https://shorturl.at/3lCHe 10 Jun 2024

---

---

=================================================================
Array Manipulators
=================================================================

```
/*
01. map():-
 - This function is used to manipulate each and every element in array
 - it returns an array

//Eg01
let arr = [10, 20, 30, 40, 50]
//multiply each element by 2
console.log(arr.map((element, index) => {
    return element * 2
}))

//Eg02
let arr = [1, 2, 3, 4, 5]
//o/p ['$1','$2','$3','$4','$5']
console.log(arr.map((element, index) => {
    return '$' + element
}))

//Eg03
let arr1 = [1, 2, 3]
let arr2 = ['one', 'two', 'three']
//o/p [ [ 1, 'one' ], [ 2, 'two' ], [ 3, 'three' ] ]
console.log(arr1.map((elemenet, index)=>{
    return [elemenet, arr2[index]]
}))

02. filter():-
 - this function creates an array based on condition

//Eg01
let arr1 = [10, 20, 30, 40, 50]
//create array with elements greater than 30
console.log(arr1.filter((elemenet, index) => {
    return elemenet > 30
}))

//Eg02
let arr2 = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]
//create an array elements greater than or equal to 100
console.log(arr2.filter((element, index) => {
    return element >= 100
}))

//Eg03
let arr3 = [10, 20, 30, 40, 50]
//o/p [300, 400, 500]
let res1 = arr3.filter((element, index)=>{
    return element >= 30
})
console.log(res1)
let res2 = res1.map((element, index)=>{
    return element * 10
})
console.log(res2)
///////////////
let res1 = arr3.filter((element, index)=>{
    return element >= 30
}).map((element, index)=>{
    return element * 10
})
console.log(res1)
//////////////////
console.log(arr3.filter((element, index)=>{
    return element >= 30
}).map((element, index)=>{
    return element * 10
}))


03. reduce()            left to right   0 -> 1
04. reduceRight()       right to left   0 <- 1

let arr = [1, 2, `3`, 4, 5]
console.log(arr.reduce((pv, nv) => {
    return pv + nv
}))

console.log(arr.reduceRight((pv, nv) => {
    return pv + nv
}))

05. forEach
06. for...of
07. for...in

08. push():- add element at end
09. unshift():- add element at beginning
10. pop():- remove element from end
11. shift():- remove element from beginning
*/
```

```
let arr = [20, 30, 40]
console.log(arr)            //[20, 30, 40]
arr.push(50)
console.log(arr)            //[ 20, 30, 40, 50 ]
arr.unshift(10)
console.log(arr)            //[ 10, 20, 30, 40, 50 ]
console.log(arr.pop())
console.log(arr)            //[ 10, 20, 30, 40 ]
console.log(arr.shift())    //10
console.log(arr)            //[ 20, 30, 40 ]


/*
12. some() :-if any one element in array satisfies the condition
 then it will return true, otherwise false.
13. every():-if all elements in array satisfy the condition
 then it will return true, otherwise false.

let arr = [10, 20, 30, 40, 50]
console.log(arr.some((element, index) => {
    return element > 10
})) //true
console.log(arr.every((element, index) => {
    return element > 10
})) //false
console.log(arr.some((element, index) => {
    return element > 50
})) //false
console.log(arr.every((element, index) => {
    return element <= 50
})) //true

14. find():-
 - this function is used to find the element in array
 - if element found, it will return the same element
 - if element not found it will return undefined
15. includes():-
 - it is boolean function used to check element present
   in array or not

let arr = [10, 20, 30, 40, 50]
console.log(arr.find((element, index)=>{
    return element == '30'
})) //30
console.log(arr.find((element, index)=>{
    return element === '30'
})) //undefined

console.log(arr.includes(30))   //true
