**NODE.JS-** server side runtime environment/platform. Language used in nodejs is JS.

Node.js = Runtime Environment + JavaScript Library

USE “ctrl+c” to get a newline if stuck somewhere in command prompter.

USE “windows+D” to go to desktop.

After installation, the path C:\program files\nodejs is already mentioned in the PATH, in environment variable, in advance system settings.  
If it’s not there, add it manually.  
  
  
>>Open command prompt(admin mode)  
>>node  
>> console.log(‘hello world’)  
  
--------------------------------------------------------------------------------------------------------------------------------------

>> Downlaod webstorm, jet BRAINS  
>> create new project, blank project OR static web   
>> directory ----> C:\Users\rahul\Desktop\Node.js\Bucky  
  
>> Added new file to the project, app.js  
--------------------------------------------------------------------------------------------------------------------------------------

>> go to app.js on top right  
>> Edit config  
>> Browse/live edit  
>> select after launch  
>>just below write… <http://localhost:8888>  
>> apply  
  
--------------------------------------------------------------------------------------------------------------------------------------

In WebStorm

File>>Settings>>Plugin>>NodeJS if plugins are not installed, else you’ll get warnings  
  
--------------------------------------------------------------------------------------------------------------------------------------

For working in sublime text and not WebStorm.  
>> create app.js in C:\Users\rahul\Desktop\Node.js\Bucky  
>>point the cmd to the directory  
>> node app.js  
-------------------------------------------------------------------------------------------------------------------------------------

---->The code in app.js is a server side code, and when you run it, it will keep on running until you explicitly stop it.  
  
---> We can store a function in a variable in JS  
var tuna= function() {………}  
  
To call this function,   
tuna();

---->Read module of Handling Multiple Request which is in handlingMultipleRequest.js in D:/node.js  
----> Read module of Handling Multiple Request part 2 similar to the first one which is in handlingMultipleRequest2.js in D:/node.js

/\*

To run this file in command prompt, point the cmd to D:\node.js

Then >> node handlingMultipleRequest.js

“node” is working here as we have included it's path in advanced system settings

\*/

19 == ’19’ //true  
19=== ’19’ //false  
  
== compare only the value and not the data type  
=== compare value as well as the data type  
  
--------------------------------------------------------------------------------------------------------------------------------------

var Bucky ={

randomFunc : function(){

console.log(“hello”);  
 console.log(this===Bucky); //true

}

}

hello  
True

Modules(or Packages)-

--->It will be tacky if you put all the code in one file, thus grouping of similar code together is done. --->These groups are called modules.   
---> They are like libraries, and are including in the beginning of files which require that module.

//module is exported as an object or more than one objects are returned from the same module

movie.js is imported in app1.js  
>>check it out

TO RUN IT  
>> point cmd to D:/node.js  
>> node app1.js

There are two types of modules  
1. Custom modules- are user defined.(“./” is used)   
2. Core modules- that already come installed with node.js. (“./” is not used)

\*\*note: for sharing objects from a custom modules use “object factory”  
  
--------------------------------------------------------------------------------------------------------------------------------------

File system core module---> fs

//CODE BELOW IS WRITTEN IN app.js  
var fs = require(‘fs’);  
fs.writeFileSync(“filename.txt”, “filename.txt will be created in the same directory of app.js and it will have this data as initial content in it”);

Path module ---> path

var path = require(‘path’);  
var webAddr=”Desktop/bucky//index.html”

**console.log(path.normalize(webAddr));**  
  
 // one of the property of path module is it finds error in the path and corrects it. There might be   
 a error in which instead of one “/” there are two, OR instead of “\”, “/” is used.

