Node.js

Created Date : 12/09/2022
Created By : GOVE ACADEMY

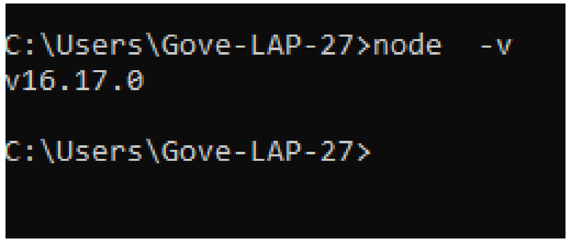
Version History

S. No	Revision Date	Version	Modifications	Modified By

Table Of Contents

1.Introduction

- Node.js is an open-source runtime environment for java script
- By using node.js java script can be used for server-side programming rather than client-side programming.
- Node.js uses common.js module and runs on v8 engine.
- Node.js works asynchronously, where node.js will eliminate the waiting process and simply continues with the next request.
- Node.js runs single-threaded, non-blocking I/O module , asynchronous programming, which is very memory efficient.
- Node.js has inbuilt modules, libraries and packages.
- To check the version of node.js in the system use node –v in the command prompt



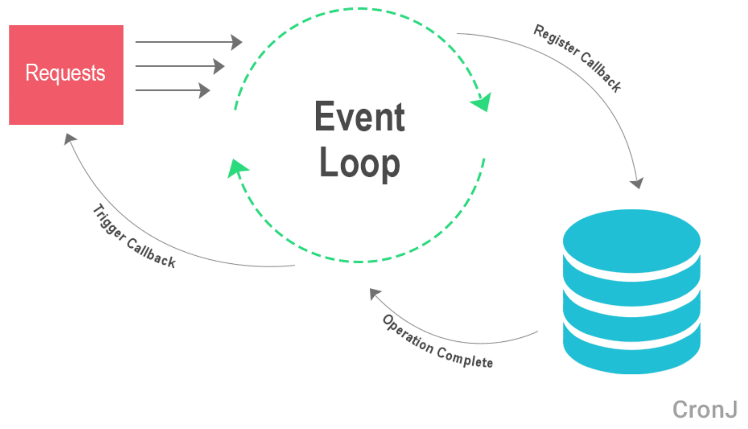
```
C:\Users\Gove-LAP-27>node  -v
v16.17.0

C:\Users\Gove-LAP-27>
```

Node.js Advantages

- Node.js is an open-source run time environment for java script built on Google Chrome's JavaScript V8 Engine.
- Node.js allows us to run JavaScript on the server.
- Node.js uses an event-driven, asynchronous non-blocking I/O model
- Node.js operates on a single thread event loop

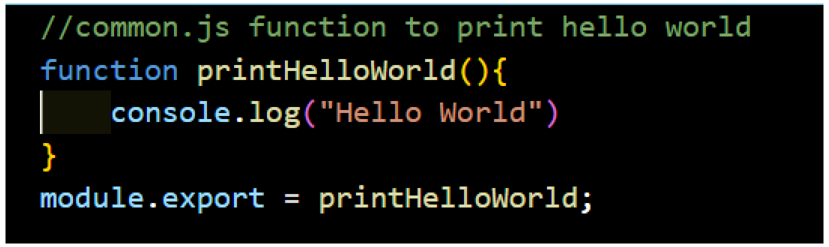
Non-blocking, I/O (Input/Output) Model

Non-blocking, I/O operations can handle and execute multiple request at the same time . Instead of the process being blocked and waiting for I/O operations to complete, the I/O operations are executed sequentially by the system, so that the process can execute the next piece of code.



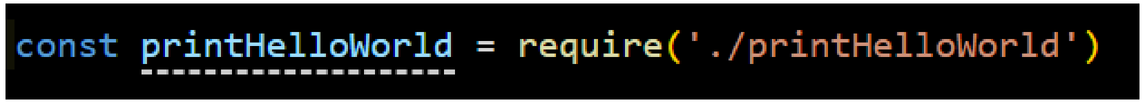
2.Common.js

- **Require** is used to load Common JS modules in node.js .Where the file can be created with the .js extension.



```
//common.js function to print hello world
function printHelloWorld(){
    console.log("Hello World")
}
module.export = printHelloWorld;
```

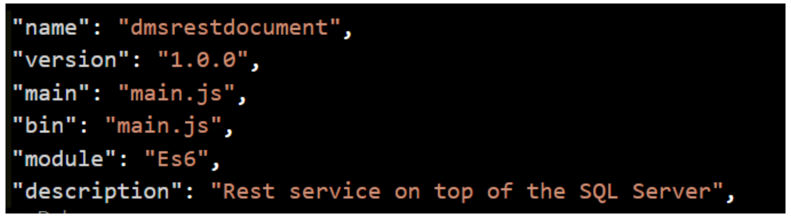
The **module.export = printHelloWorld** is what our module expose publicly to be used. We can save it in a file with .js extension and then access to the function from other modules by using ,



```
const printHelloWorld = require('./printHelloWorld')
```

3.ECMAScript

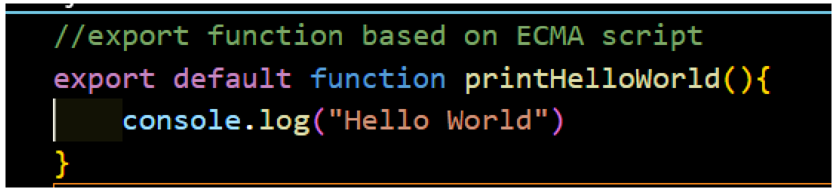
- As default the node.js support only the common js modules. Were only the common js syntax can be used.
- To run node.js by supporting the **ECMAScript modules** changes need to made in the **package.json** file.



```
"name": "dmsrestdocument",
"version": "1.0.0",
"main": "main.js",
"bin": "main.js",
"module": "Es6",
"description": "Rest service on top of the SQL Server",
```

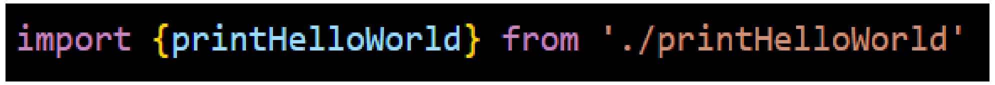
Node.js configured to support **ECMAScript** Modules

- Import is used to load **ECMAScript** modules and the file is created is using .mjs extension



```
//export function based on ECMA script
export default function printHelloWorld(){
    console.log("Hello World")
}
```

The **export default function printHelloWorld()** is what our module expose publicly to be used. We can access to the function from other modules by using .



```
import {printHelloWorld} from './printHelloWorld'
```

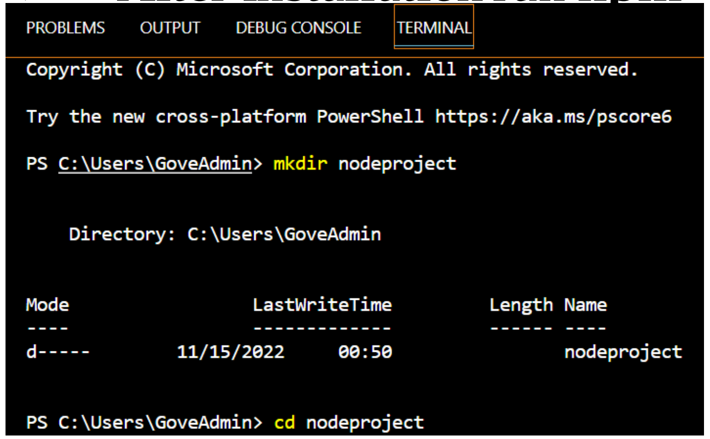
.Creating a node.js project

Step 1: Install and setup node.js in local

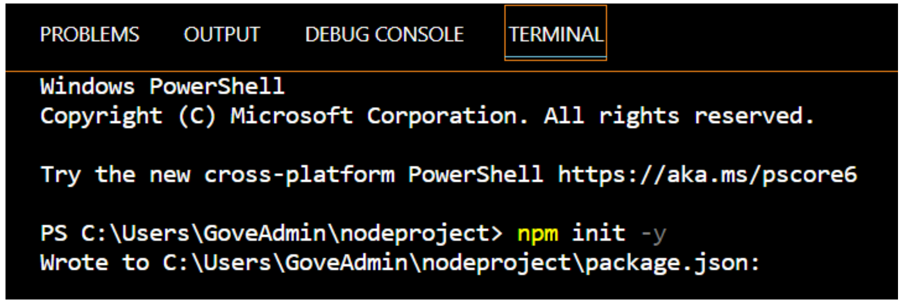
➢ For installing and setup node.js in local refer - [Install and setup]

➢ After installation run **npm –v** command in command prompt to check the **version** of the npm version.

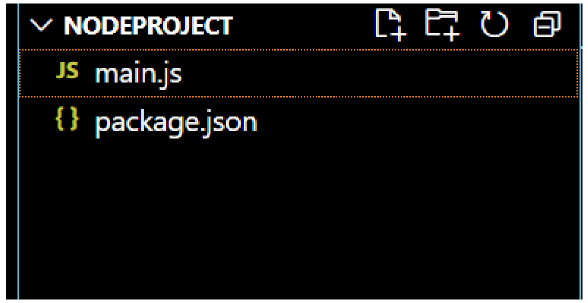
Step 2: Create a node project



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\GoveAdmin> mkdir nodeproject


    Directory: C:\Users\GoveAdmin


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/15/2022     00:50                nodeproject


PS C:\Users\GoveAdmin> cd nodeproject
```



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\GoveAdmin\nodeproject> npm init -y
Wrote to C:\Users\GoveAdmin\nodeproject\package.json:
```

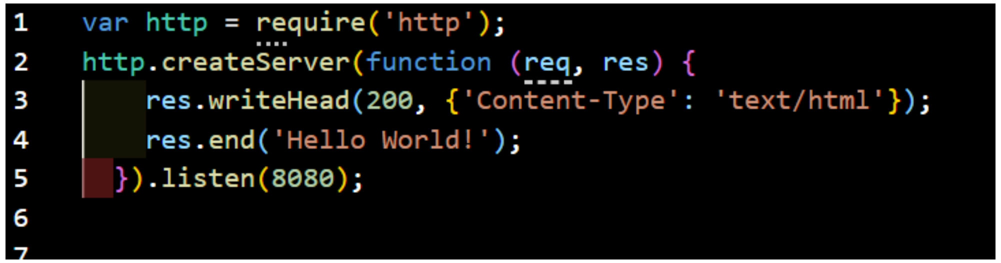


Create the required folder using

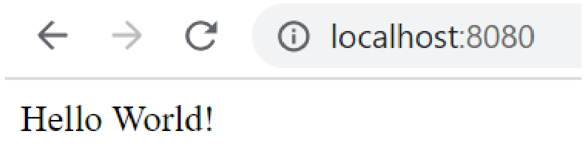
- **mkdir File Name**
- **cd File Name**
- **code .**

- Run **npm init – y** command in the terminal to create a node project.
- Once these steps are done the developer can develop, compile and run the source code for the project.

Sample Code



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



localhost:8080

Hello World!

- ❑ Below is the sample program to print hello world in the port localhost:8080.
- ❑ http module can be used to create our own server path –localhost:8080 we can run our code.
- ❑ The server should handle the request and response.
- ❑ The http module is required by using var http = require('http')
- ❑ A request(req) and response(res) is passed as parameters in a function so the server can handle it.
- ❑ The statement to be executed is written inside the function.
- ❑ While executing the code if the status is 200 then the response will be Hello World
- ❑ The localhost:8080 is used to listen the code execution.

Recap

- Node.js is an open-source runtime environment for java script.
- To check node.js Version : node –v.
- Non-blocking, I/O operations can handle and execute multiple request at the same time.
- Require is used to load Common JS modules in node.js.
- ECMAScript is the standardization for creating a scripting language.
- To check the **version** of the npm - npm –v.
- To Create a node project run **npm init**

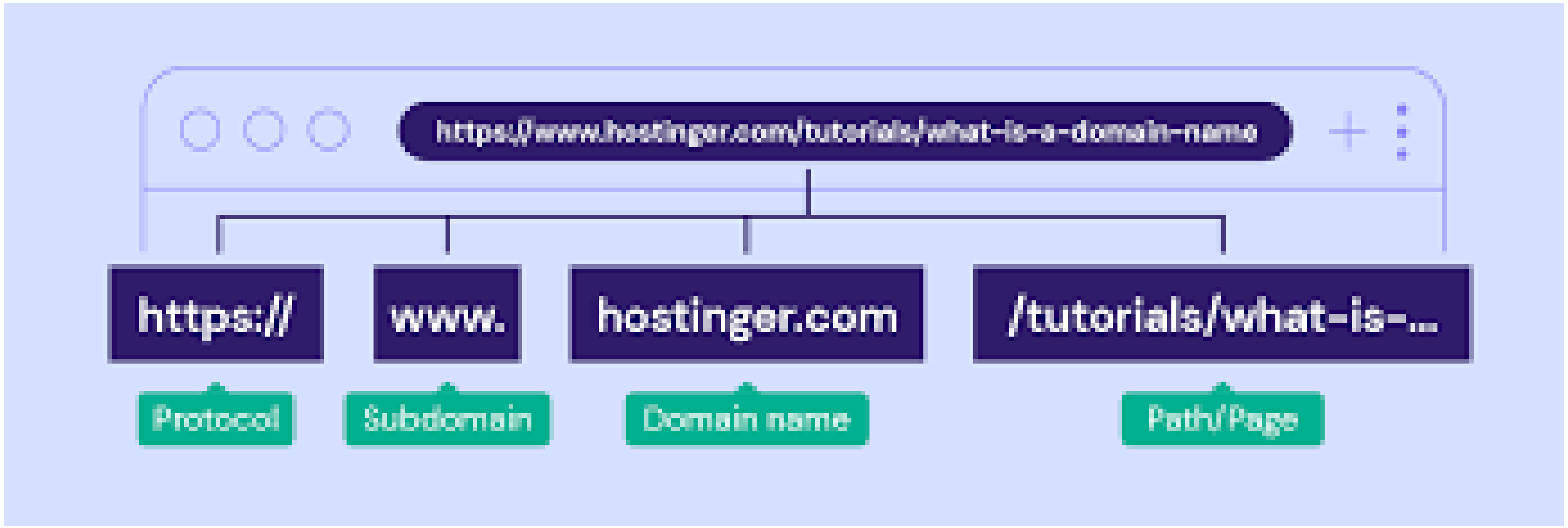
Modules

5. Modules

What are modules?