console.log(arr.includes('30')) //false

16. splice()    -> swiss army knife for arrays
https://javascript.info/array-methods
let arr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
console.log(arr)
arr.splice(5, 2)
```

```
console.log(arr)    //[10, 20, 30, 40, 50, 80, 90, 100]
//delete 80
arr.splice(5, 1)
console.log(arr)    //[ 10, 20, 30, 40, 50, 90, 100]
//delete 100
//arr.splice(6, 1)
//arr.splice(-1, 1)
arr.splice(arr.length - 1, 1)
console.log(arr)    //[ 10, 20, 30, 40, 50, 90 ]
arr.splice(2, 2)
console.log(arr)    //[ 10, 20, 50, 90 ]
//before 90 add 60, 70, 80
arr.splice(3, 0, 60, 70, 80,)
console.log(arr)    //[ 10, 20, 50, 60, 70, 80, 90 ]
//delete 50 and add 30, 40, 50
arr.splice(2, 1, 30, 40, 50)
console.log(arr)    //[10, 20, 30, 40, 50, 60, 70, 80, 90 ]
//add 100 at end
//arr.splice(9,0,100)
arr.splice(arr.length, 0, 100)
console.log(arr)    //[10, 20, 30, 40, 50, 60, 70, 80, 90, 100 ]


17. findIndex():-
 - it is used to find index of particular element

let arr = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]
let idx = arr.findIndex((element, index) => {
    return element == 30
})
console.log(arr)    //[10, 100, 20, 200, 30, 300, 40, 400, 50, 500]
console.log(idx)    //4
arr.splice(idx, 1)
console.log(arr)
key = 40
arr.splice(arr.findIndex((element, index) => {
    return element == key
}), 1)
console.log(arr)

let arr2 = [
    { p_id: 111 },
    { p_id: 1111 },
    { p_id: 222 },
    { p_id: 333 }
]
console.log(arr2)
arr2.splice(arr2.findIndex((element)=>{
    return element.p_id === 1111
}), 1)
console.log(arr2)

18. slice():-
let arr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
//in slice first include last exclude
//-ve indices supported
```

```
console.log(arr)                //[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
console.log(arr.slice(5, 7))    //[ 60, 70 ]
console.log(arr.slice(3, 6))    //[ 40, 50, 60 ]
console.log(arr.slice(5))       //[ 60, 70, 80, 90, 100 ]
console.log(arr.slice(5, -1))   //[ 60, 70, 80, 90 ]
console.log(arr.slice(5, -5))   //[]

19. copyWithin()
let arr1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
console.log(arr1)
//copy all elements at index 1 -> 1 index right
console.log(arr1.copyWithin(1))     //[10, 10, 20, 30, 40, 50, 60, 70, 80, 90
]
let arr2 = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]
console.log(arr2.copyWithin(5))     //[10, 100, 20, 200, 30, 10, 100, 20, 200,
30]
let arr3 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
//copy all elements from index 5 at index 2
console.log(arr3.copyWithin(2, 5))  //[10, 20, 60, 70,  80, 90, 100, 80, 90,
100]
let arr4 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
//copy all elements from index no 4 to 6 (excluding 6) at index 2
console.log(arr4.copyWithin(2, 4, 6))//[10, 20, 50, 60, 50, 60, 70, 80, 90,
100 ]

20. indexOf() :- don't create index for duplicate elements
let arr = [10, 20, 30, 10, 40, 20, 40, 50]
arr.forEach((element, index) => {
    console.log(element, index, arr.indexOf(element))
})
console.log(arr.filter((element, index) => {
    return arr.indexOf(element) === index
}))//this code removes duplicates
let mySet = new Set(arr)
console.log(mySet)
console.log([...mySet])
console.log(Array.from(mySet))

21. sort()
let arr = [10, 50, 20, 40, 30]
console.log(arr)    //[10, 50, 20, 40, 30]
console.log(arr.sort((num1, num2) => {
    return num1 - num2
})) //[ 10, 20, 30, 40, 50 ]
console.log(arr.sort((num1, num2) => {
    return num2 - num1
})) //[ 50, 40, 30, 20, 10 ]

22. length()
let arr = [1, 2, 3, 4, 5]
console.log(arr)            //[ 1, 2, 3, 4, 5 ]
