Getting “MEAN”

- A Practical Workshop

Node JS

What is Node JS :

- Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine).

- Node.js was developed by Ryan Dahl in 2009.

- Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications.

- Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

- Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

```
Node.js = Runtime Environment + JavaScript Library
```

Features of Node JS :

i. Asynchronous and Event Driven : All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

ii. Super Fast : Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

iii. Single Threaded but Highly Scalable Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

iv. No Buffering : Node.js applications never buffer any data. These applications simply output the data in chunks.

v. License : Node.js is released under the MIT license.

Who are you using Node.JS ?

To check who are using Node JS, Please go through this link :

(https://github.com/nodejs/node/wiki/Projects,-Applications,-and-Companies-Using-Node)

Some of the well-known companies are as follows : eBay, General Electric, GoDaddy, Microsoft, PayPal, Uber, Wikipins, Yahoo!, and Yammer, and the list go on...

Where to Use Node.js ?

Following are the areas where Node.js is proving itself as a perfect technology partner.

- JSON APIs based Applications
- Single Page Applications
- I/O bound Applications
- Data Streaming Applications

Where NOT to use Node.js ?

It is not advisable to use Node.js for CPU intensive applications.

Node JS Installation

Download :
Download the latest version of Node.js installable archive file from Node.js Downloads. (**https://nodejs.org/en/download/**)

At the time of writing this tutorial, following are the versions available on different OS.

OS	Archive Name
Windows	node-v4.5.0-x64.msi
Linux	node-v4.5.0-linux-x64.tar.xz
MAC OS X	node-v4.5.0-.pkg

Installation on UNIX / Linux / Mac OS X :

Step 1 :

Based on your OS architecture, download and extract the archive file into /temp, and then move the extracted files into /usr/local/nodejs directory.

For example:

```
$ cd /temp

$ wget http://nodejs.org/dist/v4.5.0/node-v4.5.0-linux-x64.tar.gz

$ tar xvfz node-v4.5.0-linux-x64.tar.gz

$ mkdir -p /usr/local/nodejs

$ mv node-v4.5.0-linux-x64/* /usr/local/nodejs
```

Step 2 :

Add /usr/local/nodejs/bin to the PATH environment variable.

Example :

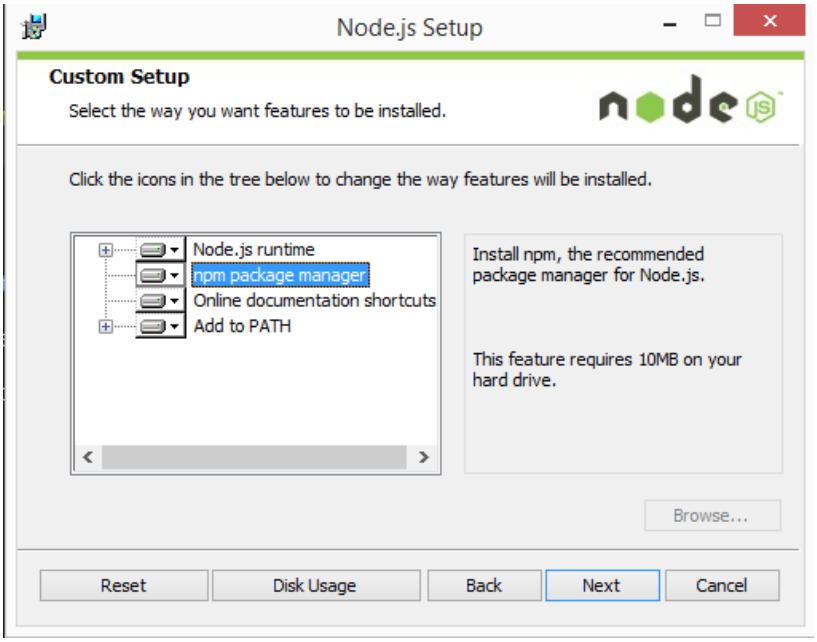
```
$ export PATH=$PATH:/usr/local/nodejs/bin
```

Installation on Windows :

Step 1 :
Download the Windows installer from the Nodes.js web site. (https://nodejs.org/en/)

Step 2 :
Run the installer (the .msi file you downloaded in the previous step.)

Step 3 :
Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings)



Step 4 :

Restart your computer. You won’t be able to run Node.js® until you restart your computer.