- Module in Node JS is a simple functionality organized in single or multiple JavaScript files which can be reused throughout the Node JS application.
- Each module in Node JS has its own context, so it will not interfere with other modules.
- Each module can be placed in a separate '.js' file under a separate folder.

URL – Uniform Resource Locator



5.1. Http Module

click here to see real time example for client Server

To create web server using http module

Syntax

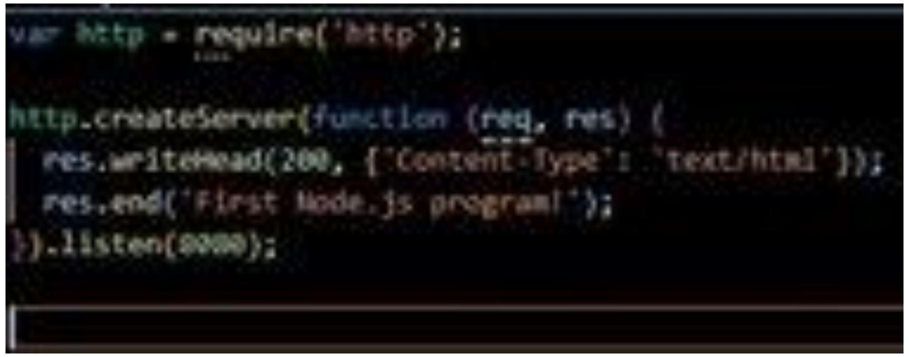
Var name http is used and require is a function to import the http module.

var http = require('http');

createServer() Method

The create Server method creates a server on local system

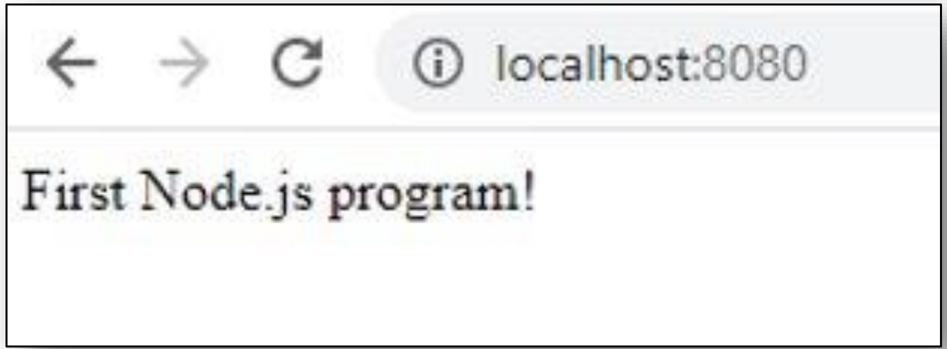
`http.createServer(`*`request Listener`*`);`



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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('First Node.js program!');
}).listen(8080);
```

This function handles request from the user, as well as response back to the user.

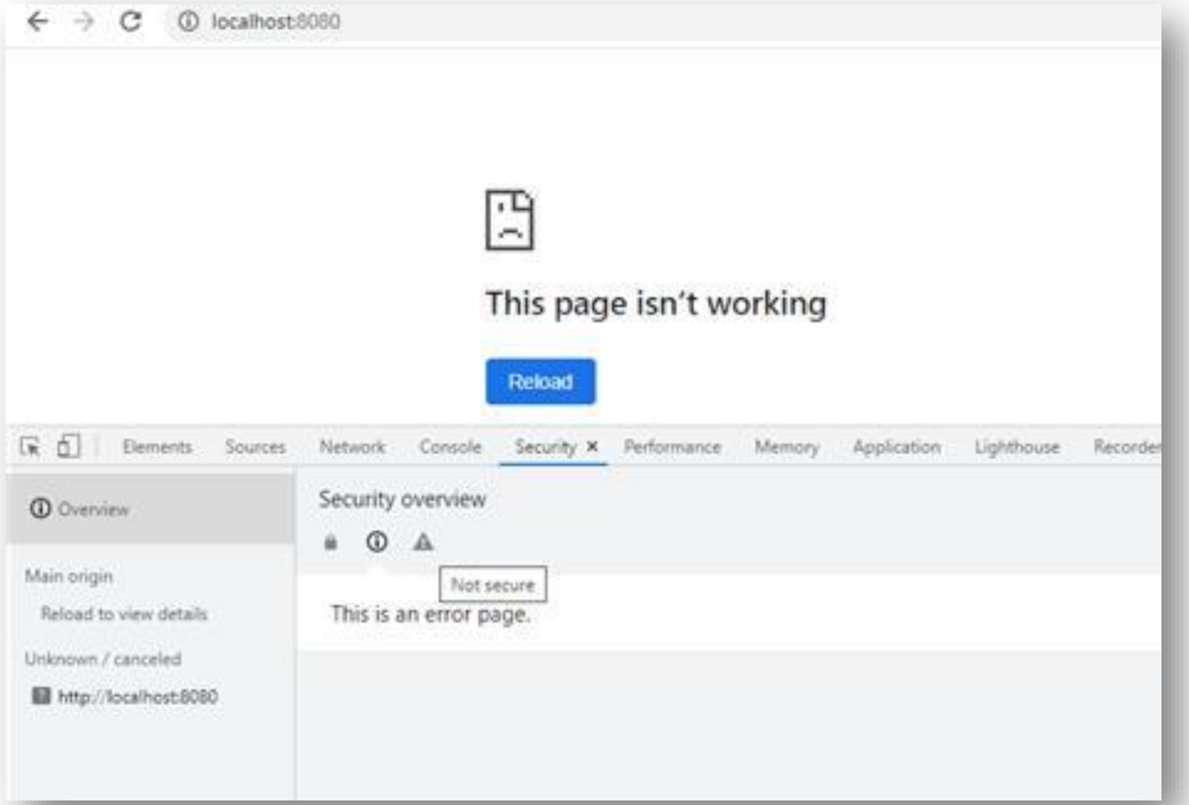
Output:



```
localhost:8080

First Node.js program!
```

- To create web server using http module

Output:


localhost:8080

This page isn't working

Security overview

Not secure

This is an error page.

Main origin

Reload to view details

Unknown / canceled

http://localhost:8080

1.Since, local host 8080 is not secure, output cannot be displayed.
2.Needs an SSL certificate to fix the https issue.

5.2. Https Module

- To make Node.js act as an HTTPS server

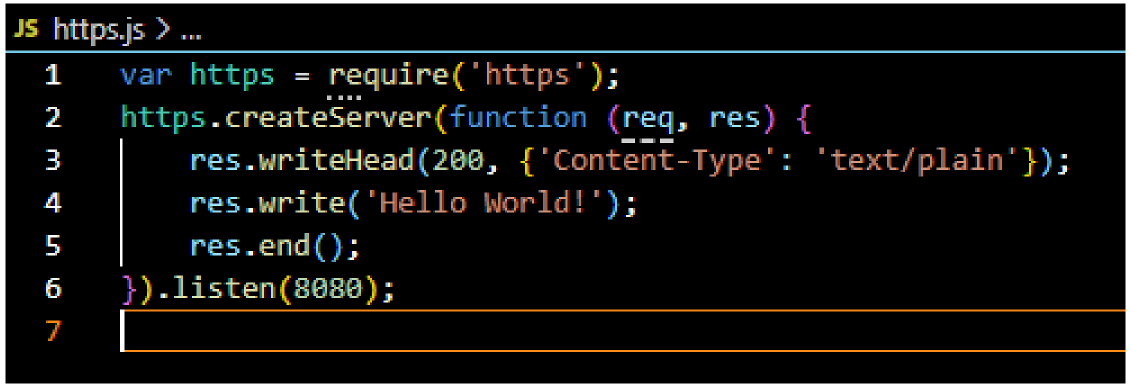
Syntax

Var name https is used and require is a function to import the https module.

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

createServer()

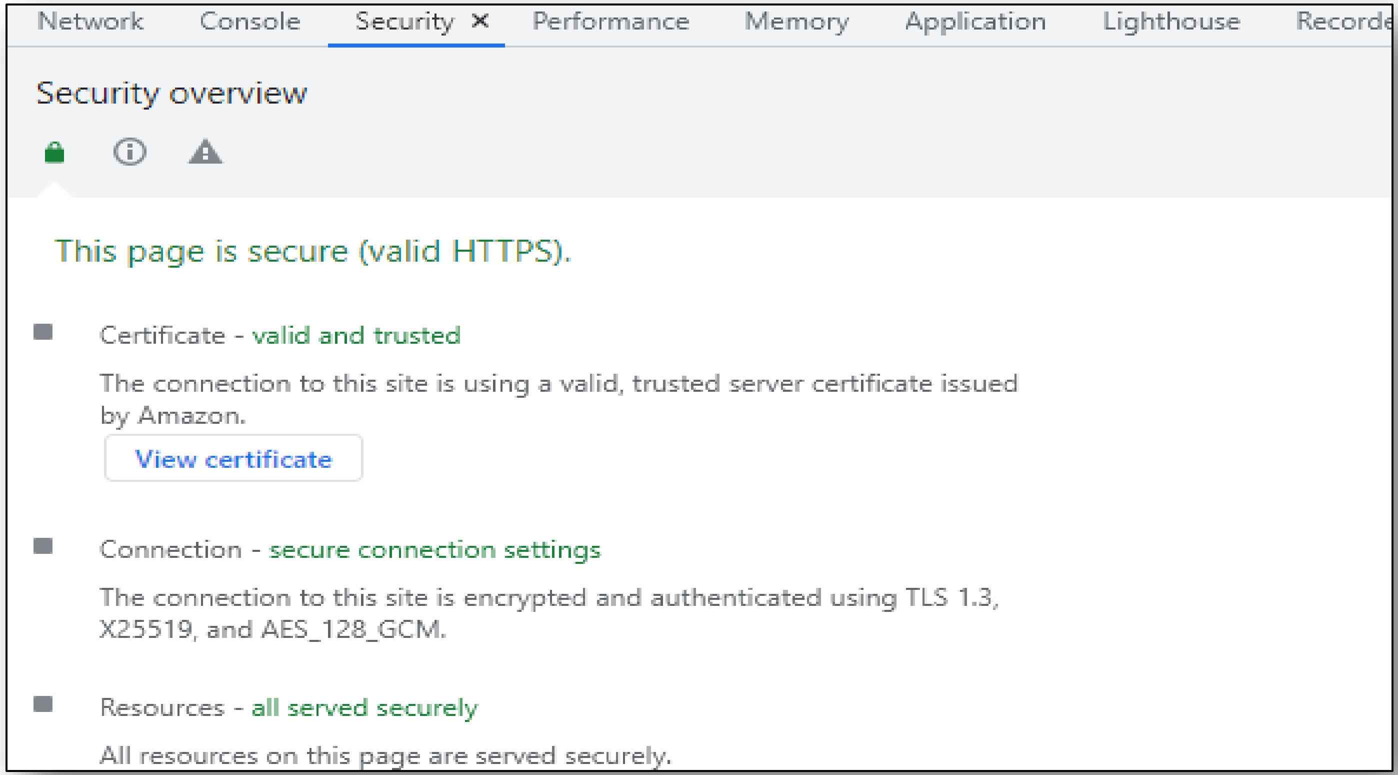
The create Server method creates a server on local system and runs in secure protocol



https.js > ...

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

Application that runs in HTTPS: (For your reference)



Security overview

This page is secure (valid HTTPS).

- Certificate - valid and trusted

  The connection to this site is using a valid, trusted server certificate issued by Amazon.

- Connection - secure connection settings

  The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.

- Resources - all served securely

  All resources on this page are served securely.

5.3. Assert ()

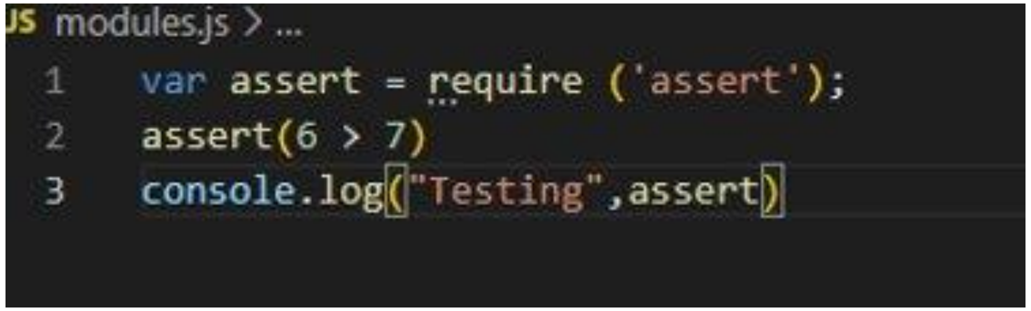
- Assert is used to check the expression in programs.
- Check whether the program is good or not
- How data is passing to request
- It is used to ensure whether the written program based on a logic is working good or not.
- Assert is mainly used for testing the code
- Assert modules contains some methods to use for testing

➢Assert ()

- ➢Using the assert() modules, you may test expressions..
- ➢Whenever expression fails. We'll terminate the program.
- ➢Examine the program and report any errors.

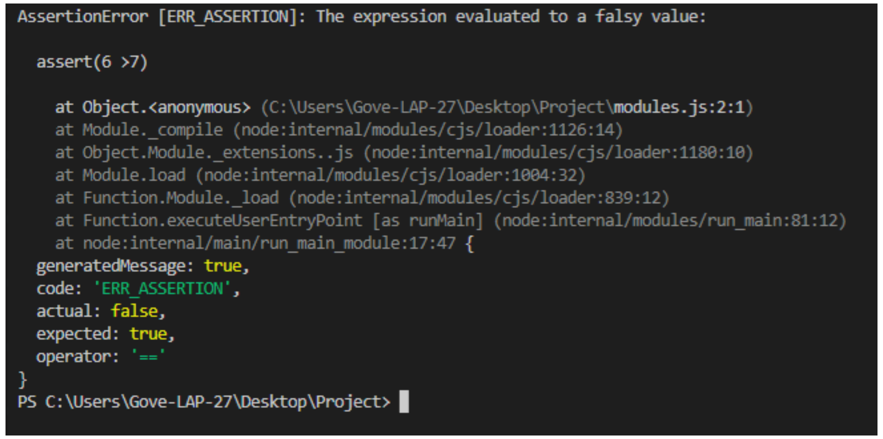
For example :

- ➢improper expression program
- ➢It fails because 6 is not more than 7
- ➢Assert () check these expressions, and then return some content to the terminal



JS modules.js > ...

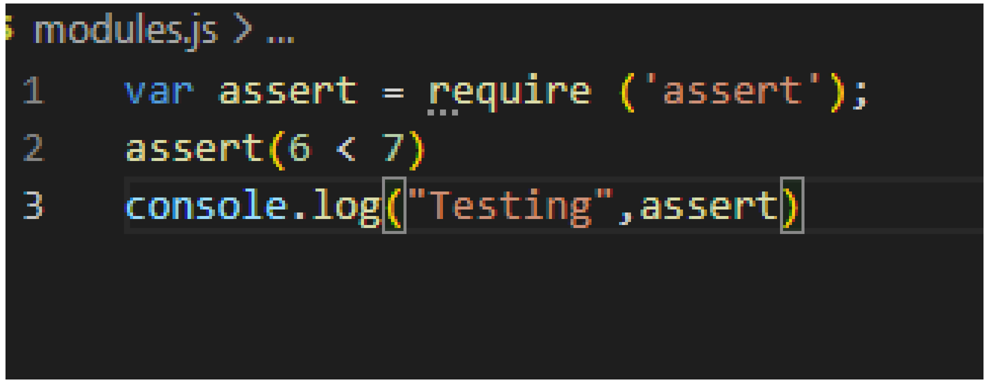
```
var assert = require ('assert');
assert(6 > 7)
console.log("Testing",assert)
```



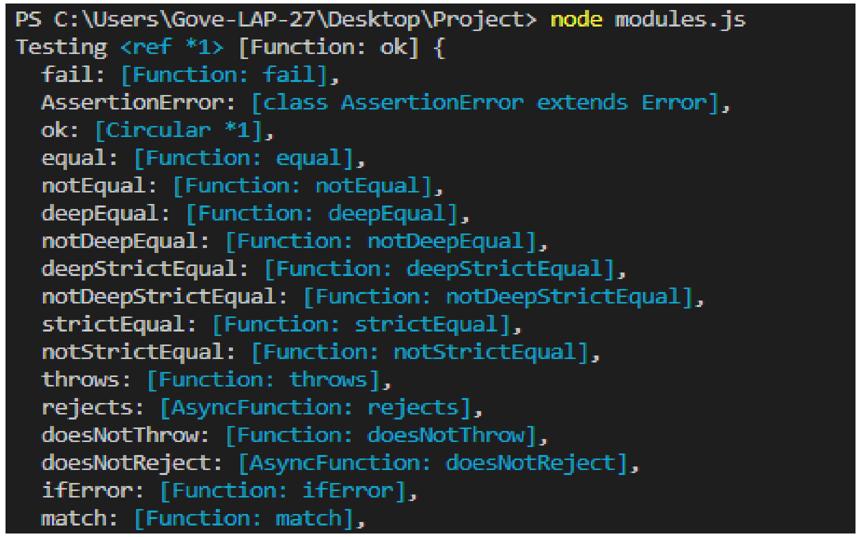
```
AssertionError [ERR_ASSERTION]: The expression evaluated to a falsy value:

