns the operating system platform of the running computer i.e.'darwin', 'win32','freebsd', 'linux', 'sunos' etc. |
| 10. | os.release() | This method returns the operating system release. |
| 11. | os.tmpdir() | This method returns the operating system's default directory for temporary files. |
| 12. | os.totalmem() | This method returns the total amount of system memory in bytes. |
| 13. | os.type() | This method returns the operating system name. For example 'linux' on linux, 'darwin' on os x and 'windows\_nt' on windows. |
| 14. | os.uptime() | This method returns the system uptime in seconds. |
| 15. | os.userinfo([options]) | This method returns a subset of the password file entry for the current effective user. |

**Tutorial 11**

**Node path modules**

The path modules provide utilities for working with the file and the directory path. It can be accessed using.

const path = require(‘path’)

|  |  |  |
| --- | --- | --- |
| **Index** | **Method** | **Description** |
| 1. | path.normalize(p) | It is used to normalize a string path, taking care of '..' and '.' parts. |
| 2. | path.join([path1][, path2][, ...]) | It is used to join all arguments together and normalize the resulting path. |
| 3. | path.resolve([from ...], to) | It is used to resolve an absolute path. |
| 4. | path.isabsolute(path) | It determines whether path is an absolute path. an absolute path will always resolve to the same location, regardless of the working directory. |
| 5. | path.relative(from, to) | It is used to solve the relative path from "from" to "to". |
| 6. | path.dirname(p) | It return the directory name of a path. It is similar to the unix dirname command |
| 7. | path.basename(p[, ext]) | It returns the last portion of a path. It is similar to the Unix basename command. |
| 8. | path.extname(p) | It returns the extension of the path, from the last '.' to end of string in the last portion of the path. if there is no '.' in the last portion of the path or the first character of it is '.', then it returns an empty string. |
| 9. | path.parse(pathstring) | It returns an object from a path string. |
| 10. | path.format(pathobject) | It returns a path string from an object, the opposite of path.parse above. |

**TUTORIAL 12**

**IMPORT EXPORT OR CREATE OUR OWN MODULES**

<https://www.geeksforgeeks.org/import-and-export-in-node-js/>

**TUTORIAL 13-14**

**NPM nodemon**



To use npm we can conclude npm init

Npm gives you piece of code that beneficial for program to program short code.

After npm init you can have package.json

Package file:- any npm package we are install that dependency will be in package.json. all details of package.

Install chalk

**TUTORIAL 15**

**Wrapper function**

*// this is called wrapper function and this can be private members*

*// behind the scene how node works*

*// this will also called as IIFE (immediately invoked function expression)*

*// all the scope under () will be in private scope.*

(function (*exports*, *require*, *module*, *\_\_filename*, *\_\_dirname*) {

*// we can require here*

    let fs = require("fs");

*//   we only know that we are only writing only this code but the behind the node they can use function wrapper*

  const name = "shiva";

  const id = 23;

  console.log(name, id);

*//   module.exports = { name, id };*

*//   we can use export here*

})();

*// if you want to run this function use just simply used*

**TUTORIAL 16**

**HTTP MODULE CREATING WEBSERVER**

To access web pages of any web application you need a webserver. The web server will handle all the http request for the web application.

Eg. IIS is a web server for ASP.NET web application and apache is a webserver for php or java web application

Node.js provides capability to create your own web server which will handle HTTP request asynchoronously. You can use IIS or apache to run node.js web application but it is recommended to use node.js web server.

**TUTORIAL 17**

**NODEJS ROUTING**

ROUTING MEANS HANDLING HTTP REQUEST AND TO HANDLE REQUEST YOU NEED TO PUT URL MODULE IN IT.

And we can simply use if else for routing.

See file index.js in httpserver folder that you find more stuf.

**TUTORIAL 18**

**JSON IN SHORT**

JSON stands for javascript object notation. JSON Is light weight format for storing and transport data.

JSON is often used when data is sent from server to the web page.

Before json we are using xml.

**TUTORIAL 19**

**CREATING SIMPLE API**

APi is the acronym for application programming interface. which is a software intermediary that allow two application to talk to each other each time you can use an application like Facebook send an instant message or check the weather on your phone you are using a api.

Or

Api is like a services which allows us to request a data.

**TUTORIAL 20**

**EVENTS IN NODES**

Nodejs has a built in module called events.

Where you can create-,fire-and listen for- your own events.

Ex1. Registering for the event to be fired only one time using once.

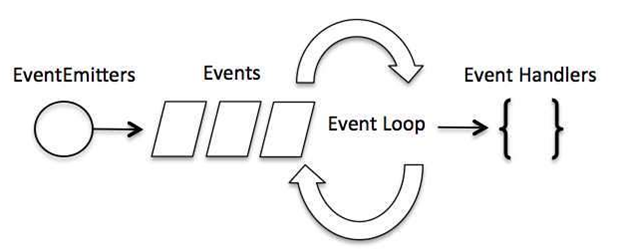
Ex2.create an event emitter instance and register a couple of feedback

Ex3. Registering for the event with callback parameter

Every action on a computer is an event. Like when a connection is made or a file is opened.

Objects in Node.js can fire events, like the readStream object fires events when opening and closing a file:

In Node.js applications, Events and Callbacks concepts are used to provide concurrency. As Node.js applications are single threaded and every API of Node js are asynchronous. So it uses async function to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and after the completion of any task, it fires the corresponding event which signals the event listener function to get executed.



Node.js uses event driven programming. It means as soon as Node starts its server, it simply initiates its variables, declares functions and then simply waits for event to occur. It is the one of the reason why Node.js is pretty fast compared to other similar technologies.

There is a main loop in the event driven application that listens for events, and then triggers a callback function when one of those events is detected.

**TUTORIAL 21**

**STREAM AND BUFFER IN NODE JS**

Streams are object that let you read data from a source or write data to a destination in continuous fashion. In node.js there are four types of streams;

Streaming in means listening music or watching video in real time. Instead of downloading a file to your computer and watching it later.

Readable- stream which is used for read operation.

Writeable- stream which is used for write operation

Duplex- stream which can be used for both read and write

Transform- a type of duplex stream where the output is computed based on input.

Each type of stream is an event emitter instance and throws several events at different instances of times. For ex, some of the commonly used event are

Data – this is fired when there is data available to read

End- this is fired when there is no more data to read

Error- this is fired when there is any error receiving or writing data

Finish- this is fired when there all data has been flushed to underlying system.

**TUTORIAL 22**

**STREAM PIPES**

Stream.pipe() the method used to take a readable stream and connect it to a writable stream.