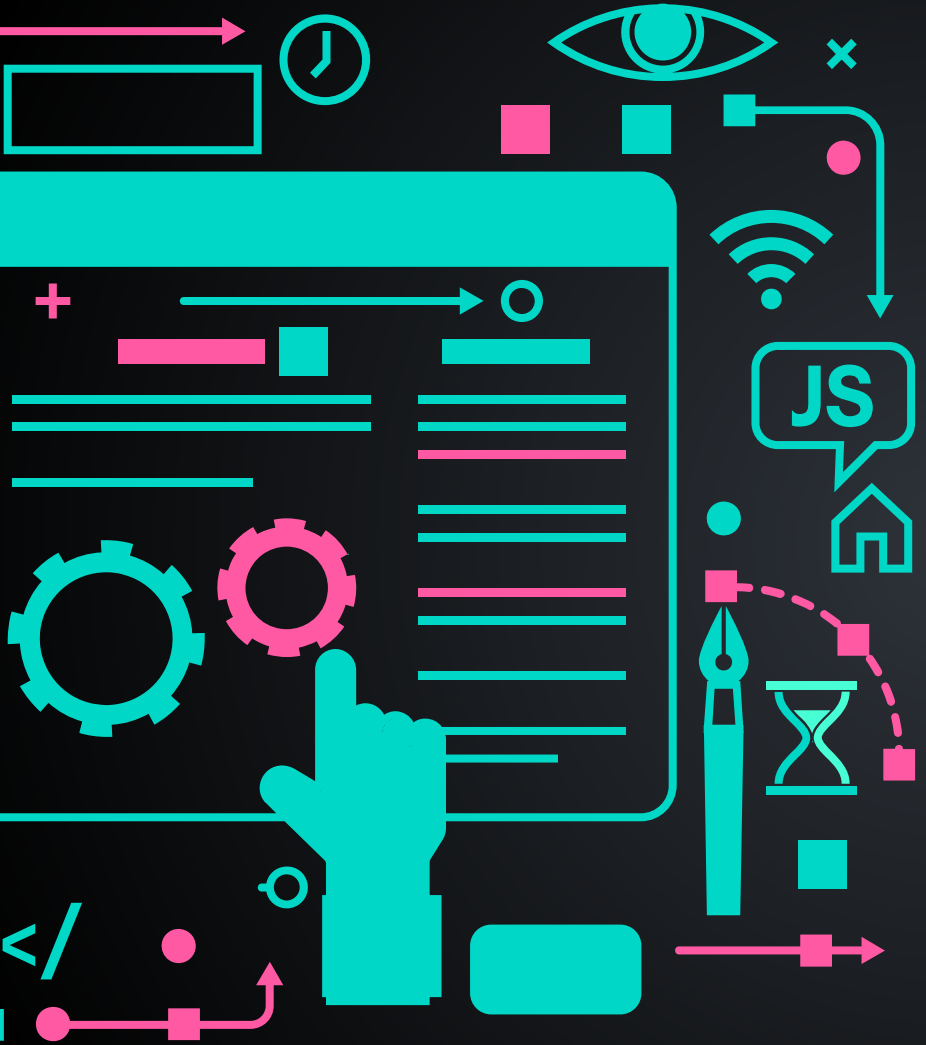
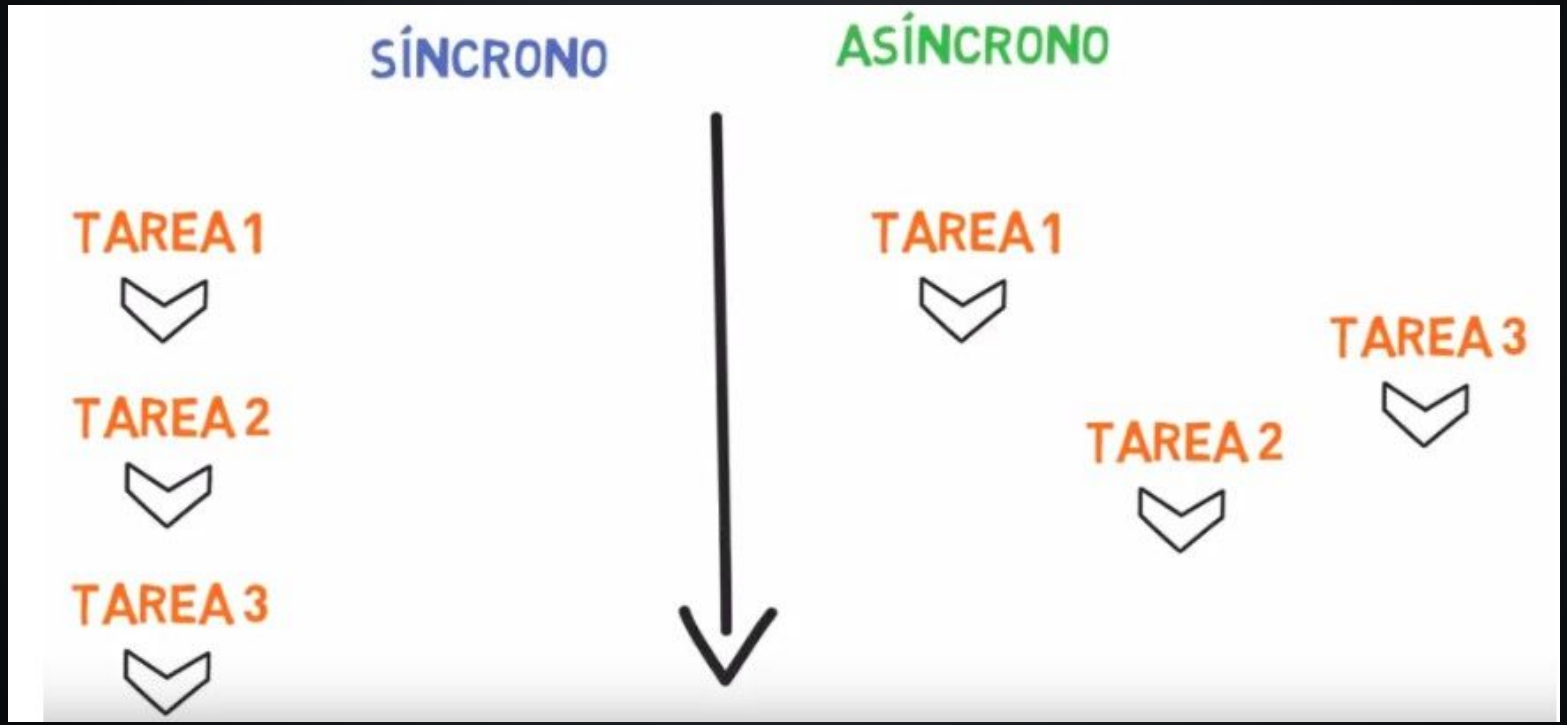