console.log(arr.length)     //5
console.log(arr[3])         //4
console.log(arr[5])         //undefined
console.log(arr[arr.length])//undefined
arr.length = 3
```

```
console.log(arr)            //[ 1, 2, 3 ]

23. delete():- element  deleted but memory not released
let arr = [1, 2, 3, 4, 5]
console.log(arr)            //[ 1, 2, 3, 4, 5 ]
console.log(arr.length)     //5
delete (arr[2])
console.log(arr.length)     //5
console.log(arr)
arr.length = 3
arr.length = 5
console.log(arr)            //[ 1, 2, <3 empty items> ]

24. from() :- string to array
25. join() :- array to string

let str = 'Hello'
console.log(str)                //Hello
console.log(Array.from(str))    //[ 'H', 'e', 'l', 'l', 'o' ]
let arr = [ 'H', 'e', 'l', 'l', 'o' ]
console.log(arr.join(""))       //Hello

26. fill() :- element replacement
let arr = [10, 20, 30, 40, 50]
console.log(arr)                //[ 10, 20, 30, 40, 50 ]
console.log(arr.fill(100))      //[ 100, 100, 100, 100, 100 ]
console.log(arr.fill(200, 2))   //[ 100, 100, 200, 200, 200 ]
console.log(arr.fill(300, 2, 4))//[ 100, 100, 300, 300, 200 ]

27. flat():-
let arr = [1, [2], [3], [4, [5]]]
console.log(arr)
console.log(arr.flat(2))
//if we dont know the levels
let arr2 = [1, [[[2]],
[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[3]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
]]]]]]]]]]]]]]]]]
console.log(arr2.flat(Infinity))


28.reduce():-
29. flatMap():-
let arr1 = [1, 2, 3]
let arr2 = ['one', 'two', 'three']
console.log(arr1.map((element, index) => {
    return [element, arr2[index]]
})) //[ [ 1, 'one' ], [ 2, 'two' ], [ 3, 'three' ] ]
console.log(arr1.flatMap((element, index) => {
    return [element, arr2[index]]
})) //[ 1, 'one', 2, 'two', 3, 'three' ]

30. entries()       :- object to array
31. fromEntries()   :- array to object

32. split()
let arr = 'Welcome to JavaScript'
```

```
console.log(arr.split())        //[ 'Welcome to JavaScript' ]
console.log(arr.split(" "))     //[ 'Welcome', 'to', 'JavaScript' ]
let str = 'mahabharat'
console.log(str.split("a"))     //[ 'm', 'h', 'bh', 'r', 't' ]
console.log(str.split("a", 3))  //[ 'm', 'h', 'bh' ]

33. lastIndexOf()
let arr = [10, 20, 10, 20, 30, 10]
console.log(arr.lastIndexOf(10))    //5
console.log(arr.lastIndexOf(20))    //3

34. concat():-

let arr1 = [10]
let arr2 = [20]
let arr3 = [30]
let arr4 = arr1.concat(arr2, arr3)
console.log(arr4)   //[ 10, 20, 30 ]

35. substr()
36. substring()

let str = `Welcome to Javascript`
//Welcome
console.log(str.substr(0, 7))
console.log(str.substring(0, 7))
//to
console.log(str.substr(8, 2))
console.log(str.substring(8, 10))
//Javascript
console.log(str.substr(11))
console.log(str.substring(11))

37. trim()
let str = ' welcome '
console.log(str.length)
let str1 = str.trim()
console.log(str1.length)
let str2 = str.trimStart()
console.log(str2.length)
let str3 = str.trimEnd()
console.log(str3.length)

38. replace() :- This function is used to complete or partial replacement of
string

//Eg01
let str = "school"
let res = str.replace("school", "college")
console.log(str)    //school
console.log(res)    //college

//Eg02
let str = "This is my school"
let res = str.replace("school", "college")
console.log(str)
console.log(res)


//Eg03
let str = "red green Red red Green Red"
let res = str.replace(/red/,'Yellow')
console.log(str)        //red green Red red Green Red
console.log(res)        //only first occurence replaced
let res1 = str.replace(/red/g,"Yellow")
console.log(res1)       //all occurences (case sensitive)
let res2 = str.replace(/red/ig,"Yellow")
console.log(res2)       //all occurences ignore cases
```

