**Own Module**

* Modules are the collection of JavaScript codes in a separate logical file that can be used in external applications on the basis of their related functionality.
* Modules are popular as they are easy to use and reusable.
* To create a module in Node.js, you will need the exports keyword.
* This keyword tells Node.js that the function can be used outside the module.

***Method 1***

**In test.js file:**

const add=(a,b)=> { return(a+b); }

**module.exports**=add;

**In another file test1.js where you want to use that module:**

var **a1**=require("./test.js");

console.log(**a1(10,15)**);

to run -> **node test1.js**

***Method 2***

**In test.js file:**

const **sub**=(a,b)=> { return(a-b); }

const **mul**=(a,b)=> { return(a\*b); }

**module.exports.s**=**sub**;

**module.exports.m**=**mul**;

**In another file test1.js:**

var **a1**=require("./test.js");

console.log(**a1**.**s**(10,5));

console.log(**a1**.**m**(10,15));

to run -> **node test1.js**

***Method 3***

**In test.js file:**

const **sub**=(a,b)=> { return(a-b); }

const **mul**=(a,b)=>{ return(a\*b); }

**module.exports**.**s**=**sub**;

**module.exports**.**m**=**mul**;

**In another file test1.js:**

var **{s,m}**=require("./test.js");

console.log(**s**(10,7));

console.log(**m**(10,12));

to run -> **node test1.js**

***Method 4***

**In test.js file:**

const **sub**=(a,b)=> { return(a-b); }

const **mul**=(a,b)=> { return(a\*b); }

const **name**="Hello"

**module.exports**=**{sub,mul,name}**;

**In another file:**

var **{sub,mul,name}**=require("./test.js");

console.log(**sub**(100,20));

console.log(**mul**(10,2));

console.log(**name**)

to run -> **node test1.js**

**Write a node.js script to create calculator using external module having a function add(), sub(), mul(), div(). This function returns result of calculation. Write all necessary .js files.**

**1.js**

exports.add = function (x, y) {

    return x + y;

};

exports.sub = function (x, y) {

    return x - y;

};

exports.mult = function (x, y) {

    return x \* y;

};

exports.div = function (x, y) {

    return x / y;

};

**2.js**

const calculator = require(‘./1.js’);

let x = 50, y = 20;

console.log("Addition of 50 and 20 is "

+ calculator.add(x, y));

console.log("Subtraction of 50 and 20 is "

+ calculator.sub(x, y));

console.log("Multiplication of 50 and 20 is "

+ calculator.mult(x, y));

console.log("Division of 50 and 20 is "

+ calculator.div(x, y));

**Output:**

**Addition of 50 and 20 is 70**

**Subtraction of 50 and 20 is 30**

**Multiplication of 50 and 20 is 1000**

**Division of 50 and 20 is 2.5**

**Write all necessary .js files to create module having a function to check numbers from 2 to 50 are prime number or not.**

**1.js**

const PrimeNo = (num) =>

{

    let temp = 0

    for(let i=2;i<num;i++)

    {

        if(num%i==0)

        {

            temp++;

        }

    }

    if(temp==0) { return true; }

    else{ return false; }

}

module.exports=PrimeNo;

**2.js**

var PrimeNumber = require("./1.js")

for(i=2;i<=50;i++)

{

let x=PrimeNumber(i);

if(x==true)

{

    console.log(i+" Prime Number");

}

else{

    console.log(i+" Not a Prime Number")

}

}

**Output:**

2 Prime Number

3 Prime Number

4 Not a Prime Number

5 Prime Number ………. upto 50

**Write a nodejs script to create own module to calculate reverse of a given number.**

**That module should be used to check given number of which square of reverse and reverse of square is same.**

**For Example,**

**12 (122 =144)**

**21 (212=441)**

**144 = reverse(441)**

c1.js

function reversenum(num)

{

let rev=0;

while(num>0)

{

rev=rev\*10+(num%10);

**//num=12 (0\*10+12%10) rev=2**

**//num=1 (2\*10+1%10) rev=21**

**//num=441 (0\*10+441%10) rev=1**

**//num=44 (1\*10+44%10) rev=14**

**//num=4 (14\*10+144%10) rev=144**

num=parseInt(num/10);

**//num 1,num=0,num=44,num==4,num=0**

}

return rev;

}

function square(num){ return num\*num; }

function checknum(num)

{

a=square(num) **//a=144**

b=square(reversenum(num)) **//b=441**

if(a==reversenum(b)) { console.log("Equal") }

else{ console.log("Not equal") }

}

module.exports.checknum = checknum

c2.js

var c1=require("./c1.js");

c1.checknum(12);

**Chalk module**

 In Node.js is the third-party module that is used for styling the format of text.

**Advantages of Chalk Module:**

1. It helps to customize the color of the output of the command-line output
2. It helps to improve the quality of the output by providing several color options like for warning message red color and many more

Steps to install chalk

**- Create one folder   
- Set proxy if required:** npm config set proxy http://192.168.10.252:808

**- To install chalk: npm install chalk** **or** **npm i chalk**

- After installating a module, package-lock.json and package.json will be created.

- Add ["type": "module"] in **package.json** file as shown below

**In package.json file:**

{

**"type": "module",**

"dependencies": {

**"chalk": "^5.2.0"**

}

}

**Note:** If chalk installed successfully and package.json and package-lock.json files are not created, then try **npm init** command inside created folder. And then try to install chalk inside folder.

**Example:**

**import ch from "chalk";**

const log=console.log;

log("LJU");

log("hello"+**ch**.**bgCyan**(" LJU ")+" GM ")

log(**ch**.**blue**.**underline**.**bgYellow**("hello")+**ch**.**red**.**bold**.**underline**.**bgWhite**("Yahoo"));

**Output:**



**Validator module**

The Validator module is popular for validation. Validation is necessary to check whether the data is in format or not, so this module is easy to use and validates data quickly and easily.

Simple functions for validation like isEmail(), isEmpty(), isLowercase() etc.

**To install validator:** npm install validator

**Example1 : Check whether given email is valid or not**

**import validator from "validator"**

let email = 'test@gmail.com'

console.log(**validator**.**isEmail**(email)) **// true**

email = 'test@'

console.log(**validator**.**isEmail**(email)) **// false**

**Example2 : Check whether string is in lowercase or not**

**import validator from "validator"**

let name = 'hellolju'

console.log(**validator**.**isLowercase**(name)) **// true**

name = 'HELLOLJU'

console.log(**validator**.**isLowercase**(name)) **// false**

**Example3: Check whether string is empty or not**

**import validator from "validator"**

let name = ''

console.log(**validator**.**isEmpty**(name)) **// true**

name = 'helloLJU'

console.log(**validator**.**isEmpty**(name)) **// false**

**Wrapper function**

* NodeJS does not run our code directly, it wraps the entire code inside a function before execution.
* This function is termed as Module Wrapper Function. Before a module’s code is executed, NodeJS wraps it with a function wrapper that has the following structure:

(function (exports, require, module, \_\_filename, \_\_dirname) {

//module code

});

* The five parameters — exports, require, module, \_\_filename, \_\_dirname are available inside each module in Node.
* Though these parameters are global to the code within a module yet they are local to the module (because of the function wrapper as explained above). These parameters provide valuable information related to a module.
* The variables like **\_\_filename** and **\_\_dirname**, that tells us the module’s absolute filename and its directory path.

**Example:**

(

function()

{

console.log(\_\_filename);

console.log(\_\_dirname);

}

)();

**Output:**

D:\Trynode\ex1.js //returnd path of current file

D:\Trynode //returned path till current file (folder)