Venimos de el método for each y vamos hacia muchos más porque los arreglos tienen muchos métodos, podemos utilizar muchas funciones pre construidas para trabajar, para iterar sobre los datos de un arreglo, de veras hay muchos, vamos a ver los principales, además de for each son map y filter pero hay muchos otros, algunos de los cuales vamos a ver también aquí. Antes de continuar con más métodos de los arreglos arrays o vectores vamos a chequear que efectivamente hayan entendido esta línea, esto que llamamos template strings o string interpolatión básicamente consiste en utilizar back tics que son estas comillas invertidas, está del principio y está del final para poder mezclar texto con variables, esto es exactamente lo mismo que escribir esto que van a ver ustedes acá, voy a comentar esta línea y ahora esto vamos a escribirlo como si fuera, digamos, la manera más tradicional o antigua, comillas, aquí estamos creando un elemento de tipo "p", luego +, el nombre de una variable, otra vez + y otra vez entre comillas un elemento "p,"si ahora guardamos ustedes verán a la izquierda que no cambia nada porque esto funciona exactamente igual, aquí tenemos un texto que es interpretado por el método wright del document, como un elemento de tipo "p", ok, y luego tenemos una referencia a una variable que es esta variable que estamos iterando en cada pasada, en cada recorrida del for each. Escribirlo de esta manera pues nos puede resultar un tanto más sencillo o no, depende, ustedes pueden utilizar lo que quieran, aquí me evito tener que andar controlando cuántos espacios en blanco hay, con los + cerrar comillas, etcétera, pero estamos haciendo exactamente lo mismo, quiero que eso quede claro, ahora sí, vamos a continuar con los métodos de los arreglos, veamos unos cuantos, hay muchos, vamos a ver bueno pues unos cuantos reitero. Comencemos por el método find de un array que va a encontrar si un valor efectivamente existe en un arreglo o no, podríamos probarlos aquí debajo, podríamos escribir por ejemplo top songs.find, bueno el primero es el método include en realidad, aquí, creo que ya lo utilizamos para un ejercicio pero no está de más revisarlo, vamos a ver si de top songs incluye por ejemplo aquí un string, busquemos y lo escribimos exactamente igual, vamos a hacer de esto un console-log, oops, vamos a escribirlo y esto nos va a devolver el elemento si es que lo encuentra, vamos a abrir aquí la consola de desarrollo para ver los resultados, si nos devuelve true, ok, vamos a limpiar aquí, perdón vamos a limpiar recargando nuestra página, ahora sí, corremos nuevamente y nos dice true, porque efectivamente está encontrando este valor, aquí se está corriendo la línea 16, ha perdón lo teníamos hecho arriba, bien miren, lo escribimos dos veces, vamos a borrarlo de aquí arriba y esta también vamos a comentarla por ahora y ahora aquí ustedes pueden ver que nos devuelve verdadero porque este valor se encuentra dentro de nuestro arreglo, de hecho aquí está, pero qué ocurre si lo escribimos de otra manera, lo escribimos con minúsculas nos va a retornar false puesto que no lo encuentra, lo mismo que si escribiesemos cualquier cosa, un valor que efectivamente no se encuentra, ok, bien, también podríamos correr el método find y veremos que si el método find no encuentra un valor, vamos aquí a indicar, tenemos que pasar una función al método find, el método find va a recibir un, recuerdan en el concepto de los callbacks, una función que se le pasa como parámetro a otra función, ok, entonces aquí por ejemplo, canción y podríamos aquí preguntar si canción es igual a cualquiera no importa, aquí, y nos va a decir andy find, ok, porque no encuentra precisamente este valor ya que la "p" está con minúscula, entonces cuando no encuentra el valor el método find retorna and andy find a diferencia de include que nos retornaba flase, veamos qué ocurre si lo escribimos

