Contents

[Creating Simple Web Server using Node.js 2](#_Toc126673076)

[Built-in Modules(Core Moudel) 2](#_Toc126673077)

[2](#_Toc126673078)

[Local Modules 3](#_Toc126673079)

[3](#_Toc126673080)

[3rd Party Modules 3](#_Toc126673081)

[3](#_Toc126673082)

[Node.js Global Objects 4](#_Toc126673083)

[Events 5](#_Toc126673084)

[Read File with File System(fs) Module 5](#_Toc126673085)

[Create File with fs module 6](#_Toc126673086)

[Update File with fs module 6](#_Toc126673087)

[Delete File with fs module 7](#_Toc126673088)

[7](#_Toc126673089)

[Rename File with fs module 7](#_Toc126673090)

[7](#_Toc126673091)

[URL MOUDLE 8](#_Toc126673092)

[Path Moudule 9](#_Toc126673093)

[9](#_Toc126673094)

[Serving Static Files with Http ,URL and File Module 10](#_Toc126673095)

[Basic Routing 11](#_Toc126673096)

[11](#_Toc126673097)

[Template Engines 12](#_Toc126673098)

[12](#_Toc126673099)

[13](#_Toc126673100)

# Creating Simple Web Server using Node.js

**var** http **=** require**(**'http'**);** // import http module

// create server with createServer methods

http**.**createServer**(function** **(**req**,** res**)** **{** // req : request , res : respone

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});** // set res headar

res**.**end**(**'<h1>Hello World!</h1>'**);** // respone html

**}).**listen**(**8080**,function(){**

console**.**log**(**'Server runging on port 8089'**)**

**});**

//access : http://localhost:8080/

//Console http module

**const** http **=** require**(**'http'**);**

console**.**log**(**http**);**

**{**

\_connectionListener**:** **[**Function**:** connectionListener**],**

METHODS**:** **[**

'ACL'**,** 'BIND'**,** 'CHECKOUT'**,**

'CONNECT'**,** 'COPY'**,** 'DELETE'**,**

'GET'**,** 'HEAD'**,** 'LINK'**,**

**....**

**],**

STATUS\_CODES**:** **{**

'100'**:** 'Continue'**,**

'101'**:** 'Switching Protocols'**,**

'102'**:** 'Processing'**,**',

'511'**:** 'Network Authentication Required'

**....**

**},**

# Built-in Modules(Core Moudel)

# 

/\* Built-in Modules(Core Moudel)

- To use a module, use the require() function with the name of the modul

- Node.js has a set of built-in modules which you can use without any further installation.

assert : Provides a set of assertion tests

buffer : To handle binary data

child\_process : To run a child process

cluster : To split a single Node process into multiple processes

crypto : To handle OpenSSL cryptographic functions

dgram : Provides implementation of UDP datagram sockets

dns : To do DNS lookups and name resolution functions

domain : Deprecated. To handle unhandled errors

events : To handle events

fs : To handle the file system

http and https : To make Node.js act as an HTTP server and HTTPS server

net : To create servers and clients

os : Provides information about the operation system

path: To handle file paths

punycode : Deprecated. A character encoding scheme

querystring : To handle URL query strings

readline : To handle readable streams one line at the time

stream : To handle streaming data

string\_decoder : To decode buffer objects into strings

timers : To execute a function after a given number of milliseconds

tls : To implement TLS and SSL protocols

tty : Provides classes used by a text terminal

url : To parse URL strings

util : To access utility functions

v8 : To access information about V8 (the JavaScript engine)

vm : To compile JavaScript code in a virtual machine

zlib : To compress or decompress files

\*/

# Local Modules

**var** http **=** require**(**'http'**);** // import http module

// create server with createServer methods

http**.**createServer**(function** **(**req**,** res**)** **{** // req : request , res : respone

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});** // set res headar

res**.**end**(**'<h1>Hello World!</h1>'**);** // respone html

**}).**listen**(**8080**,function(){**

console**.**log**(**'Server runging on port 8089'**)**

**});**

//access : http://localhost:8080/

//Console http module

**const** http **=** require**(**'http'**);**

console**.**log**(**http**);**

**{**

\_connectionListener**:** **[**Function**:** connectionListener**],**

METHODS**:** **[**

'ACL'**,** 'BIND'**,** 'CHECKOUT'**,**

'CONNECT'**,** 'COPY'**,** 'DELETE'**,**

'GET'**,** 'HEAD'**,** 'LINK'**,**

**....**

**],**

STATUS\_CODES**:** **{**

'100'**:** 'Continue'**,**

'101'**:** 'Switching Protocols'**,**

'102'**:** 'Processing'**,**',

'511'**:** 'Network Authentication Required'

