Promesas y Callbacks

dev.f

desarrollamos(personas);

¿Que es un Callback?

- Es una función que recibe como parámetro otra función.
- La función “callback” por lo regular va a realizar algo con los resultados de la función que la está ejecutando.
- Es una forma de ejecutar código de forma “asíncrona” ya que una función va a llamar a otra.

Ejemplo

```
function successCallback() {
// Do stuff before send
}

 function successCallback() {
// Do stuff if success message received
}

 function completeCallback() {
// Do stuff upon completion
}

 function errorCallback() {
// Do stuff if error received
}
```

Ventajas y desventajas de los callbacks

Ventajas

- Son faciles de usar.
- Pueden solucionar problemas de flujo de una aplicación
- Ayudan a manejar excepciones.
- Son útiles cuando quieres hacer consultas a una BD o servicio web

Desventajas

- A Veces el concepto es confuso
- Si se usa demasiado se puede caer en algo denominado “callback hell”
- El uso excesivo puede afectar el performance.
- Para programadores novatos no es muy fácil leer y entender qué hacen las funciones callback.

Promesas

- Es usado para interacciones asíncronas
- Se compone de dos aspectos:
  - Resolve: Se ejecuta cuando el objetivo de la promesa se efectuó de manera correcta
  - Reject: Se ejecuta cuando el objetivo de la promesa ocasionó un error o no se llegó a cumplir.
- Se utiliza la palabra reservada ***“Promise”***
- En las promesas se tienen siempre tres estados:
  - Pendiente: Estado inicial de la promesa
  - Resuelta: Todo se ejecutó correctamente
  - Rechazada: Hubo un problema

En otras palabras..

dev.f

Ejemplo

```
const promise = new Promise((resolve, reject) => {
    const number = Math.floor(Math.random() * 10);

    setTimeout(
        () => number > 5
            ? resolve(number)
            : reject(new Error('Menor a 5')),
        1000
    );
});

promise
    .then(number => console.log(number))
    .catch(error => console.error(error));
```

Ejemplo

```
function randomDelayed(max = 10, expected = 5, delay =  1000) {
    return new Promise((resolve, reject) => {
        const number = Math.floor(Math.random() * max);

        setTimeout(
            () => {number > expected ? resolve(number):
                reject(new Error('número menor al esperado'));}, delay
        );
    });
}

randomDelayed(100, 75, 2500)
    .then(number => console.log(number))
    .catch(error => console.error(error));
```