bien, nos devuelve el mismo elemento, es decir, si lo encuentra lo retorna, si no lo encuentra retorna andy find, veamos ahora find index, find index va a funcionar parecido pero nos indicará el índice del arreglo en el cual se encuentra el elemento, si es que el elemento se encuentra, aquí nos retorna 1 y eso es correcto, 0, 1 si aquí escribiesemos hard woman, nos diría 9 y eso es correcto 1, 2, 3, 4, 5, en fin, 9, ahora si por el contrario escribimos algo que aquí no existe nos va a retornar -1, noten que funcionan de un modo similar pero los valores de retorno son distintos, ok, supongamos que queremos ordenar nuestro arreglo, ok, entonces eso podríamos hacerlo de la siguiente manera, vamos a recuperar aquí esta porción de código para renderizar la lista, cuando rendericemos la lista estará desordenada, es decir, estará en el orden en que nosotros indicamos aquí las canciones, los elementos, si pueden notar es igual, queremos ordenarlo, para ordenarlo podemos utilizar el método .sort, top songs.sort y esto nos ordena el arreglo original, noten que aquí estoy cambiando, mutando el orden en el arreglo original, ok, si nosotros hacemos cualquier operación con top songs antes de haberlo ordenado nos va a dar el orden original, pero si por ejemplo aquí abajo hacemos un console-log the top songs veremos en la consola qué bueno no está definido porque lo escribí mal, puse tops, veremos aquí que está ordenado, ok, porque una vez que corremos la función de ordenamiento el array original cambia de orden, ténganlo en cuenta, ahí se produce una mutación del orden original, perfecto, si esta función la hubiésemos corrido después, van a notar lo siguiente, por consola van a estar las canciones ordenadas pero aquí en la pantalla no, por qué, porque aquí en pantalla estamos renderizando el contenido o imprimiendolo en pantalla antes del ordenamiento, miren, ok, aquí arriba está en el mismo orden pero aquí debajo luego de haberlo ordenado el orden cambio, ok, también podríamos ordenarlo de aquí está digamos este modo ascendente como podríamos ordenarlo de manera descendente, bueno, vamos a ponerlo nuevamente aquí para que se vea en la pantalla y aquí podríamos utilizar otra función del otro métodos de los arrays concatenado al ordenamiento, es decir, porque podemos concatenar métodos, después de hacer el ordenamiento además podemos pasarle el método reverse y esto lo va a ordenar y luego lo va a invertir, bien, ahora noten lo siguiente, si queremos ordenar un arreglo de números, aquí tenemos números, vamos a tener un problema, el problema es que el método de ordenamiento toma el contenido como si fuera un string, es decir, lo toma primero ordena por la primera por el primer valor, en este caso la primera letra y luego de ordenar por la primera letra, ordena por la segunda y así sucesivamente, bueno, tal cual ordenamos las palabras en un diccionario, por ejemplo no, o en cualquier lista donde queremos ordenar palabras de manera ascendente o descendente, eso para los números no sirve ya van a ver porque, supongan que queremos a nuestro arreglo nums que está aquí abajo ordenar y vamos a mostrar por pantalla cómo queda esto, console.log.sort y aquí verán aquí abajo, bueno, comienza bien -4,1, 10, 20 y como luego un 3, y un 310 y un 43 noten que está ordenando por el primer término, primero ordena el 1 entonces 1, 10 y eso es correcto según el algoritmo de ordenamiento, aunque nosotros sabemos que no está ordenando de mayor a menor o de menor a mayor adecuadamente, está ordenando como si fueran letras, ok, para resolver este dilema podemos pasarle a nuestra un método sort, un callback de ordenamiento, una función de comparación, vamos a decir que compare los términos A con B, o no importa aquí el nombre, es lo de menos sí, y luego diremos que esto retorne aquí el ratón es implícito no necesito escribirlo, esto va a retornar A-B y ahora vamos a ver que los números están ordenados correctamente y si quisiéramos