**....**

**},**

**var** http **=** require**(**'http'**);** // import http module

// create server with createServer methods

http**.**createServer**(function** **(**req**,** res**)** **{** // req : request , res : respone

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});** // set res headar

res**.**end**(**'<h1>Hello World!</h1>'**);** // respone html

**}).**listen**(**8080**,function(){**

console**.**log**(**'Server runging on port 8089'**)**

**});**

//access : http://localhost:8080/

//Console http module

**const** http **=** require**(**'http'**);**

console**.**log**(**http**);**

**{**

\_connectionListener**:** **[**Function**:** connectionListener**],**

METHODS**:** **[**

'ACL'**,** 'BIND'**,** 'CHECKOUT'**,**

'CONNECT'**,** 'COPY'**,** 'DELETE'**,**

'GET'**,** 'HEAD'**,** 'LINK'**,**

**....**

**],**

STATUS\_CODES**:** **{**

'100'**:** 'Continue'**,**

'101'**:** 'Switching Protocols'**,**

'102'**:** 'Processing'**,**',

'511'**:** 'Network Authentication Required'

**....**

**},**

# 

**/\* Local Modules (created locally in your Node.js application)**

- create flie

- export that

- require exported moudel

- use that

\*/

touch myMoudle**.**js // create flie

exports**.**calSquare **=** **function** **(**w**,**h**)** **{** exports **return** w**\***h**;};** //export that

**var** http **=** require**(**'http'**);** //require file

**var** myMoudle **=** require**(**'./myMoudle'**);** //require expro file

http**.**createServer**(function** **(**req**,** res**)** **{**

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});**

res**.**write**(**"calculate area : " **+** myMoudle**.**calSquare**(**2**,**3**));**

res**.**end**();**

**}).**listen**(**8080**);**

# 3rd Party Modules

**var** http **=** require**(**'http'**);** // import http module

// create server with createServer methods

http**.**createServer**(function** **(**req**,** res**)** **{** // req : request , res : respone

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});** // set res headar

res**.**end**(**'<h1>Hello World!</h1>'**);** // respone html

**}).**listen**(**8080**,function(){**

console**.**log**(**'Server runging on port 8089'**)**

**});**

//access : http://localhost:8080/

//Console http module

**const** http **=** require**(**'http'**);**

console**.**log**(**http**);**

**{**

\_connectionListener**:** **[**Function**:** connectionListener**],**

METHODS**:** **[**

'ACL'**,** 'BIND'**,** 'CHECKOUT'**,**

'CONNECT'**,** 'COPY'**,** 'DELETE'**,**

'GET'**,** 'HEAD'**,** 'LINK'**,**

**....**

**],**

STATUS\_CODES**:** **{**

'100'**:** 'Continue'**,**

'101'**:** 'Switching Protocols'**,**

'102'**:** 'Processing'**,**',

'511'**:** 'Network Authentication Required'

**....**

**},**

**var** http **=** require**(**'http'**);** // import http module

// create server with createServer methods

http**.**createServer**(function** **(**req**,** res**)** **{** // req : request , res : respone

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});** // set res headar

res**.**end**(**'<h1>Hello World!</h1>'**);** // respone html

**}).**listen**(**8080**,function(){**

console**.**log**(**'Server runging on port 8089'**)**

**});**

//access : http://localhost:8080/

//Console http module

**const** http **=** require**(**'http'**);**

console**.**log**(**http**);**

**{**

\_connectionListener**:** **[**Function**:** connectionListener**],**

METHODS**:** **[**

'ACL'**,** 'BIND'**,** 'CHECKOUT'**,**

'CONNECT'**,** 'COPY'**,** 'DELETE'**,**

'GET'**,** 'HEAD'**,** 'LINK'**,**

**....**

**],**

STATUS\_CODES**:** **{**

'100'**:** 'Continue'**,**

'101'**:** 'Switching Protocols'**,**

'102'**:** 'Processing'**,**',

'511'**:** 'Network Authentication Required'

**....**

**},**

# 

/\***3rd Party Modules**

- 3rd Party Modules (express, gulp, lodash, async, socket.io, mongoose, underscore, pm2, . bower, q, debug, react,mocha etc)

- Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls . Node.js packages in your application

\*/

**>**npm install express **//install package local**

**>**npm install **-**g express **//install package globally`**

**>**npm update express **//Update Package**

**>**npm uninstall express **//uninstall pacekage**

# Node.js Global Objects

/\* **Node.js Global Objects**

- Node.js global objects are built-in objects,that are part of the JavaScript . they are available in all modules and no need to include or import in your . application so you can use directly.