39. search() :- This function returns the index of the first match string
returns -1 for unsuccessful search

```
let str = "Sound mind in sound body"
console.log(str.search("sound"))    //14
console.log(str.search("Sound"))    //0
console.log(str.search(/sound/i))   //0
console.log(str.search('refresh'))  //-1
```

40. toLocaleLowerCase()
41. toLocaleUpperCase()

these functions are similar to toLowerCase() and toUpperCase() respectively,
the difference is
that toLocaleLowerCase() and toLocaleUpperCase()
functions produce outputs depend on local
language of that particular region (i.e. in browser's local language)

```
let str = 'istanbul'
console.log(str.toLocaleUpperCase('tr'),str.toUpperCase())
```

42. charCodeAt():- this function returns unicode of the character at the
specified index in a string.
//http://www.columbia.edu/kermit/ucs2.html

```
let str = "aAbB"
console.log(str.charCodeAt(0))      //97
console.log(str.charCodeAt(1))      //65
console.log(str.charCodeAt(2))      //98
console.log(str.charCodeAt(3))      //66
```

43.valueOf():-returns the primitive value of String object

```
var str = new String("ABC")
var res = str.valueOf(str)
console.log(str)
console.log(res)
```

44. toString() :-
String.toString()   -> converts String object to string
number.toString()    -> method converts a number to a string with base as
argument (from 2 to 36)

```
var str = new String("ABC")
var res = str.toString()
```

```
console.log(str)    //[String: 'ABC']
console.log(res)    //ABC
let num = 99
console.log(num.toString(2))    //1100011
console.log(num.toString(8))    //143
console.log(num.toString(16))   //63

45. match():-this function accepts regular expression as argument and
returns array of matches and returns null if match not found
*/
var str = "Importance is given to Portfolio"
console.log(str.match(/port/g))     //[ 'port' ]
console.log(str.match(/port/ig))    //[ 'port', 'Port' ]
```

**All students have to gather data in following JSON format for Capstone project**

```
[
	{
		"p_id": ,
		"p_name": "",
		"p_cost": ,
		"p_cat": "",
		"p_desc": "",
		"p_img": "IMAGE URL"
	},
]
```

**for "IMAGE URL" make sure that all images are with same resolution**
**Gather 100 data**

=================================================================
Interaction with MongoDB
=================================================================

- MongoDB is a lightweight NoSQL database.
- MongoDB follows client server architecture.
- MongoDB follows the 'mongodb' protocol.
- Mongoserver running on port no 27017.
- MongoDB supports JSON.

Installation of MongoDB

1. Download and install mongodb(community edition) (Prefer v5.x)
	https://www.mongodb.com/try/download/community

*** For windows OS ***

2. Create directory structure
	c:/data/db

3. set path for environment variables
	-> computer
	-> properties

-> advanced system settings
-> advanced tab
-> environment variables
-> click on path
-> edit
-> new
-> copy and paste path of installation of mongodb
  where mongo.exe and mongod.exe located

For cloud database
1. Login to mongodb.com (atlas)
- Perform relevant network access and database access settings
2. Browse collections
3. Create database 'nodedb' with collection 'products'
   - Add my own data
4. Insert documents
5. Click on databases from left panel
6. Click on connect
7. click on compass
8. Copy url and follow steps below url from web page

mongodb+srv://admin:admin@mdb.vtkja.mongodb.net/

MongoDB Queries (Local Database)
 - Open command prompt
 >mongo
 - create and switch to database
   >use nodedb;
 - create collection
   >db.createCollection("products")
 - insert data
   >db.products.insert({"p_id":111, "p_name":"P_one", "p_cost":10000})
 - fetch data
   >db.products.find()
 - show databases
   >show dbs
 - delete database
   >db.dropDatabase()