ordenar de mayor a menor podríamos acá correr un reverse, por ejemplo, qué ocurriría, no ocurriría lo que nosotros esperamos, bueno tal vez alguno me dirá es que el método debe ir dentro, bueno, puede ser, esta es una buena manera de resolverlo pero hay otra, otra manera de resolverlo sin llamar a un nuevo método, es decir, sin concatenar métodos es aquí, en vez de indicar A-B es indicar B-A y hace exactamente lo mismo, ok, tenemos resueltos entonces unos cuántos de nuestros de nuestros métodos de arreglo, podríamos concatenar dos arreglos, bueno si, podríamos concatenar, arreglos por ejemplo aquí podríamos tener otro arreglo, names y esto podría tener, vamos a hacer pocos valores para que no demore demasiado, Silvia, Adriana y Miguel, perfecto y ahora podríamos concatenar dos arreglos, vamos a ir borrando esto, bueno, este lo vamos a dejar, vamos a borrar estos de aquí abajo para que no ensucien tanto la consola, tenemos demasiada información al mismo tiempo, no quiero que nos confundamos, entonces aquí podríamos indicar que queremos concatenar en un tercer arreglo, por ejemplo, podríamos indicar const, array combinado, no importa el nombre, ok, sería igual a tops song. concat, aunque bueno en realidad podríamos hacerlo así miren, mucho más fácil, podríamos hacer const, voy a dejarlo como estaba, no quiero no quiero complicarles la vida, vamos a hacer un array combinado, va a ser igual a nuestro primera, nuestro primer arreglo más nuestro segundo arreglo, names, aquí podríamos imprimir el resultado de conbined, aquí tenemos combinado nuestro arreglo, están todas las canciones y luego vienen los nombres, bien, simplemente estamos sumando arreglos, aquí los valores pueden ser de cualquier tipo, pueden ser números, pueden ser strings, en fin, lo que ustedes deseen, bien, ahí tenemos otra de las posibilidades contact, también podemos hacer, preguntar por algunos de los valores con el método soom, soom aquí, que hace el método soom, bueno ejecuta una función como for each por cada elemento de un arreglo, ok, si la función retorna a true entonces el método soom retorna a true, es decir, encontró un valor al menos y detiene la ejecución, podríamos probarlo con un ejemplo clásico, podríamos tener ages o edades, edad, edades mejor y sería un arreglo con valores numéricos 10, 15, 40, 3, 9, en fin, 70, etcétera y queremos averiguar si hay alguna persona mayor de 18 años, por ejemplo, entonces podríamos someter a análisis este arreglo con el método soom, cuál sería la sintaxis, bien, podríamos preguntar aquí en esta sintaxis sí algunos de los, las edades, no importa aquí el nombre podríamos poner elementos, valores, vamos a poner el por elementos, bien, si alguno de los valores es mayor a 18, bien, y esto que va a ser, bueno, esto va a retornar, si vamos a mostrarlo para que vean exactamente lo que retorna, console.log y esto va a retornar true porque va a retornar true, bueno, porque hay efectivamente algún valor que es más grande que 18, de hecho tenemos 2, 40 y 70 cuando el método soom encuentra 40 retorna true y deja de analizar, ya nos sigue analizando porque, porque simplemente encuentra el primero, o sea, por eso es soom, alguno, busca algún valor que cumpla la condición, la condición es que el elemento sea mayor que 18 y aquí el elemento es cada uno de los números porque esto es un bucle, una iteración, como la mayoría de los métodos de array, toman un arreglo y lo procesan, ejecutan una suerte de bucle for y hacen algo con los valores, ejecutan a alguna acción con cada uno de los valores, es el caso lo que estamos haciendo aquí. Esto es fácil de probar si aquí no ponemos ningún valor mayor que 18, bueno, va a ir hasta el final, los va a recorrer a todos y como no encuentra un elemento que cumpla la condición porque el 18 no es mayor que 18 es igual, no no cumple la condición retorna false, es decir, no ha encontrado aquí dentro ningún valor que satisfaga esta condición, ok, podríamos indicar el método join,

