APUNTES

Día 1 parte I

Como inicializar cualquier proyecto o ejercicio NODEJS

npm init

nombre del proyecto ENTER

versión ENTER.

description ENTER.

Entry point: (index.js) ENTER.

Test command: ENTER “Definiríamos el comando del testing”

Git repository: ENTER.

Keywords: ENTER.

Author: Alfonso Jiménez ENTER.

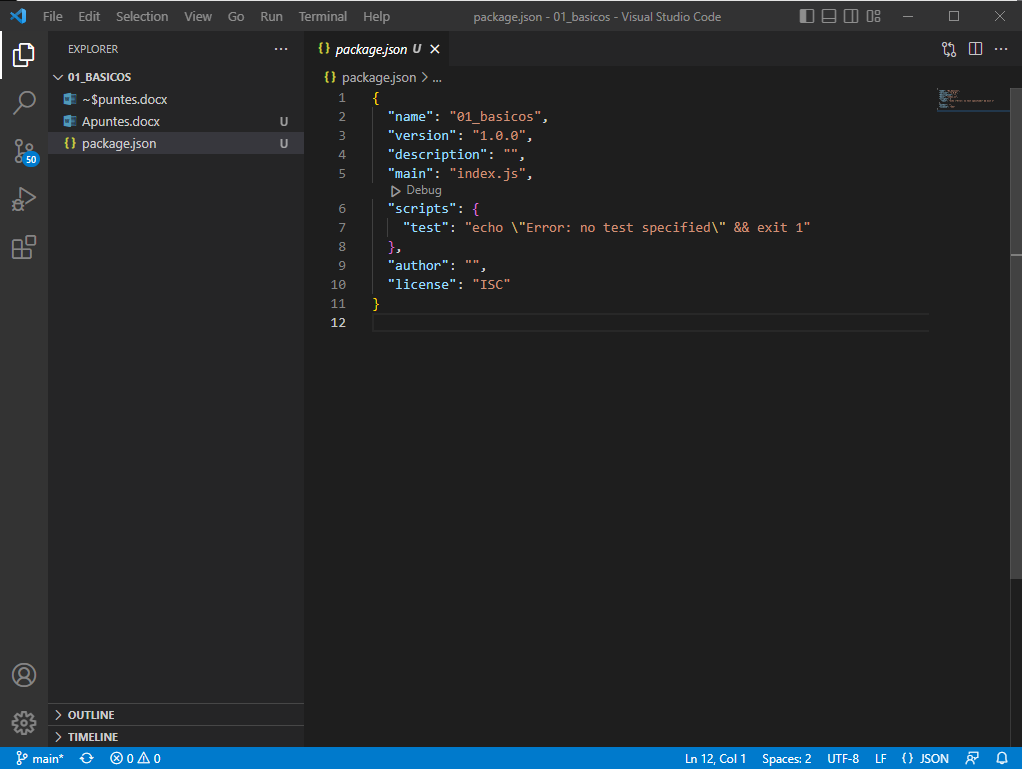
License: (ISC) ENTER.

Is this OK? (yes)

Yes

NOTA: Ya nos ha creado el fichero package.json con un DIR

Abrimos el proyecto en visual code .



Ahora mismo no hay dependencias, más adelante lo veremos.

* Generamos el fichero index.js
  + Console.log (“Hola Node”)

Terminal

Node index.js

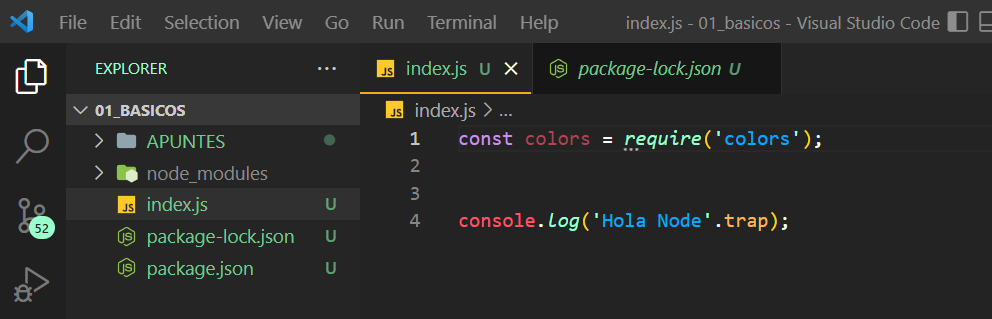
Resultado

* Instalar mi primera librería COLORS (Colores)
* Terminal

Npm install colors

Automáticamente me ha añadido al proyecto la carpeta node\_module y en el package.json una dependencia llamada colors, esto nos dice que para que mi proyecto funcione es imprescindible que tenga instalada esta dependencia.

