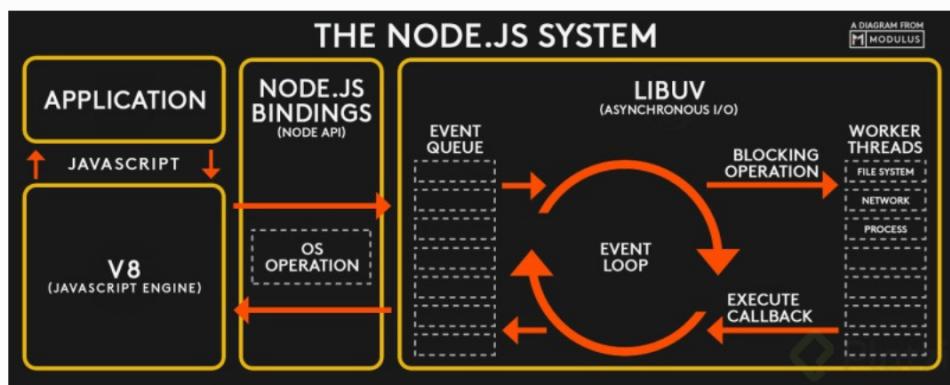


Event Loop

Es un proceso con un bucle que gestiona de forma asíncrona todos los eventos de la APP.

- NO se bloquea.



NOTA: Cuando ocurre un error dentro de los hilos y no se encuentra controlada (catch), node detiene todos los hilos de ejecución. Esto puede ser muy peligroso.

CALL BACKS:

Una función callback es una función que es pasado como argumento a otra función, para ser llamada en otro momento.

La función que recibe como argumento otras funciones es llamada función de orden superior (Higher-order function).

CALLBACK HELL:

SE DAN CUANDO SE COMPIEZA A PASAR UNA FUNCIÓN COMO PARÁMETRO QUE A SU VEZ LLAMA A OTRA FUNCIÓN COMO PARÁMETRO, Y ASÍ HASTO X. Una estrategia para trabajar con estas estructuras tan monótonas es usar ESTRUCTURAS DE CONTROL y FUNCIONES RECURSIVAS.

Las funciones recursivas se llaman así misma y mediante la estructura de control de doble cuenta veces puedo llamar.

PROMESAS

Viniendo de los callbacks, pero las promesas lo que hacen es dar un estado, son una clase global que podemos llamar de donde sea.

ASYNC / AWAIT

Es Azucar sintactico, es decir, una forma muy legible y entendible de realizar código.

Un Async/Await no deja de ser una función Asincrona. La diferencia es que al usar esta sintaxis se podra ver un código mas legible.

Para definir una función asincrona declaramos la función con `Asyme . & con await` "esperamos" a que el evento termine.

GLOBALS

Los módulos globales son módulos del Core. Es una de las funciones muy usadas en node son:

- Set Interval
- Clear Interval

En node contiene los métodos y propiedades básicas que usamos

`Global = THIS en el navegador`

`THIS == Global // TRUE`

Algunos métodos son:

- Set Time Out : Llama a otra función después de un tiempo
- Set Interval : Llama a una función cada intervalo de tiempo
- Set Immediate : Equivalente a Set Time Out pero con tiempo 0
- Clear Time Out : Detiene el setTimeout
- Clear Interval : Detiene el setInterval

FILE SYSTEM

Provee un API para interactuar con el sistema de archivos cerca del Estándar POSIX.

POSIX es el estándar para interfaces de comando shell, las palabras significan

"Interfaz de sistema operativo portátil". La X de Posix es por UNIX.

El file system nos permite acceder al archivo del sistema, leer, modificar, escribirlos.
Es muy útil para precompiladores.

" USE READFILE siempre que sea posible, READFILE SYNC bloquea el hilo mientras la
Solicitud es resuelta"

CONSOLE

Podemos imprimir todo tipo de variables:

- console.log = recibe cualquier tipo y lo muestra en la consola
- console.error = Equivalente a los pero usado para errores
- console.info = solo se usa para informar
- console.warn = solo se usa para warnings.
- console.table = Muestra una tabla apartir de un objeto
- console.count = Inicia un contador automático
- console.countReset = Reinicia el contador a 0
- console.time = Inicia el cronómetro en ms
- console.timeEnd = Finaliza el cronómetro
- console.group = Permite agrupar errores mediante indentación
- console.groupEnd = Finaliza la agrupación
- console.clear = limpia la consola.