Verify Installation of Node JS :

Download the example1.js file from provided github link (https://github.com/virtualSharif/gettingMEAN/blob/master/node/examples/example1.js)

File Contains below code :

```
console.log("Hello, Node.js!");
```

Now open the Command Prompt and execute the example1.js using Node.js interpreter to see the result : i.e.

```
node example1.js
```

If everything is fine with your installation, it should produce the following result:

```
Hello, Node.js!
```

First Application

First Application in Node.JS

Before creating an actual "Hello, World!" application using Node.js, let us see the components of a Node.js application. A Node.js application consists of the following three important components:

1. Import required modules: We use the require directive to load Node.js modules.

2. Create server: A server which will listen to client's requests similar to Apache HTTP Server.

3. Read request and return response: The server created in an earlier step will read the HTTP request made by the client which can be a browser and return the response.

Creating first Application :

Step 1: Require module.

We use the require directive to load the http module and store the returned HTTP instance into an http variable as follows:

```
var http = require("http");
```

Step 2 : Create server

We use the created http instance and call http.createServer() method to create a server instance and then we bind it at port 1991 using the listen method associated with the server instance. Pass it a function with parameters request and response. Write the sample implementation to always return "Hello, Node.js!".

```
http.createServer(function (request, response){

        //create response head
        response.writeHead(200, {'Content-type': 'text/plain'});

        //send the response Body
        response.end('Hello Node.js ! \n');
})
//listening to the port
.listen(1991);
```

Step 3 : Testing request and response.

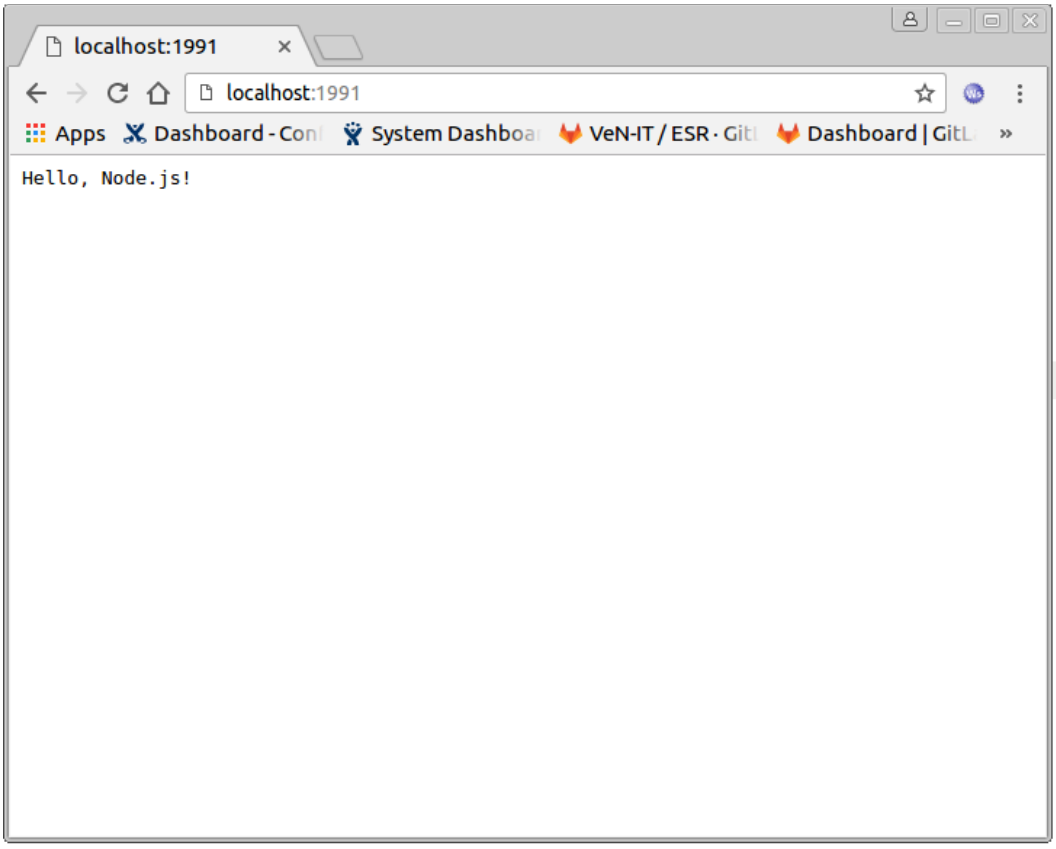
Combine step 1 and step 2, code. Or You can download the source code from here
(https://github.com/virtualSharif/gettingMEAN/blob/master/node/examples/example2.js)

Verify the output :

```
$ node example2.js
Server running at http://127.0.0.1:1991
```

Make a request to Node.js Server using any browser :

Open http://localhost:1991/ or http://127.0.0.1:1991, and observe the below result.