Javascript: Procesos asíncronos

Procesos síncronos y asíncronos



¿Cómo resolvemos los procesos asíncronos ?

- Con el uso de callbacks
- Con el uso de promesas
- Async y await

¿Qué son los callbacks?

Una función de callback es una función que se pasa a otra función como un argumento

```
const saluda = (callback) => {
  ...
};

saluda((resultado) => {
  console.log(resultado);
});
```

Ejercicio

- Crea una función con el nombre saludar la cual recibe un parámetro de tipo string y otro que será una función callback.
- Dentro de la función saludar, el parámetro de tipo string debe ser convertido a mayúsculas y se deberá pasar como argumento al callback
- En la función callback, recupera el resultado e imprimelo en consola.

Ahora bien manda llamar tu función saludar

Ejercicio 2

- Del ejercicio anterior, agrega en la función saludar un setTimeOut de 4000(4 seg).
- Dentro del setTimeout debes agregar el resultado que se pasa al callback
- Mandar llamar la función saludar
- Debajo de la función saludar agrega un console.log el cual debe decir final.

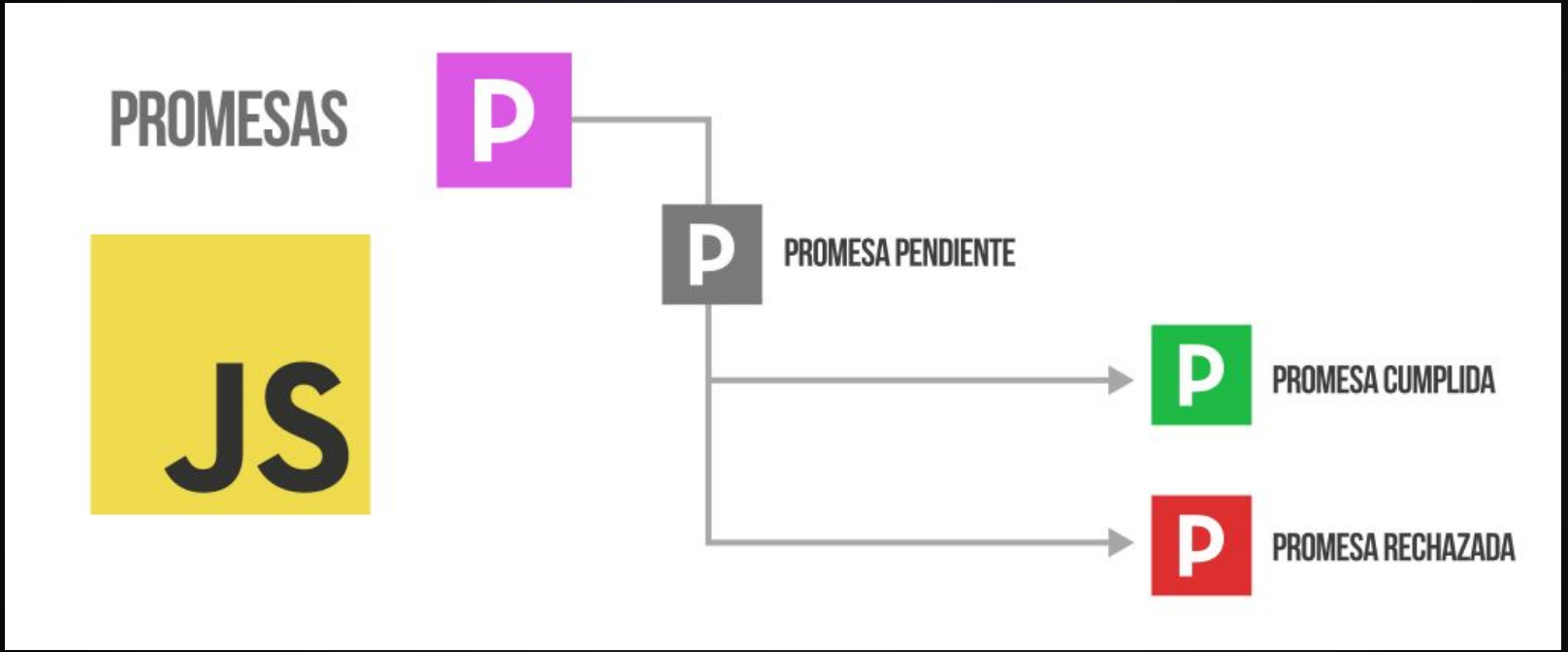
Dinos cual es el último mensaje mostrado en consola

Promesas

¿Qué son las promesas?

Las promesas son un concepto para resolver el problema de asincronía de una forma mucho más elegante y práctica

Promesas


PROMESAS
JS
P
P
PROMESA PENDIENTE
P
PROMESA CUMPLIDA
P
PROMESA RECHAZADA

Crear una promesa

```
const resolver = new Promise((resolve, reject) => {
 ....Código
resolve();
reject()
});

resolver
 .then((resultado) => {
   Si la promesa se resolvió correctamente
 })
 .catch((error) => {
   Si la promesa no se resolvió correctamente
 })
 .finally(() => {
   Se ejecuta cuando finaliza la promesa sin importar su resultado
 });
```