- Some of these objects aren't actually in the global scope but in the module . . scope

\*/

/\* **global** : The global namespace. Setting a property to this namespace makes it globally visible within the running process.In the browser, the top-level property is the global scope. Therefore within the browser, the var var\_name will define a global variable. In Node var var\_name inside a module is the local to that module i.e. top-level scope is not the global scope \*/

/\* **console** : is used to print stdout and stderr. \*/

/\* **process** : provides interaction with the current Node process and can be accessed from anywhere.\*/

/\* **Class:Buffer** : It deals with binary data directly and can be constructed in a variety of ways. \*/

/\* **require()** : used to include other modules that exist in separate files,a string specifying the module to load.\*/

/\* **require.resolve()** : used to search the location of a module and returns the resolved filename \*/

\* **require.cache** : Modules are cached (contains all the current loaded modules) in this object. You can clear module by deleting a key value from this object. You can load it again using require() \*/

/\* **filename** : \_\_filename contains the absolute path of the currently executing file. \*/

/\* **dirname** : \_\_dirname contains the path to the root directory of the currently executing script \*/

/\* **module** : A reference to the current module. In a particular program, a module becomes available through require().It is not global but rather local to each module. \*/

/\* **exports** : A reference to the module.exports that is shorter to type. exports isn't actually a global but rather local to each module\*/

/\* **setTimeout(cb, ms)** : the setTimeout() calls a function (cb) after a specified number of milliseconds (ms) \*/

/\* **clearTimeout(t)** : The clearTimeout() is used to cancel a timeout that was set with setTimeout(). The callback will not execute.\*/

/\* **setInterval(cb, ms)** : setinterval() calls a fun(cb) repeatedly at specified intervals (in milliseconds (ms)) \*/

/\* **clearInterval(t)** : clearInterval() is used to stop a timer that was set with setInterval(). callback will not execute.\*/

# Events

**/\*Events**

- Every action() on a computer is an event. Like when a connection is made or a file is opened or keypress.etc..

- Node.js has a built-in module, called "Events", where you can create-, fire-, and listen for- your own events.

- Objects in Node.js can fire events, like the readStream object fires events when opening and closing a file

\*/

**const** events **=** require**(**'events'**);** // import events

**const** eventsEmitter **=** **new** events**.**EventEmitter**();** // create EventEmitte object

eventsEmitter**.**on**(**'myEvent'**,function(){** // assign event handler

console**.**log**(**'myEvetn working...'**)**

**})**

eventsEmitter**.**emit**(**'myEvent'**)** // fire event

**//Console events moduel**

**const** eve **=** require**(**'events'**);**

console**.**log**(**eve**);**

**[**Function**:** EventEmitter**]** **{**

once**:** **[**Function**:** once**],**

on**:** **[**Function**:** on**],**

EventEmitter**:** **[**Circular**],**

usingDomains**:** **false,**

captureRejectionSymbol**:** Symbol**(**nodejs**.**rejection**),**

captureRejections**:** **[**Getter**/**Setter**],**

errorMonitor**:** Symbol**(**events**.**errorMonitor**),**

defaultMaxListeners**:** **[**Getter**/**Setter**],**

init**:** **[**Function**],**

listenerCount**:** **[**Function**]**

**}**

console**.**log**(new** eve**.**EventEmitter**())**

# Read File with File System(fs) Module

**/\* File System(fs) Module**

- Fs module allows you to work with the file system on your computer.

\*/

**/\* Read File**

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

\*/

**const** http **=** require**(**'http'**);** // http moudle

**const** fs **=** require**(**'fs'**);** // file module

http**.**createServer**(function** **(**req**,** res**)** **{** // create server

fs**.**readFile**(**'home.html'**,** **function(**err**,** data**)** **{** // reading file : home.html

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});** // set respone header

res**.**write**(**data**);** // respone data

**return** res**.**end**();** // end response process

**});**

**}).**listen**(**8080**,function(){**

console**.**log**(**'Server running on port 8080'**);**

**});** // port listen

# Create File with fs module

**/\* Create File**

- In Nodejs to create file we can use several method such as fs.appendFile(),fs.open(),fs.writeFile()

