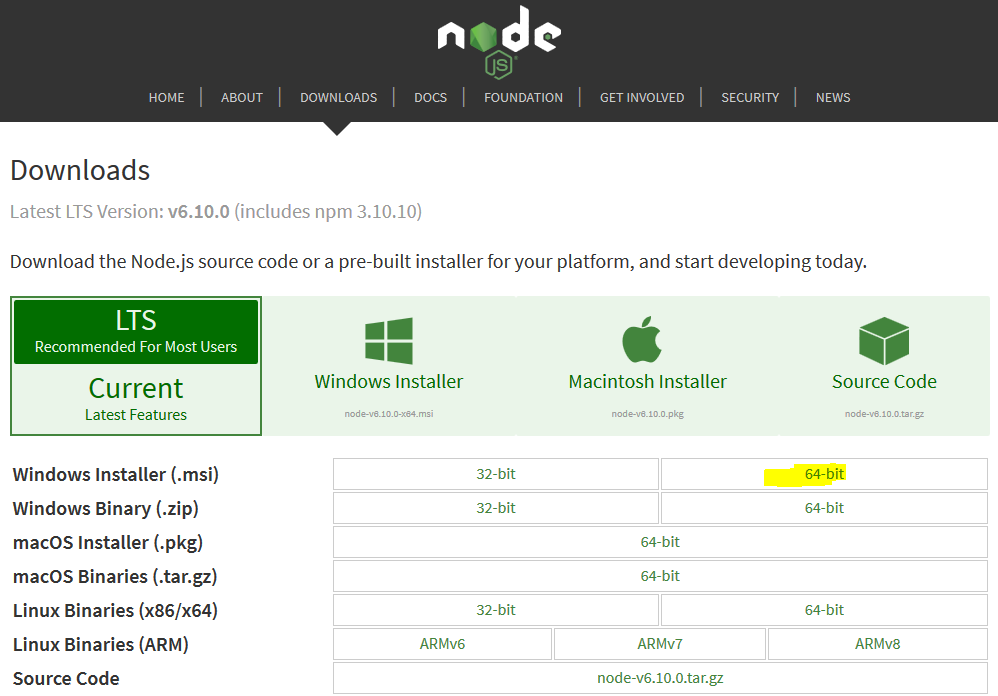
Download and Install Node JS from the following link:

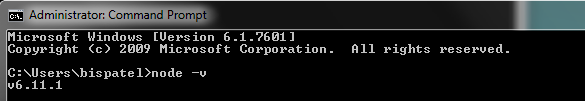
<https://nodejs.org/en/download/>





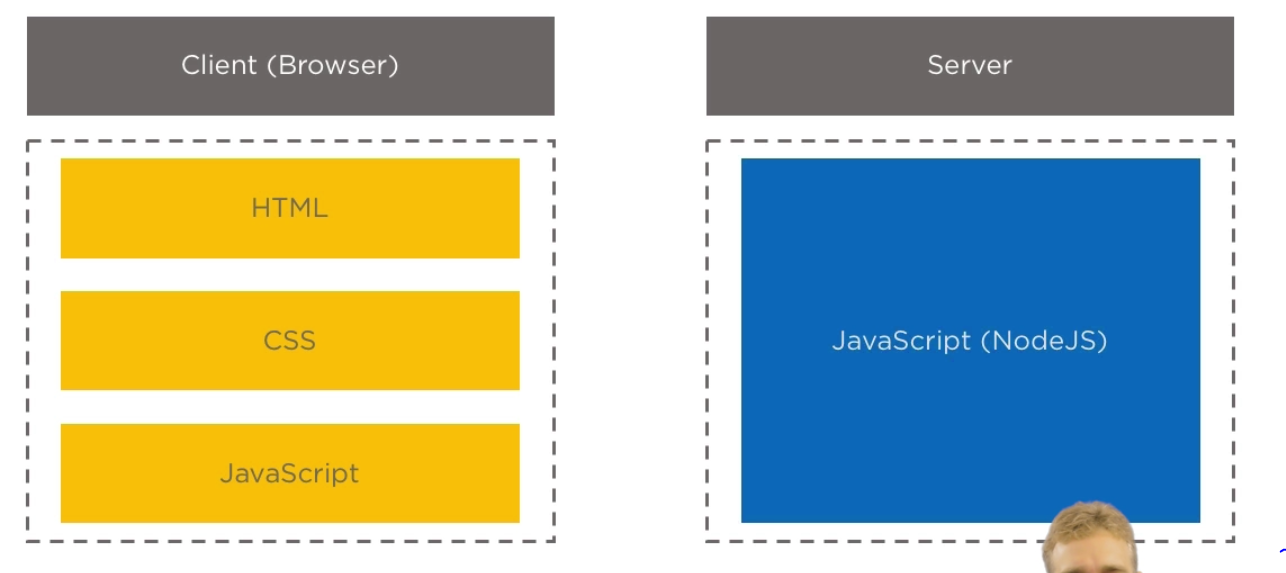
Node JS is a JavaScript runtime built on Chrome’s V8 JavaScript engine. Node.js uses an event driven, non-blocking model that makes it lightweight and efficient. Node.js’ package ecosystem, npm, is the largest ecosystem of open source libraries.

