Node JS

Introduction

Node JS is a JavaScript runtime which uses a v8 engine. A V8 engine is an open source JavaScript engine written in C++ that takes a JavaScript code and compiles to a machine code. It is used inside of Node JS and inside of a Chrome browser.

Node is a platform to run any JavaScript program outside of any browser.

For plain JavaScript there is a v8 engine installed in chrome browser, and we used to execute the JavaScript programs in chrome. Now the same v8 engine is extracted and wrapped around with a C++ environment and installed inside our system just like any other language and name as node js.

Now we can execute the JavaScript programs just like any other browser.

Ex:

console.log('Hello World');

in the browser, we have a global object like ‘window’ and document to manipulate the DOM on the browser and deal with the dynamic nature of any browser. But here we don’t have window and document objects.

Here we have an object called ‘global’ just like ‘window’ in browser.

Here we have an object called ‘process’ just like ‘document’ in browser.

|  |  |
| --- | --- |
| Java Script | Node JS |
| Window object | Global |
| Document object | Process object |

Global object of node js indicates the global object of node , all the activities we can able to perform with node js will be captured by ‘global’ object.

The Process object indicates each process we run for node js application.

How open a node terminal to execute Node JS

Open cmd -> type node

How to exit from the node terminal is

Press CTRL + C twice (or) type ‘.exit’

Why to Use NODE JS

Node js is very popular backend processing logic of any application. The Companies like Netflix , Uber , Walmart uses node js in the production.

Node JS uses a non-blocking IO model of communicating with database and other environments.(This means it’s a multithreaded and multi tasking IO).

Node JS serves all the requests simultaneously without blocking any other request by using its callback functions of JavaScript.

Node JS is having a largest package ecosystem to make use of third party libraries. And Node JS uses package manager called ‘npm’. This is the world’s largest ecosystem of open source libraries which is used to inject these libraries to our applications.

Almost all kind tasks there are one or other library is available for us in the npm eco system and we can make use of it using the npm package manager.

This is almost like a maven tool for Java Applications.

Ex:

[www.npmjs.com](http://www.npmjs.com)

Using the plain JavaScript we cannot write each and every boilerplate code for each specific task. We can go ahead and search for some of the libraries in npm and if available for our requirement, we can install it and make use of that library in our node js application.

Ex:

Validate js -> for validating our JavaScript Objects

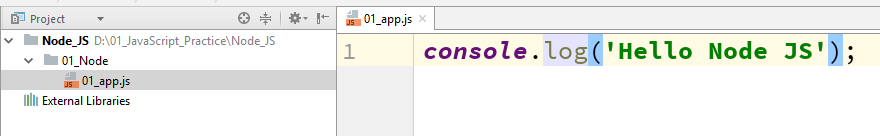
Lodash -> for an ocean of utility methods

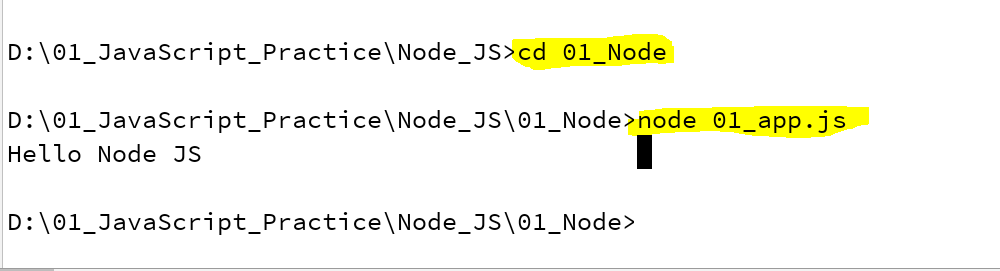
Yargs -> to read the command line args in a better way

Note: All the libraries in npm are written in a non-blocking fashion. We should not worry about is that blocking io or non-blocking io.