Congratulations! you have your first HTTP server up and running which is responding to all the HTTP requests at port 1991

NPM

NPM :

Node Package Manager (NPM) provides two main functionalities:

1. Online repositories for node.js packages/modules which are searchable on search.nodejs.org

2. Command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

Note : Now, We don't have to separate install npm. NPM comes bundled with Node.js.

If you are running an old version of NPM, then it is quite easy to update it to the latest version.

Just use the following command from root:

```
$ sudo npm install npm -g

/usr/bin/npm -> /usr/lib/node_modules/npm/bin/npm-cli.js
/usr/lib
`-- npm@3.10.6
  +-- glob@7.0.5
  | `-- minimatch@3.0.2
  +-- npm-user-validate@0.1.5
  `-- rimraf@2.5.3

$ npm –version
3.10.6
```

Installing Modules through NPM :

There is a simple syntax to install any Node.js module:

```
$ npm install <module name>
```

Example : following is the command to install a famous Node.js web framework module called express:

```
$ npm install express
```

Now you can use this module in your js file as following:

```
var express = require('express');
```

Global v/s Local Installation :

- By default, NPM installs any dependency in the **local mode**.
- Here local mode refers to the package installation in **node_modules** directory lying in the folder where Node application is present.
- Locally deployed packages are accessible via **require()** method.

Example :
When we installed express module, it created node_modules directory in the current directory where it installed the express module.

Try this command, and check the current directory for the change:

```
$npm install express
```

After completing the installation of express, In the current directory, node_modules directory will be created and under that you can find the express directory (interconnected directories).

Note: you can also use npm ls command to list the local node_ modules.
Verify the Output :

```
sharif@earth:~/mean/node/example$ npm ls
/home/sharif/mean/node/example
`-- express@4.14.0
  +-- accepts@1.3.3
  | +-- mime-types@2.1.11
  | | `-- mime-db@1.23.0
  | `-- negotiator@0.6.1
  +-- array-flatten@1.1.1
  +-- content-disposition@0.5.1
 ...(the list goes ahead)
```

- **Globally installed** packages/dependencies are stored in system directory.
- Such dependencies can be used in CLI (Command Line Interface) function of any node.js but cannot be imported using require() in Node application directly.

- Now let's try installing the express module using global installation.

```
$ npm install -g express
```

This will produce similar result but the modules will be installed globally.

Here the first lines shows the module version and the location where it is getting installed.

```
express@4.12.2 /usr/lib/node_modules/express
├── merge-descriptors@1.0.0
├── utils-merge@1.0.0
├── cookie-signature@1.0.6
├── methods@1.1.1
├── fresh@0.2.4
├── cookie@0.1.2
├── escape-html@1.0.1
├── range-parser@1.0.2
├── content-type@1.0.1
├──
.....(the list goes on)
```

Note : You can use npm ls -g command to check all the modules installed globally in your system.

Using package.json

- package.json is present in the root directory of any Node application/module and is used to define the properties of a package.

- Once we proceed with the workshop you will get to know the more about the package.json.

Attributes of package.json :

name - name of the package
version - version of the package
description - description of the package
homepage - homepage of the package

author - author of the package
contributors - name of the contributors to the package
dependencies - list of dependencies. NPM automatically installs all the dependencies mentioned here in the node_module folder of the package.
repository - repository type and URL of the package
main - entry point of the package
keywords – keywords

For example : You can check the package.json under express directory of node_modules.

Uninstalling a Module:

Use the following command to uninstall a Node.js module.

```
$npm uninstall express
```

There are many npm commands for search , update and so on. You can also create your own module using npm commands and publish it server for public use. (for more details, please go throught the https://www.npmjs.com/)

Callback Concept

Callback :

- Callback is an asynchronous equivalent for a function.

- A callback function is called at the completion of a given task.

- Node makes heavy use of callbacks. All the APIs of Node are written in such a way that they support callbacks.

Example :
A function to read a file may start reading a file and return the control to the execution environment immediately so that the next instruction can be executed.
Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as a parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process a high number of requests without waiting for any function to return results.