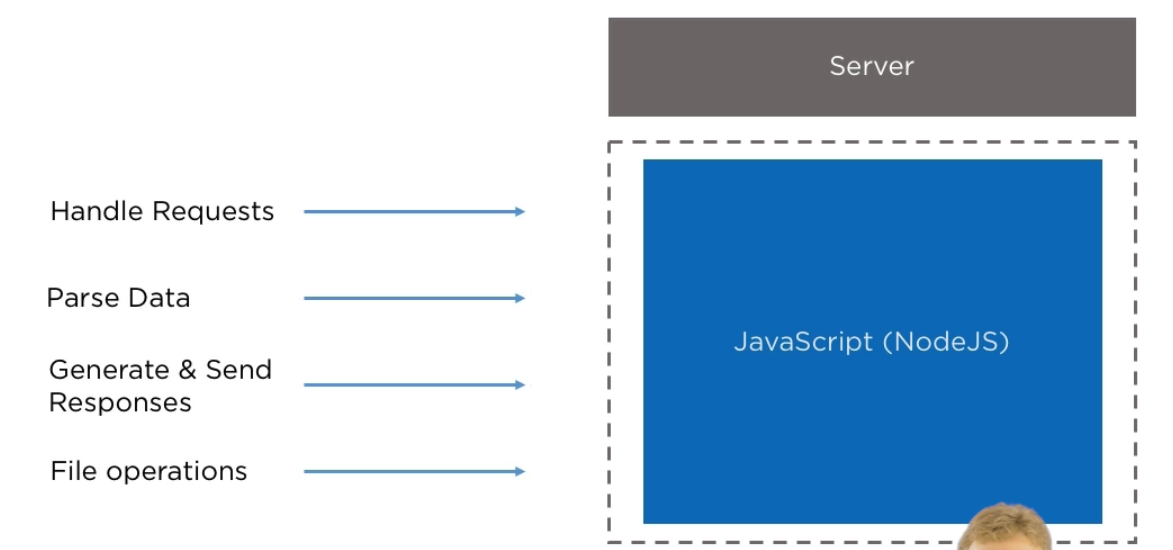
Once installed the version can be checked by using the command: **node -v**;



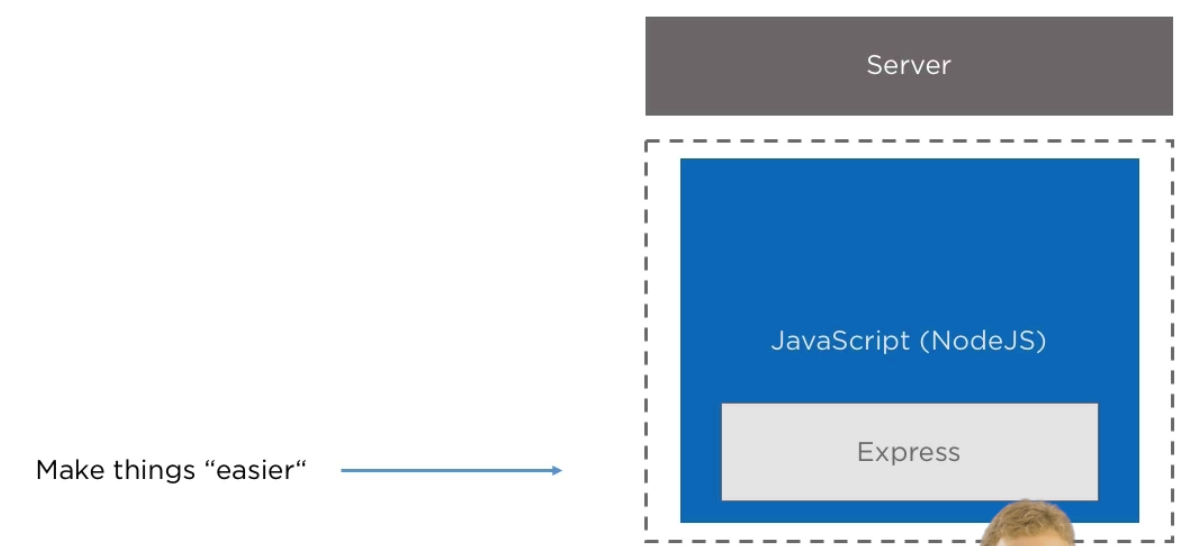
**What is Node.Js**

1. A Platform which allows us to run JavaScript on a computer/server.
2. Read, delete and update files.
3. Easily communicate with a database.





For Node JS the popular framework to handle all the request is Express. Express JS wraps the Node JS.



**Why Node.Js**

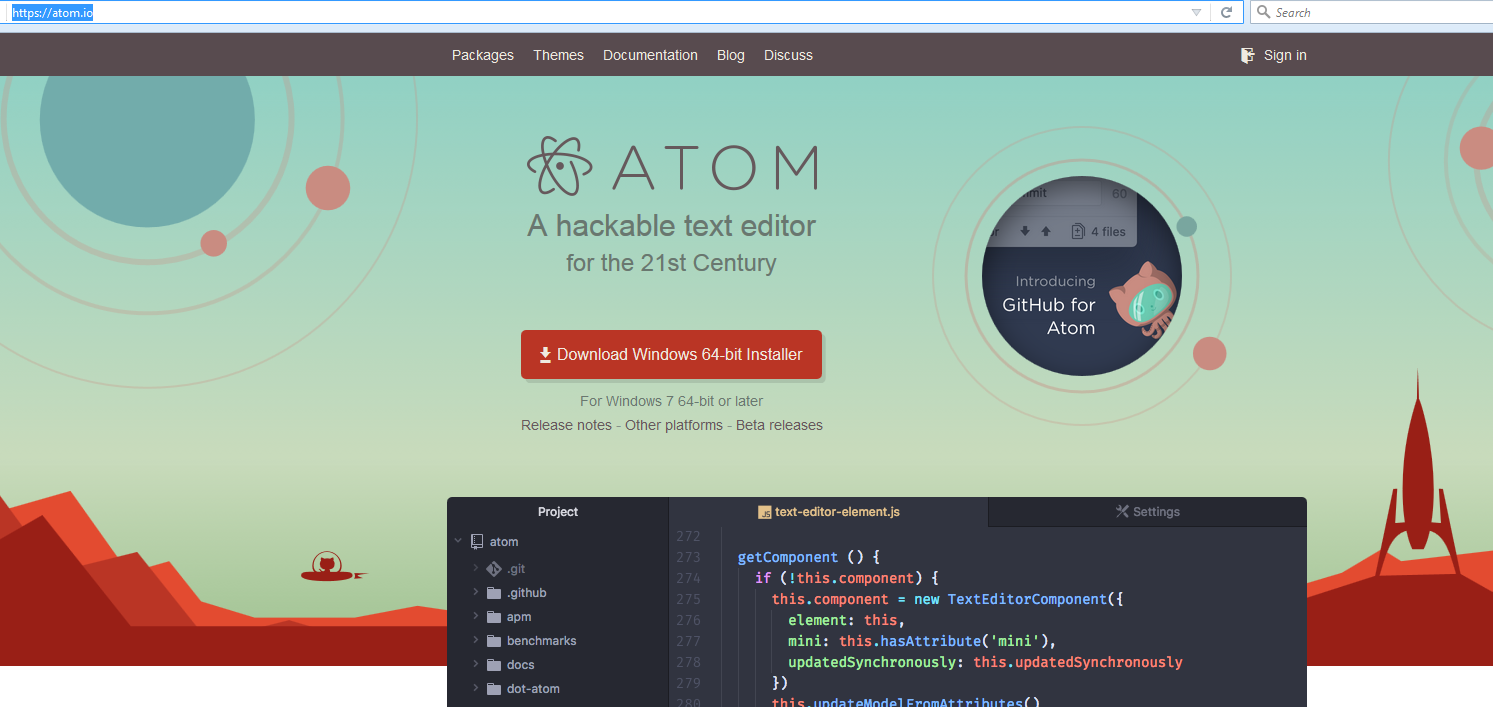
1. It uses JavaScript
2. Very Fast (runs on the V8 engine and uses non-blocking code)
3. Huge ecosystem of open source packages (npm)
4. Great for real-time services (like chats)