* Dentro de index.js
  + Donde queramos utilizar esta librería lo primero es requerirla



Recordar que si borramos la carpeta node\_module o si directamente nos descargamos un proyecto cualquiera para que todo funcione correctamente lo primero es hacer un npm install.

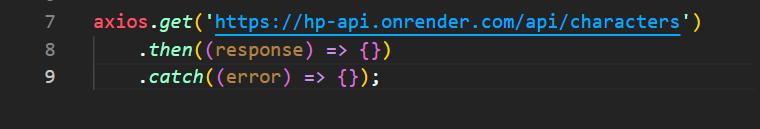
* Instalar mi segunda librería AXIOS (Peticiones WEB)
  + Terminal

Npm install axios.

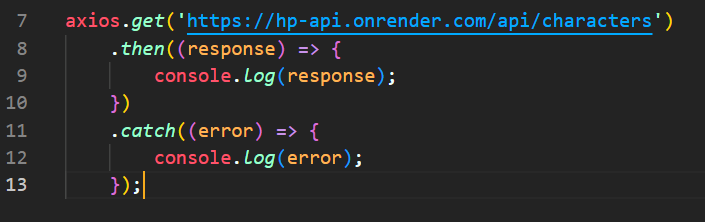
* + - Nuestro objetivo es traernos los datos, por ejemplo, del api de LOS SIMPSONS

https://api.sampleapis.com/simpsons/episodes

* Promesa
  + Si la promesa se resuelve positivamente
    - .then()
  + Si la promesa se resuelve negativamente
    - .catch()



* Consumir la promesa



* Ahora mismo nos hemos traído todo el api con sus datos.
* Ahora vamos a recoger esos datos, meterlos en un array y sacar solamente las imágenes.

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

const axios = *require*('axios').default;

console.*log*('Hola Node'.trap);

axios.*get*('https://api.sampleapis.com/simpsons/episodes')

    .*then*((response) => {

        const personajes = response.data;

        const imagenes = [];

*for*(let personaje of personajes){

            imagenes.*push*(personaje.thumbnailUrl);

        }

        console.*log*(imagenes);

    })

    .*catch*((error) => {

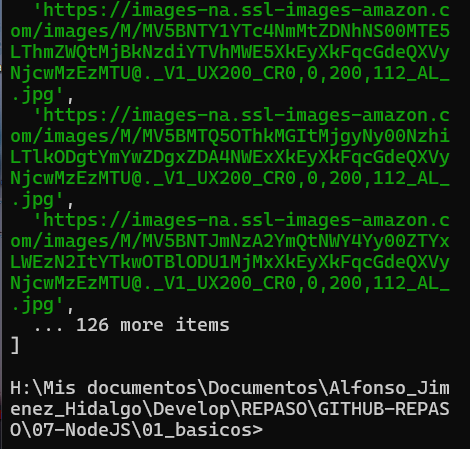
        console.*log*(error);

    });

* + Cuando trabajamos con AXIOS, a la hora de ir accediendo a los datos es desde .data, para ir profundizando e ir sacando por ejemplo las imágenes en este caso se llama .thumbnailUrl

Terminal

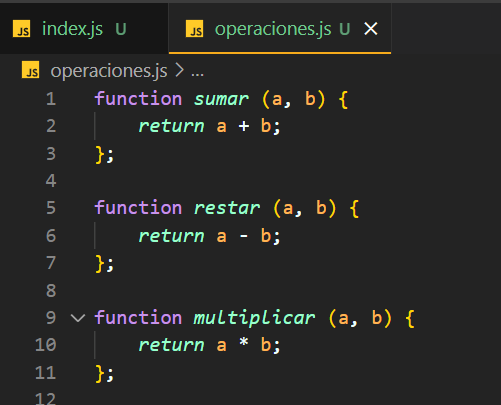
node index.js



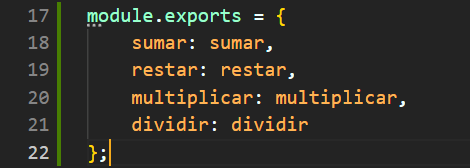
Día 1 parte II

FUNCIONES

* Vamos a separar funciones para poder tener el trabajo más organizado.
* Creamos el fichero operaciones.js
  + Creamos las funciones sumar, restar y multiplicar.