  assert(6 >7)

    at Object.<anonymous> (C:\Users\Gove-LAP-27\Desktop\Project\modules.js:2:1)
    at Module._compile (node:internal/modules/cjs/loader:1126:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1180:10)
    at Module.load (node:internal/modules/cjs/loader:1004:32)
    at Function.Module._load (node:internal/modules/cjs/loader:839:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47 {
  generatedMessage: true,
  code: 'ERR_ASSERTION',
  actual: false,
  expected: true,
  operator: '=='
}
PS C:\Users\Gove-LAP-27\Desktop\Project>
```

Correct Expression

- Program expression is correct it return ok
- Checks if a value is true. Same as assert .ok()



```
modules.js > ...
var assert = require ('assert');
assert(6 < 7)
console.log("Testing",assert)
```



```
PS C:\Users\Gove-LAP-27\Desktop\Project> node modules.js
Testing <ref *1> [Function: ok] {
  fail: [Function: fail],
  AssertionError: [class AssertionError extends Error],
  ok: [Circular *1],
  equal: [Function: equal],
  notEqual: [Function: notEqual],
  deepEqual: [Function: deepEqual],
  notDeepEqual: [Function: notDeepEqual],
  deepStrictEqual: [Function: deepStrictEqual],
  notDeepStrictEqual: [Function: notDeepStrictEqual],
  strictEqual: [Function: strictEqual],
  notStrictEqual: [Function: notStrictEqual],
  throws: [Function: throws],
  rejects: [AsyncFunction: rejects],
  doesNotThrow: [Function: doesNotThrow],
  doesNotReject: [AsyncFunction: doesNotReject],
  ifError: [Function: ifError],
  match: [Function: match],
```

5.3.1. Assert - deepEqual() Method

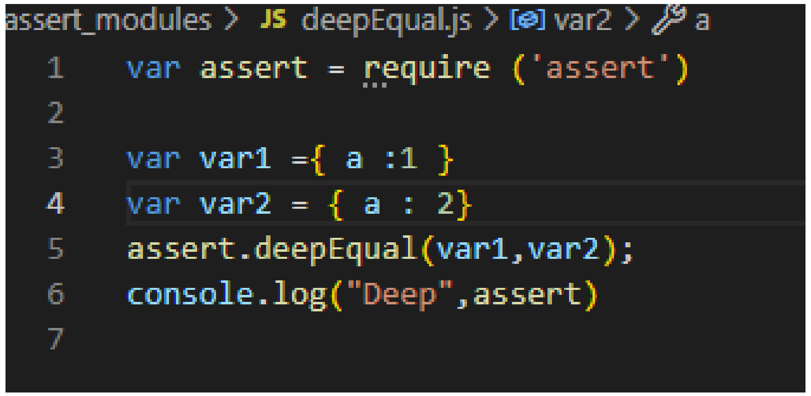
- whether an expression is correct, use assert.
- deepEqual is one of the method in Assert module
- deepEqual is check two variable objects
- This Method check only the values

e.g. : incorrect program using object

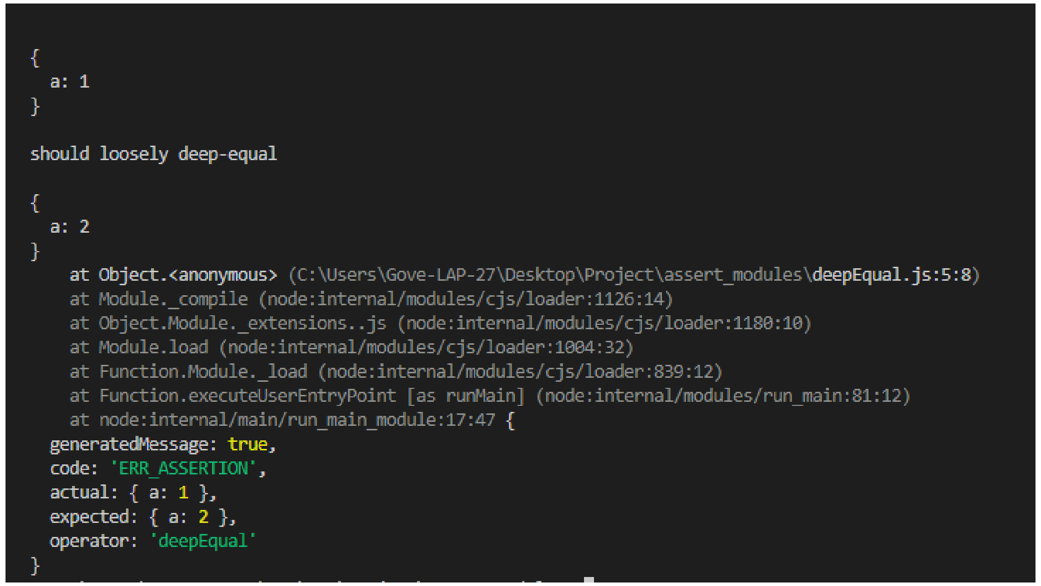
Stored the different object in variables .

Check the variable using deepEqual ()

Returns the error



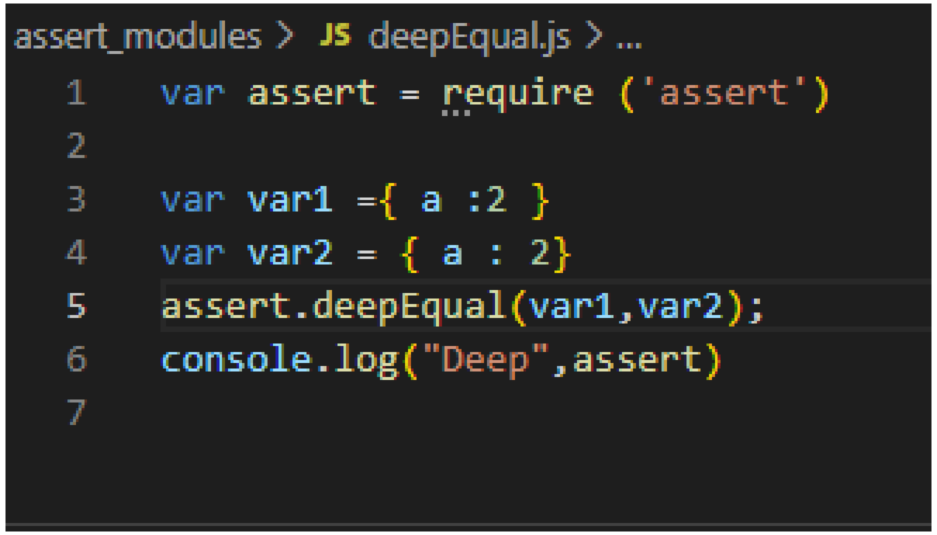
```
assert_modules > JS deepEqual.js > var2 > a
var assert = require ('assert')

var var1 ={ a :1 }
var var2 = { a : 2}
assert.deepEqual(var1,var2);
console.log("Deep",assert)

```



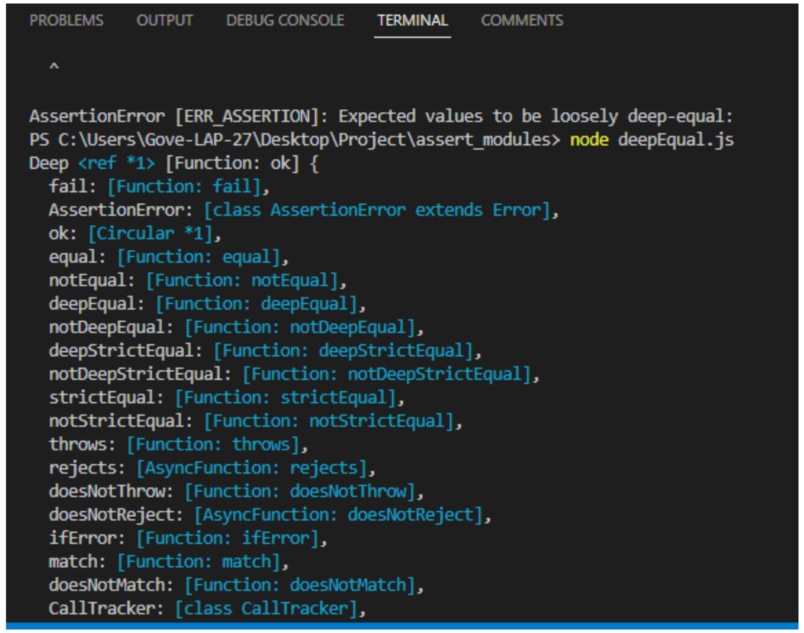
```
{
  a: 1
}

should loosely deep-equal

{
  a: 2
}
    at Object.<anonymous> (C:\Users\Gove-LAP-27\Desktop\Project\assert_modules\deepEqual.js:5:8)
    at Module._compile (node:internal/modules/cjs/loader:1126:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1180:10)
    at Module.load (node:internal/modules/cjs/loader:1004:32)
    at Function.Module._load (node:internal/modules/cjs/loader:839:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47 {
  generatedMessage: true,
  code: 'ERR_ASSERTION',
  actual: { a: 1 },
  expected: { a: 2 },
  operator: 'deepEqual'
}
```

DeepEqual()

- Checks if two values are equal



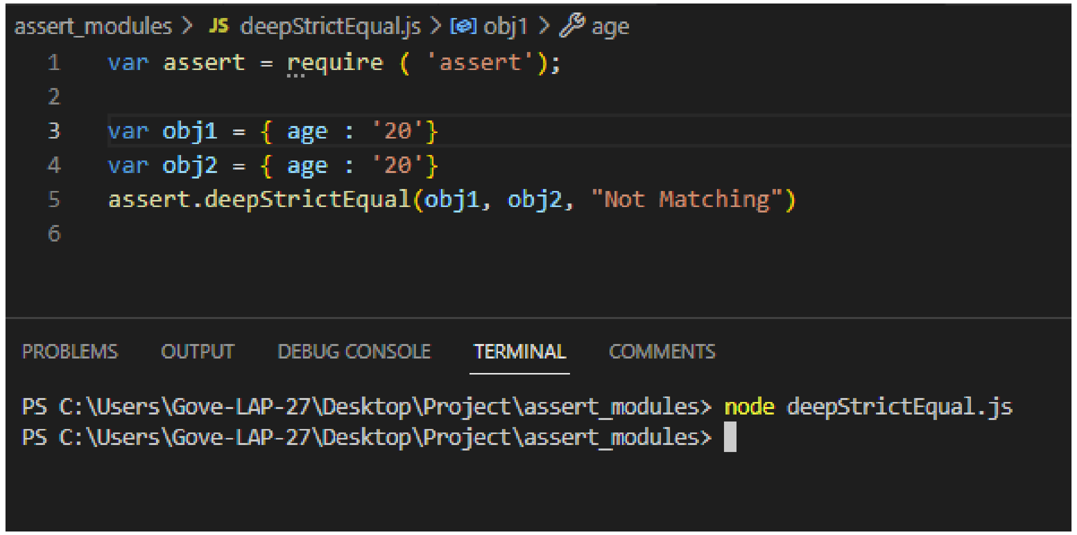
```js
var assert = require ('assert')

var var1 ={ a :2 }
var var2 = { a : 2}
assert.deepEqual(var1,var2);
console.log("Deep",assert)
```



