MODULE AND EXPORTS

**Stuff.js file:**

//stuff.js

var counter=function(arr){

return "This array has " + arr.length + " Elements"

}

var adder=function(a,b){

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

}

module.exports.counter=counter;

module.exports.adder=adder;

**Export pattern 2:**

//stuff.js

var counter=function(arr){

return "This array has " + arr.length + " Elements"

}

var adder=function(a,b){

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

}

module.exports={

counter:counter,

adder:adder,

}

**app.js file:**

//app.js

var stuff=require("./stuff")

console.log(stuff.counter(["Siddharth","Rahul","Himanshu"]))

console.log(stuff.adder(5,5))

FUNCTION EXPRESSIONS

**Function Expression:**

function sayHi(name){

console.log("Hi " + name)

}

sayHi("siddharth");

var sayBye=function(name){

console.log("Bye " + name)

}

sayBye("siddharth")

**Function as parameters:**

function callfunction(fun){

fun()

}

function sayHi(){

console.log("Hi")

}

callfunction(sayHi)

READ AND WRITE FILES

**Synchronous Methods:**

const http = require('http');

const fs=require("fs");

//syntax fs.readFileSync(path[, options])

var readFile=fs.readFileSync("./readme.txt",'utf8');

//syntax fs.writeFileSync(file, data[, options])

var writeFile=fs.writeFileSync("./writeme.txt",readFile);

**Synchronous Methods:**

const http = require('http');

const fs=require("fs");

//syntax fs.readFileSync(path[, options])

var readFile=fs.readFileSync("./readme.txt",'utf8');

//syntax fs.writeFileSync(file, data[, options])

var writeFile=fs.writeFileSync("./writeme.txt",readFile);

**readFileMethods:**

const http = require('http');

const fs=require("fs");

//syntax fs.readFile(path[, options], callback)

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

console.log(data)

})

**writeFile Methods:**

**writeFileMethods:**

const http = require('http');

const fs=require("fs");

//syntax fs.writeFile(file, data[, options], callback)

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

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

})

**writeFile Methods:**

HTTP SERVER CREATIONS

const http = require('http');

const server=http.createServer(function(req,res){

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

res.end("Hi Siddharth")

});

server.listen(3000,'127.0.0.1')

console.log("server listening on port 3000")

STREAMS AND BUFFERS

**Read Stream:**

//syntax s.createReadStream(path[, options])

const MyReadStream=fs.createReadStream(\_\_dirname + "/readme.txt" , 'utf8');

MyReadStream.on('data',function(chunk){

console.log("New Chunk Recived");

console.log(chunk)

})

**Write Stream:**

const http = require('http');

const fs=require("fs");

//syntax s.createReadStream(path[, options])

const MyReadStream=fs.createReadStream(\_\_dirname + "/readme.txt" , 'utf8');

//syntax fs.createReadStream(path[, options])

const MyWriteStream=fs.createWriteStream(\_\_dirname + "/write.txt" , 'utf8');

MyReadStream.on('data',function(chunk){

MyWriteStream.write(chunk)

})

PIPES IN NODE

**Example of pipes:**

const http = require('http');

const fs=require("fs");

//syntax s.createReadStream(path[, options])

const MyReadStream=fs.createReadStream(\_\_dirname + "/readme.txt" , 'utf8');

//syntax fs.createReadStream(path[, options])

const MyWriteStream=fs.createWriteStream(\_\_dirname + "/write.txt" , 'utf8');

MyReadStream.pipe(MyWriteStream)

SERVING HTML

**Example:**

const http = require('http');

const fs=require("fs");

const server=http.createServer(function(req,res){

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

var myReadStrim=fs.createReadStream(\_\_dirname+ '/readme.txt' , 'utf8');

myReadStrim.pipe(res)

});

server.listen(3000,'127.0.0.1')

console.log("server listening on port 3000")

SERVING JSON

**Example:**

const http = require('http');

const fs=require("fs");

const server=http.createServer(function(req,res){

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

var myObj={

name:'Siddharth',

job:'UI/UX',

age:29

}

res.end(JSON.stringify(myObj))

});

server.listen(3000,'127.0.0.1')

console.log("server listening on port 3000")