Blocking Code Example :
You can download the source code from here (https://github.com/virtualSharif/gettingMEAN/blob/master/node/examples/example3.js) OR Create a text file named **input.txt** with the following content:

```
This is first MEAN workshop.
Any fool can know , the point is to understand!!!!!
```

Create a js file named **example3.js** with the following code:

```
var fs = require("fs");
var data = fs.readFileSync('input.txt');
console.log(data.toString());
console.log("Program Ended");
```

Now run the **example3.js** to see the result:

```
$ node example3.js
```

Verify the Output :

```
This is first MEAN workshop.
Any fool can know , the point is to understand!!!!!
Program Ended
```

Non-Blocking Code Example :
You can download the source code from here (https://github.com/virtualSharif/gettingMEAN/blob/master/node/examples/example4.js) OR

Create a text file named **input.txt** with the following content:

```
This is first MEAN workshop.
Any fool can know , the point is to understand!!!!!
```

Create a js file named **example4.js** with the following code:

```
var fs = require("fs");

fs.readFile('input.txt', function (err, data) {
	if (err) return console.error(err);
	console.log(data.toString());
});
console.log("Program Ended");
```

Now run the **example4.js** to see the result:

```
$ node example4.js
```

Verify the Output :

```
Program Ended
This is first MEAN workshop.
Any fool can know , the point is to understand!!!!!
```

The above two examples explain the concept of blocking and non-blocking calls.
- The first example shows that the **program blocks until it reads the file** and then only it proceeds to end the program.
- The second example shows that the **program does not wait for file reading** and proceeds to print "Program Ended" and at the same time, the program without blocking continues reading the file.

Thus, a blocking program executes very much in sequence. From the programming point of view, it is easier to implement the logic but non-blocking programs do not execute in sequence.

In case a program needs to use any data to be processed, it should be kept within the same block to make it sequential execution.

Web Module

Web Module

- A Web Server is a software application which handles HTTP requests sent by the HTTP client, like web browsers, and returns web pages in response to the clients.

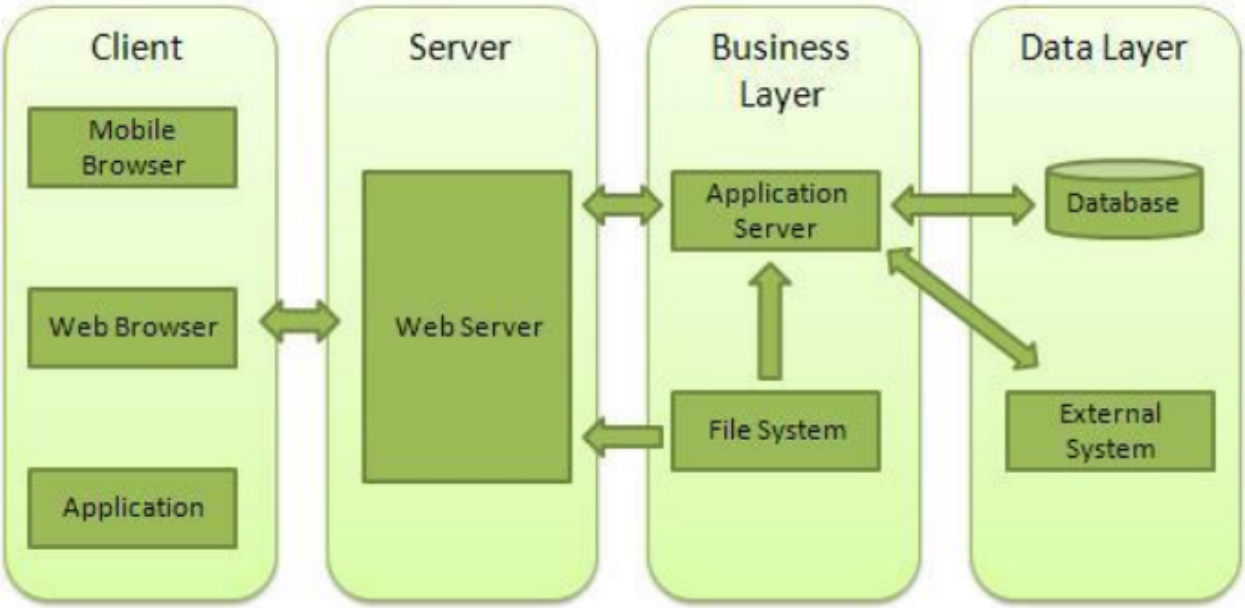
- Web servers usually deliver html documents along with images, style sheets, and scripts.

- Most of the web servers support server-side scripts, using scripting languages or redirecting the task to an application server which retrieves data from a database and performs complex logic and then sends a result to the HTTP client through the Web server.

- Apache web server is one of the most commonly used web servers. It is an open source project.