```
  ^

AssertionError [ERR_ASSERTION]: Expected values to be loosely deep-equal:
PS C:\Users\Gove-LAP-27\Desktop\Project\assert_modules> node deepEqual.js
Deep <ref *1> [Function: ok] {
  fail: [Function: fail],
  AssertionError: [class AssertionError extends Error],
  ok: [Circular *1],
  equal: [Function: equal],
  notEqual: [Function: notEqual],
  deepEqual: [Function: deepEqual],
  notDeepEqual: [Function: notDeepEqual],
  deepStrictEqual: [Function: deepStrictEqual],
  notDeepStrictEqual: [Function: notDeepStrictEqual],
  strictEqual: [Function: strictEqual],
  notStrictEqual: [Function: notStrictEqual],
  throws: [Function: throws],
  rejects: [AsyncFunction: rejects],
  doesNotThrow: [Function: doesNotThrow],
  doesNotReject: [AsyncFunction: doesNotReject],
  ifError: [Function: ifError],
  match: [Function: match],
  doesNotMatch: [Function: doesNotMatch],
  CallTracker: [class CallTracker],
```

5.3.2 Assert - deepStrictEqual ()

- DeepStrictEqual is one method in assert module
- DeepStrictEqual is check the object values and data type
- DeepStrictEqual objects are stored in variables . Then check the expression using deepStrictEqual( )
- Variable object values and datatypes are equal . Program will not terminate
- It Execute the program

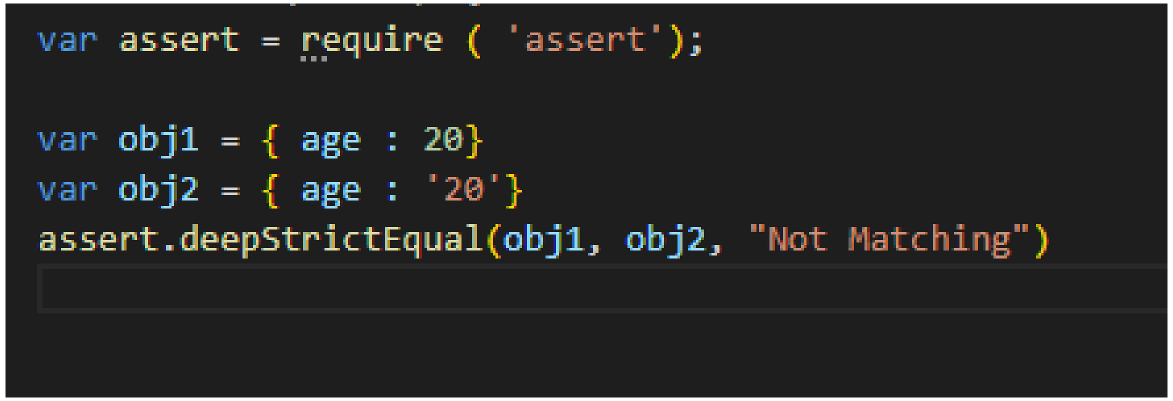


```javascript
var assert = require ( 'assert');

var obj1 = { age : '20'}
var obj2 = { age : '20'}
assert.deepStrictEqual(obj1, obj2, "Not Matching")
```

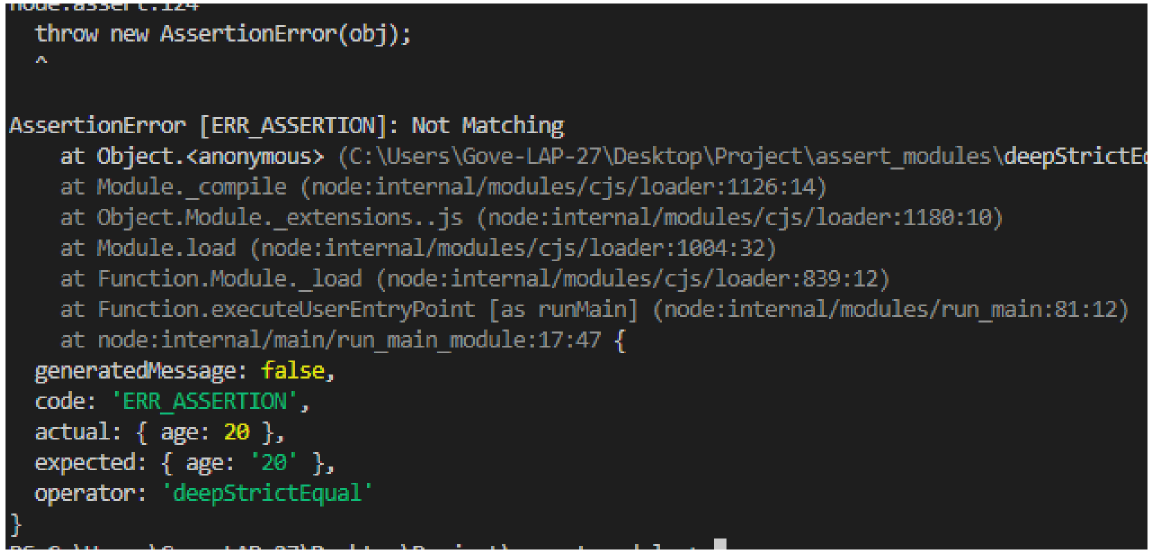
```
PS C:\Users\Gove-LAP-27\Desktop\Project\assert_modules> node deepStrictEqual.js
PS C:\Users\Gove-LAP-27\Desktop\Project\assert_modules>
```

Incorrect

Data Type and values are not matching



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

var obj1 = { age : 20}
var obj2 = { age : '20'}
assert.deepStrictEqual(obj1, obj2, "Not Matching")
```



```
node:assert:124
  throw new AssertionError(obj);
  ^

AssertionError [ERR_ASSERTION]: Not Matching
    at Object.<anonymous> (C:\Users\Gove-LAP-27\Desktop\Project\assert_modules\deepStrictE
    at Module._compile (node:internal/modules/cjs/loader:1126:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1180:10)
    at Module.load (node:internal/modules/cjs/loader:1004:32)
    at Function.Module._load (node:internal/modules/cjs/loader:839:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47 {
  generatedMessage: false,
  code: 'ERR_ASSERTION',
  actual: { age: 20 },
  expected: { age: '20' },
  operator: 'deepStrictEqual'
}
```

5.4. Fs() - Module

- ❑fs Modules is a file system ,it handles the files .
- ❑file system module allows you to work with the file system on your computer
- ❑file system contains some methods to use.
- ❑Working with files is as common for development purposes
- ❑In programmatically manipulate files with the built-in fs module.

Uses of fs module

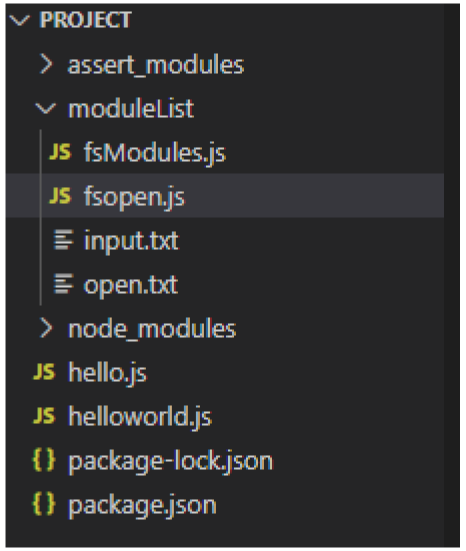
- ❖ Read files
- ❖ Create files
- ❖ Update files
- ❖ Delete files
- ❖ Rename files

5.4.1.Fs.open()

Fs.open()- File Open

If Open the file in write mode, The file is created if doesn't exist.

Existing File it will be read the input file



```
PROJECT
  > assert_modules
  v moduleList
      fsModules.js
      fsopen.js
      input.txt
      open.txt
  > node_modules
    hello.js
    helloworld.js
    package-lock.json
    package.json
```

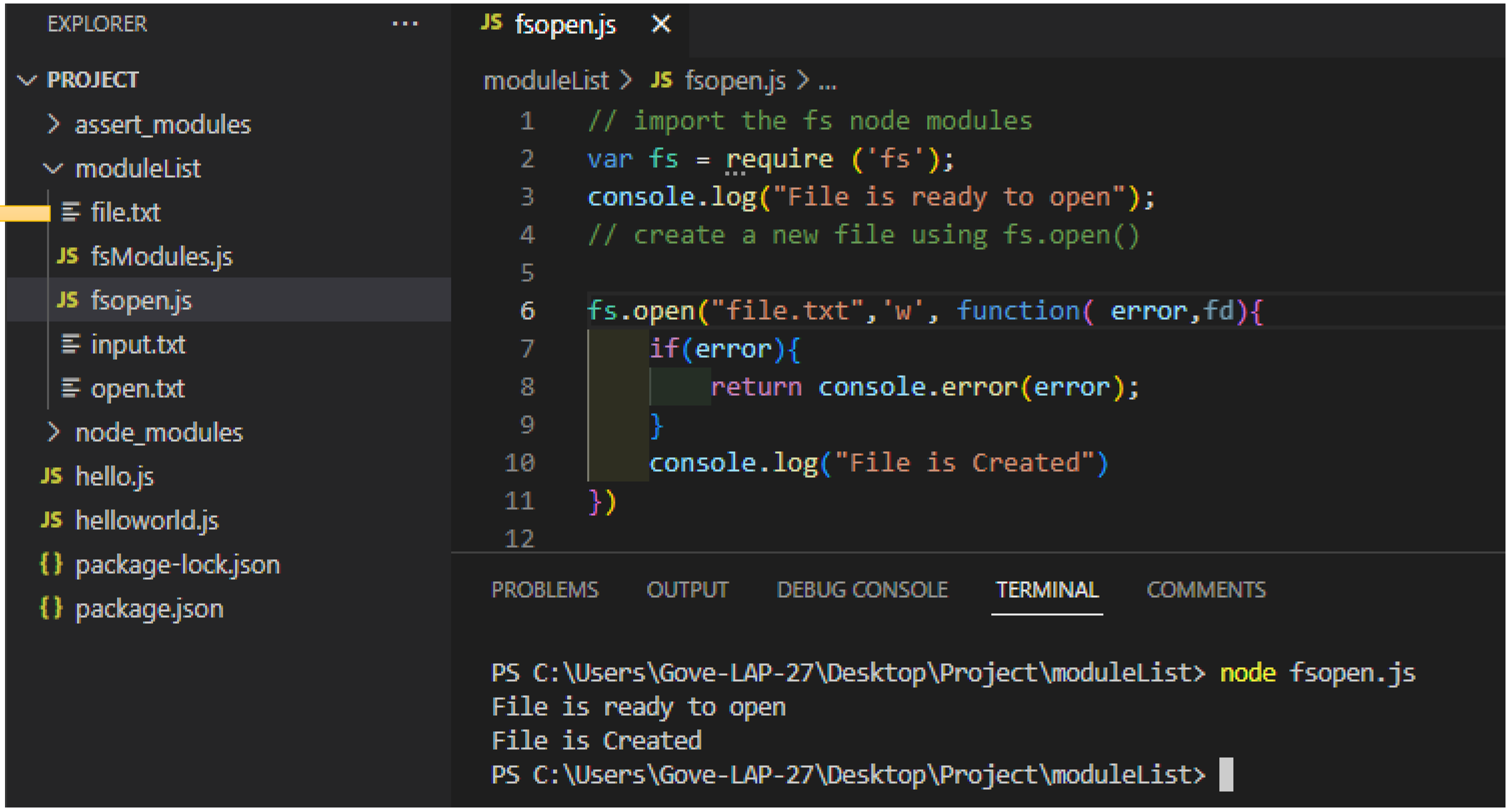
File is created

This approach looks at the mention filename first.

If the file name doesn't already exist, it will be created.