Express is a very popular node package.

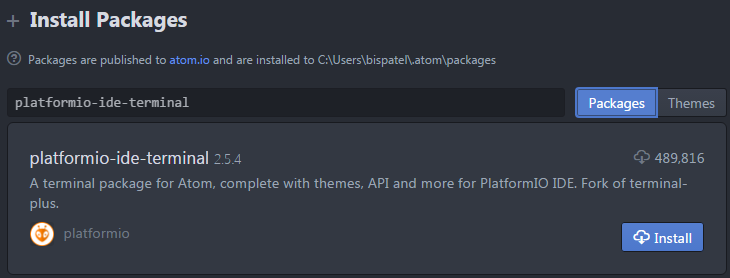
**Node JS Text Editor**

**Link:** <https://atom.io/>

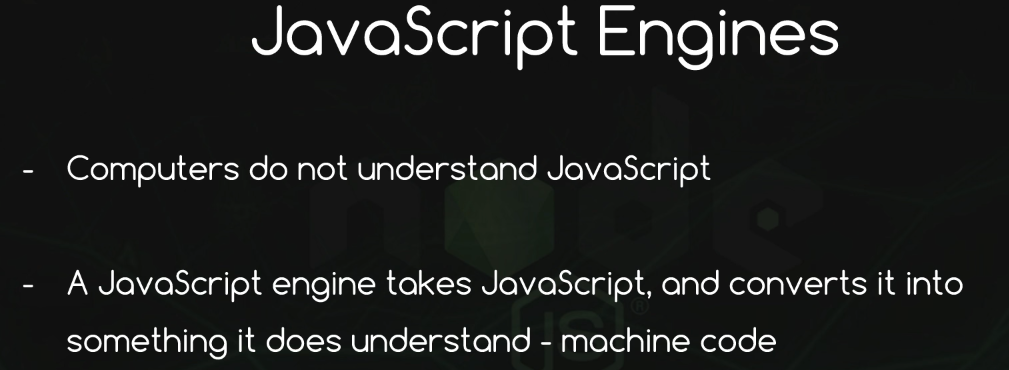


**To execute the node command in the atom tool itself follow the below steps**

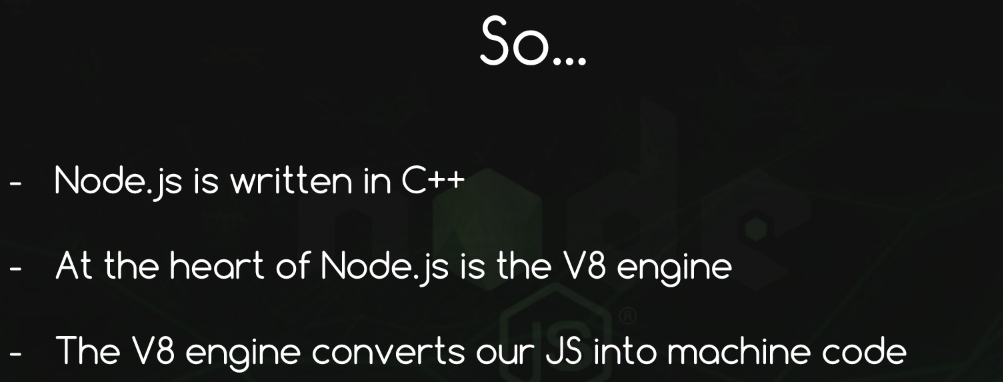
File->Settings->Install and search for platformio-ide-terminal

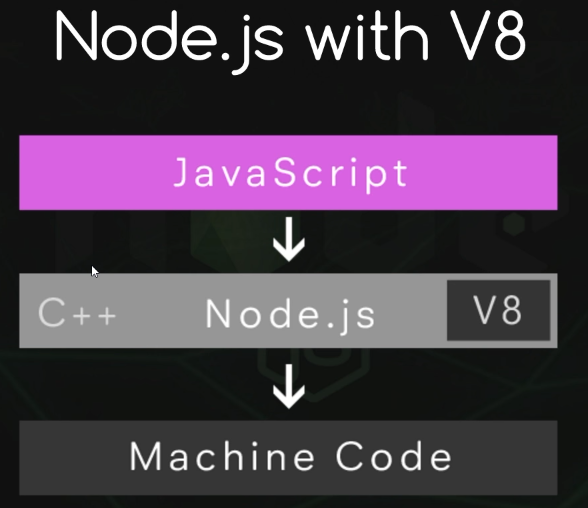


**V8 Engine**



Node Js is written in C++. Reason is Node Js uses the V8 JavaScript engine created by Googlewhich is also written in C++.