First Application of Node JS

Create any ‘.js’ file in the project folder and write a console.log() and execute it in the cmd window with the command ‘node filename.js’

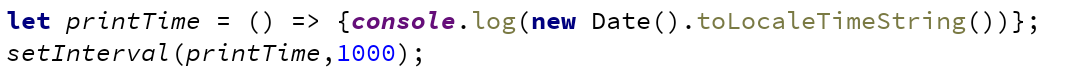




Example 2 :



Example 3:



Node JS Fundamentals

Node JS is a runtime environment to run a javascript program outside of a browser. Node js is comes with whole lot of modules to deal with the underlying file system and os and other IO operations.

All the modules of node js will be available in the official node js website.

<https://nodejs.org/dist/latest-v10.x/docs/api/>

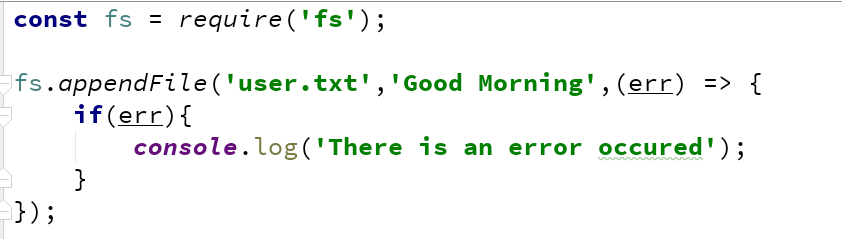
Examples of Node JS Modules are as follows,

1. File System -> The file system module is used to deal with the local file structure. We can read the data of a file and write some data to a file and creating new file and updating the content of a files.
2. Operating system –> This module is dealing with the underlying operating system, using this we can fetch the memory specific data , user info , host information , uptime of os and etc.

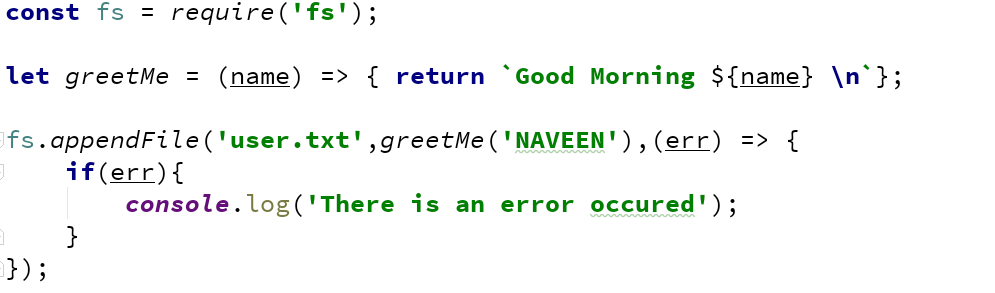
We can load any module to our node js application using ‘require’ statement.

Once it is loaded we can start using the module in our application.