\*/

**/\* fs.appendFile()**

- fs.appendFile() method appends content to a file. If the file does not exist, the file will be created: `

- fs.apppendFile method take three parameter,filename,content,cb(err)

\*/

**const** fs **=** require**(**'fs'**);**

fs**.**appendFile**(**'about.html'**,**"<h1 style='text-aling:center'>About Page</h1>"**,function(**err**){**

**if** **(**err**)** **throw** err**;**

console**.**log**(**'Created File and added content!'**);**

**});**

**/\* fs.open()**

- Open for writing.If the file does not exist, the file will be created

- The fs.open() method takes a "flag" as the second argument, if the flag is "w" for "writing"

- fs.open method take three parameter,filename,w,cb(err,file)

\*/

**const** fs **=** require**(**'fs'**);**

fs**.open(**'contact.html'**,**'w'**,function(**err**,**file**){**

**if** **(**err**)** **throw** err**;**

console**.**log**(**'opend file and writed content'**);**

**})**

**/\* fs.writeFile()**

- 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

\*/

**const** fs **=** require**(**'fs'**);**

fs**.**writeFile**(**"login.html"**,**"<h1 style='text-aling:center'>Login Page</h1>"**,function(**err**){**

**if** **(**err**)** **throw** err**;**

console**.**log**(**'Create login.html file and added content !'**);**

**})**

# Update File with fs module

**/\* Update(fs.appendFile(),fs.writeFile())**

- The File System module has methods for updating filesfs.appendFile() and fs.writeFile()

\*/

**/\* fs.appendFile() method**

- fs.apppendFile() content to a file. If the file does not exist, the file will be created `

\*/

**const** fs **=** require**(**'fs'**);**

fs**.**appendFile**(**'about.html'**,**"<h1 style='text-aling:center'>About Page</h1>"**,function(**err**){**

**if** **(**err**)** **throw** err**;**

console**.**log**(**'Created File and added content!'**);**

**});**

**/\* fs.writeFile()**

- 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:

\*/

**const** fs **=** require**(**'fs'**);**

fs**.**writeFile**(**"login.html"**,**"<h1 style='text-aling:center'>Login Page</h1>"**,function(**err**){**

**if** **(**err**)** **throw** err**;**

console**.**log**(**'Create login.html file and added content !'**);**

**})**

# Delete File with fs module

# 

**/\* Delete(fs.unlink()) File**

- fs.unlink method can use to delete a file

\*/

**const** fs **=** require**(**'fs'**);**

**if(**fs**.**existsSync**(**'test.html'**)){** // check file with sync

fs**.**unlink**(**'test.html'**,function(**err**){**

**if** **(**err**)** **throw** err**;**

console**.**log**(**'test.html File deleted /> '**);**

**});**

**}else{**

console**.**log**(**'test.html file not found ,Place create test.html /> '

**)}**

# Rename File with fs module

# 

**/\* Rename(fs.rename(finame,rename,cb(err)))**

- To rename a file use the fs.rename() method.

\*/

**const** fs **=** require**(**'fs'**);**

**if(**fs**.**existsSync**(**'text.txt'**)){**

fs**.**rename**(**'text.txt'**,**'text2.txt'**,function(**err**){**

**if(**err**)** **throw** err**;**

console**.**log**(**'renamed file/> '**);**

**})**

**}else{** console**.**log**(**'text.txt file not found ,Place create text.txt /> '**)}**

**//Console Fs module**

**const** fs **=** require**(**'fs'**);**

console**.**log**(**fs**);**

appendFile**:** **[**Function**:** appendFile**],**

appendFileSync**:** **[**Function**:** appendFileSync**],**

access**:** **[**Function**:** access**],**

……

# URL MOUDLE

**/\* URL MOUDLE**

- Nodejs built-in URL Module is provided to wrok operations with address URLs like split a web address

- URL string may carry information such as https, the hostname, query string values, and more.