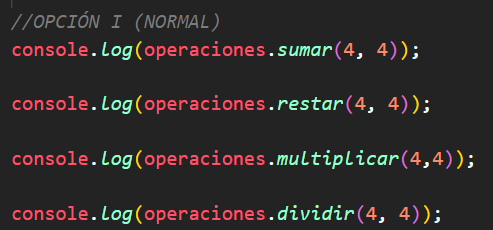
* Para que nosotros podamos utilizar estas funciones en cualquier parte del proyecto, para ello vamos a importar y exportar cada función IMPORTANTISIMO.
  + Exportar nuestras funciones operaciones.js

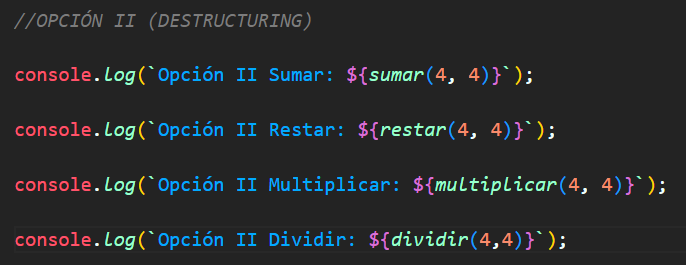


* + - Recordar -> CLAVE : VALOR
      * Clave -> nombre con el que queremos exportar
      * Valor -> Nombre exacto que exporto.
  + OPCIÓN I NORMAL .Importamos en index.js
  + OPCIÓN II DESTRUCTURING. Importamos con destructuring

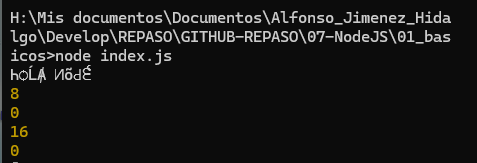


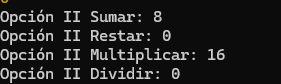
* Como lo consumimos con la OPCION I



* Como lo consumimos con la OPCION II

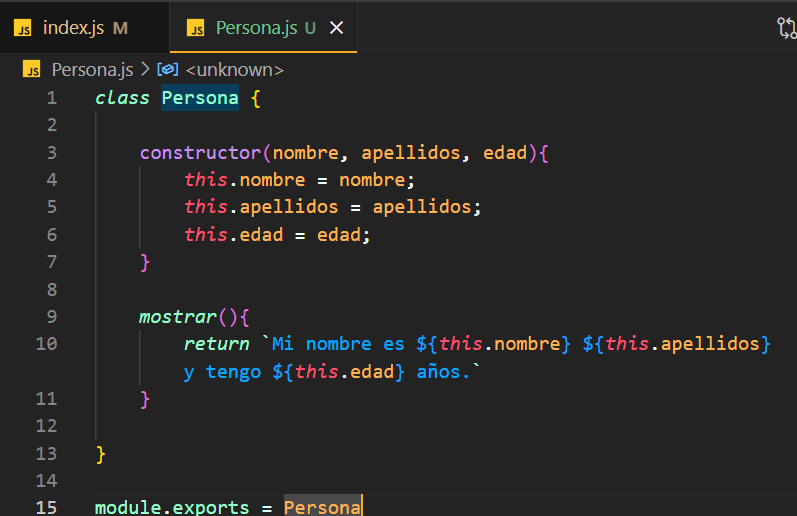
Terminal





CLASES

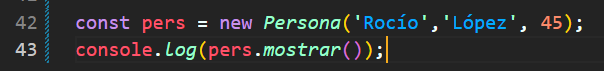
* Creamos fichero de CLASES Persona.js



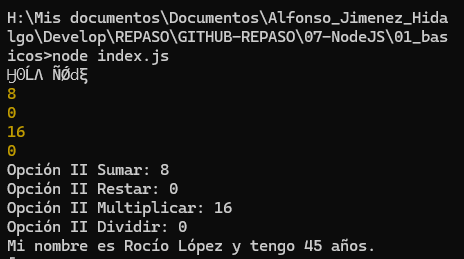
* Importamos en index.js



* + Consumimos la clase persona que es creando un objeto tipo Persona.



Terminal



LIBRERIAS PROPIAS DE NODEJS

NODE CORE (Propio de NODE)

Node tiene sus propias funciones, variable.. y no nos hace falta importar nada para utilizarlas porque ya vienen por defecto.

VARIABLES PROPIAS DE NODE

Para ver el directorio de donde se ejecuta el proyecto



Para ver la ruta del fichero donde se ejecuta el fichero.



Objeto process donde aparecen todos los procesos



LIBRERÍA FILE SYSTEM (fs)