================================================================
MongoDB Advanced Operations
================================================================
Find particular record/s ?

```
db.products.find(
	{
		$where:function()
		{
			return (
                                        this.p_name === “p_five”
                              )
                    }
          }
)
```

Sorting mongodatabase (Note:- only display)

```
>db.products.find().sort({p_name:1})
          where          p_name is sort key
                              1         -> ascending order sort
                              -1        -> descending order sort
```

pretty() method

```
>db.products.find().sort({p_name:1}).pretty()
```

Limiting Records

```
>db.products.find().sort({p_name:-1}).limit(2)
```

where we can access only two records

MongoDB task
Create a mongodb database namely college
create collection Student
insert atleast 20 documents
- Student Name
- Branch
- percentage
- gender

1. Find topper of CS branch
2. Find College topper
3. Find Topper from girls
4. Find Topper from boys
5. Sort according to percentage
6. Find Top 3 students
7. Find slow learners (less percentage)

```
{"name":  ,"branch":  , "percentage":  , "gender": }
```

Sample for insert many

```
>use college
>db.createCollection("students")
>db.students.insertMany([{},{},{}])
```

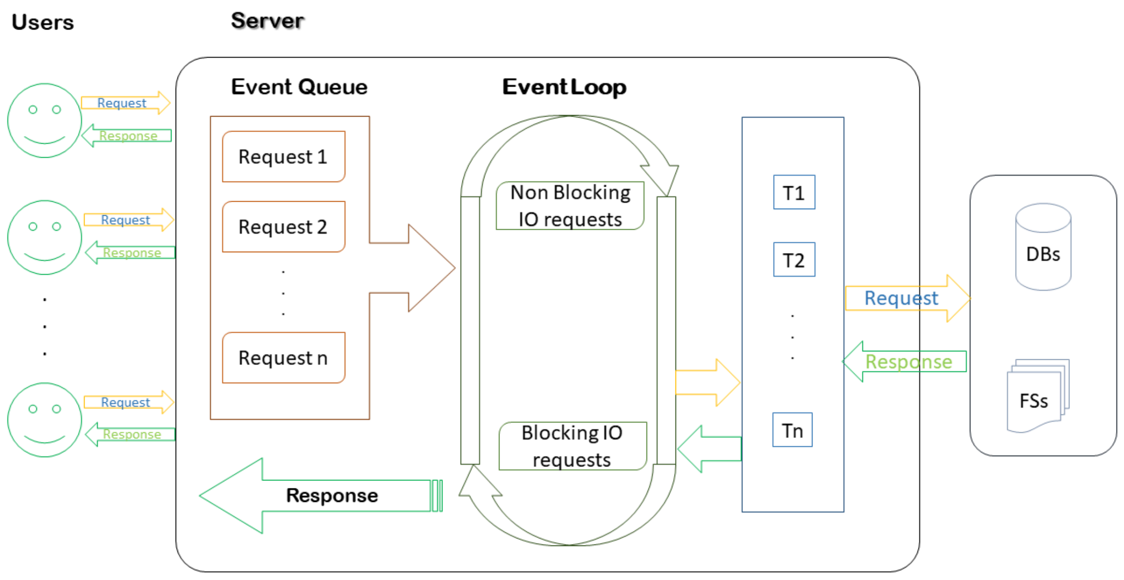
=========================================================

## Node.JS Intro

=========================================================

- Node.JS is open source, cross platform, Javascript server side runtime environment.
- Node applications can be developed using either Javascript or Typescript.

- Node.JS was released by Ryan Dahl on 27th May 2009, at netscape.
- Latest version of Node.JS is v22.2.0 2024-05-15
- Current stable version is v20.14.0 2024-05-28
  https://nodejs.org/en/about/previous-releases
- The applications (servers) developed by Node.JS are called 'Single Threaded Event Loop' applications.