Examples on File System Module

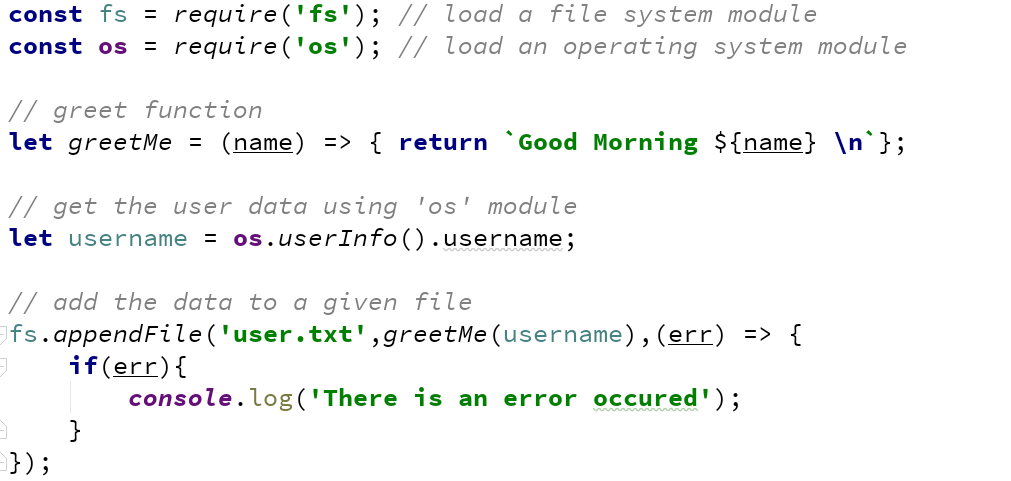


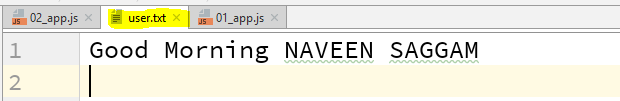
Fs.appendFile() function is used top append any data to the given file name, if the file is not exists, it creates a new file and if the file already exists in the given location, it append the given information to the file. In this ‘err’ callback is mandatory for this function.



In the above function, it adds the Good Morning Message to the given function.

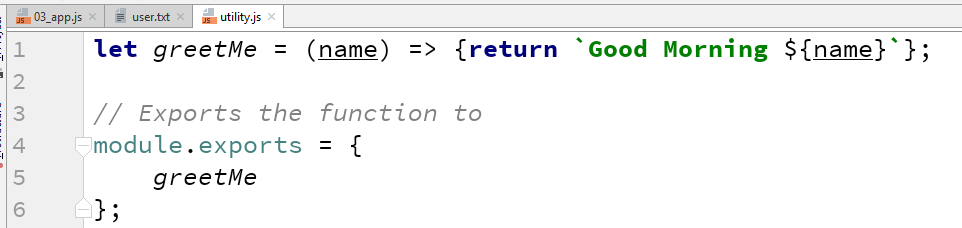
It prints the static User Name on the console. In the next example we will fetch the user information from operating system module and add that to the given file.





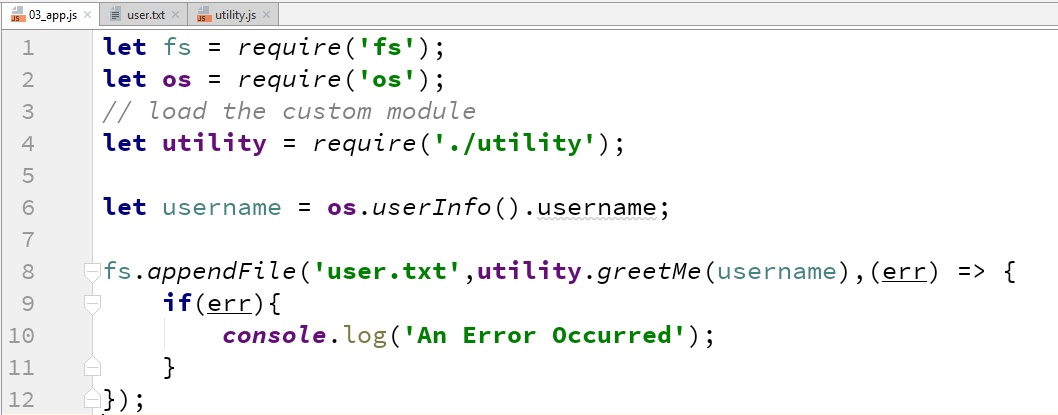
How to load custom modules in our Node JS application

In the above example the ‘greetMe()’ function is available in the same file and we are also calling the function from the same file. Now we are going to keep this function in some other JavaScript file and export that file as a module to another JavaScript file and make use of that function.

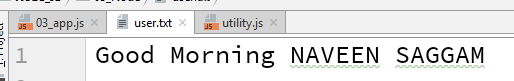


We have to write the functions in any other JavaScript files and export that functions using ‘Module.exports’ statement. So that other JavaScript files load this JavaScript files then these exported functions will be available to use them.

