Es momento de hablar sobre las promesas en JavaScript pero vale la pena decir que las ponencias es un

tema bastante amplio.

Lo que yo quiero darles acá por lo menos es una introducción de cómo se utilizan y cómo funcionan.

Por lo menos de forma básica las usaremos bastante en el curso y eso les va a ayudar a ustedes a afianzar

esos conocimientos pero las promesas es un tema bastante grande que podrían ser varias clases para hablar

de ellas.

Pero expliquemos por lo menos lo básico que a mi me interesa que ustedes comprendan las promesas básicamente

no sirven para ejecutar un código sin bloquear el código de mi aplicación.

Eso sonará rarísimo.

Déjenme mostrarlo como un ejemplo.

Voy a hacer aquí un lo que diga inicio y al final voy a poner otro conchudos que diga fin y aquí en

medio voy a definir una promesa.

Vamos a ver cómo lo hacemos.

Voy a definirlo con la palabra reservada const como si quisiera crear cualquier otra constante o variable

se va llamar PROM EPROM uno de promesa y esto va a ser igual Agnew New promesas parentesis punto y coma.

Aquí posiblemente les esté marcando un error que dice que la promesa solo se refiere un tipo pero estado

o está siendo usado aquí como si fuera bla bla bla.

Y esto es porque las promesas realmente no tienen una forma de traducirse directamente al ECMAscript

5 sin necesidad de tener una librería que me permita a mí procesarlas.

O sea Tapscott no puede pasar la promesa a su sinónimo del ECMAscript 5 porque literalmente no hay nada

que se pueda hacer en ese caso por lo cual habría que implementar una Libreville de terceros para hacer

más o menos el funcionamiento de una promesa.

Pero ustedes no se deben de preocupar de eso porque estamos trabajando con angular angular de instada

Poly Fils que son básicamente esas librerías o funciones que me va a permitir a mí trabajar con promesas

en otras versiones de JavaScript pero para fines educativos les voy a pedir que vayamos aquí al diez

config puntu Jayson y cambiemos el estándar no vamos a apuntar el 5 vamos a apuntar al 6 el estándar

del ECMAScript 6 es el que incorporó las promesas por las cuales sí vamos a poder trabajarlas directamente.

Entonces pongo aquí ECMAScript 6 cierran y acá ya deberíamos de tener otro ROP.

El error aquí ya me dice que se esperaba por lo menos un argumento pero se recibió cero.

Ok esto parecerá un poco extraño aquí adentro de la promesa.

Necesito mandar una función que recibe dos argumentos.

Voy a poner aquí una función de flecha orillero enrollables.

Eso es lo que hice.

Dentro de estos argumentos necesito mandar dos funciones una que se llama usualmente rizó y otra que

se llama reject.

El free shop es lo que nosotros vamos a retornar cuando todo funciona cuando todo sucede correctamente

y el reject es el que vamos a llamar si sucede algún error.

Las promesas son bastante utilizadas también cuando se hacen peticiones a servicios web y ustedes necesitan.

Bueno si se resuelve toda la información correctamente llamará al resort y si algo sale mal llaman el

billete para manejar el error.

En este caso vamos a hacer algo sencillo voy a colocar aquí en un set un septima out que ya habíamos

visto y poner en un segundo y después del segundo lo único que quiero hacer es llamar el rizó con algún

mensaje que diga se terminó el Time Out.

Ok estoy llamando el rizó y estoy mandándole un argumento puntico.

Todavía no estamos usando el reject perfecto.

Ahora bajemos un poco justo antes del fin que habíamos puesto acá.

Después de crearme la promesa Voy a llamar a esa promesa prom 1 punto.

Y aquí ustedes van a notar dos cosas.

De hecho deberían aparecer tres.

Pero bueno tenemos dos cosas uno que es el de.

Y el otro que es el catch lo que es el símbolo es básicamente un identificador.

Pero esto no lo vamos a utilizar a lo largo del curso o menos todavía no la parte del dén.

Es lo que yo quiero ejecutar cuando se realiza todo exitosamente y el catch es lo que yo quiero ejecutar.

Cuando sucede un error en este caso voy a seleccionar el de parentesis y yo recibiría el valor que estoy

mandando acá como por ejemplo el mensaje mensaje mensaje ok el mensaje opera una función de flecha con

el punto del mensaje no lo voy a poder puntico más todavía.

Noten que el mensaje es de tipo no quiere decir que taques que no puede inferir el tipo de manera dinámica

aunque aquí estamos revisando un string y se puede ver que quizá mandando un no pero otros sabemos que

eso son string no logra identificar por porque nosotros vamos a tener que definirle cuál es el tipo

de lo que resuelve esta promesa ya lo vamos a ver en breve en el peor de los casos ustedes pueden poner

esto entre paréntesis.

Poner aquí que es de tipo String.

Si ustedes quieren saber el tipado estrictamente aquí me está marcando que el 1 no es asignarle de tipo

string ya vamos a resolver esto en breve.

No se preocupe yo voy a grabar los cambios y vamos a ver qué sucede.

El punto en se va a disparar cuando se resuelva la promesa Voy a grabar los cambios voy a recargar el

navegador web note que aparece rápidamente inicio y fin un segundo después se terminó el taimados genial.

Si yo regreso acá noté que se disparó primero al inicio se definió la promesa se empezó a ejecutar la

promesa pero como este código no es bloqueante entonces siguió ejecutando el fin y un segundo después

cuando se resuelve el tema se dispara el dén excelente.

Pero qué pasaría si en lugar de llamar el Bristow llamo el billete pero lo escribimos bien el reject

es este de acá voy a grabar los cambios voy a revisar navegador web recargo y se dispara el inicio del

fin y un segundo después se dispara un error y lo importante es que noten que es un coff es decir no

se atrapo el error el no manejar errores en las promesas puede detener el código de su programa por

lo cual es importante que nosotros manejemos el caché es decir lo que puede ser que suceda si se dispara

un error y en este caso lo que voy a hacer acá es bajar un poco justo después del dén presionaron enter

y poner el punto cache parentesis aquí recibiríamos un error función de flecha y Conso punto del error.

Ahora sí pero en vez de un solo pongamos un Consol Worm para que se mire en amarillo.

Los cambios entonces básicamente sería ejecutar esa línea de código rápidamente en la imprime establece

y crea la promesa Yo se podría pensar que es como una función asíncrona.

Luego de definir la promesa aquí abajo ya la estoy ejecutando aquí le estoy diciendo por favor inicia

el proceso con el punto.

Pero como el proceso no responde inmediatamente sigue ejecutando la línea del código.

Encuentra el Consol 2.5 del delfín lo imprime y quedó esperando la resolución o el error de mi promesa.

Agravarlos cambio ser del navegador web vamos a ver inicio y fin rápidamente y luego se terminó el Time

Out pero sale en amarillo porque lo estoy manejando aquí el error y no sale un COD o sea aquí lo estoy

manejando yo sé que esta clase tal vez fue un poco extraña pero el concepto de las promesas es algo

indispensable en el uso de JavaScript de hoy en día.

En la siguiente clase voy a hacer otro ejercicio de las promesas foro para que tenga algo más de sentido

porque este ejercicio es muy abstracto pero por lo menos nos ayuda a ver cómo funcionan las promesas.

Entonces voy a copiar el PP voy a pegar aquí adentro.

Voy a nombrar esto.

Promesas 1.

Promesas.

1 0 1 Enter.

Y en la siguiente clase vamos a trabajar con o con las promesas de otra manera.