Node.js notas:

# Parte I

1. requires(./nombre\_archivo) esto para importar librerías
2. exports.nombre = función; hay que exportar las funciones que serán llamadas por el requires
3. crear un objeto const Math = {};
4. exportar ese objeto  
     
   Math.add = add;

Math.substract = substract;

Math.multiply = multiply;

Math.divide = divide;

module.exports = Math;

1. Documentación <https://nodejs.org/dist/latest-v16.x/docs/api/>
2. Llamado de modulo pre instalados: const os = require('os');
3. Modulo sistema operativo  
   const os = require('os');

console.log(os.platform());

console.log(os.release());

console.log('free mem', os.freemem());

1. Manejo de archivos:  
     
   const fs = require('fs');

fs.writeFile('./texto.txt', 'la persona que me gusta es: ', function(err){ // ubicacion del archivo, mensaje a escribir, funcion callback para aber el error

    if(err){

       console.log(err);

    }

    else{

        console.log('No se habla de bruno nonono');

    }

});

console.log('ultima linea de codigo');

fs.readFile('./texto.txt',  function(err,data){ // ubicacion del archivo,  funcion callback para saber el error o leer lo datos

    if(err){

       console.log(err);

    }

    else{

        console.log(data.toString());

    } });

1. http

const http = require('http');

http.createServer(function (req, res ){ // peticion request req, respuesta response res

res.writeHead(200, {'content-type': 'text/html'});// cabecera, e pone un código de estado http y el contenido de lo //////que se se espera (texto html)

res.write('<h1>Hola mundo desde node js</h1>'); // escibir

res.end();

}).listen(3000);

10)otro modo:  
  
const handleserver = function (req, res ){ // peticion request req, respuesta response res

res.writeHead(200, {'content-type': 'text/html'});// cabecera, e pone un codigo de estado http y el contenido de lo que se se espera (texto html)

res.write('<h1>Hola mundo desde node js</h1>'); // escibir

res.end();

}

const server = http.createServer(handleserver);

server.listen(3000, function(){// crea una funncion callback despues de acceder al servidor

console.log('server on port 3000');

});

11)usando npm  
  
EN la consola escribir

npm install colors

12) uso colors   
const color = require('colors');

console.log('server on port 3000', .green);

13) npm init // sirve para hacer una descripción del curso

Crea un package.jon

14) dentro del paquete json como decirle cual es el sitio inicial:

"scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "node index.js"

luego puedo iniciar el proyecto con el codigo npm start

15)correr otra categoría del script npm run nombre de la categoría creada

16)instalar express

npm i express

"dependencies": {

    "colors": "^1.4.0",

    "express": "^4.17.2"

  },

17)mismo codigo con express

const express = require('express');

const server  = express();

server.get('/', function(req, res){

     res.send('<h1>Hola MUndo</h1>');

     res.end();

});

server.listen(3000,function(){

    console.log('server on port 3000'.green)

} );

# PARTE 2: EXPRESS

1. routing

server.get('/about', (req,res)=>{ // creando la ruta about

    res.send('About me');

});

1. npm install nodemon: framework que me permite introducer cambios inmediatos a node reiniciar el servidor  
     
   uso de nodemon: npx nodemon index.js
2. permitir que el servidor use json

server.use(express.json()); // hace que express entienda el formato json

4)método post(dinámico)

Si por ejemplo a parte de /user s quiere agregar algo mas (ejemplo <http://localhost:5000/user/456>)

server.post('/user/:id',(req,res)=>{ //mostando lo recibido del post en consola

    console.log(req.body);

    console.log(req.params);

    res.send('REQUEST POST');

});

5)

Acceder a los parámetros de la URL

server.put('/aput/:id',(req,res)=>{

    console.log(req.body);

    res.send('User ${req.params.id} updated'); // para acceder al parametro en la url

});

6)

server.all('/user', (req,res,next)=>{

    console.log('por aqui paso');

    next();

 });

Ojo este “all” tendría que estar de primero”

# Parte 3 Middleware

1) )los middleware funcionan para procesar datos antes de llegar a la ruta

function logger(req,res,next){ //middleware

    console.log('Request Recieved');

    next();

}

Uso:  
  
server.use(logger);

2) para saber el protocolo (http, https,ftb) el host (ip, dirección, localhost) y el puerto:

console.log( `Request Recieved: ${req.protocol}://${req.get('host')}${req.originalUrl}`);

el middleware debe ser definido antes de las rutas

3) Morgan

Sirve para dar la información del request

const morgan = require('morgan'); //llamado modulo morgan npm i morgan

uso

server.use(morgan('dev'));

respuesta:

GET /user 304 6.534 ms - -

4) static files (se usa por defecto en caso que no se acceda a ninguna ruta)

server.use(express.static('public'));// uso de express static en caso de que no se acceda a ninguna ruta

la carpeta public debe tiene un index.html que accede a una carpeta css o cualquier otro archivo javascript que se acceda por defecto

5) java script timer

setTimeout(() => {

    document.getElementById('loquesea').innerHTML = 'Cambiado desde inner html';

}, 3000);

# Parte IV settings