**Node JS enables writing Server side code with JavaScript.**

**Node JS Documentation Link:** <https://nodejs.org/api/globals.html>

console.log(\_\_dirname); **//To print the current directory**

console.log(\_\_filename); **//To print the current file**

**Functions can be created in different way in Node Js**

1. **//normal function statement**

function sayHi () {

console.log("Hi");

}

sayHi (); ===========Will print Hi

1. **//Functional Expression**

var sayBye=function () {

console.log("Bye");

}

sayBye (); =================Will print Bye

1. //passing Function to a function

function callFunction(fun) {

fun ();

}

callFunction(sayBye); ========Will print Bye

**Modules and require ()**

**Using require we can use the function written on other js**

**Ex: Count.js**

var counter=function(arr){

return 'There are '+ arr.length+' elements in the array';

};

console.log(counter(['A','B','C']));

**app.js**

require('./count');

cmd->node app======The output will be “There are 3 elements in the array”;

We can expose a module using the command

Module.exports = counter -------Use in count.js

In the app.js

Var output=require(./count);

Console.log(output([‘A’,’B’,’C’])); ==== The output will be “There are 3 elements in the array”;

How to export a module with different function

**Stuff.js**

var counter=function(arr){

return 'There are '+ arr.length+' elements in the array';

};

var adder =function(a,b){

return `The sum of 2 numbers is ${a+b}`;

}

var pi=3.142;

module.exports ={

counterFun:counter,

adderFun:adder,

piVar:pi

}

**App.js**

var stuff =require('./stuff');

console.log(stuff.counterFun(['A','B','C','D']));

console.log(stuff.adderFun(5,6));

console.log(stuff.piVar);

**Reading and Writing Files Using Node JS**

**We require fs module for this.**

**Ex:**

**Synchronous method**

var fs= require('fs');

var readMe=fs.readFileSync('Test.txt','utf8');

**For Writing to a file:**

fs.writeFileSync('writeFile.txt',readMe);

param1: file name

param2: content

**Asynchronous Method:** This code will not block the execution of other methods in the JS

fs.readFile('Test.txt','utf8',function(err,data){

console.log(data);

fs.writeFile('writeFile.txt',data);

});

**Create and Remove Directory:**

**//deleting a file. Make sure it exists**

fs.unlink('writeFile.txt');

**//Create Directory synchronously**

fs.mkdirSync('stuff');

**//Delete Directory synchronously**

fs.rmdirSync('stuff');

**//Create Directory Asynchronously**

fs.mkdir('stuff',function(){

console.log("Created Directory");

fs.readFile('Test.txt','utf8',function(err,data){

fs.writeFile('./stuff/writeMe.txt',data);

});

});

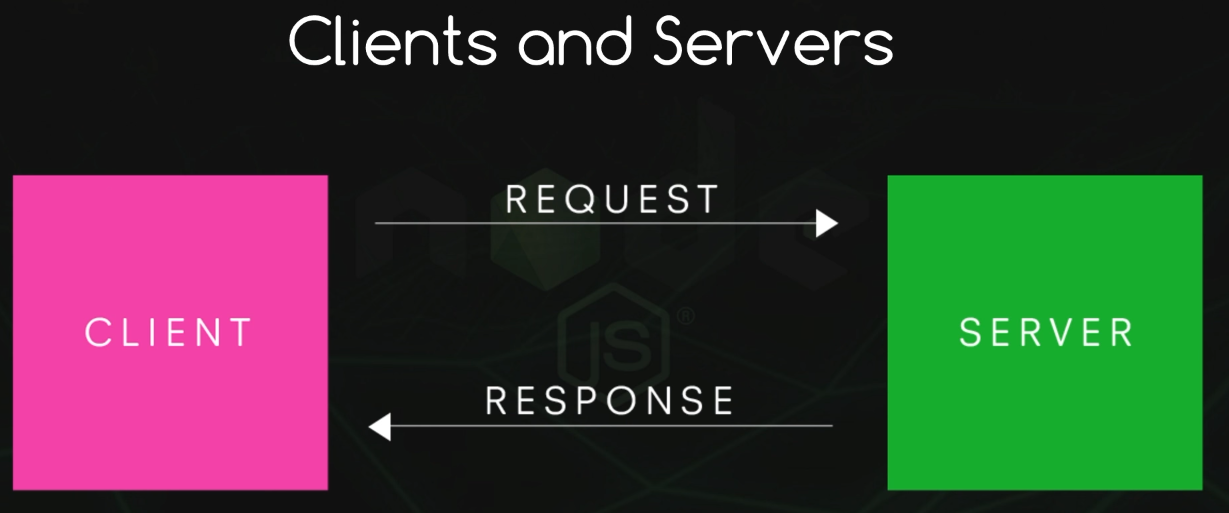
**//delete the file in the folder first followed by deleting the folder**

fs.unlink('./stuff/writeMe.txt',function(){

fs.rmdir('stuff');

});

