**Node js:**

Node.js is an open source server environment.

Node.js allows you to run JavaScript on the server.

Example:

var http =require('http');

http.createServer(function (req,res){

        res.writeHead(200,{'Content-Type':'text/plain'});

        res.end('hello wolrd');

}).listen(8080);

**Node Modules:**

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

**Three type of module**

1. Built-in Module
2. Local Module
3. Third party Module

**1.Built-in-Module:**

Node.js has a set of built-in modules which you can use without any further installation.

Look at our Built-in Modules Reference for a complete list of modules.

How to include module:

var http = require('http');

**1.Local Module:**

You can create your own modules, and easily include them in your applications.

For Example:

Create datefile.js

exports.todaysDate = function(){

    return Date();

}

Import in file

var http =require('http');

var dt =require('./datefile.js')

http.createServer(function (req,res){

        res.writeHead(200,{'Content-Type':'text/plain'});

        res.end('hello wolrd'+dt.todaysDate());

}).listen(8080);

**The Built-in HTTP Module**

To make HTTP requests in Node.js, there is a built-in module HTTP in Node.js to transfer data over the HTTP. To use the HTTP server in the node, we need to require the HTTP module. The HTTP module creates an HTTP server that listens to server ports and gives a response back to the client. Use the createServer() method to create an HTTP server: The function passed into the http.createServer() method, will be executed when someone tries to access the computer on port 8080.

Syntax:

const http = require('http');

## Add an HTTP Header

## If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type:

## var http = require('http'); http.createServer(function (req, res) { **res.**writeHead**(**200**, {**'Content-Type'**:**'text/html'**});**   res.write('Hello World!');   res.end(); }).listen(8080);

## Read the Query String

var http= require('http');

http.createServer(function(req,res){

    res.writeHead(200,{'Content-Type':'text/html'});

    res.write(req.url);

    res.end();

}).listen(8080);

## Split the Query String:

var http = require('http');

var url = require('url');

http.createServer(function(req,res){

    res.writeHead(200,{'Content-Type':'text/html'});

    var q=url.parse(req.url,true).query;

    res.end(q.id);

}).listen(8080);

**Node.js File System**

To handle file operations like creating, reading, deleting, etc., Node.js provides an inbuilt module called FS (File System). Node.js gives the functionality of file I/O by providing wrappers around the standard POSIX functions. All file system operations can have synchronous and asynchronous forms depending upon user requirements. To use this File System module, use the require() method:

var fs = require('fs');

Common use for File System module:

* Read Files
* Write Files
* Append Files
* Close Files
* Delete Files

Create a Node.js file that reads the HTML file, and return the content:

**Fs.redaFile()**

var fs= require('fs');

var http= require('http');

http.createServer(function(req,res){

    fs.readFile("hello.html",function(err,data){

        res.writeHead(200, {'Content-Type': 'text/html'});

        res.write(data);

        return res.end();

  });

}).listen(8080);

**fs.appendFile()**

  fs.appendFile('hello.html', '<h2>Hello content!</h2>', function (err) {

    if (err) throw err;

    console.log('Saved!');

  });

**fs.writeFile()**

var fs = require('fs');

fs.writeFile('data.txt',"Tops Technology!!!!!!!!!!!",function(err){

    if(err) throw err;

    console.log("File created");

});

**fs.unlink()**

var fs = require('fs');

fs.unlink('data.txt', function (err) {

  if (err) throw err;

  console.log('File deleted!');

});

**fs.rename()**

var fs = require('fs');

fs.rename('hello.html', 'first.html', function (err) {

  if (err) throw err;

  console.log('File Renamed!');

});

**Nodemon**

Nodemon is a popular tool that is used for the development of applications based on node.js. It simply restarts the node application whenever it observes the changes in the file present in the working directory of your project.

Additionally, nodemon does not need any specific modifications to code or the mode of development. It acts as a facilitator in the node by replacing the wrapper for it. To use nodemon, you will simply need to replace the word node on the CLI while you are about to execute your script.

**Installation:**

Command:

* npm install -g nodemon
* First, make a directory with the below command as shown.
  + mkdir nodemon -exp
* After making a directory, you will need to initialize the package.json using the command.
  + npm init -y
* install express using the below command whether you have yarn or node installed in your system.
  + npm install express --save
* create index.js file
* const app = require('express')();
* app.use('/', (req, res) =>
* {
* res.status(300).send('Hello JavaTpoint!');
* });

Run server

Nodemon server

**How to read command line arguments in Node.js ?**

Command-line arguments (CLI) are strings of text used to pass additional information to a program when an application is running through the command line interface of an operating system. We can easily read these arguments by the global object in node i.e. process object.

var arglist = process.argv;

function squareFind(number){

    return number\*number;

}

var sq= squareFind(arglist[2]);

console.log("square of "+arglist[2]+" = "+sq);

# **Node.js Yargs Module**

Yargs module is used for creating your own command-line commands in node.js and helps in generating an elegant user interface. This module makes command-line arguments flexible and easy to use.

**Installation of yargs module:**

**npm install yargs**

const yargs = require('yargs')

yargs.version('1.1.0');

yargs.command({

    command :'add',

    describe:'Adds two number',

    builder: {

        firstNum:{

            describe:'First Number',

            demandOption:true,

            type:'number'

        },

        secNum:{

            describe:'First Number',

            demandOption:true,

            type:'number'

        }

    },

    handler(argv){

        console.log("addition:",(argv.firstNum+argv.secNum));

    }

});

yargs.argv

**Node.js URL Module**

The URL module splits up a web address into readable parts.

To include the URL module, use the require() method:

var url = require('url');

var link= "http://localhost:8080/hello.html?name=megha&subject=nodejs";

var querystring= url.parse(link,true);

console.log("hostname= "+querystring.host);

console.log("filename = "+querystring.pathname);

console.log("serach for ="+ querystring.search);

var qst= querystring.query;

console.log(qst.name);

console.log(qst.subject);

**Node.js Validator:**

**Intsall : npm install validator**

const validator = require('validator')

var url ="https://careercenter.tops-int.com/";

var mail="megha@gmail.com";

console.log(validator.isURL(url));

console.log(validator.isEmail(mail));