\*/

┌─────────────────────────────────────────────────────────────────────────────┐

│ href │

├──────────┬┬───────────┬─────────────────┬───────────────────────────┬───────┤

│ protocol ││ auth │ host │ path │ hash │

│ ││ ├──────────┬──────┼──────────┬────────────────┤ │

│ ││ │ hostname │ port │ pathname │ search │ │

│ ││ │ │ │ ├─┬──────────────┤ │

│ ││ │ │ │ │ │ query │ │

" http: // user:pass @ host.com : 8080 /p/a/t/h ? query=string #hash "

│ ││ │ │ │ │ │ │ │

└──────────┴┴───────────┴──────────┴──────┴──────────┴─┴──────────────┴───────┘

**//Console url module**

**const** url **=** require**(**'url'**);**

console**.**log**(**url**)**

**/\* url.parse()**

- Parse an address with the `url.parse()` method, and it will return a URL object with each part of the address as properties

\*/

**const** url **=** require**(**'url'**);** //import url moudle

**let** url\_adr **=** 'http://localhost:8080/'**;**

**let** query **=** url**.**parse**(**url\_adr**,true);** //split address with url.parser( ) method

console**.**log**(**query**);**

Url **{**

protocol**:** 'http:'**,**

slashes**:** **true,**

auth**:** **null,**

host**:** 'localhost:8080'**,**

port**:** '8080'**,**

hostname**:** 'localhost'**,**

hash**:** **null,**

search**:** '?user=10&info=shie'**,**

query**:** **[**Object**:** **null** **prototype]** **{** user**:** '10'**,** info**:** 'shie' **},**

pathname**:** '/index.html'**,**

path**:** '/index.html?user=10&info=shie'**,**

href**:** 'http://localhost:8080/index.html?user=10&info=shie'

**}**

**/\*url.format()**

- The url.format() method returns a formatted URL string derived from urlObject

\*/

**const** url **=** require**(**'url'**);**

**let** url\_adr **=** 'http://localhost:8080/index.html?user=10&info=shie'**;**

**let** query **=** url**.**parse**(**url\_adr**,true)**

console**.**log**(**url**.**format**(**url\_adr**))**

http**:**//localhost:8080/index.html?user=10&info=shie

/\* Rename(fs.rename(finame,rename,cb(err)))

- To rename a file use the fs.rename() method.

\*/

**const** fs **=** require**(**'fs'**);**

**if(**fs**.**existsSync**(**'text.txt'**)){**

fs**.**rename**(**'text.txt'**,**'text2.txt'**,function(**err**){**

**if(**err**)** **throw** err**;**

console**.**log**(**'renamed file/> '**);**

**})**

**}else{** console**.**log**(**'text.txt file not found ,Place create text.txt /> '**)}**

//Console Fs module

**const** fs **=** require**(**'fs'**);**

console**.**log**(**fs**);**

appendFile**:** **[**Function**:** appendFile**],**

appendFileSync**:** **[**Function**:** appendFileSync**],**

access**:** **[**Function**:** access**],**

……

/\* Rename(fs.rename(finame,rename,cb(err)))

- To rename a file use the fs.rename() method.

\*/

**const** fs **=** require**(**'fs'**);**

**if(**fs**.**existsSync**(**'text.txt'**)){**

fs**.**rename**(**'text.txt'**,**'text2.txt'**,function(**err**){**

**if(**err**)** **throw** err**;**

console**.**log**(**'renamed file/> '**);**

**})**

**}else{** console**.**log**(**'text.txt file not found ,Place create text.txt /> '**)}**

//Console Fs module

**const** fs **=** require**(**'fs'**);**

console**.**log**(**fs**);**

appendFile**:** **[**Function**:** appendFile**],**

appendFileSync**:** **[**Function**:** appendFileSync**],**

access**:** **[**Function**:** access**],**

……

# Path Moudule

# 

**/\* Path Moudule**

- Provides a way of working with directories and file paths.

\*/

basename**():** Returns the last part **of** a path

delimiter**:** Returns the delimiter specified **for** the platform

dirname**()** **:** Returns the directories **of** a path

extname**()** **:** Returns the file extension **of** a path

format**()** **:** Formats a path object into a path string

isAbsolute**():** Returns **true** **if** a path is an absolute path**,** otherwise **false**

join**()** **:** Joins the specified paths into one

normalize**()** **:** Normalizes the specified path

parse**()** **:** Formats a path string into a path object

posix **:** Returns an object containing POSIX specific properties and methods

relative**():** Returns the relative path from one specified path to another specified path

resolve**()** **:** Resolves the specified paths into an absolute path

sep **:** Returns the segment separator specified **for** the platform

win32**:** Returns an object containing Windows specific properties and methods

**//Console path moduel**

**const** path **=** require**(**'path'**);**

console**.**log**(**path**)**

**/\* basename(path, extension:optional)**

returns the filename part of a file path.\*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**basename**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**);**

console**.**log**(**filename**);** // 011-path.js

**/\*delimiter**

- return the delimiter used on the server( ;(semicolon) for Windows), : (colon) for POSIX) \*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**delimiter**;**