ERRORS TRY/CATCH

cuando se genera un error, modo propaga el error hacia arriba, hasta que este es capturado, si el error no se captura modo se detiene.

Los errores se notifican por hilos, es decir, si el error sucede en el hilo principal del event loop, es decir, el evento queve, el error se avisara desde este mismo hilo.

```
TRY {  
  //...  
} catch (e) {  
  //...  
}
```

Proceso Hijo

El módulo de procesos secundarios en node.js (CHILD_PROCESS), tiene 2 funciones

SPAWN y EXEC, mediante las cuales podemos iniciar un proceso secundario para ejecutar otros programas en el sistema.

La diferencia entre SPAWN y EXEC esta en que SPAWN devuelve un stream y EXEC y buffer

- Usa SPAWN cuando quieras que el proceso hijo devuelva datos binarios enormes a node.
- Usa EXEC cuando quieras que el proceso hijo devuelva mensajes en estados simples.
- Usa SPAWN cuando quieras recibir datos desde que arranca el proceso
- Usa EXEC cuando solo quieras recibir datos al final de la ejecución.

HTTP

Node ofrece el módulo HTTP el cual nos permite principalmente crear un servidor en nuestro computador. En este módulo encontraremos todo lo necesario para crear un sistema de rutas, uno de los principales métodos es CREATE SERVER.

HTTP Status Codes Cheat Sheet

Successful Requests	
200 OK	Request was successful.
201 Created	Request was successful and something new was created based on that request.
202 Accepted	Request was received successfully, but may not be acted on immediately.
203 Non-authoritative Information	Request was successful, but information sent to the client about the response comes from a 3rd party server.
204 No Content	Request was successful, but no data is sent back.
205 Reset Content	Request from the server to reset the information sent, such as form data.
206 Partial Content	Response to a successful request for only part of a document.
Client Errors	
400 Bad Request	The server did not understand the request.
401 Unauthorized	Client must be authorized before access, typically through some kind of login.
403 Forbidden	Client does not have permission to access the requested document. If this shows up on a document that should be available, it is likely a permissions issue.
404 Not Found	Requested document was not found at the URL given and there is no new location specified for the document. Does not mean that the document is permanently missing from the given URL.
405 Method Not Allowed	Request method was not allowed for the specified document.
406 Not Acceptable	Requested document cannot be sent in a way that the client can understand.
407 Proxy Authentication Required	Client must be authorized by the proxy before the requested document can be sent.
408 Request Timeout	Amount of time for the request exceeded the amount of time that the server is set to wait for a request.
409 Conflict	Requested document could not be sent because of a conflict in the request.
410 Gone	Similar to a 404, except that it means the document is permanently gone from the URL and there is no new location specified.
411 Length Required	Request refused because content length must be specified by client.
412 Precondition Failed	At least one condition of the request has failed.
413 Request Entity Too Large	Request was larger than the server was able to handle. A common example of this would be if a file is sent through a posted form and its larger than the server settings allow for a post.
414 Request URI Too Long	The URL was longer than what the server could handle. A URL resulting in this error is normally thousands of characters long, thus it is rarely an issue.
415 Unsupported Media Type	Indicates that the format of at least part of the request is unsupported.
416 Requested Range Not Satisfiable	Request could not be fulfilled. Can occur if client requests a part of a document that doesn't exist.
417 Expectation Failed	The server could not fulfill the requirements sent in the "Expect" header field.
Redirects	
300 Multiple Choices	Response indicating that what was requested has moved or that there are multiple options that match the request.
301 Moved Permanently	Requested document was moved permanently and response contains the URL for that new location. Important to use when changing domain names or URLs of existing documents.
302 Found	Requested document was temporarily moved to a different location. The URL of the new location is sent back with the response. 303 and 307 are more specific versions of this type and were implemented as of HTTP/1.1.
303 See Other (HTTP/1.1)	Requested document found and responds with the URL where the document can be currently be found.
304 Not Modified	Requested document has not changed since the last time it was requested. Client loads the document from the cache.
305 Use Proxy	Requested document can only be accessed through a specified proxy.
307 Temporary Redirect (HTTP/1.1)	Requested document can temporarily found at a different URL, which is given in the response. This is a more pure version of what a 302 is normally meant to be.
Server Errors	
500 Internal Server Error	Generic error message meaning that something broke but nothing more specific can be sent.
501 Not Implemented	Server does not support what is required to fulfill the request.
502 Bad Gateway	Server acting as a gateway or proxy received response from an upstream server that was deemed invalid.
503 Service Unavailable	Server is currently unavailable due to high load, maintenance, or other temporary situation.
504 Gateway Timeout	Server acting as a gateway or proxy did not receive response within the amount of time that the server is set to wait for a response.
505 - HTTP Version Not Supported	Server does not support the HTTP protocol used by the client for the request.

