Carlos Azustre:

Simular asincronía con la función “set time out day time out en javascript nos permite ejecutar una función pasado por el tiempo.

Los datos están simulando ese retraso con set time out que recibe una función.

El segundo parámetro es el tiempo, vamos a poner 1.500 milisegundos un segundo.

Cuando se resuelva con then tendremos hay una función de callback entonces en then tendremos el primer argumento que escribe la función de cual va a ser a lo que devuelva ese reason en ese caso que los datos aquí lo puedo llamar como quieras

Asincronía en JS

Es una maneras de escribir esto pero en lugar con el punto, sería una forma más síncrona a la hora de escribir lo más secuencial pero por debajo sigue siendo asícrono sigue habiendo promesas como se haría esto. Podemos utilizar la palabra reservada “await getDatos. El resultado de Await lo podemos guardar en cualquier variable o lo que sea entonces podemos tener aquí con sus datos igual await de datos aquí, si te das la cuneta arriba podríamos acceder a los datos que devolvía a la promesa después de ejecutar delay y tendríamos una función de callback que es esta de aquí que he puesto como una función y ahí tendríamos acceso a los datos y ya podríamos lo que fuera imprimirlos y hacer una transformación, lo que sea y de esta manera queda más digamos secuencial.

Antes ya es como de hecho se puede es más legible aquí

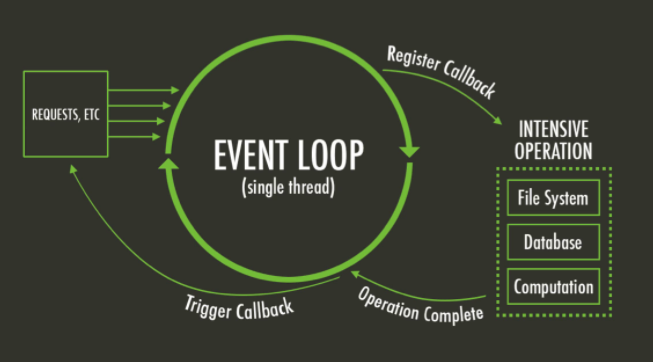
Javascript Asíncrono:

La asincronía es uno de los pilares fundamentales de Javascript, ya que es un lenguaje de programación de un sólo subproceso o hilo (single thread), lo que significa que sólo puede ejecutar una cosa a la vez.

Imagina que solicitas datos de una API. Dependiendo de la situación, el servidor puede tardar un tiempo en procesar la solicitud mientras bloquea el hilo principal y hace que la página web no responda.

Ahí es donde entra en juego la asincronía que permite realizar largas solicitudes de red sin bloquear el hilo principal.

**Javascript** usa un **modelo** **asíncrono** y **no bloqueante**, con un **bucle** de **eventos** implementado en un sólo hilo, (**single thread**) para operaciones de entrada y salida (**input/output**).



Antes de explicar como funciona el modelo de *JavaScript* es importante entender algunos conceptos:

* Procesamiento *Single thread* (Hilo único) y *Multi thread* (Hilos múltples).
* Operaciones de *CPU* y Operaciones de *I/O* (Entrada y Salida).
* Operaciones Concurrentes y Paralelas.
* Operaciones Bloqueantes y No Bloqueantes.
* Operaciones Síncronas y Asíncronas.

#### Single thread y Multi thread

Un hilo la unidad básica de ejecución de un proceso, cada que abres un programa como el navegador o tu editor de código, se levanta un proceso en tu computadora e internamente este puede tener uno o varios hilos (threads) ejecutándose para que el proceso funcione.

#### Operaciones de CPU y de Entrada y Salida

* Operaciones CPU: Aquellas que pasan el mayor tiempo consumiendo Procesos del CPU, por ejemplo, la escritura de ficheros.
* Operaciones de Entrada y Salida: Aquellas que pasan la mayor parte del tiempo esperando la respuesta de una petición o recurso, como la solicitud a una API o BD.

**Concurrencia y Paralelismo**

* Concurrencia: cuando dos o más tareas progresan simultáneamente.
* Paralelismo: cuando dos o más tareas se ejecutan, al mismo tiempo.

**Bloqueante y No Bloqueante**

Se refiere a como la fase de espera de las operaciones afectan a nuestra aplicación:

* Bloqueante: Son operaciones que no devuelven el control a nuestra aplicación hasta que se ha completado. Por tanto el *thread* queda bloqueado en estado de espera.
* No Bloqueante: Son operaciones que devuelven inmediatamente el control a nuestra aplicación, independientemente del resultado de esta. En caso de que se haya completado, devolverá los datos solicitados. En caso contrario (si la operación no ha podido ser satisfecha) podría devolver un código de error.

**Síncrono y Asíncrono**

* Síncrono: La respuesta sucede en el presente, una operación síncrona esperará el resultado.
* Asíncrono: La respuesta sucede a futuro, una operación asíncrona no esperará el resultado.

Con lo anterior en *JavaScript* podemos tener:

* Código síncrono y bloqueante o
* Código asíncrono y no bloquenate.