## Asincronia

Al momento de realizar tareas, para nuestro sitio, nosotros contamos con dos tipos de ejecucion: sincronica o asincronica.

Cuando hablamos de funciones asincronicas hablamos de tareas ejecutadas una detras de otra.Es decir que espera el resultado de una para iniciar la siguiente.

En cambio las funciones sincronas , son aquellas que hacen multiples tareas a la vez, y no requieren del resultado de una previa.

Para poder hacer que las acciones se ejecuten de forma consecutiva, hace falta poner en espera el inicio de la siguiente tarea para que aguarde el resultado de la anterior.

Para ello, contamos con tres formas de hacerlo:

-Por medio de una funcion **Callback**

-Por medio de **Promesas**

-Por medio de las palabras **async** y **await**

En este punto solamente veremos las dos primeras formas.

## CALLBACK:

Llamamos ***callback*** a funciones que queremos que se ejecuten al finalizar una tarea. Este tipo de funciones son argumentos de una funcion.Esto nos permite anidar tareas y no tener que preocuparnos de tener que esperar que termine una para arrancar la siguiente, ya que de forma automatica van a ejecutarse al finalizar la tarea inicial.

Tener en cuenta que esa funcion callback es igual que cualquier otra funcion y puede tomar el nombre que querramos tomarla, callback no es mas que una definicion para indicar que esas funciones se ejecutarn al finalizar otra.

```
function myDisplayer(info) {
    let texto= document.querySelector("p");
        texto.innerText=info;
};

function myCalculator(num1, num2, myCallback) {
    let sum = num1 + num2;
    myCallback(sum);
};

myCalculator(5, 5, myDisplayer);
```

En este ejemplo podemos ver que la funcion myDisplayer() se va a comportar como la funcion callback de myCalculator.
Esto quiere decir que solo se ejecutara myDisplayer una vez hecha la suma .

Vamos a ver que esto por ahora no tiene mucho sentido para que utilizar los callback, pero pensemos cuando tenemos que llamar informacion de otro lado, necesitamos esperar a que vuelva esa informacion para poder manipularla, si ejecutariamos todo seguido se produciria un error porque va a querer manipular esa informacion que no tiene .

## PROMESAS

Las ***promesas*** son un concepto para resolver el problema de asincronía de una forma practica; y como su nombre lo indica, es algo que, en principio pensamos que se cumplirá, pero en el futuro pueden ocurrir que no se cumpla o que quede en un estado incierto indefinidamente .

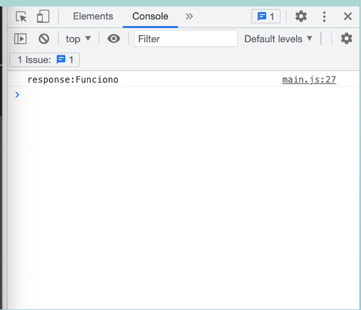
Con esto en mente podemos decir que una promesa no es mas que un objeto devuelto al cuál se adjuntan funciones callback segun el resultado de ese objeto.

Las promesas hacen uso de ***metodos*** que nos permitiran armar la estructura necesaria para indicarle al programa como actuar tanto en el caso que la promesa se cumpla , como en caso que halla algun error por lo que la promesa no se cumpla.

Estos metodos son **.then() ,**que contiene la funcion callback a ejecutar en caso que la promesa se cumpla; y **.catch() ,** que contiene la funcion callback en el caso que surja algun error o falla.

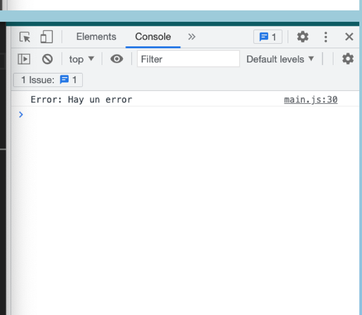
```
let promesa = new Promise((resolve,reject)=>{
 let todoOk=true
    if (todoOk == true) {
        resolve("Funciono")
    } else {
        reject('Hay un error')
    }
});

promesa
    .then((response)=>{
    console.log("response:" + response);
    })
    .catch((error)=>{
        console.log('Error:',error);
    })
```




```
let promesa = new Promise((resolve,reject)=>{
 let todoOk=false
    if (todoOk == true) {
        resolve("Funciono")
    } else {
        reject('Hay un error')
    }
});

promesa
    .then((response)=>{
    console.log("response:" + response);
    })
    .catch((error)=>{
        console.log('Error:',error);
    })
```




Como vemos en el siguiente ejemplo estamos ejecutando una promesa en donde vemos que el metodo **.then()** ejecuta la función callback **resolve()** cuando la promesa se cumple ; y el metodo .**catch()** ejecuta la función callback **reject()** cuando la promesa no se cumple

En este caso la promesa se encuentra en el condicional **if()** donde evalua si la variable **todoOk** es **true**