File already exists in folder; try to open

empty file created

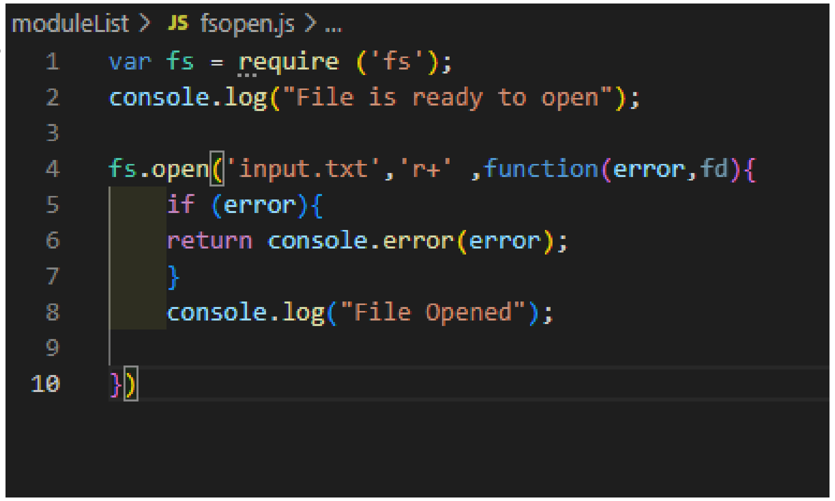


```javascript
// import the fs node modules
var fs = require ('fs');
console.log("File is ready to open");
// create a new file using fs.open()

fs.open("file.txt",'w', function( error,fd){
    if(error){
        return console.error(error);
    }
    console.log("File is Created")
})
```

```
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList> node fsopen.js
File is ready to open
File is Created
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList>
```

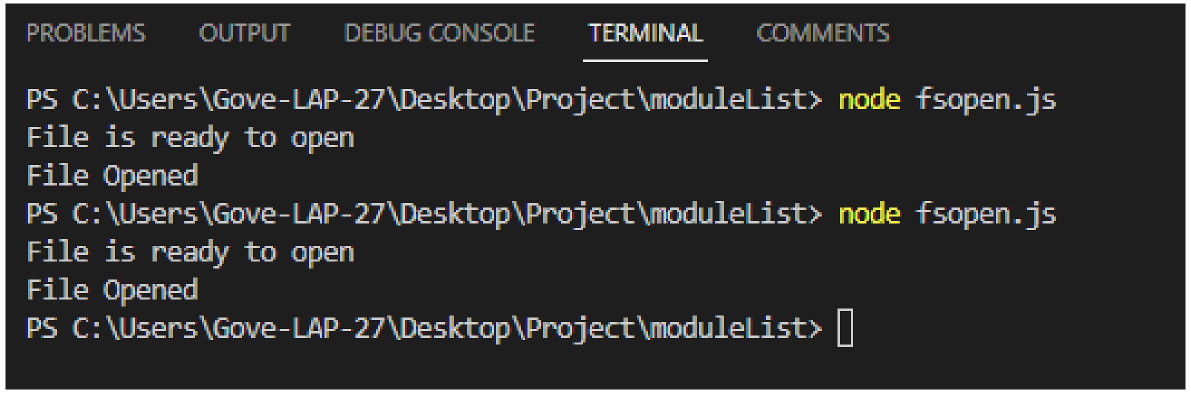
Fs.open() - fs modules

- Fs.open () means open the file.
- Some parameters are fixed to open like path(filename, flag ,callback)
- Path --------> filename
- Flag –-------> mode(like read or write)
-



moduleList > JS fsopen.js > ...

```
var fs = require ('fs');
console.log("File is ready to open");

fs.open('input.txt','r+' ,function(error,fd){
    if (error){
    return console.error(error);
    }
    console.log("File Opened");

})
```

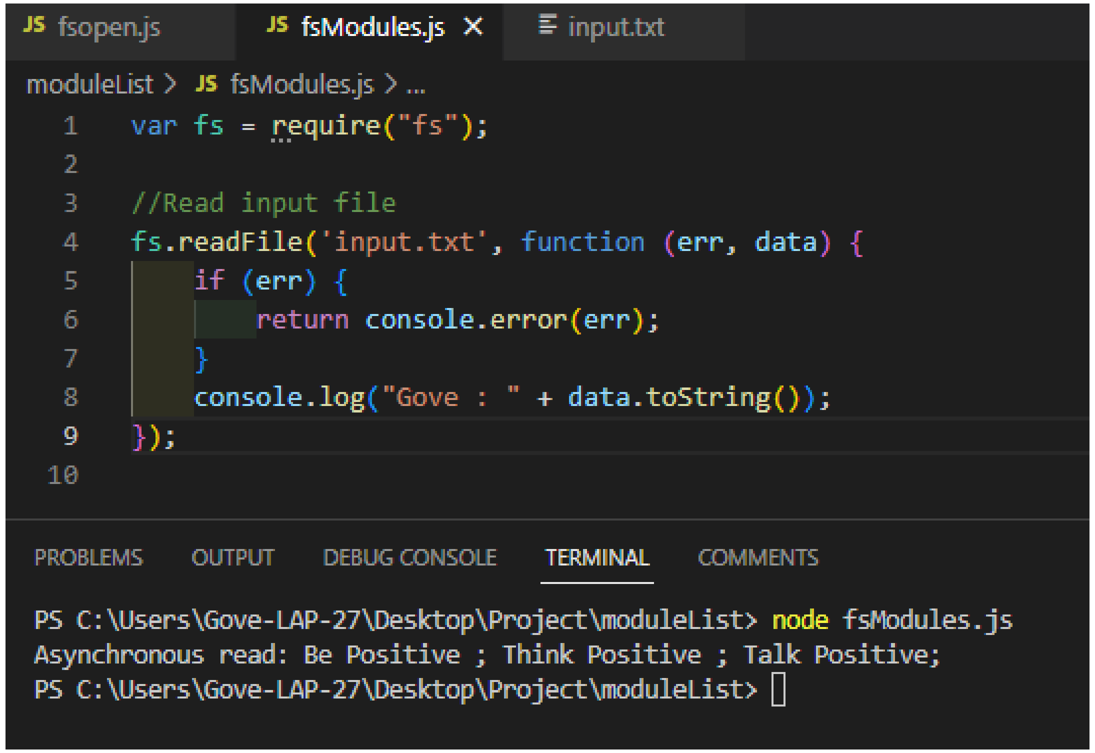


```
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList> node fsopen.js
File is ready to open
File Opened
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList> node fsopen.js
File is ready to open
File Opened
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList> 
```

Fs.read() - File Read

fs.readFile() method is used to read files on your computer

'Input.txt' is input file with some content



fsopen.js | fsModules.js | input.txt

moduleList > fsModules.js > ...

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

//Read input file
fs.readFile('input.txt', function (err, data) {
    if (err) {
        return console.error(err);
    }
    console.log("Gove : " + data.toString());
});
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | COMMENTS

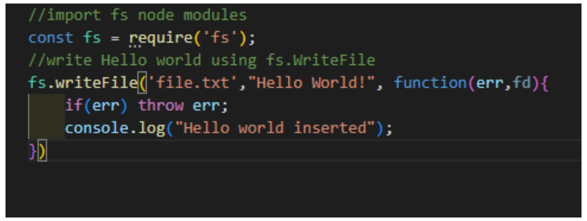
```
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList> node fsModules.js
Asynchronous read: Be Positive ; Think Positive ; Talk Positive;
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList>
```

5.4.2.Fs.write()

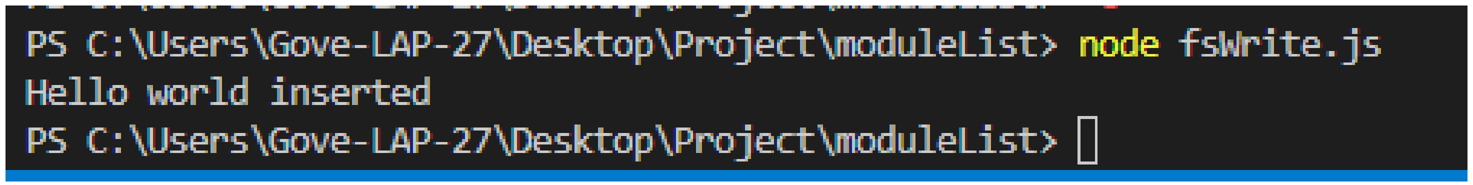
Fs.writeFile () method replaces the specified file and content if it exists.

If the file does not exist, a new file, containing the specified content, will be created:

"Request Content is inserted into file.js



```
//import fs node modules
const fs = require('fs');
//write Hello world using fs.WriteFile
fs.writeFile('file.txt',"Hello World!", function(err,fd){
    if(err) throw err;
    console.log("Hello world inserted");
})
```



```
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList> node fsWrite.js
Hello world inserted
PS C:\Users\Gove-LAP-27\Desktop\Project\moduleList> 
```

Recap

- Modules - Functionality organized in single or multiple files, can be reused throughout the Node JS application.
- HTTP Module - To create web server
- HTTPS Module – To create web server as secure
- Assert () is used to check the expression in programs.
- deepEqual () is check two variable objects
- DeepStrictEqual () is check the object values and data type
- fs Modules is a file system ,it handles the files .
- Fs.open()- File Open
- Fs.read() - File Read
- Fs.writeFile () method replaces the specified file and content if it exists

Think !

15. Write the uses of fs Module:

16. what are all the parameters is used for opening the file in the file module?

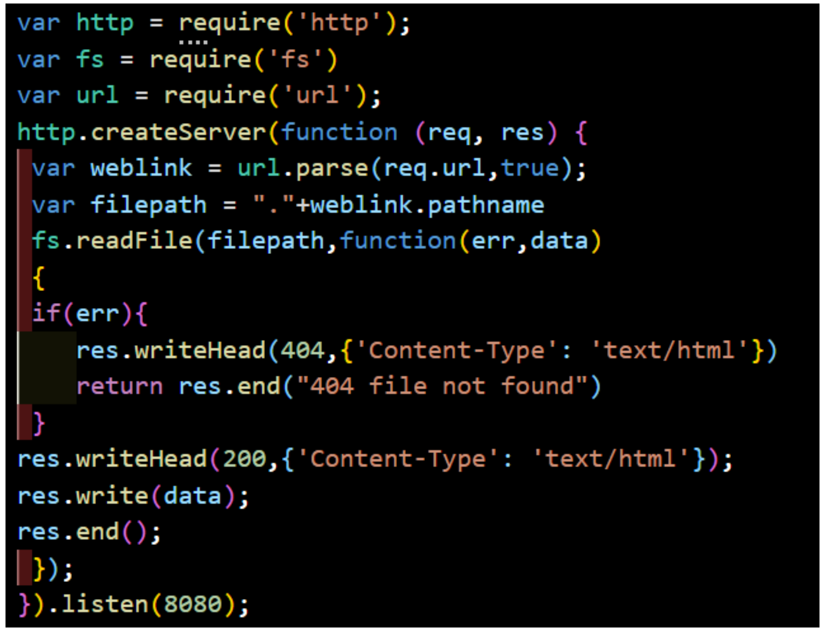
5.5.URL - Module

- The URL module provides a way of parsing the URL string and extract the href property.
- URL modules split up web address into readable parts.
- Parse an address with url.parse() method and it will split a URL object with each part of address as properties.
- Basically, the local web server for all systems is localhost:8080.
- Web server usually respond with html document which includes images, style sheet and scripts.
- syntax for including the url module is var url = require('url');

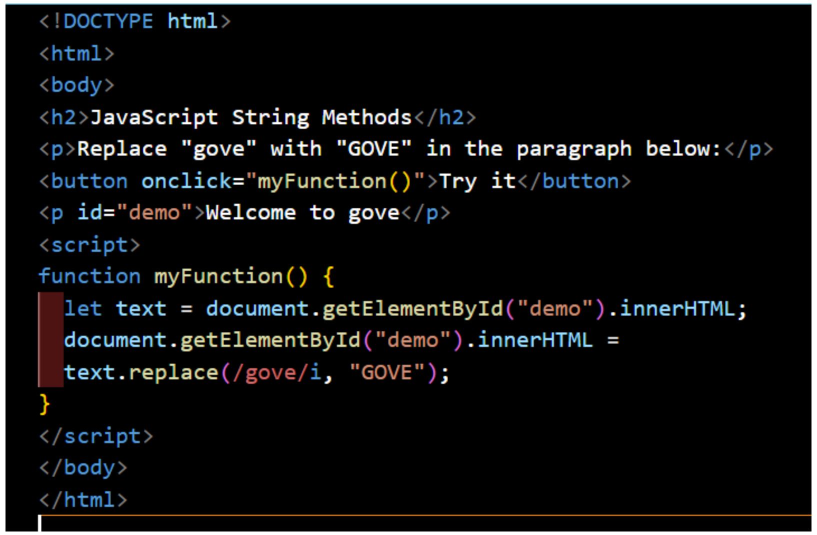
URL Methods

Method	Description
url.format()	Returns a formatted URL string
url.parse()	Returns a URL object
url.resolve()	Resolves a URL

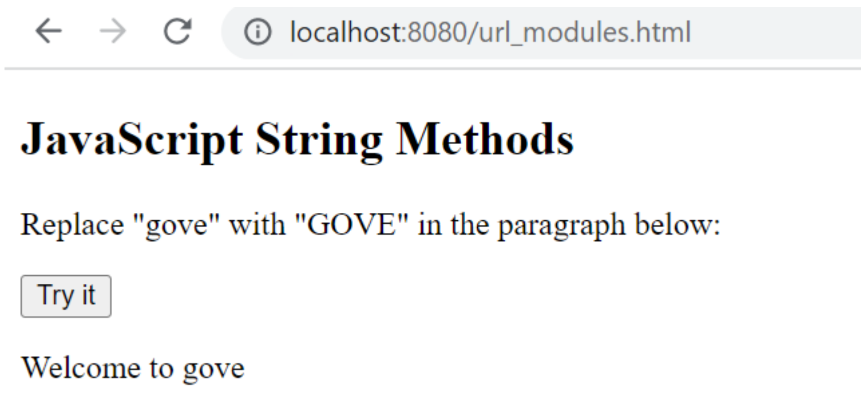
Sample Node.js URL module code using url.parser(). Which will render the url_module.html as html document html in the localhost:8080 port



```
var http = require('http');
var fs = require('fs')
var url = require('url');
http.createServer(function (req, res) {
 var weblink = url.parse(req.url,true);
 var filepath = "."+weblink.pathname
 fs.readFile(filepath,function(err,data)
 {
 if(err){
     res.writeHead(404,{'Content-Type': 'text/html'})
     return res.end("404 file not found")
 }
res.writeHead(200,{'Content-Type': 'text/html'});
res.write(data);
res.end();
 });
}).listen(8080);
```