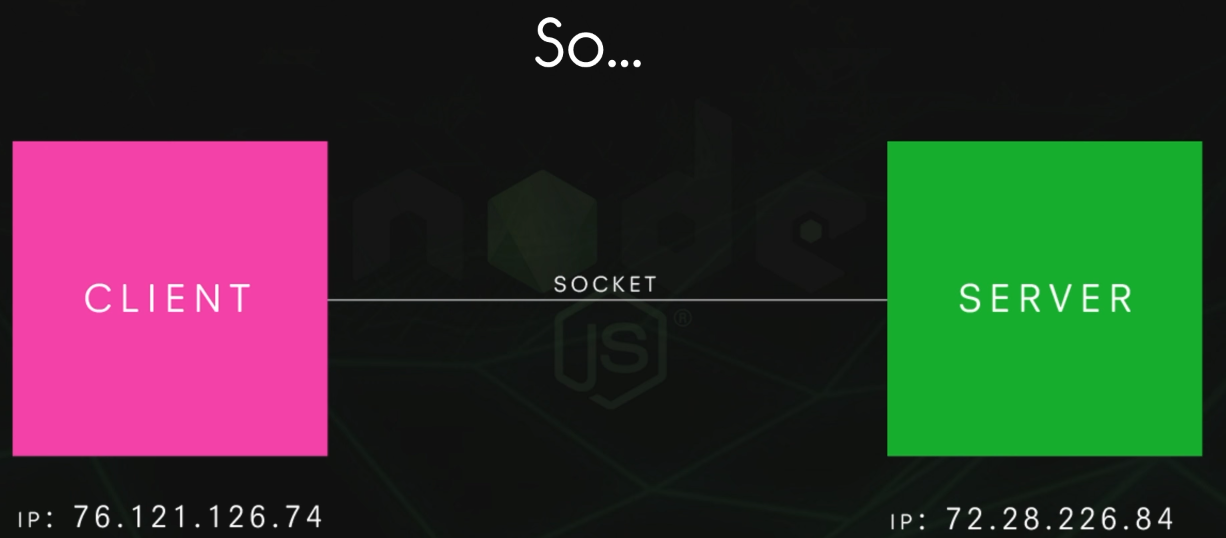
Client and Server in Node JS



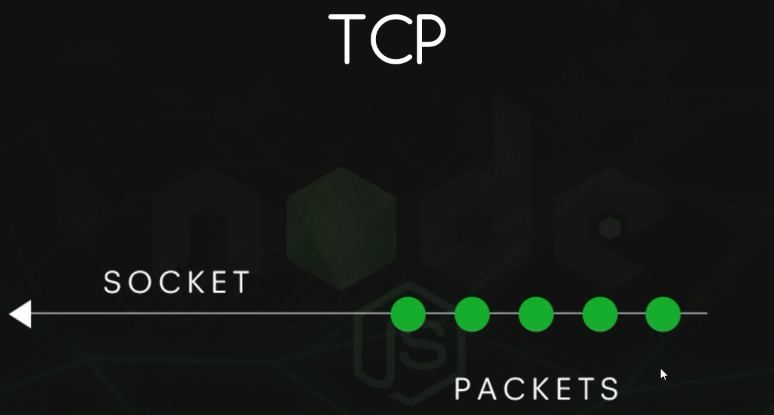
**Protocols:**

A set of communication rules, that two sides agree to use when communicating.

Whenever client and server communicate with each other a connection will be created and data will be transferred via the socket.

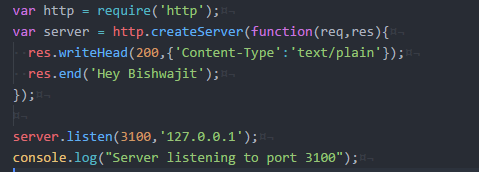


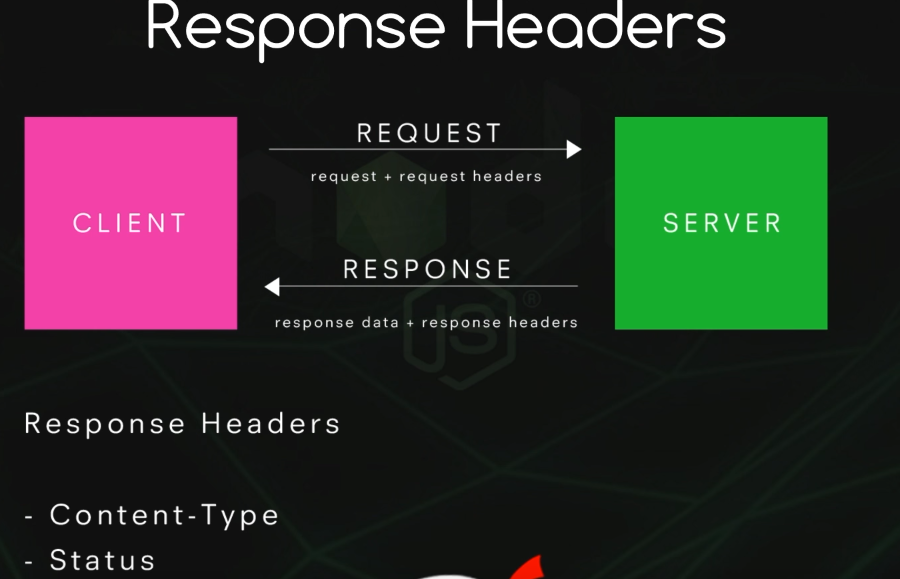
Data will be transferred via this using different protocol (Http or FTP)