console**.**log**(**filename**);**

**/\* path.dirname()**

- returns the directories of a file path. \*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**dirname**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**);**

console**.**log**(**filename**);** // D:/Desktop/note/Nodejs/codes

**/\* path.extname()**

- returns the extension of a file path. \*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**extname**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**);**

console**.**log**(**filename**);** // .js

**/\* path.format()**

- formats a path object into a path string. \*/

**const** path **=** require**(**'path'**);**

**const** file\_obj **=** **{** dir**:** 'D:\\Desktop\\note\\odejs\\codes'**,** base**:** '011-path.js' **}**

**const** fp **=** path**.**format**(**file\_obj**);**

console**.**log**(**fp**);** // D:\Desktop\note\odejs\codes\011-path.js

**/\* path.isAbsolute()**

- returns true if the specified path is an absolute path, otherwise false \*/

**const** path **=** require**(**'path'**);**

console**.**log**(**path**.**isAbsolute**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**));** // true

console**.**log**(**path**.**isAbsolute**(**'codes/011-path.js'**));** // false

# Serving Static Files with Http ,URL and File Module

//home.html

**<**body**>**

**<**h1 style**=**"text-align: center;padding-top:50px;"**>** Home page**</**h1**>**

**</**body**>**

//about.html

**<**body**>**

**<**h1 style**=**"text-align: center;padding-top:50px;"**>**About page**</**h1**>**

**</**body**>**

//App.js

**const** http **=** require**(**'http'**);**

**const** url **=** require**(**'url'**);**

**const** fs **=** require**(**'fs'**);**

**const** path **=** require**(**'path'**);**

http**.**createServer**(function(**req**,**res**){**

**const** url\_adr **=** url**.**parse**(**req**.**url**,true)**

**const** filename **=** "." **+** url\_adr**.**pathname**;**

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/html'**});**

fs**.**readFile**(**filename**,** **function(**err**,** data**)** **{**

**if** **(**err**)** **{**

**return** res**.**end**(**"<h1 style='text-aling:center'>Page Not Found</h1>"**);**

**}**

res**.**write**(**data**);**

**return** res**.**end**();**

**});**

**}).**listen**(**8080**,()=>** console**.**log**(**'server running on port 8080'**))**

/\* Path Moudule

- Provides a way of working with directories and file paths.

\*/

basename**():** Returns the last part **of** a path

delimiter**:** Returns the delimiter specified **for** the platform

dirname**()** **:** Returns the directories **of** a path

extname**()** **:** Returns the file extension **of** a path

format**()** **:** Formats a path object into a path string

isAbsolute**():** Returns **true** **if** a path is an absolute path**,** otherwise **false**

join**()** **:** Joins the specified paths into one

normalize**()** **:** Normalizes the specified path

parse**()** **:** Formats a path string into a path object

posix **:** Returns an object containing POSIX specific properties and methods

relative**():** Returns the relative path from one specified path to another specified path

resolve**()** **:** Resolves the specified paths into an absolute path

sep **:** Returns the segment separator specified **for** the platform

win32**:** Returns an object containing Windows specific properties and methods

//Console path moduel

**const** path **=** require**(**'path'**);**

console**.**log**(**path**)**

/\* basename(path, extension:optional)

returns the filename part of a file path.\*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**basename**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**);**

console**.**log**(**filename**);** // 011-path.js

/\*delimiter

- return the delimiter used on the server( ;(semicolon) for Windows), : (colon) for POSIX) \*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**delimiter**;**

console**.**log**(**filename**);**

/\* path.dirname()

- returns the directories of a file path. \*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**dirname**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**);**

console**.**log**(**filename**);** // D:/Desktop/note/Nodejs/codes

/\* path.extname()

- returns the extension of a file path. \*/

**const** path **=** require**(**'path'**);**

**const** filename **=** path**.**extname**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**);**

console**.**log**(**filename**);** // .js

/\* path.format()

- formats a path object into a path string. \*/

**const** path **=** require**(**'path'**);**

**const** file\_obj **=** **{** dir**:** 'D:\\Desktop\\note\\odejs\\codes'**,** base**:** '011-path.js' **}**

**const** fp **=** path**.**format**(**file\_obj**);**

console**.**log**(**fp**);** // D:\Desktop\note\odejs\codes\011-path.js

/\* path.isAbsolute()