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript String Methods</h2>
<p>Replace "gove" with "GOVE" in the paragraph below:</p>
<button onclick="myFunction()">Try it</button>
<p id="demo">Welcome to gove</p>
<script>
function myFunction() {
  let text = document.getElementById("demo").innerHTML;
  document.getElementById("demo").innerHTML =
  text.replace(/gove/i, "GOVE");
}
</script>
</body>
</html>
```



localhost:8080/url_modules.html

JavaScript String Methods

Replace "gove" with "GOVE" in the paragraph below:

Try it

Welcome to gove

Exercise 3:

Topic: Modules (HTTP, fs & URL)

1. Create a HTML File using simple tags

2. Read a created html file using fs module

3. Open the Created HTML file and append URL Information as an object using URL Module.

4. Send the html data to the browser when access the url using http module.

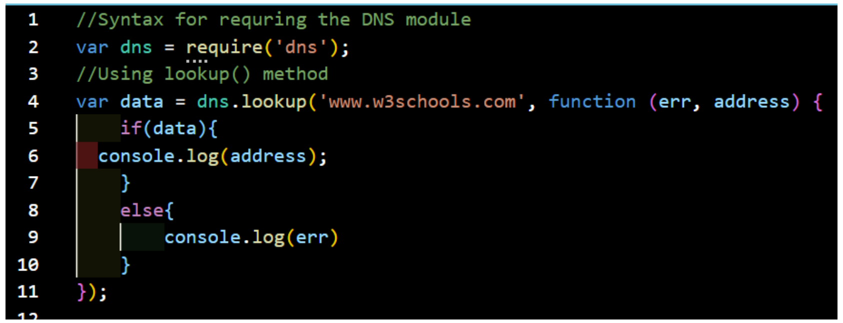
5.6.DNS Module

- ❖ DNS, or the Domain Name System, translates human readable domain names (for example, www.gove.co) to machine readable IP addresses (for example, 192.0.2.44).
- ❖ The Node.js DNS module contains methods to get information of given hostname.
- ❖ DNS modules is used to translate network name to addresses and addresses to network name.

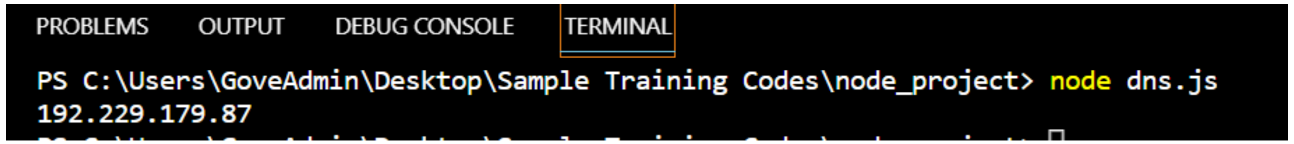
Sample DNS Module code

There are various methods() in DNS module for getting the host information.

lookup()-Commonly used method, it's a callback function containing information about the hostname,



```
//Syntax for requring the DNS module
var dns = require('dns');
//Using lookup() method
var data = dns.lookup('www.w3schools.com', function (err, address) {
    if(data){
  console.log(address);
    }
    else{
        console.log(err)
    }
});
```

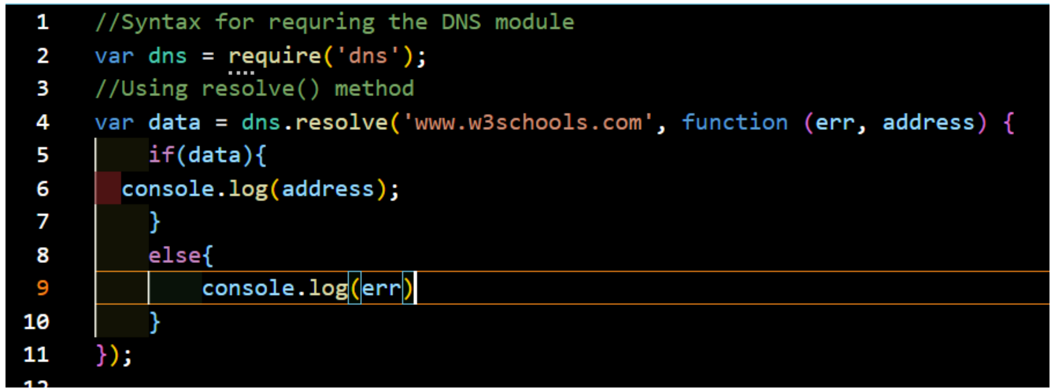


```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node dns.js
192.229.179.87
```

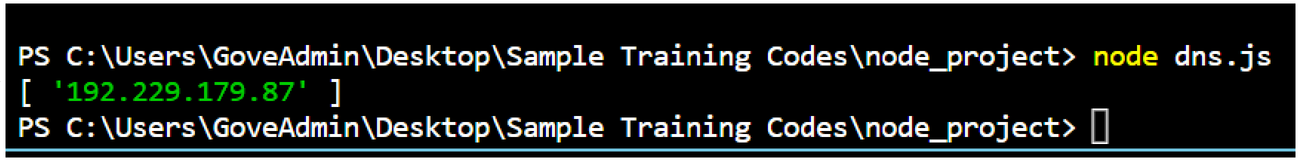
By using the DNS module lookup() method the IP address of the network name "www.w3schools.com" is printed on the console.

resolve()

This method returns an array of record types belonging to the specified hostname



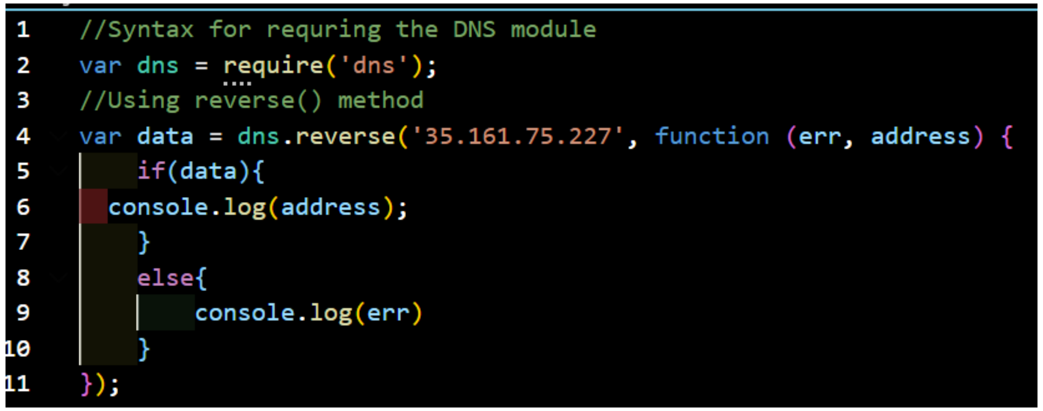
```
//Syntax for requring the DNS module
var dns = require('dns');
//Using resolve() method
var data = dns.resolve('www.w3schools.com', function (err, address) {
    if(data){
  console.log(address);
    }
    else{
        console.log(err)
    }
});
```



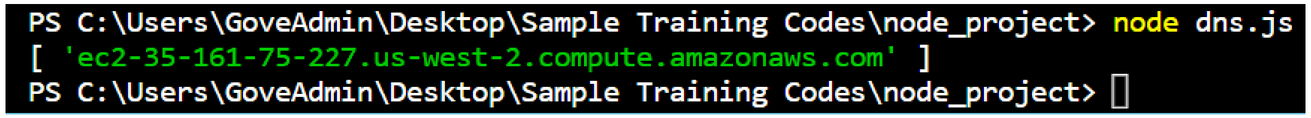
```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node dns.js
[ '192.229.179.87' ]
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> 
```

reverse()

Reverses an IP address into an array of hostnames



```
//Syntax for requring the DNS module
var dns = require('dns');
//Using reverse() method
var data = dns.reverse('35.161.75.227', function (err, address) {
    if(data){
  console.log(address);
    }
    else{
        console.log(err)
    }
});
```



```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node dns.js
[ 'ec2-35-161-75-227.us-west-2.compute.amazonaws.com' ]
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> 
```

5.7.Query String-Module

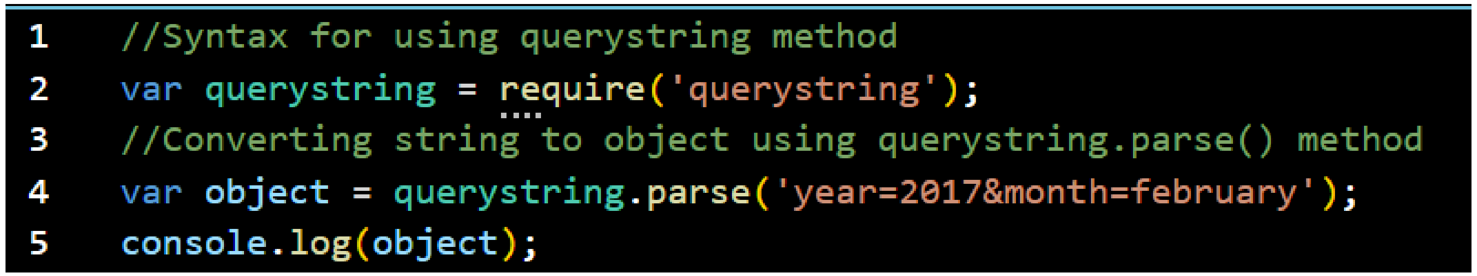
- Query string can be used to convert string to objects and vice versa.

Syntax for using query string

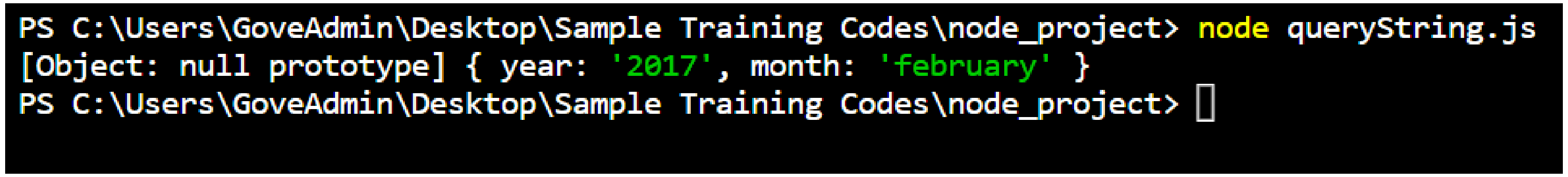
- var querystring = require('querystring');

Query String Methods

1.querystring.parse()- This method is used to convert string to object.



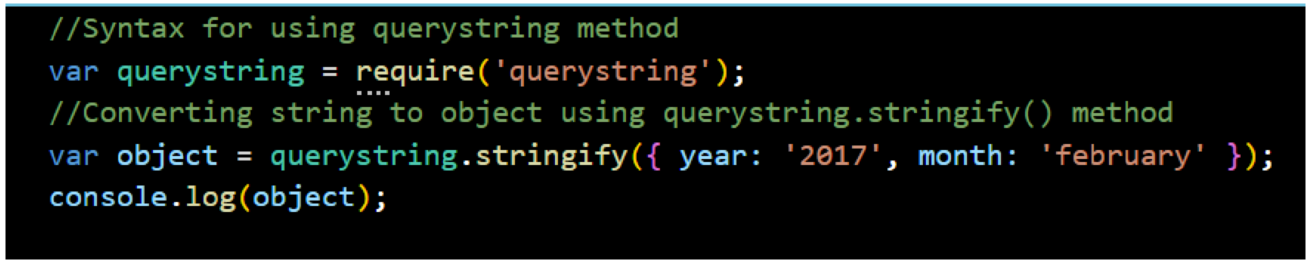
```
//Syntax for using querystring method
var querystring = require('querystring');
//Converting string to object using querystring.parse() method
var object = querystring.parse('year=2017&month=february');
console.log(object);
```



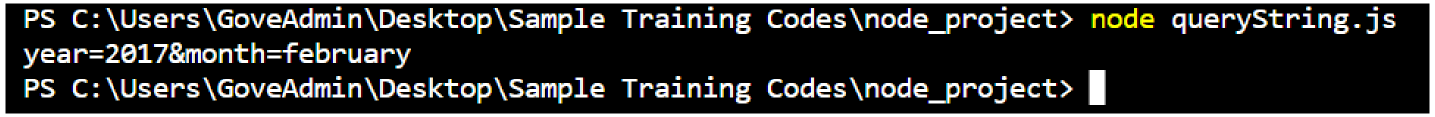
```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node queryString.js
[Object: null prototype] { year: '2017', month: 'february' }
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project>
```

2.querystring.stringify()

This method is used to convert object to string .