**console.log(path.dirname(webAddr));**

//it specifies the directory excluding the index.html file

--------------------------------------------------------------------------------------------------------------------------------------

setTimeout( function() {…….} , 5000 );  
//Runs only once after 5 seconds.

setInterval( function () {…….} , 5000);  
//runs after every 5 seconds.

console.log(\_\_dirname) will give you the directory without the file app.js  
console.log(\_\_filename) will give you the directory with the file app.js

CREATING A BASIC SERVER  
>> check D:/node.js/server.js

SIMPLE WEB FILE SERVER  
>> check D:/node.js/server2.js  
(to run it in webStorm, change the Edit Configuration of server2.js)  
(to run it in cmd, point cmd to D:/node.js and then node server2.js )  
>> index2.html is also part of it.

CONNECT- server framework  
<http://stackoverflow.com/questions/5284340/what-is-node-js-connect-express-and-middleware>

>> point cmd to C:\Users\rahul\Desktop\Node.js\Bucky (AND D:\node.js) i.e where the project files are stored  
>> npm install connect

WEB SERVER USING CONNECT  
>> D:\node.js\server3  
>> D:\node.js\server4   
--------------------------------------------------------------------------------------------------------------------------------------  
EXPRESS- web/application framework(some functionalities are already written).  
<http://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm>

Entire structure is already exists for us to modify them according to our needs.  
  
SETUP  
>>file-->new project  
>>node.js express app  
>>LOCATION- C:\Users\rahul\Desktop\Node.js\megan  
>> TEMPLATE- EJS  
  
\*\*note- megan project files are copied for reference and is copied to the hard disk to understand the code. To run the project follow the above steps again in webStorm or someother IDE.  
  
localhost:3000 for this app

>>bin/www>> run as server  
--->www is the startup script, it kicks app.js which has all the main stuff, it is the core foundation

--->node\_modules folder comprises all the custom modules. Core modules are present in external library folder.  
--->public folder contains file which can be accessed by the user  
--->each page in the application has its own route which is configured in the route folder  
--->view folder contains ejs files which a user view, code is written in html kind of format





**Understanding Default app.js**

1.  
 app.use(‘/’, routes ) ----> var routes =require(‘./routes/index’) ---->   
  
Now index.js will have different path routes(and it exports it in the last line, that is why require is used to access it in the statement “var routes=require…”)  
  
Example  
router.get(‘/’, function(req,res,next){…res.render(‘index’…}  
or  
router.get(‘/mongoTrial’, function(req,res,next) {…res.render(‘mongoTrial’…}  
  
whenever localhost:3000/ is called,   
routes is called,   
which directs to index.js,  
where index.ejs view is rendered.   
  
whenever localhost:3000/mongoTrial is called,   
routes is called,   
which directs to index.js,  
where mongoTrial.ejs view is rendered.

2.  
app.use(‘/about’, about ) ----> var about =require(‘./routes/about’) ---->   
  
Now about.js will have different path routes  
example  
router.get(‘/’, function(req,res,next){…res.render(‘about’…}  
or  
router.get(‘/random’, function(req,res,next) {…res.render(‘random’…}  
  
whenever localhost:3000/about is called,   
 is about is called,   
which directs to about.js,  
where about.ejs view is rendered.   
  
whenever localhost:3000/about/random is called,   
about is called,   
which directs to about.js,  
where random.ejs view is rendered.

3.  
var app=express(); is similar to   
var app=connect();

Express does to Connect what Connect does to the http module.  
So all of the functionality of Connect is there, plus view rendering and a handy DSL for describing routes.

4.  
// catch 404 and forward to error handler

app.use(function(req, res, next) {  
var err = new Error('Not Found');  
err.status = 404;  
next(err);  
});

// error handlers

This is used to catch errors and forward it to other more specific error handler which can be made by us in the future in place of “//error handlers”

5.  
//development error handler  
gives a detailed information about the error so that it can be handled by the developer.

//production error handler  
doesn’t give information to the user as it gives bad aesthetic value.