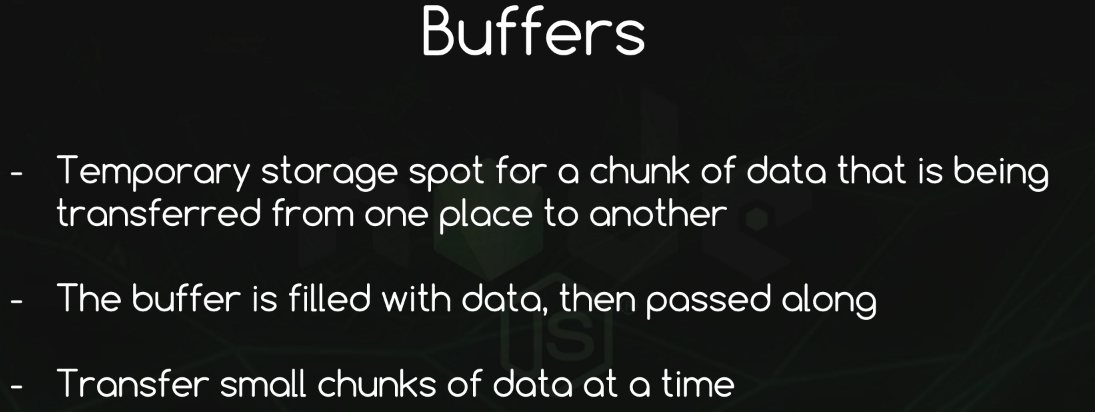


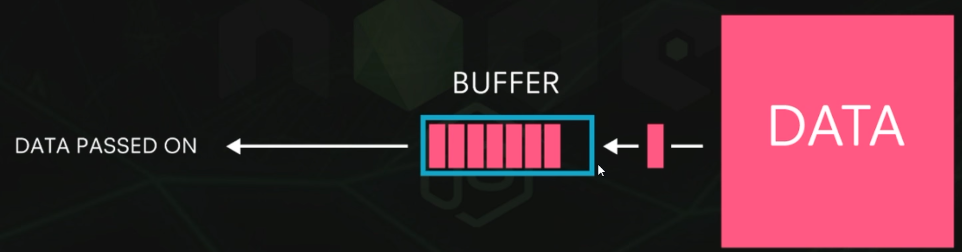
**Creating the Server in Node JS**

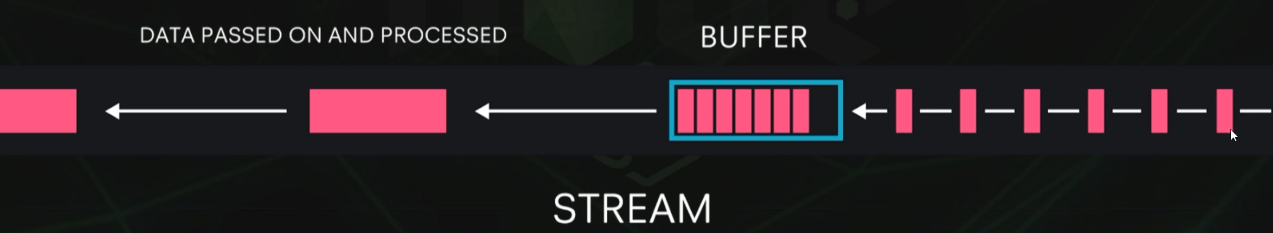




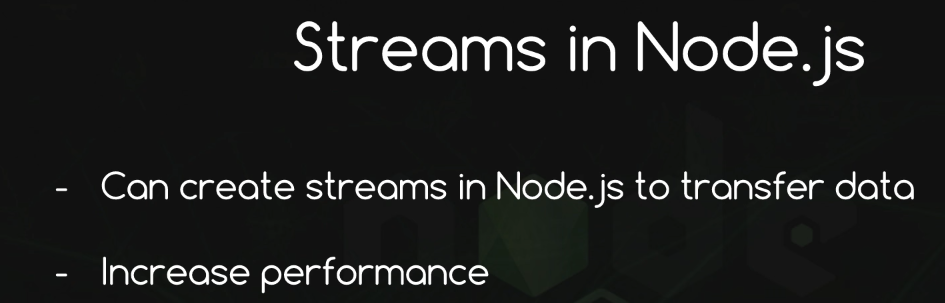
**Streams and Buffers**

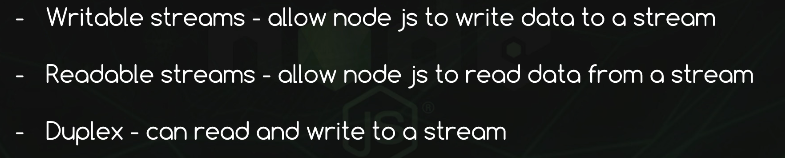




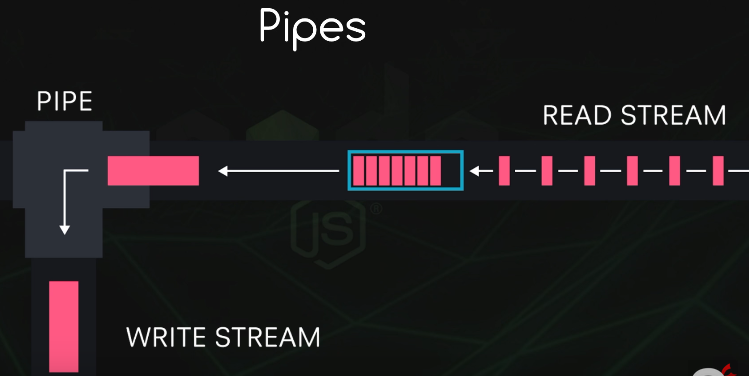


When Buffer is full then chunk of data will be send to client for reading. This process is manual.





**Pipes**



**Routing**

Based on the URL Node will send the proper output to Client.

**Node Package Manager (NPM)**

Express helps in creating a web application on node. It is just a bunch of command line tools which helps us install third party packages. Express is one of the package and there are many more.

It helps us perform certain set of task within Node JS.

Ex:

Express Package helps us in routing and templating.

NPM website: <https://www.npmjs.com/>

Install express using the command: npm install express

**Package.json File creation**

It will keep track of all the packages that the node is dependent on.

We can manually create it or using the command in node js the same can be created.

Cmd->npm init

**Install Nodemon**

nodemon will watch the files in the directory in which nodemon was started, and if any files change, nodemon will automatically restart your node application.

nodemon does **not** require *any* changes to your code or method of development. nodemon simply wraps your node application and keeps an eye on any files that have changed. Remember that nodemon is a replacement wrapper for node, think of it as replacing the word "node" on the command line when you run your script.

npm install -g nodemon====-g is global. So, in future if we create any other application then, no need to install nodemon again.