Modules in Node.JS

- We can download all third party modules by using either 'npm' or 'yarn' tool.
- npm stands for 'Node Packaging Manager'.
- npm is an integrated tool for NodeJS.
- yarn is the latest tool used to download 'Node modules'
- yarn is faster as compared to npm.
- All node modules will be downloaded to the 'node_modules' folder in the current path.
- we can start node server in 3 ways
- >node server
- >npm start
- >nodemon server (monitoring / watch mode)

Environmental setup

1. Download and install nodejs
   https://nodejs.org/en/
2. Download and install git
   https://git-scm.com/
3. Download and install VSCode

https://code.visualstudio.com/
4. Download and install postman
https://www.postman.com/downloads/
5. Install yarn tool using following command
```
>npm install -g yarn
        npm :- node packaging manager
        -g  :- global installation
```

- Create a folder rename it as 'NodeJS'
- in that folder create one more folder 'modules eg'
- open 'modules eg' folder in VSCode
- create server.js file there

Download following modules using yarn tool

1. express
2. mysql
3. mongodb@2.2.32
4. multer
5. jwt-simple

initialise project
```
>npm init -y
```

```
1. express              >yarn add express --save
2. mysql                >yarn add mysql --save
3. mongodb@2.2.32       >yarn add mongodb@2.2.32 --save
4. multer
5. jwt-simple
```

Combine 4 and 5
```
>yarn add multer jwt-simple --save
```

==============================================================
Implementing HTTP server
==============================================================
- 'http' is the predefined module used to create http servers.
- http is the native module, so no need to download it.
- 'require()' is used to import.
- Eg let http = require('http')
- 'createServer()' is the predefined function in the http module.
- createServer is used to create http server.
- The argument to createServer is the arrow function.
- to this arrow function there are two arguments, 'req' and 'res'.
- request and response objects provided by node engine respectively.
- req object is used to store client data.
- res object is used to send response to client.
- 'writeHead(-,-)' is the predefined function in res object.

-	The writeHead function is used to set the MIME type.
-	First argument to writeHead function is status code (200 - ok)
-	Second argument to the writeHead function is a JSON object.
-	The JSON key is 'content-type' and the value is 'text/html'.
-	'write(-)' is the predefined function in res object.
-	write function is used to append response to res object.
-	'end()' is the predefined function in the res object used to lock the response.

```
//import http module
let http = require('http')
//create server
let server = http.createServer((req, res)=>{
    //set MIME type
    res.writeHead(200,{'content-type':'text/html'})
    //append response
    res.write('<h1>Welcome to http server</h1>')
    //lock response
    res.end()
})
//assign port no
server.listen(8080)
console.log('Server listerning port no 8080')
/*
    start server
    >node server
    url http://localhost:8080
*/
```

====================================================================
HTTP get parameters
====================================================================
-	'url' is the predefined module in node.
-	url module is used to read get parameters in http server.

***server.js***
```
//import http module
let http = require('http')
//import url module
let url = require('url')
//create server
let server = http.createServer((req, res) => {
    let obj = url.parse(req.url, true).query
    let uname = obj.uname
    let upwd = obj.upwd
    //set MIME type
    res.writeHead(200, { 'content-type': 'text/html' })
    if (uname == 'admin' && upwd == 'admin')
        res.write('<h1>Login Success</h1>')
    else
        res.write('<h1>Login Failed</h1>')
    //lock response
    res.end()
})
```

```
//assign port no
server.listen(8080)
console.log('Server listerning port no 8080')
/*
    start server
    >node server
    url :- http://localhost:8080/?uname=admin&upwd=admin
*/
```

***index.html***

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
        <title>Http GET parameters</title>
    </head>
    <body>
        <form action="http://localhost:8080" method="get" class="box">
            <h1>Login</h1>
            <input type="text" placeholder="Enter Username" name="uname">
            <input type="password" placeholder="Enter Password" name = "upwd">
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

***style.css***

```
body
{
    background: radial-gradient(circle, white, black);
    font-family: sans-serif;
}
h1
{
    color: white;
    text-transform: uppercase;
    font-weight: normal;
}
.box
{
    background-color: black;
    width: 300px;
    margin: 50px auto;
    padding: 40px;
    border-radius: 20px;
    text-align: center;
}
input
{
    margin: 20px auto;
    text-align: center;
    padding: 14px 10px;
    width: 200px;
    border-radius: 24px;
    background: none;
}
input[type = "text"], input[type = "password"]
{
    border: 2px solid skyblue;
    color : lightyellow;
}
input[type = "submit"]
{
    border: 2px solid burlywood;
    color:white;
    cursor: pointer;
}
```