```
//Syntax for using querystring method
var querystring = require('querystring');
//Converting string to object using querystring.stringify() method
var object = querystring.stringify({ year: '2017', month: 'february' });
console.log(object);
```



```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node queryString.js
year=2017&month=february
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project>
```

5.8.Path Module

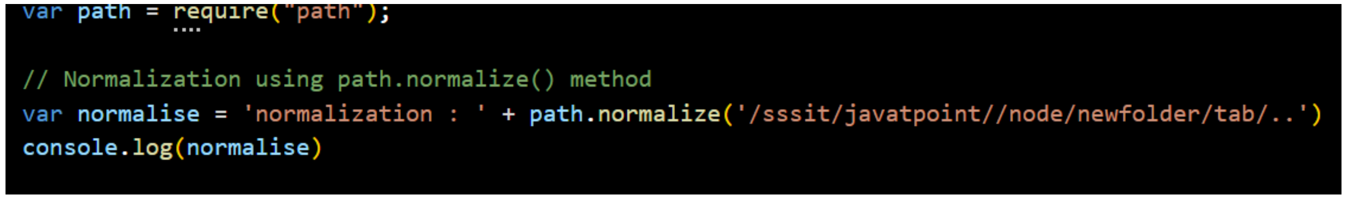
- The Node.js path module is used to handle and transform file paths.
- Path modules can be used to require paths, join paths and can get the extension of the file etc..

Syntax for using path module

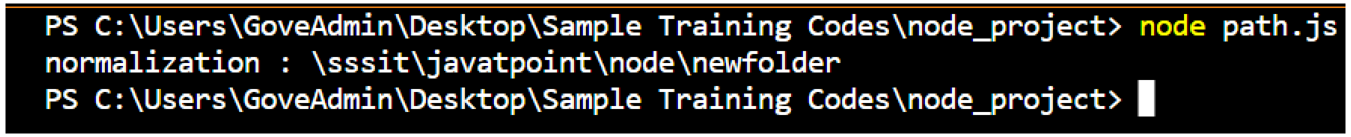
- var path = require ("path")

Path Methods.

5.8.1.path.normalize() - It is used to normalize a string path by using the '..' and '.' parts.



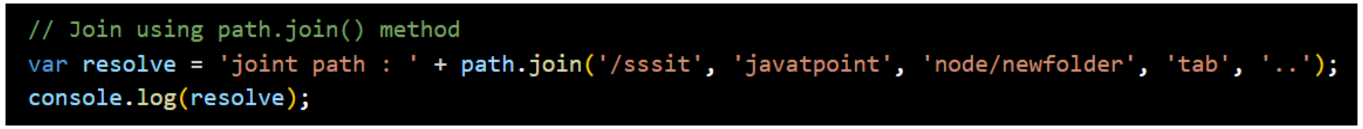
```
var path = require("path");

// Normalization using path.normalize() method
var normalise = 'normalization : ' + path.normalize('/sssit/javatpoint//node/newfolder/tab/..')
console.log(normalise)
```



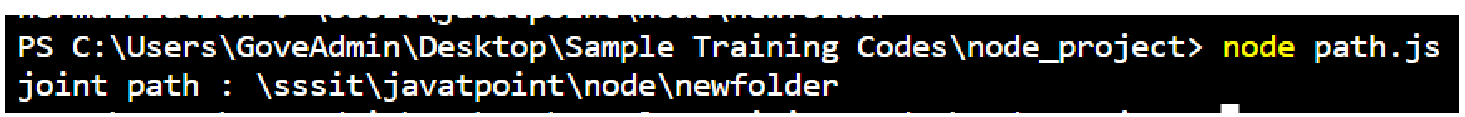
```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node path.js
normalization : \sssit\javatpoint\node\newfolder
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project>
```

5.8.2.path.join()

It is used to join all arguments together and normalize the resulting path.



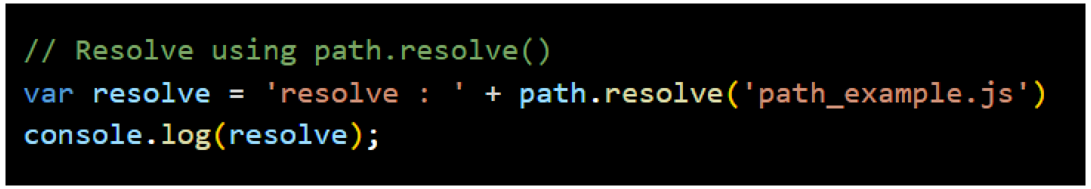
```
// Join using path.join() method
var resolve = 'joint path : ' + path.join('/sssit', 'javatpoint', 'node/newfolder', 'tab', '..');
console.log(resolve);
```



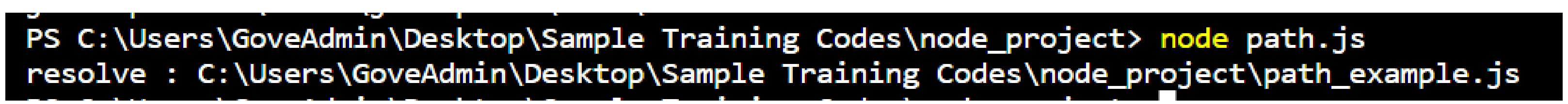
```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node path.js
joint path : \sssit\javatpoint\node\newfolder
```

5.8.3.path.replace()

This method is used to resolve an absolute path.



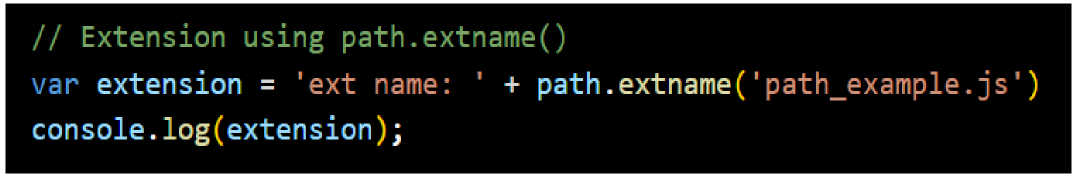
```
// Resolve using path.resolve()
var resolve = 'resolve : ' + path.resolve('path_example.js')
console.log(resolve);
```



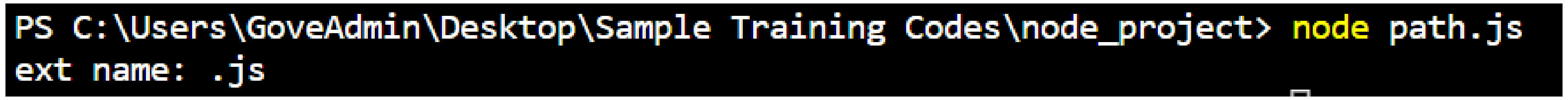
```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node path.js
resolve : C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project\path_example.js
```

5.8.4.path.extname()

It returns the extension of a file in specified path, from the last '.' to end of string in the last portion of the path. if there is no '.' in the last portion of the path or the first character of it is '.', then it returns an empty string.



```
// Extension using path.extname()
var extension = 'ext name: ' + path.extname('path_example.js')
console.log(extension);
```



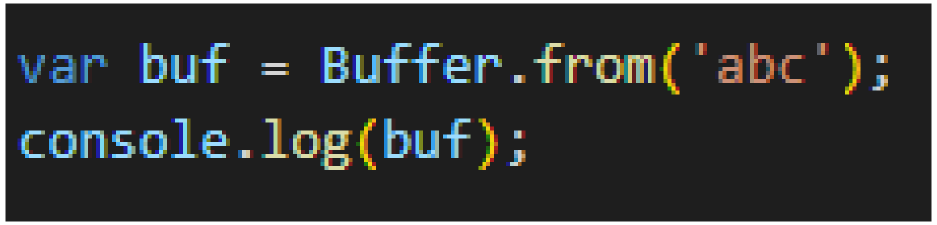
```
PS C:\Users\GoveAdmin\Desktop\Sample Training Codes\node_project> node path.js
ext name:  .js
```

Recap

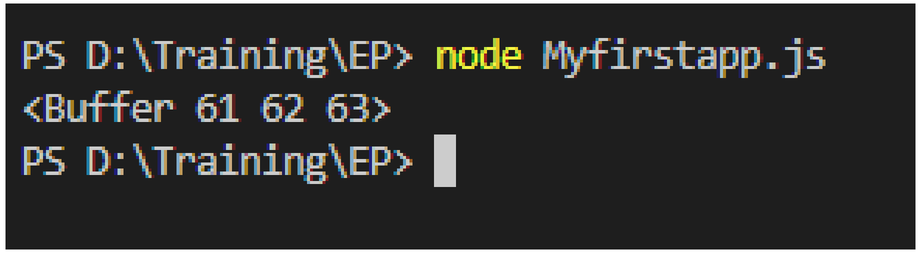
- URL modules split up web address into readable parts.
- DNS translates human readable domain names to machine readable IP addresses using **lookup()**.
    - resolve()
    - reverse()
- Query string can be used to convert string to objects
    - querystring.parse()
    - querystring.parse()
- Path modules can be used to require paths, join paths and can get the extension of the file
    - path.normalize()
    - path.join()
    - path.replace()
    - path.extname()

5.9.Buffer-Module

- Buffer module is used to convert strings to binary format are passed in arrays for the computer to understand it easily.



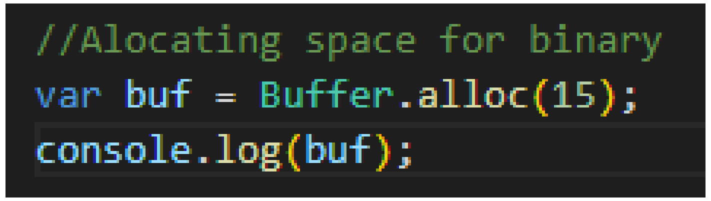
```
var buf = Buffer.from('abc');
console.log(buf);
```



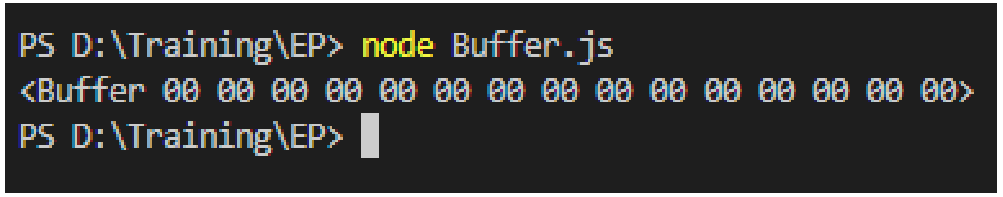
```
PS D:\Training\EP> node Myfirstapp.js
<Buffer 61 62 63>
PS D:\Training\EP>
```

5.9.1.alloc()

Creates a Buffer object of the specified length. By default, the value allocated will be 00. It can be changed other than 00 by allocating a value to it.

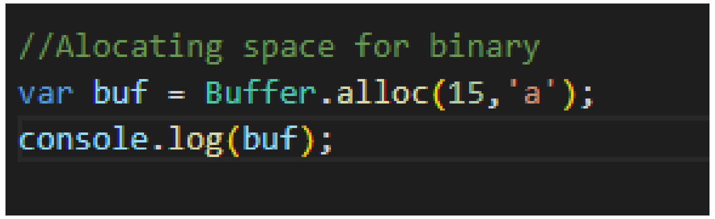


```
//Alocating space for binary
var buf = Buffer.alloc(15);
console.log(buf);
```

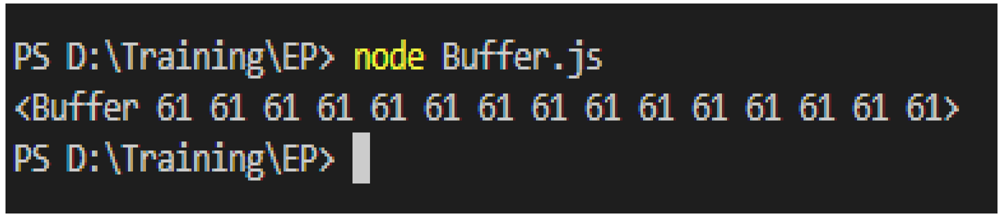


```
PS D:\Training\EP> node Buffer.js
<Buffer 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>
PS D:\Training\EP>
```

Default Value will be assigned as 00



```
//Alocating space for binary
var buf = Buffer.alloc(15,'a');
console.log(buf);
```

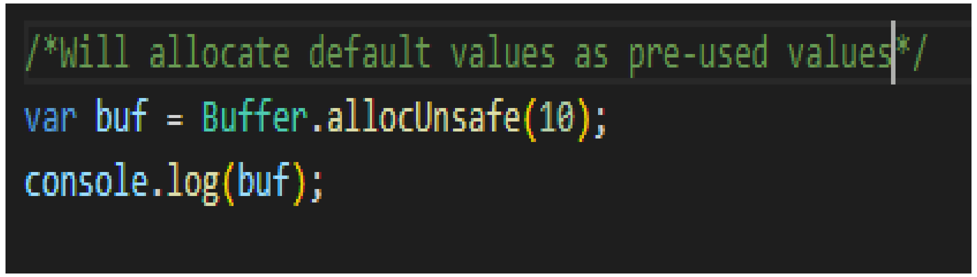