- returns true if the specified path is an absolute path, otherwise false \*/

**const** path **=** require**(**'path'**);**

console**.**log**(**path**.**isAbsolute**(**'D:/Desktop/note/Nodejs/codes/011-path.js'**));** // true

console**.**log**(**path**.**isAbsolute**(**'codes/011-path.js'**));** // false

# Basic Routing

# 

**/\* Basic Routing \*/**

**const** http **=** require**(**'http'**);**

**const** fs **=** require**(**'fs'**);**

http**.**createServer**((**req**,**res**)=>{**

res**.**writeHead**(**200**,{**'Content-Type' **:** 'text/html'**});**

**if(**req**.**url **===** '/' **||** req**.**url **===** '/home' **||** req**.**url **===** '/home' **||** req**.**url **===** '/index'**){**

**const** readStream **=** fs**.**createReadStream**(**'home.html'**,**'utf-8'**)**

readStream**.**pipe**(**res**)**

**}else** **if(**req**.**url **===** '/about'**){**

**const** readStream **=** fs**.**createReadStream**(**'about.html'**,**'utf-8'**)**

readStream**.**pipe**(**res**)**

**}else** **if(**req**.**url **===** '/contact'**){**

**const** readStream **=** fs**.**createReadStream**(**'contact.html'**,**'utf-8'**)**

readStream**.**pipe**(**res**)**

**}else{**

**const** readStream **=** fs**.**createReadStream**(**'404.html'**,**'utf-8'**)**

readStream**.**pipe**(**res**)**

**}**

**}).**listen**(**8080**,()=>{** console**.**log**(**'Server running on port 8080'**);})**

**//Serving Json**

**const** http **=** require**(**'http'**);** // import fs module

**const** fs **=** require**(**'fs'**);** // import fs module

http**.**createServer**((**req**,**res**)=>{** // create server

**const** jsonData **=** fs**.**createReadStream**(**'api.json'**,**'utf-8'**);** // create read stream

res**.**writeHead**(**200**,** **{**'Content-Type'**:** 'text/json'**});** // wirte respone header

jsonData**.**pipe**(**res**);** // respone with readable stream pipe

**}).**listen**(**8080**,()=>** console**.**log**(**'Server running on port 8080'**))** // listen port

# Template Engines

# 

**/\*Template Engines**

- also known as a template processor or template parser

- At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client

\*/

/\* JavaScript Template engines

- Ejs

- PUG

- Underscore

- Embedded

- Mustache

- Handlebars

- Jade

- JSRENDER

- DOT

- Nunjucks

\*/

**/\* Ejs**

- EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. \*/

**//Install**

npm install **--**save ejs

**//Usage**

ejs**.**renderFile**(**filename**,** data**,** options**,** **function(**err**,** str**){**

// str => Rendered HTML string

**});**

**/\* Using ejs in express**

- we need to do is to set EJS as our templating engine with Express

\*/

// app.js

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

**var** app **=** express**();**

app**.**set**(**'view engine'**,** 'ejs'**);** // Set EJS as templating engine

app**.**get**(**'/'**,** **(**req**,** res**)=>{**

res**.**render**(**'home'**);**

**});**

**/\*View folder**

- EJS looks into the ‘views’ folder for the templates to render by default

- make a ‘views’ folder in our main node project folder

- make a file named “filname.ejs”

\*/

// views/home.ejs

**<**body**>** **<**h1**>**Home page**</**h1**></**body**>**

**/\* Dynamic data**

- To add dynamic content this render method takes a second parameter which is an object

\*/

//app.js

app**.**get**(**'/user'**,** **(**req**,** res**)=>{**

res**.**render**(**'user'**,{**name**:**'shine shine'**});**

**});**

// views/user.ejs

**<**body**>** **<**h1**>**Hellow **<%=**user**%></**h1**></**body**>**

# 

**/\* Looping**

- To add dynamic content this render method takes a second parameter which is an object

\*/

// app.js

app.get('/all-users', (req, res)=>{

const all\_users = ['shine shine','Tun Tun','Aung Aung'];

res.render('user',{users:all\_users});

});

// view/user.ejs

<body>

<% users.forEach((item)=>{ %>

<li><%=item%></li>

<% }); %>

</body>

**//condtiional runding**

app.get('/all-users', (req, res)=>{ // app.js

const all\_users = {

isShow : true,

users : ['shine shine','Tun Tun','Aung Aung']

}

res.render('user',{users:all\_users});

});