Métodos de una promesa

- then(). Indica si una promesa se resolvió de forma correcta
- catch(). Indica si una promesa no se resolvió de forma correcta
- finally(). Indica cuando una promesa ha finalizado independientemente de tu estatus

Ejercicio 3

- Crea una promesa con el nombre de saludar
- Dentro de la promesa agrega un setTimeOut de 4000(4 seg).
- Dentro del setTimeOut, debes agregar en el resolve un string que diga Hola mundo. Soy llamado desde una promesa.
- Por medio de la función then obtén el resultado de la promesa e imprímelo en un console.log

Ejercicio 5

- Del ejercicio anterior, ahora en el setTimeOut manda llamar reject y pasa como argumento. Ha fallado la promesa
- Ahora por medio del método catch, obtén el valor resuelto por parte de la promesa e imprimelo con un console.log

Ejercicio 6

- Crea una promesa que se llame status. Dentro de la promesa deberás crear una variable que se llame valor la cual debe ser igual a 5.
- Dentro de la promesa deberás validar si la variable valor es mayor o menor a 5.
- En caso de ser mayor deberás llamar al método resolver con el resultado Es mayor a 5.
- En caso contrario deberás llamar al método reject el cual y pasar el argumento Es menor a 5

Promise all

Devuelve una promesa que termina correctamente cuando todas las promesas en el argumento iterable han sido concluidas con éxito

```
Promise.all([Array de promesas]).then((resultado) => {
  ...
});
```

Promise en cascada

Asyn await

Async define una función asíncrona

La palabra clave await hace que JavaScript espere hasta que esa promesa se establezca y devuelva su resultado.

```
async function miFunction(){
  const resultado = await miPromesa
}
```

Ejercicio 8

Ver repositorio

Ejercicio grupal

Ver repositorio