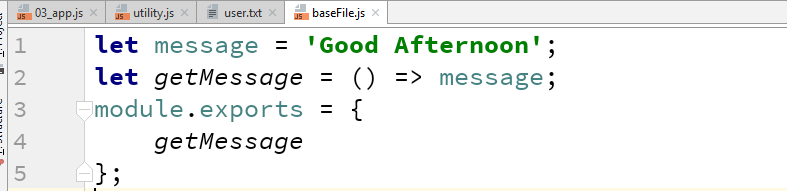
If any file is not exported, then even though we load the JavaScript file, we cannot be able to access to that functions which are not exported.

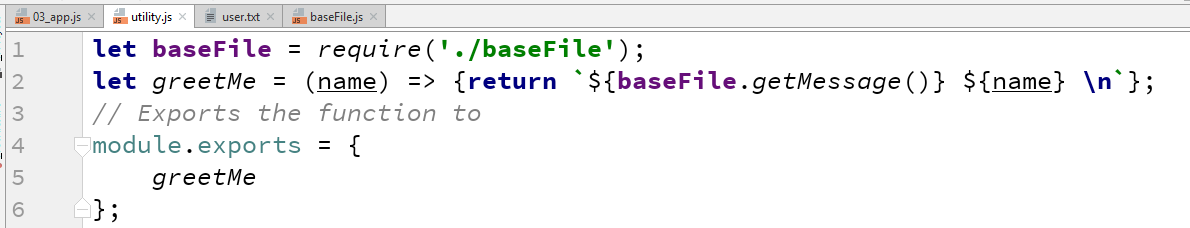


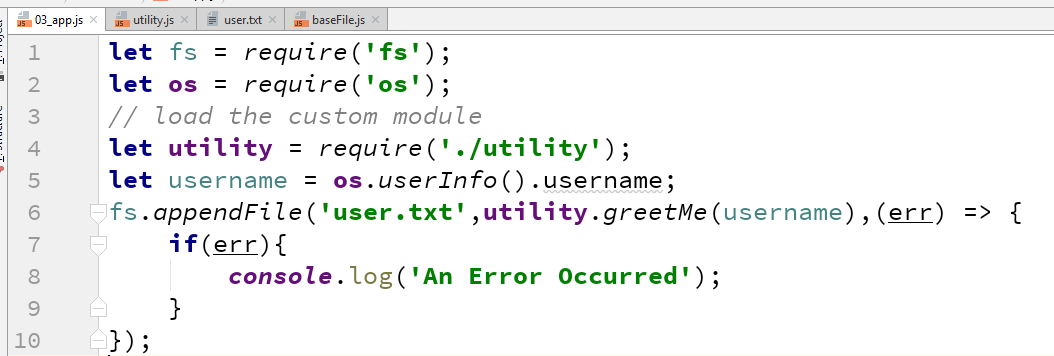
We can load the custom module using the same require() statement as mentioned in the above program.



Let’s make the custom module is dependent on another module.