OS

El módulo OS (OPERATIVE SYSTEM) nos permite ejecutar acciones de más bajo nivel en nuestro sistema operativo, permitiéndonos conocer una gran variedad de detalles del mismo, como la memoria disponible que tiene, el total de la memoria, la interfaz de red, etc.

PROCESS

El objeto process es una instancia del Event Emitter, podemos suscribirnos a él para escuchar eventos de modo.

- Uncaught Exception: Permite capturar cualquier error que fue capturado previamente. Este evento a que mode cierra todos los hilos al encontrar un error no manejado.
- EXIT: Se ejecuta cuando mode detiene el evento loop y cierra su proceso principal.
- BEST EXIT: Es para enviar algo antes que pare un proceso
- Uncaught Rejection: Permite capturar cualquier error de promesas que se han rechazado.

REQUIRE & IMPORT

En NODE tenemos una forma de importar módulos el cual es con el método require, el cual es la forma por defecto de importar módulos, ya sean nuestros propios módulos o como los de otros en nuestro proyecto JS. Pero suele haber mucha confusión con el Import.

Import es la forma de importar módulos en Ecma Script, el cual es un estandar para la web, Esta forma de importar módulos no lo soporta, pero gracias a babel nosotros podemos usar estos módulos de Ecma en nuestro código para cuando se ejecute, se transforme a código que sea aceptada por node.js.

MJS → Para usar Import

BUFFER

Es un espacio en memoria (RAM) en el que se almacenan datos de manera temporal. Esta forma más cruda en la que se puede almacenar los datos. (Se guardan en bytes y no especifica el tipo de dato).

Ej: HOLA => 48 6F 6C 61

BUFFER NO ALMACENA DATOS EN BINARIO Y A QUE CADA ESPACIO TIENE 2 DIGITOS.

STREAM

Es el proceso de ir consumiendo datos al tiempo en que se reciben. Por ejemplo cuando vemos un video en youtube estamos consumiendo datos por medio de stream yo que lo veo al mismo tiempo en el que este se va descargando.

Los streams son colecciones de datos, como matrices o cadenas. La diferencia es que las transacciones pueden no estar disponibles.

BENCHMARKING:

```
console.time('START PROCESS')
for (let i=0; i<1000; i++) {
    console.log(i)
}
console.timeEnd("START PROCESS")
```

BEDBUGGER:

Node.js ya viene integrado con un modo de depuración para poder conectarnos desde cualquier herramienta de inspección de código a nuestro código en nodejs.

Podemos usar en la terminal --inspect con nodejs

```
node --inspect app.js
```

ERROR FIRST CALLBACK:

Un patrón que se sigue siempre en cualquier lenguaje y programa de devs es ERROR FIRST CALLBACK, Esto quiere decir que siempre que tengamos un callback el primer parámetro debería ser un error.

```
function name(cb) {
    // ...
}

name((err, dato) => {
    // ...
})
```

SCRAPING:

Es una técnica utilizada mediante programas de software para extraer información de sitios web. Usualmente estos programas simulan la navegación de una persona en la web ya sea utilizando el protocolo HTTP manualmente o insertado en el navegador en una aplicación.