**Proxy setting while downloading the node packages:**

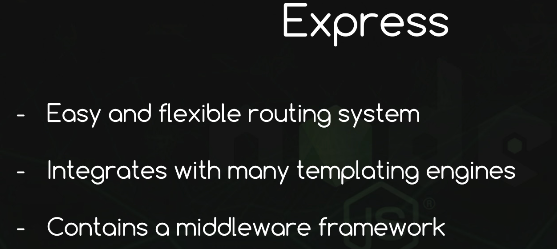
npm config set https-proxy http://username:password@hostname:port

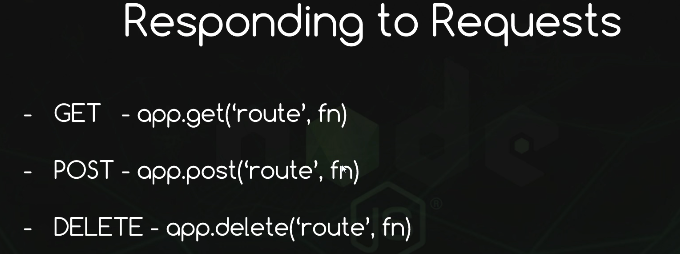
**>npm config set https-proxy** <http://148.87.19.20:80>

We don’t have to restart the server if anything is changed with nodemon.

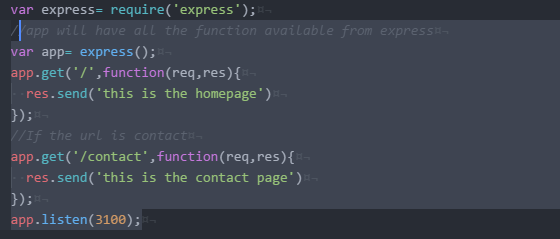
Introduction to Express

One of the most popular node packages.





Express will automatically know the content type of the message. So, no need to write this explicitly



**Route Parameters**



**Template Engine:**

Using Template Engine, we can embed data in JavaScript code into html code.

For this we are going to use EJS

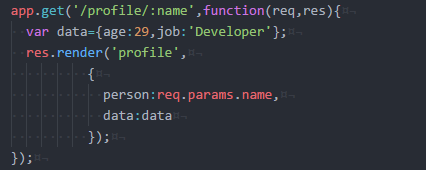
**Link**: <http://www.embeddedjs.com/>

To install EJS following command is used: npm install ejs -save

In express we can set the view engine using the code:



How to inject data to views:



Injecting person and data field to profile page.

Profile template will be searched inside the views folder of current directory.



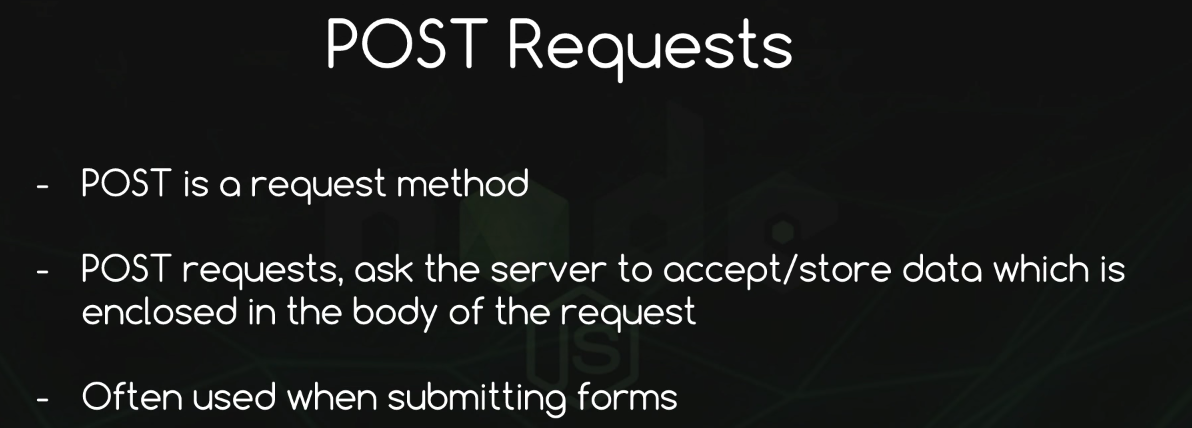
**rs** command is used restart the server

**MiddleWare :**

MiddleWare is the code that runs in between the request and response.



req.query===it is used to get the query parameter.



To parse the form data, we use Body parser:

<https://www.npmjs.com/package/body-parser>

installation command: npm install body-parser

To send Email we can use nodemailer package.

**Create a Node JS application from scratch**

**Step1**: Create the package.json file by executing the following command.

* **npm init()**

It will ask a series of questions.

**Step2**: Install express

* npm install express -save

-save will register the version of express in package.json

**Step3**: Install ejs (template engine)

* npm install ejs -save

**Step4**: Install the body-parser which will parse the post request.

* Npm install body-parser -save

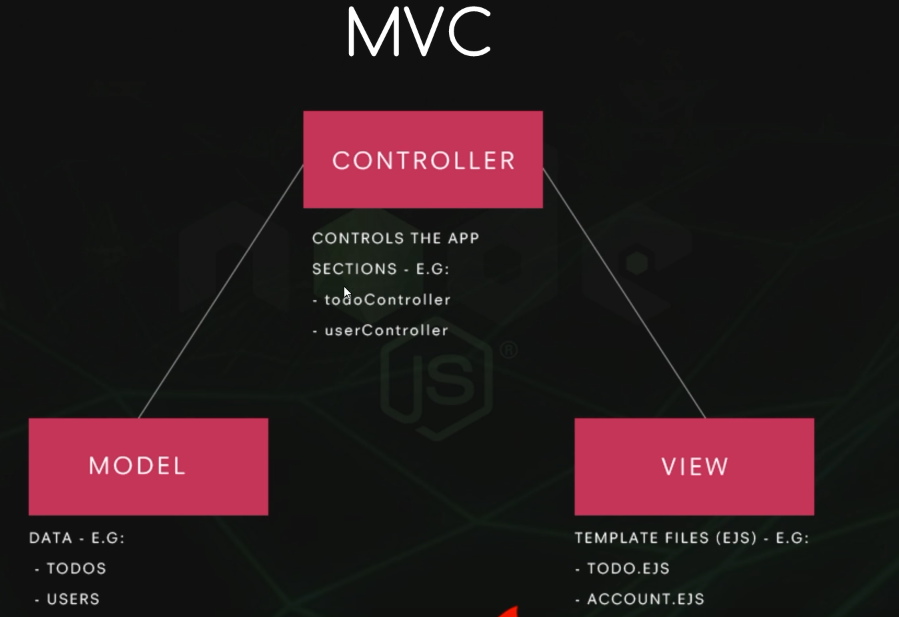
**Step5**: Install mongoose to interact with the No-Sql database