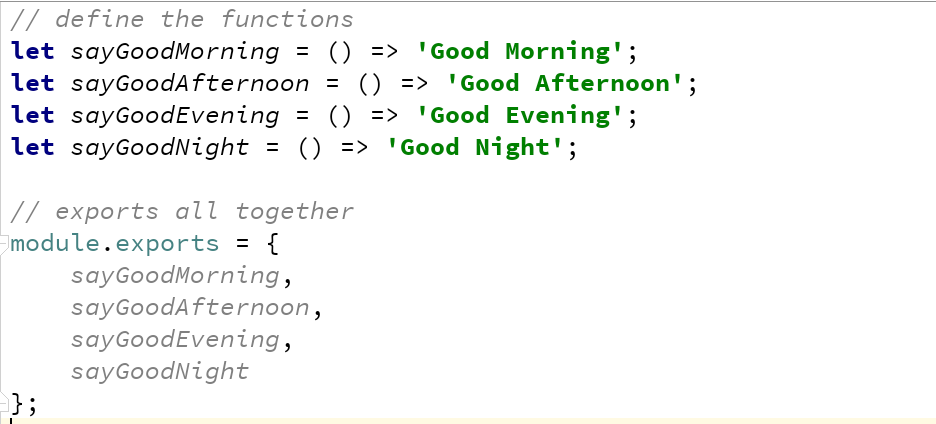






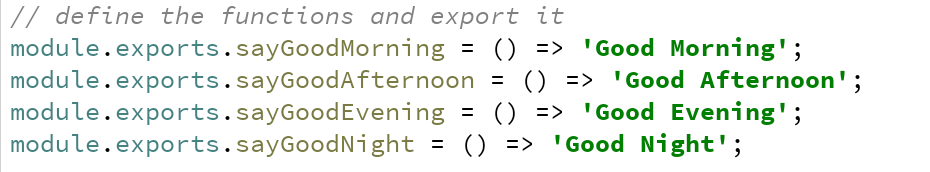
We can export each module in two ways,

1. First we define all functions and later we can export all the functions together.



In this approach we may miss some of the functions to be exported.

1. We can export the function as soon as it is declared or defined as follows,



Note: Why we create a custom modules separately is if any utility function we defined, if we wanted to use it many number of times, we must not define that function every time. We have to create once and use it any number of times.

Using 3rd Party Modules in Node JS

To use any third party module, we should have a package.json file in the project folder. Using this we can install any package of npm landscape.

To create a package.json file, we use the command called ‘npm init’ go ahead with the default changes, the package.json file will be created.

Now we can go to the project folder and install any package using the following command.



The above command will install the required package and creates in inside a ‘node\_modules’ folder. Now we can starts using this inside of our node JS application.

Here we use a ‘lodash’ for utility functions.

Same like live-server for live reload of a page, here we can use a 3rd party module called ‘nodemon’ for live-reload of the JavaScript code.

We can install this nodemon package using the following command



We can run the javascript file using this nodemon command as follows



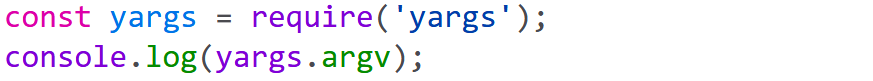
How to read command line arguments using Node JS

To read all the command line arguments, we ‘process.argv’ in node js

This returns an array of all the command line args.

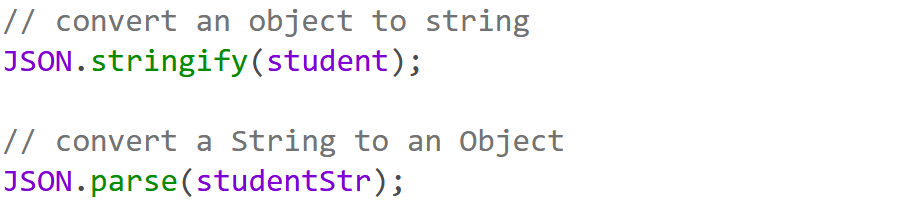
To read the command line arguments in a proper way we have to use ‘yargs’ 3rd party module.

We can capture the command line arguments using the following statement.



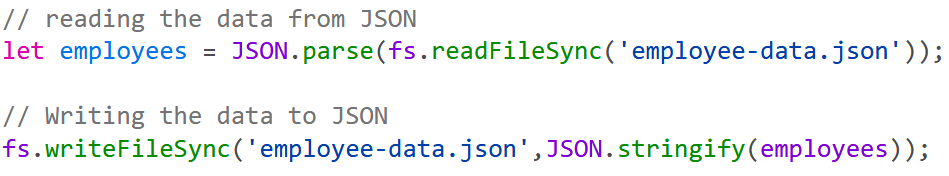
Using JSON in Node JS

We can convert a javascript object to a string using the following function



While storing the data into a json file, first convert an object to a string format and write to a json file using fs module.

While retrieving data from json, first convert to an object using JSON.parse() and use it.



Debugging a Node JS Application

For debugging a node js application, we can use Chrome debug tool.