Web Application Architecture :

A Web application is usually divided into four layers:



Client - This layer consists of web browsers, mobile browsers or applications which can make HTTP requests to the web server.

Server - This layer has the Web server which can intercept the requests made by the clients and pass them the response.

Business - This layer contains the application server which is utilized by the web server to do the required processing. This layer interacts with the data layer via the database or some external programs.

Data - This layer contains the databases or any other source of data.

Creating a Web Server using Node :

Node.js provides an http module which can be used to create an HTTP server.
Following is the bare minimum structure of the HTTP server which listens at 1991 port.

You can download the source code from here (https://github.com/virtualSharif/gettingMEAN/blob/master/node/examples/example5.js) OR Create file name as example5.js.

File example5.js :

```
var http = require('http');
var fs = require('fs');
var url = require('url');

//Create the Server
http.createServer(function (request, response){

        //Parse the request containing filename
        var pathname = url.parse(request.url).pathname;

        // Print the name of the file for which request is made.
        console.log("request for "+ pathname + " received");

        // Read the requested file content from file system
        fs.readFile(pathname.substr(1), function (err, data) {

                if(err){
                        console.log(err);

                        // HTTP Status: 404 : NOT FOUND
                        response.writeHead(404, {'Content-Type': 'text/html'});

                } else {
                        //HTTP status : 200 : OK
                        response.writeHead(200, {'Content-Type': 'text/html'});
                        response.write(data.toString());
                }

                //send response body
                response.end();
        });


}).listen(1991);

console.log("Server is running at http://localhost:1991");
```

File index.html :

```
<html>
        <head>
                <title>First HTML Page</title>
        </head>
        <body>
                Hello, Node.js user!
        </body>
</html>
```

Now, you can run the example5.js to see the result :

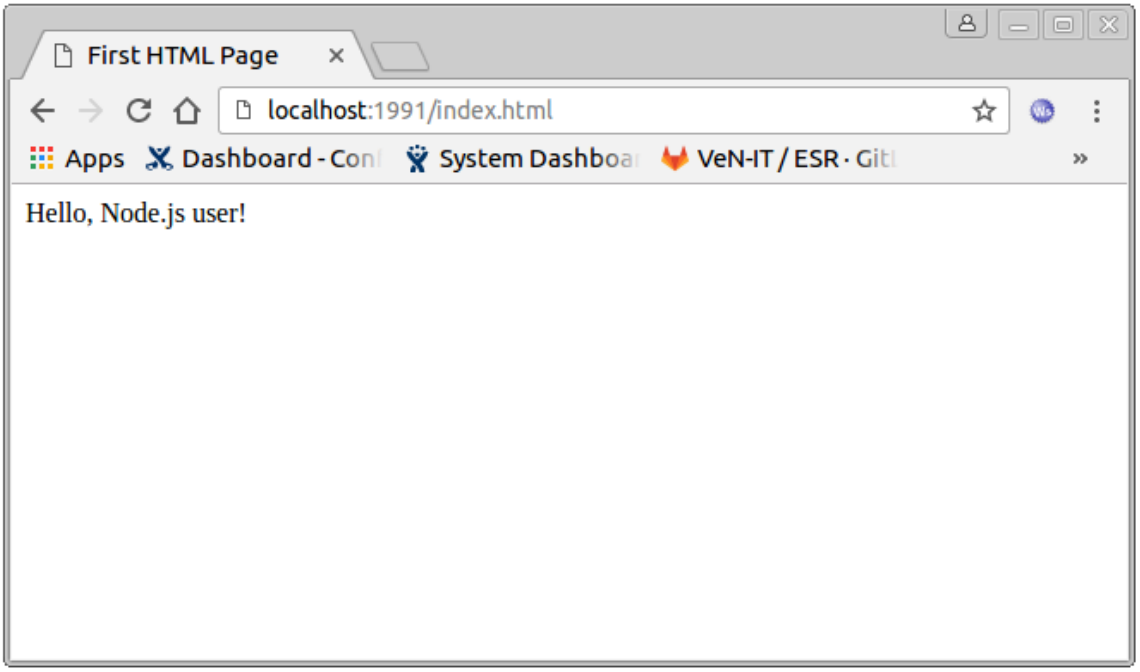
$ node example5.js

Verify the output :

Server is running at http://localhost:1991

Make a request to server by using any browser :

open http://localhost:1991/index.html in any browser to see the following result.



Verify the Output at the server end:
Server is running at http://localhost:1991
request for /index.html received