Nuestra aplicación está casi terminada, aunque hay muchas cosas que nosotros podríamos hacer, como

implementar Low Dings, entre otras cosas.

Pero hay un tema muy importante que si quiero tocar con ustedes, vamos a revisar esta pantalla, propiamente

la de película.

Voy a regresar aquí.

Y cuando estamos leyendo el argumento del BRL donde obtenemos el Heidy de la película, hacemos dos

llamados a dos servicios a lo que es nuestra película.

Dos al detalle la película y a obtener el cast.

Ambos son observables y lo podemos saber porque aquí tenemos un subscribe.

Aquí tenemos otro subscribe.

Entonces, en este caso, si está trabajando es de manera independiente.

Es decir, este servicio puede que dé un error.

Puede que no, o este puede que dé un error o no, o puede que esto demore mucho más o no que o que

este sea mucho más rápido que el otro.

Y esto va a ser que primero se cargue un pedazo y luego se cargue el otro pedazo.

Y bueno, ustedes me entienden.

Hay muchas cosas o muchos panoramas que pueden variar dependiendo de la velocidad, internet y del servidor.

Entonces, qué les parece si aprovecho el momento de explicarles algo genial que tiene RX JS para combinar

observables y que este código quede aún más corto?

Bueno, que quede corto porque la verdad es que es mucho código.

Vamos a ver cómo lo resolvemos.

Voy a presionar un par de.

Bueno, adentres aquí abajo y hacer el código aquí.

Escriban el siguiente código Combined leares Tab.

Yo sé que aquí está marcado como obsoleto, pero es la forma como se usa.

Si yo lo uso directamente con un observable perdón como un operador, eso está obsoleto.

Pero la firma que vamos a utilizar funciona perfectamente.

A ver Whipper y administrar Yards y se quita el obsoleto o la depreciación.

Aquí adentro ustedes van a colocar cualquier cantidad de obsérvalos.

El convine leres es otro observable como ustedes pueden ver aquí tengo un suscrip y aquí yo tendría

la información de la.

Bueno, voy a poner aquí una un objeto, voy a poner ese objeto función de flecha y lo voy a imprimir

en consola para que lo miremos ok?

Entonces el convoys leer recibe una cantidad X de observables y regresa un objeto que en realidad es

un arreglo con todas las respuestas de los observados cuando ya han emitido por lo menos un valor todos.

Y eso es genial.

Es exactamente lo que yo necesito.

Entonces podría cortar todo este código hasta aquí antes de subscribe y lo voy a colocar aquí adentro

porque esto es un observable.

Si yo lo dejo así, es un observó anónimo, o sea, sin nombre.

Ver eso está bien.

Y voy a hacer lo mismo.

Copiar esto hasta antes de suscrip y lo voy a pegar aquí adentro.

Todo lo demás.

Bueno, todo este código que está aquí lo puedo comentar porque ya no va a funcionar de esa manera.

Obviamente no vamos a ver los detalles, pero sí en consola.

Deberíamos de ver los resultados de ambos.

Ya van a ver.

Aquí tenemos la película como tal.

El primer, la primera posición de este observador del Convent Lainez regresa a la película y el segundo,

el cast.

Entonces, en pocas palabras, si ustedes pusieran a primero el cast, entonces la primera posición

va a ser el cast y luego la película.

Y esto también es genial, porque les ayuda a ustedes a saber cuál es el primero ya identificados.

Ahora yo podría hacer aquí objeto, punto o película y en el segundo objeto punto Kaz.

Pero qué pereza, qué pereza!

Podemos hacer lo siguiente Voy a poner aquí un paréntesis y voy a desestructurar.

O esa es la desdesu tud desestructuración de arreglos y de extraer de ahí lo que es el primer argumento

que se va a llamar película.

Y el segundo se va a llamar cast.

Si iba con consol loga ahora de película, Comcast gravó los cambios.

Regreso aquí, van a ver que ahí ya los tengo.

Y yo tengo mi película y aquí tengo mi cast.

Fácil, súper fácil.

Lo único es que esta desestructuración de argumentos es un poco rara la primera vez, pero es muy fácil

cuando ya produce mucho código, cuando ya sabemos.

Otra cosa sería que ustedes lo hicieran a la manera tradicional que aquí tendríamos sujeto y tendríamos

que si yo e la película que sería igual al objeto object objeto, voy a ponerle VJ o BJ en la posición

cero y luego tendríamos la película o el cast que sería en la posición 1.

Todo esto que acabo de escribir es literalmente lo mismo que hacer la desestructuración en el argumento

es lo mismo.

Y ahora hagamos lo siguiente hagamos las mismas reglas de validación.

Por ejemplo, esta de aquí.

Déjenme copiarlas.

Voy a pegar acá.

Le puse, le puse Booby y entonces cambiémoslo por película o muy o como ustedes quieran ponerle.

Entonces tengo aquí si la película no viene, le saco el zoom y listo.

Pero si tengo la película, entonces ya se la seteo a la propiedad que tengo en la clase.

Ahora voy a hacer lo mismo con el cast percas.

Solo voy a copiar esta línea y la voy a pegar aquí abajo.

Entonces, de esta manera, todo este código equivale a todo el código que tenemos acá.

Es mucho más simple de leer esto estoy de comprender y si tuviéramos aún más, observamos.

Es más fácil trabajar con esto que hacer muchas suscripciones independientes.

Voy a grabar los cambios y ustedes van a observar que la aplicación funciona exactamente igual.

Ok.

Todo funciona exactamente igual.

Bueno, eso sería todo lo que quería cubrir por este lado.

Y nos vemos en el siguiente video.