**EJS- embedded JS**  
It is html with embedded JS code in it. It is somewhat similar to JSP.  
  
<%= %> to print variable  
  
<% %> to write JS code  
   
  
>> res.render(‘index’, {title: “Express”})  
 variable title is defined equal to “Express” in index.js  
  
   
EXAMPLE of <% %>  
  
<ul>  
 <% for(var i=0;i<5;i++) { %>  
 <li>Tuna</li>  
 <% } %>  
</ul>

\*\*note- you can also completely ignore JS also in EJS file as EJS=HTML+JS

-------------------------------------------------------------------------------------------------------------------------------------

**ADDING PAGES**   
  
>>rightClick(view)--->new--->directory--->templates  
  
>>rightClick(templates)--->new--->file--->header.ejs  
  
>>rightClick(view)--->new--->file--->about.ejs  
  
>>rightClick(route)--->new--->file--->about.js  
  
  
CHECK the project megan.  
  
--------------------------------------------------------------------------------------------------------------------------------------

**USING JSON DATA**>>rightClick(megan)--->new--->file--->videodata.json  
  
>> copy content from   
<https://github.com/buckyroberts/Source-Code-from-Tutorials/blob/master/Node.js/021_NodeJs/videodata.json>to videodata.json

THERE ARE 2 APPROACHES

1. Creating global variable in app.js using  
   app.locals.points= ”8900”;  
     
   And it can be accessed anywhere in .ejs using  
   <p><%= points %></p>

SIMILARLY  
app.locals.videodata=require(‘./videodata.json’);  
  
ACCESS in index.ejs  
<body>

<% include template/header.ejs %>

<h1><%= videodata.categoryName %></h1>

<% videodata.categories.forEach(function(item){%>

<li><%= item.categoryName%></li>

<%});%>

</body>

1. PASSING JSON data through routes  
   (better technique than the earlier approach, app.locals.videodata was present in app.js, which is a central managing file and keeping something which specific to some .ejs file is not a very efficient way.)

---> in index.js create new variable  
var vd=require(‘../videodata.json’) ;  
  
---> adding object in index.js  
videodata: vd  
  
\*\*note- index.ejs is the same in both  
  
--------------------------------------------------------------------------------------------------------------------------  
  
**NPM (node package manager)-** to install packages/modules  
  
To install a module  
>> npm install <mdule\_name>  
  
  
**package.json** file exists for every module/package and the application in totality. It defines the properties of them.  
Example- for express, package.json will be in node\_modules/express/package.json  
---------------------------------------------------------------------------------------------------------------------------  
  
Few npm commands  
  
>>npm uninstall <module\_name>  
>>npm ls  
>>npm update <module\_name>

For creating a module/package  
>>npm init  
(creates the package.json, necessary details must be entered which will be reflected in package.json)

CALLBACKS ---> non-blocking programs --->non-sequential-->asynchronous  
<http://www.tutorialspoint.com/nodejs/nodejs_callbacks_concept.htm>

Form action in server5.js and index3.html

>>open command prompt   
>>cd D:/node.js  
>> node server5.js  
>> localhost:8081/index3.html

REST API  
<http://www.tutorialspoint.com/nodejs/nodejs_restful_api.htm>  
  
CHECK serverRest.js AND users.json  
serverRest.js has server side code and users.json has json data.  
(later similar kind of API can be generated using mongoDB rather than a users.json).



Check serverTrial.js for mongodb data in command prompt

MONK  
  
**1. Installing monk to megan project**  
>>package.json of your project  
(project location is C:\Users\rahul\Desktop\Node.js\megan)  
>> ADD “monk”:”\*” in dependencies  
>> cd to your project location  
>>npm install  
 OR  
  
>> cd to your project location  
>>npm install monk

**2.Installing monk to trial project**  
>>cd D:/node.js  
>>npm install monk

To read from mongodb database and display in browser  
Read app.js, index.js and mongoRead.ejs  
--------------------------------------------------------------------------------------------------------------------------------------

To write in mongodb database from the browser browser  
Read app.js, index.js and mongoInsert.ejs

REST API on mongoDB

app.js and api.js for viewing/listing data