\*PARA TRABAJAR CON EL TEMA DE ARCHIVOS\*

Nuevo proyecto 02-FILESYSTEM

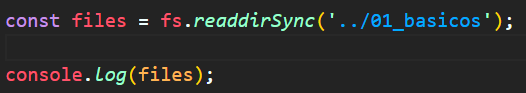
* + Terminal

npm int -y

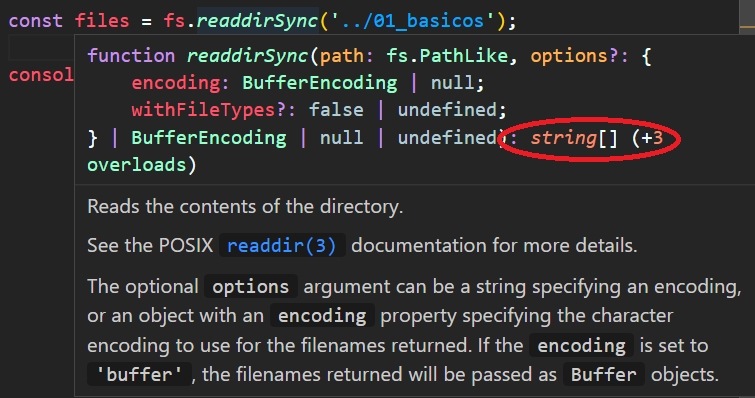
Requiero la librería fs



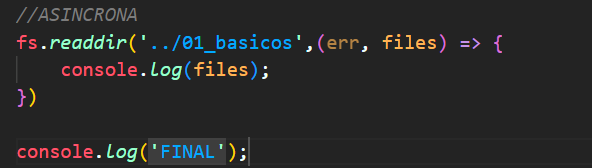
Ejecuto “SINCRONA”



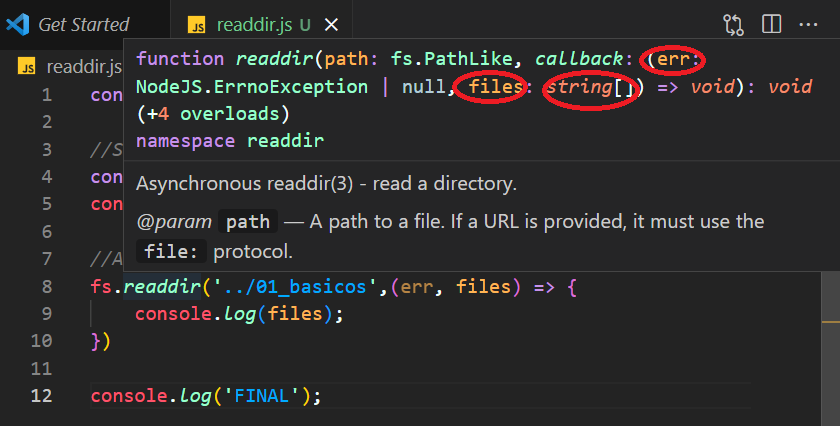
NOTA: la función (Síncrona) readdirSync Esto nos devuelve un array de string



Ejecuto “ASINCRONA”



NOTA: la función (Asíncrona) readdir tiene un callback (err, files) y nos devuelve un array de string



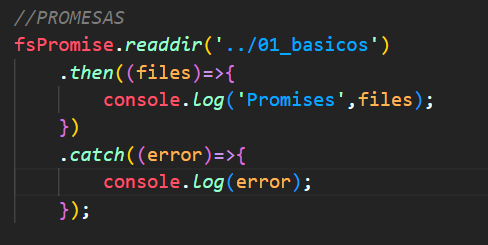
Promesas

\*TRABAJAR CON LA MANIPULACIÓN DE ARCHIVOS\*

Requiero la librería propia de promesas.



Ejecuto



NOTA: como lo voy a ejecutar con promesa, es lo mismo pero con el proceso de promesa.

.then() -> Si se resuelve

.catch() -> Si no se resuelve

REPASO DE CALLBACK

Es una función que nosotros pasamos por parámetro en la ejecución de otra función.

PROMESAS

Las promesas siempre tienen la misma gestión

* + ¿Cómo gestionarlo?

.THEN

.CATCH

O Async Await

.try

.catch

LIBRERIAS

Cualquier librería que queramos utilizar, lo primero de todo es requerir

const fs = require(‘fs’);

IMPORTANTE: Siempre que se una función o método SINCRONO hay que meterlo en una variable.

Requerir la librería para utilizar promesas

const fsPromise = *require*('fs/promises');

Async – Await

*//Delante de la ejecución de la promesa, colocamos la palabra AWAIT.*

*//2. En una variable almacenamos el mismo valor que recogeríamos del THEN*

*//3. Colocamos el ASYNC delante del ámbito de función donde estamos ejecutando la promesa.*

RESUMEN APUNTES

\*\* LEER FICHEROS \*\*\*\*\* readFile

* + - * + 4 FORMAS

Síncrona

Asíncrona

Promesas

.then() .catch()

Async - await

\*\* LEER CARPETAS \*\*\*\* readdir

* + - * + 4 FORMAS

Síncrona

Asíncrona

Promesas

.then() .catch()

Async – await

* LIBRERÍAS “GESTION DE ARCHIVOS Y CARPETAS”
  + const fs = require(“fs”);
  + const fsPromise = require(“fs/promises”)