```
PS D:\Training\EP> node Buffer.js
<Buffer 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61>
PS D:\Training\EP>
```

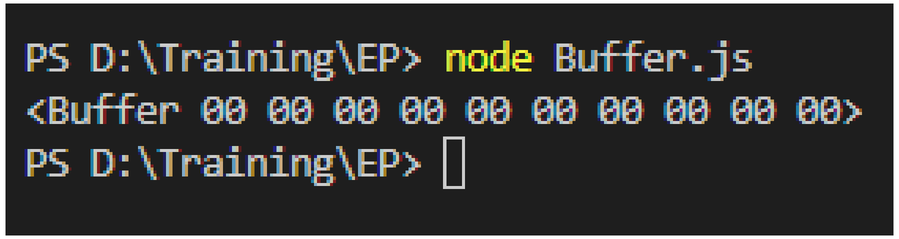
Default Value can changed as a - 61

5.9.2.allocUnsafe()

- Creates a non-zero-filled Buffer of the specified length. The value of an unsafe buffer is not emptied, and can contain data from older buffers



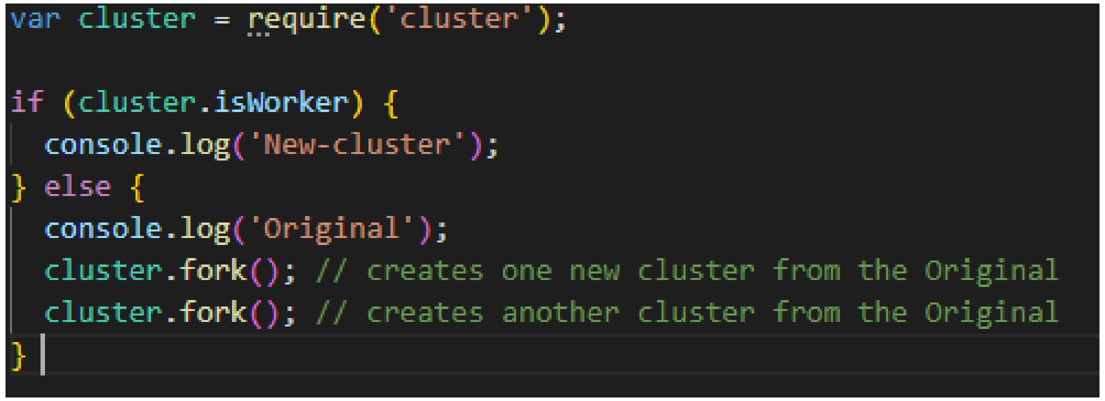
```
/*Will allocate default values as pre-used values*/
var buf = Buffer.allocUnsafe(10);
console.log(buf);
```



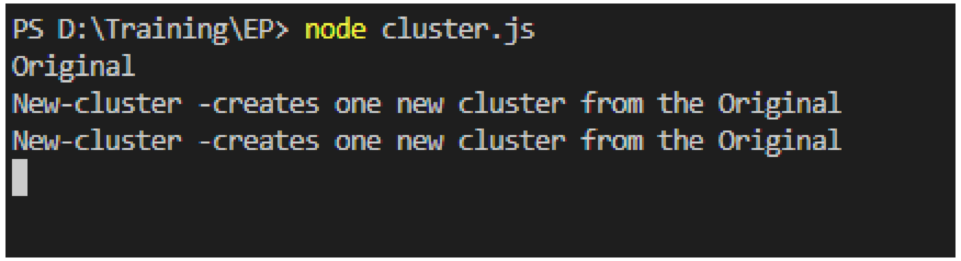
```
PS D:\Training\EP> node Buffer.js
<Buffer 00 00 00 00 00 00 00 00 00 00>
PS D:\Training\EP> 
```

5.10.Cluster Module:

- The cluster module provides a way of creating child processes that run simultaneously and share the same server port.
- Node JS run in single thread only , by using computers multi-core systems, cluster modules create child-process that runs on their own thread to balance the load .



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

if (cluster.isWorker) {
  console.log('New-cluster');
} else {
  console.log('Original');
  cluster.fork(); // creates one new cluster from the Original
  cluster.fork(); // creates another cluster from the Original
}
```



```
PS D:\Training\EP> node cluster.js
Original
New-cluster -creates one new cluster from the Original
New-cluster -creates one new cluster from the Original
```

5.11.Events Module

Event: Events are actions that happen in the system programming.

Call back Function: when task get completed ,get into other tasks like asynchronous.

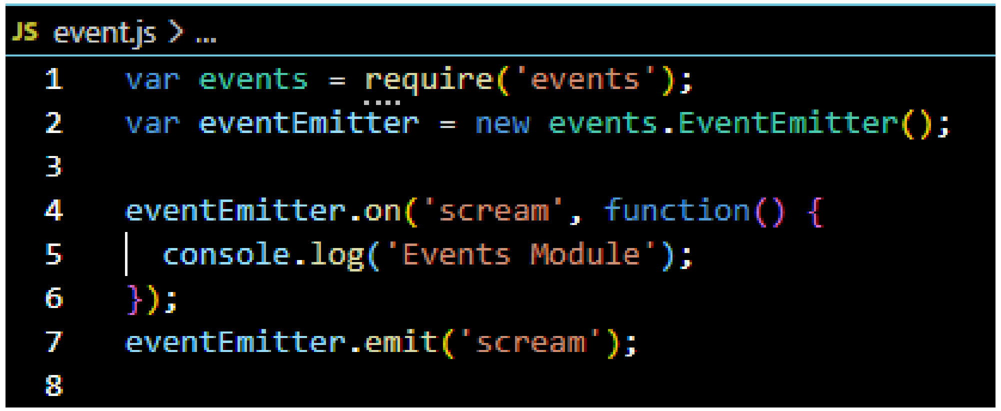
Event Listeners: The event listener code is a callback function that takes a parameter for the data and handles it.

Node.js uses events module to create and handle custom events.

Syntax:

var events = require('events');

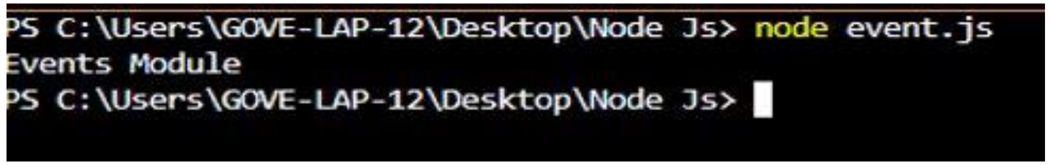
var eventEmitter = new events.EventEmitter();



```
event.js > ...
var events = require('events');
var eventEmitter = new events.EventEmitter();

eventEmitter.on('scream', function() {
    console.log('Events Module');
});
eventEmitter.emit('scream');

```

Using on Method: Adds the specified listener

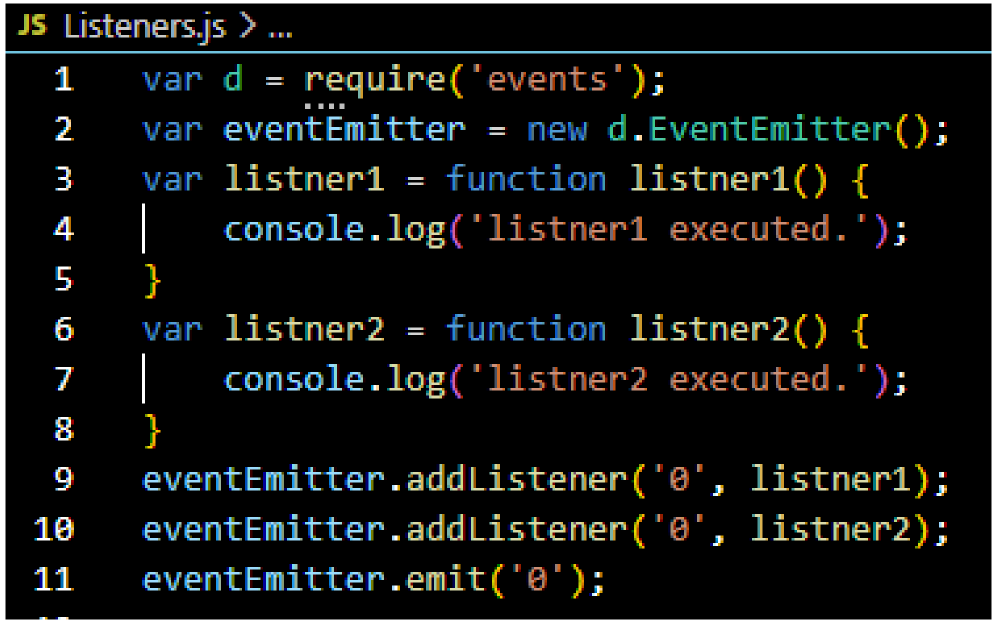
Output:



```
PS C:\Users\GOVE-LAP-12\Desktop\Node Js> node event.js
Events Module
PS C:\Users\GOVE-LAP-12\Desktop\Node Js> 
```

5.11.1.addListener() Method:

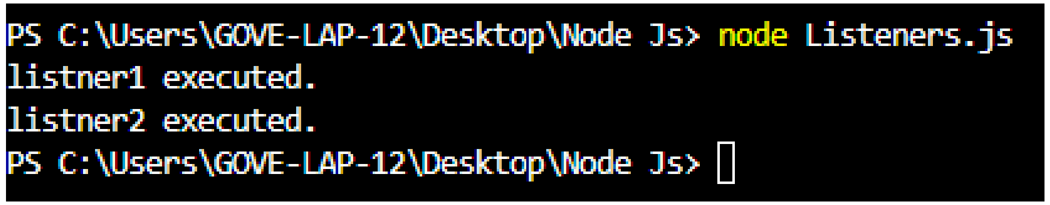
1. Add listener method listens to the event and display the output.
2. Two functions is created as listener1() & listener2()
3. addlistener method contains ('string', event name);



Listeners.js > ...

```
var d = require('events');
var eventEmitter = new d.EventEmitter();
var listner1 = function listner1() {
    console.log('listner1 executed.');
}
var listner2 = function listner2() {
    console.log('listner2 executed.');
}
eventEmitter.addListener('0', listner1);
eventEmitter.addListener('0', listner2);
eventEmitter.emit('0');
```

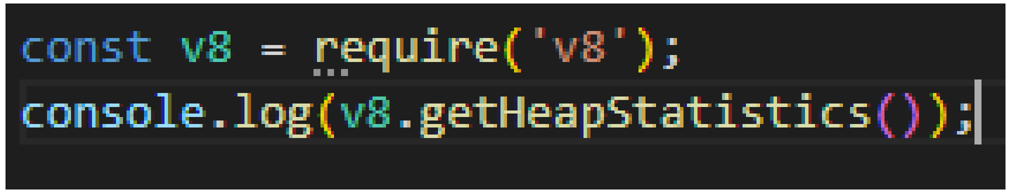
Output:



```
PS C:\Users\GOVE-LAP-12\Desktop\Node Js> node Listeners.js
listner1 executed.
listner2 executed.
PS C:\Users\GOVE-LAP-12\Desktop\Node Js> 
```

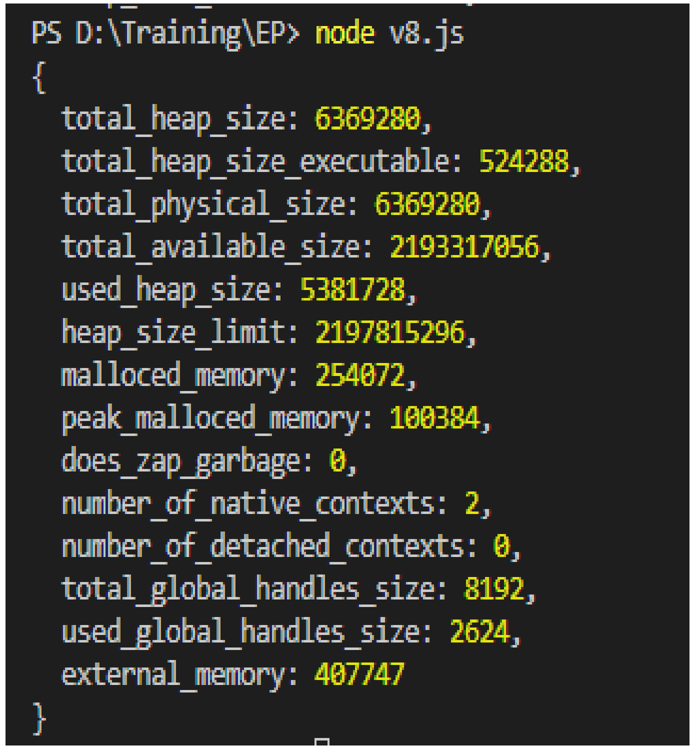
5.12.Node-V8:

- V8 is an open-source JavaScript engine developed by the Chromium project for the Google Chrome web browser. It is written in C++.
- To use this module, you need to use require('v8').
- **Syntax :** const v8 = require('v8');
- console.log(v8.getHeapSpaceStatistics()); and console.log(v8.getHeapStatistics());



```
const v8 = require('v8');
console.log(v8.getHeapStatistics());
```

Output:



```
PS D:\Training\EP> node v8.js
{
  total_heap_size: 6369280,
  total_heap_size_executable: 524288,
  total_physical_size: 6369280,
  total_available_size: 2193317056,
  used_heap_size: 5381728,
  heap_size_limit: 2197815296,
  malloced_memory: 254072,
  peak_malloced_memory: 100384,
  does_zap_garbage: 0,
  number_of_native_contexts: 2,
  number_of_detached_contexts: 0,
  total_global_handles_size: 8192,
  used_global_handles_size: 2624,
  external_memory: 407747
}
```

By default, v8 has a memory limit of 512Mb on 32-bit and 1Gb on 64-bit systems. We can raise the limit by setting --max-old-space-size to a maximum of ~1Gb for 32-bit and ~1.7Gb for 64-bit systems. But it is recommended to split our single process into several workers if you are hitting memory limits.

Recap

- Buffer module is used to convert strings to binary format
    - alloc()
    - allocUnsafe()
- The cluster module provides a way of creating child processes.
- addListener() Method - listens to the event and display the output.
- V8 is an open-source JavaScript engine

Think!

1. Node.js is an open-source runtime environment for _______.
2. Node.js operates on what engine?
3. Write a Command to check the node version.
4. Node.js works on Synchronous or Asynchronous?
5. Node.js operates on a _______ thread event loop
6. How to import a js file in node.js?
7. To run a node.js, what changes needs to be made in the package.json file?
8. Write a command to check the npm version
9. Whether Modules can be reused throughout the node.js application - yes/No.

10. What function is used to import the In-built Modules?

11. _____________Method is used to create a server on a local system.

12. Which module is used to check the expression in the programs & also act as a testing code?

13. In Assert Module, If the program expression is correct then it returns _______________ as output.

14. Which method is used to check the object values and data type in the assert module?

Option1: DeepEqual

Option2:deepStrictEqual()

15. Write the uses of fs Module:

16. what are all the parameters is used for opening the file in the file module?

17. Which Module translates the IP address to the Human readable form?

18. __________ method is Used to get the Host name Information.

19. Which Method is used to convert Ip address into Host name?

Option1:resolve()

Option2:reverse()

20. What module is used to convert query to string?

21. _________ method is used to convert object to string.

22. List out some path methods in path Module:

23. _________ module is used to convert strings to binary format

24. Which module creates a child process that run simultaneously and share the same server port.

25. Choose the syntax for the addlistener() method:

Option1:addListener ('string', event name);

Option2:addListener ('event name','string');

Exercise 4:

1. Convert a String to object using Query string module

2. Convert an Object to String using Query string module

3. Using Buffer module, Convert a String to binary format

Happy Coding!