//view/user.ejs

<body>

<% if(all\_users.isShow){ %>

<% all\_users.users.forEach((item)=>{ %>

<li><%=item%></li>

<% }); %>

<% }); %>

</body>

**//Layouts**

<%- include('header'); -%>

<h1>Title</h1>

<p>My page</p>

<%- include('footer'); -%>

**// Includes**

<%- include('user/show', {user: user}); %>

**/\* Tags**

- `<%` 'Scriptlet' tag, for control-flow, no output

- `<%\_` ‘Whitespace Slurping’ Scriptlet tag, strips all whitespace before it

- `<%=` Outputs the value into the template (HTML escaped)

- `<%-` Outputs the unescaped value into the template

- `<%#` Comment tag, no execution, no output

- `<%%` Outputs a literal '<%'

- `%>` Plain ending tag

- `-%>` Trim-mode ('newline slurp') tag, trims following newline

- `\_%>` ‘Whitespace Slurping’ ending tag, removes all whitespace after it

\*/

**//Layouts**

**<%-** include**(**'header'**);** **-%>**

**<**h1**>**

Title

**</**h1**>**

**<**p**>**

My page

**</**p**>**

**<%-** include**(**'footer'**);** **-%>**

# Environment variables

/\* **Environment variables**

- To manage the configuration of our applications separately from our codebase.

- Separating configurations makes it easier for our application to be deployed in different environments.

- An environment variable is a dynamic-named value.

- Wherever your app needs configuration such as HTTP port, database connection string, you use environment variables

- to reduces hard code

\*/

**/\* process.env**

- That can be use read or set environment variables from Node.js

- "process" does not require a “require”, it’s automatically available.

\*/

**const** port **=** process**.**env**.**PORT **||** 8080

**const** hostname **=** process**.**env**.**HOST**;**

**const** database **=** process**.**env**.**DATABASE**;**

**/\* Dotenv(.env)**

- "Dotenv" is a 3rd Party Modules that use to access environment variables in our node application

- By convention, the variables are written in UPPERCASE letters.

- The ".env" file should never be committed to the source code repository. We must place the file into the .gitignore" file. (When using git.)

\*/

npm install dotenv **--**save **//Install**

require**(**'dotenv'**).**config**();** **// Loading environment variables**

**//Create .env file and then add environment variables**

HOST **=** localhost

DATABASE **=** dbname

PORT **=** 8080

TOEKN **=** ajfLK8932lkad9wlks0k

**// Access environment variables**

**const** http **=** require**(**'http'**);**

http**.**createServer**((**req**,**res**) => {**

res**.**end**(**process**.**env**.**TOKEN**)**

**}).**listen**(**8080**,()=>** console**.**log**(**'Server running on port' **+** process**.**env**.**PORT **))**

# readline Module

/\* **readline Module**

- Allows the reading of input stream line by line

- This module wraps up the process standard output and process standard input objects.

- Readline module makes it easier for input and reading the output given by the user

- the createInterface() method takes two arguments. The first argument will be for the standard input and the second one will be for reading the standard output.

- rli.question() method is used for asking questions

- rli.close() method is used close the interface

\*/

**const** readline **=** require**(**'readline'**);** // import readline moudle

**const** rli **=** readline**.**createInterface**(**process**.**stdin**,**process**.**stdout**);** // interface for input output

// or const rli.createInterface({input : process.stdin,output : process.stdout });

rli**.**question**(**'What is your age? '**,** **(**age**)** **=>** **{** // aking question

console**.**log**(**'Your age is: ' **+** age**);**

rli**.close();**

**});**

**//Open a file and return the content line by line with readline module**

**const** readline **=** require**(**'readline'**);**

**const** fs **=** require**(**'fs'**);**

**const** rli **=** readline**.**createInterface**({**input**:** fs**.**createReadStream**(**'home.html'**)});**

**let** linenumber **=** 0**;**

rli**.**on**(**'line'**,function(**line**){**

linenumber**++;**

console**.**log**(**linenumber **+**' ' **+** ' ' **+** ' ' **+** ' ' **+** line**)**

**});**

**/\* Prompt**

- setPrompt() method : to set the particular statement to the console.

- prompt() method : for displaying the statement which is set in setPrompt() Method.

- rli.on('line',cb(data)) method : listen event

- rl.on() method takes the first argument as line event.

\*/

**const** rl **=** require**(**'readline'**)**

**const** rli **=** rl**.**createInterface**(**process**.**stdin**,**process**.**stdout**);**

rli**.**setPrompt**(**'Enter Name.. '**);**

rli**.prompt();**

rli**.**on**(**'line'**,** **(**name**)** **=>** **{**

console**.**log**(**`Your name : ${name}`**);**

rli**.close()**

**});**