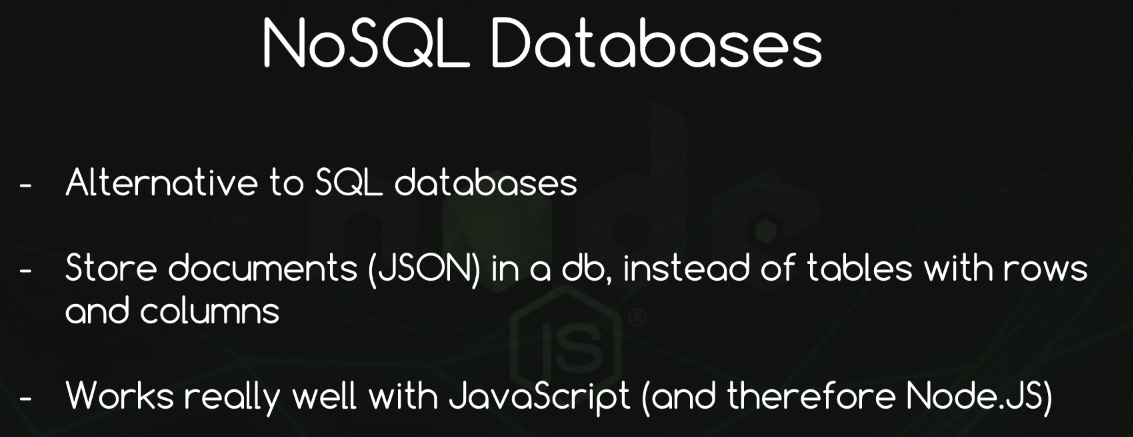
--to-do

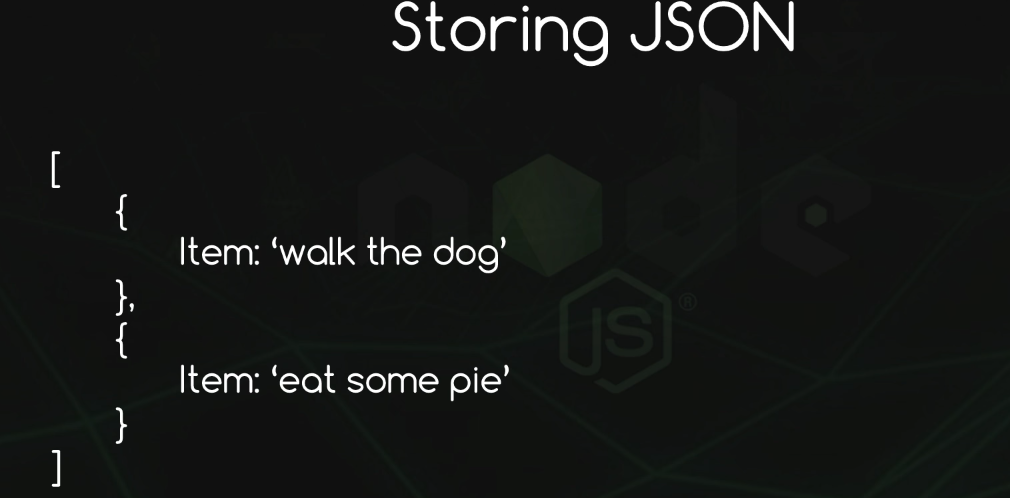
**Model**- It is just the data

**View** -It is what we send to the user

**Controller**: Any kind of data manipulation from Model to view will be managed by the Controller.



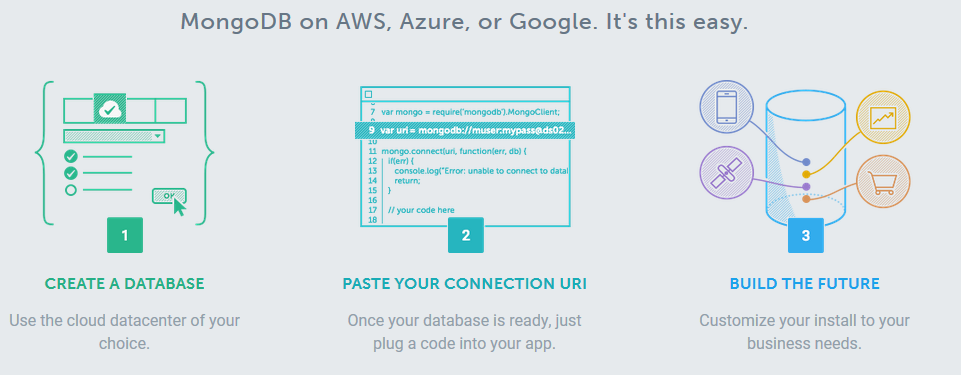




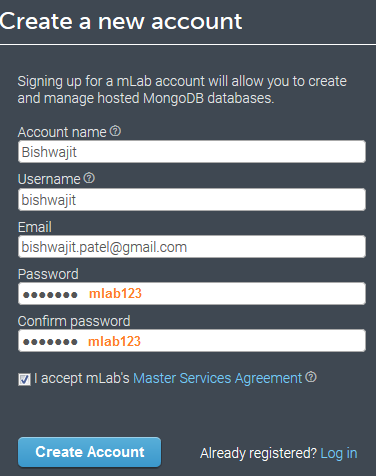
Mongo DB can be downloaded from: <https://www.mongodb.com/careers>

MongoDB hosted version: <https://mlab.com/>

We can create a database on the cloud and use it in our application.



Account details:



To use mongo db we can use the package: mongoose

This can be installed by command > npm install mongoose -save

