**Vue**

**Promesas.**

**Documentación:**

[**https://v3.vuejs.org/guide/introduction.html**](https://v3.vuejs.org/guide/introduction.html)

[**https://vuejs.org/v2/guide/**](https://vuejs.org/v2/guide/)

Las promesas no es mas que una tarea asíncrona, este tipo de tareas nos permite ejecutar un código en diferentes momentos en el tiempo. Para las promesas no hay que importar nada, porque las promesas es parte de JS.

**Una promesa se compone de la siguiente manera:**

1. Recibe un callback o alguna función que se va a ejecutar, esto es conocido como el cuerpo de la promesa.
2. Usualmente los argumentos que recibe tienen el nombre de resolve y reject.
   1. Resolve: Todo sale bien.
   2. Reject: Ver si la promesa disparó algún tipo de error.
   3. Tanto el resolve como el reject son funciones.

**Así luce una promesa. Ejemplo:**

console.log("Inicio");

new Promise( (resolve, reject) => {

    console.log("Cuerpo de la promesa");

});

console.log("Fin");

Hay que tener algo en cuenta, y es que el cuerpo de la promesa se ejecuta inmediatamente, pero la resolución nosotros lo podemos manejar de manera independiente o por defecto se maneja en un hilo independiente. **Aunque en realidad lo que hará JS va a ejecutar el cuerpo de la promesa y su resolución la va a colocar en una pila de callbacks, para que cuando termine de hacer el procedimiento principal, entonces va a llamar las promesas o lo que se encuentre en esta pila de resoluciones o de promesas y las va a ejecutar en el mismo orden en el cual fueron ingresadas a esa pila.**

Cuando tenemos una promesa, al ejecutar la función de resolve que se refiere a que todo salió bien, podemos poner el “**.then()”** esto hace referencia a cuando se hace correctamente la promesa, este .then() puede recibir una serie de argumentos opcionales y esto será todo lo que se le haya pasado por la función resolve. Aunque se recomienda que sean un máximo de 3 argumentos y si serán mas de 3 argumento se recomienda enviar un objeto.

**Ejemplo:**

console.log("Inicio");

new Promise( (resolve, reject) => {

    console.log("Cuerpo de la promesa");

    resolve({estado:"Ok", mensaje: "Promesa resuelta correctamente"});

}).then( ({respuesta}) => {

    console.log(respuesta); // {estado: 'Ok', mensaje: 'Promesa resuelta correctamente'}

});

console.log("Fin");

**Nota importante.**

Aquí empieza lo interesante porque en la salida a consola primero tenemos el inicio, después el cuerpo de la promesa y luego tenemos el fin a pesar de que en nuestro código primero esta .then() de la promesa. La razón de esto, es porque el código del .then() se va ejecutar hasta que el procedimiento o el hilo principal de nuestro archivo sea terminado, termina el archivo de ejecutarse y luego JS va a ir a revisar en la pila de las promesas resueltas o también de los que errores que hayan sucedido, va a revisar si hay algo pendiente de ejecutar, que serían los callbacks “las funciones de los .then()”.

Para atrapar el error de la promesa, utilizaremos un .catch() donde por medio de sus argumentos recibiremos la información de las variables.

**Ejemplo:**

console.log("Inicio");

new Promise( (resolve, reject) => {

    console.log("Cuerpo de la promesa");

    const exito = false;

    if( exito )

        resolve({estado:"Ok", mensaje: "Promesa resuelta correctamente"});

    else

        reject({estado:"Error", mensaje: "Promesa resuelta con error"});

}).then( (respuesta) => {

    console.log(respuesta); // {estado: 'Ok', mensaje: 'Promesa resuelta correctamente'}

}).catch( (error) => {

    console.log(error); // {estado: 'Error', mensaje: 'Promesa resuelta con error'}

});

console.log("Fin");

**Tip JS.**

Por último cuando nosotros tenemos una función o callback en la resolución del éxito o error de la promesa, si esos argumentos los queremos mandar a una función que se ejecuta dentro de la función de la resolución de la promesa, entonces se puede resumir, sin especificar los argumentos y solo poniendo el puro nombre de la función. Ejemplo:

// Promesas

console.log("Inicio");

new Promise( (resolve, reject) => {

    console.log("Cuerpo de la promesa");

    const exito = false;

    if( exito )

        resolve({estado:"Ok", mensaje: "Promesa resuelta correctamente"});

    else

        reject({estado:"Error", mensaje: "Promesa resuelta con error"});

})

.then( console.log ) // {estado: 'Ok', mensaje: 'Promesa resuelta correctamente'}

.catch( console.log ); // {estado: 'Error', mensaje: 'Promesa resuelta con error'}

console.log("Fin");

En este ejemplo la resolución de la promesa se mandará directamente a la función del console.log, esto hará que tengamos menos líneas de código.

**Promesas con argumentos.**

import {obtenerHeroePorID} from './bases/08-imports-exports';

const obtenerHeroPorIdAsincrono = ( id ) => new Promise( (resolve, reject) => {

    setTimeout(() => {

        const heroe = obtenerHeroePorID( id );

        if( heroe )

            resolve( heroe )

        else

            reject("El héroe no existe");

    }, 1000);

});

obtenerHeroPorIdAsincrono( 4 )

.then( (heroe) => {

    console.log( `El héroe es: ${heroe.nombre}` );

})

.catch( console.log )