=====================================================================
HTTP post parameters
=====================================================================

- 'querystring' is the predefined module in nodejs.
- querystring module is used to read post parameters in http server

```
//import http module
let http = require('http')
//import querystring
let qs = require('querystring')
//create server
let server = http.createServer((req, res) => {
    //set MIME type
    res.writeHead(200, { 'content-type': 'text/html' })
    let body = ""
    //listen to post parmateres
    req.on("data", (result) => {
        body = body + result
    })
    //end callback node engine
    req.on("end", () => {
        let obj = qs.parse(body)
        let uname = obj.uname
        let upwd = obj.upwd
        if (uname === 'admin' && upwd === 'admin')
            res.write("<h1 style = 'color:green'>Login success</h1> ")
        else
            res.write("<h1 style = 'color:red'>Login failed</h1>")
        //lock response
        res.end()
    })
})
//assign port no
server.listen(8080)
console.log('Server listerning port no 8080')
/*
    start server
    >node server
*/
```

***index.html***

```
<!DOCTYPE html>
```

```
<html>
    <head>
        <link rel="stylesheet" href="style.css">
        <title>Http POST parameters</title>
    </head>
    <body>
        <form action="http://localhost:8080" method="post" class="box">
            <h1>Login</h1>
            <input type="text" placeholder="Enter Username" name="uname">
            <input type="password" placeholder="Enter Password" name = "upwd">
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

***style.css***

```
body
{
    background: radial-gradient(circle, white, black);
    font-family: sans-serif;
}
h1
{
    color: white;
    text-transform: uppercase;
    font-weight: normal;
}
.box
{
    background-color: black;
    width: 300px;
    margin: 50px auto;
    padding: 40px;
    border-radius: 20px;
    text-align: center;
}
input
{
    margin: 20px auto;
    text-align: center;
    padding: 14px 10px;
    width: 200px;
    border-radius: 24px;
    background: none;
}
input[type = "text"], input[type = "password"]
{
    border: 2px solid skyblue;
    color : lightyellow;
}
input[type = "submit"]
{
    border: 2px solid burlywood;
    color:white;
    cursor: pointer;
```

}

====================================================================

Express

====================================================================

- Download express module
> yarn add express --save

```
//initialyse project
//>npm init -y

//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express() //where app rest object
//create get request
app.get("/", (req, res) => {
    console.log('Default get request')
    //res.send({'message':'default get requrest'})
    res.json({ 'message': 'default get requrest' })
})
//create one more get request
app.get("/fetch",(req,res)=>{
    res.send({'message':'fetch get requrest'})
})
//craete post request
app.post("/",(req,res)=>{
    res.send({'message':'default post request'})
})
//create one more post request
app.post('/login', (req, res) => {
    res.send({ 'message': 'login post request' })
})
//create port
let port = 8080
//assign port no
app.listen(port, () => {
    console.log('Server listening port no ', port)
})
/*
start server
    >node server
Test urls with postman
    http://localhost:8080          Default GET
    http://localhost:8080/fetch    fetch GET
    http://localhost:8080          Default POST
    http://localhost:8080/login    login POST
*/
```

====================================================================

Reading get parameters in express
=====================================================================

//url :- http://localhost:8080/login/?uname=admin&upwd=admin

```
//initialyse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//create port
let port = 8080
//create rest api
app.get("/login",(req,res)=>{
    //query is the predefined key in req object
    //query is used to read get parameters
    let uname = req.query.uname
    let upwd = req.query.upwd
    if (uname === 'admin' && upwd === 'admin')
        res.json({ 'login': 'success' })
    else
        res.json({ 'login': 'failed' })
})
//assign port no
app.listen(port, () => {
    console.log('Server listening port no ', port)
})
```