en el método join lo que hace es, bueno, también unir un arreglo pero lo retorna como si fuera un string, como si fuera un conjunto de palabras, ok, aquí en este caso podríamos tener Cecilia, Adriana y miguel y si quisiéramos trabajar allí con esos valores aquí podríamos indicar o aquí abajo no hay problema, podríamos indicar names. join simplemente y podríamos hacer de esto un console-log para que nos muestre exactamente lo que esta haciendo, lo que va a hacer es que transformará el contenido de, mejor dicho, la estructura de names y me va a retornar, ok, me lo va a retornar como un string por eso aquí ya no estamos viendo un arreglo, fíjense que esto es un string, si nosotros aquí hacemos un console-log de names ustedes verán claramente que me devuelve la estructura de un arreglo, pero si hacemos names.join me va a devolver simplemente un texto, esto es una cadena de texto, además aquí podríamos indicar, por ejemplo, dentro de join que queremos que los elementos se separen no por una coma sino con un espacio y entonces aquí tendríamos como separador un espacio o podríamos haber indicado un guión, más un espacio y obtendríamos lo siguiente, son algunas posibilidades de estos métodos, hay más métodos por supuesto, son muchos, les estamos mostrando, bueno pues algunos de los que ustedes pueden utilizar, hay algunos que son métodos digamos que capitales, súper importantes, que son map y filter esos son métodos muy importantes y son los que vamos a ver a continuación para cerrar este capítulo de arreglos, que hace map, map nos devuelve un nuevo arreglo, nos va a devolver un nuevo, crea un nuevo arreglo, o sea, el no muta el arreglo original sino que crea uno nuevo después de llamar a una función por cada elemento del array original, eso ya lo hemos visto por ejemplo en el for each y en otros, ok, entonces supongamos que aquí tenemos bueno, estos números, numeros, vamos a usar números y hagamos lo siguiente, vamos a mapearlo, vamos a decir aquí por ejemplo, const nuevo nums, porque, porque map retorna a un arreglo nuevo, por lo tanto lo vamos a almacenar en una variable que se llama nuevo nums y aquí entonces vamos a hacer lo siguiente, nums, map y aquí viene nuestra función, es decir, vamos a correr esta función una vez por cada elemento y que queremos hacer bueno, por ejemplo, vamos a ponerle N a cada elemento, no importa el nombre recuerden y aquí lo que estamos haciendo, lo que estamos haciendo referencia es a cada elemento por cada una de las veces que recorre, que itera nuestro arreglo, va a iterar 1,2,3,4 en fin, todas las veces y aquí lo que vamos a retornar es lo que ustedes quieran pero por ejemplo podríamos decir N X 2, bien, y ahora en nuevos nosotros deberíamos tener en vez de 1, 1 X 2, en vez de 10, 10 X 2, 310 X 2, vamos a revisarlo, console-log, nuevo nums y aquí tenemos los valores 2, 20, 6, 20 pero recuerden maps no destruye el arreglo original, no lo muta, crea un nuevo arreglo, retorna un nuevo arreglo, por lo tanto si aquí hacemos un console-log de nums, del arreglo original, veremos que los valores no han sido tocados, bien, y otro método super importante que también este debemos aprender es el método filter y con esto vamos a terminar por hoy nuestro repaso de los métodos de los arrays. Filter lo que hace es bueno un filtrado, filtra precisamente, retorna al igual que map retorna un nuevo array, no es destructivo, no muta el arreglo original nos va a retornar en un nuevo arreglo los elementos que pasen el filtro, que pasen la condición de la función que le vamos a pasar, por ejemplo aquí, podríamos indicar un proceso de corte, podríamos decir, por ejemplo, vamos a ver const-filter en nums y podríamos esto sería igual a nums.filter y al igual que map vamos a pasar aquí un callback, una función que por cada número, también le voy a poner N otra vez va a retornar lo siguiente N mayor que supongamos 30 y ahora en nuestro nuevo arreglo, en filters nums tendremos solamente los

números que pasan esa condición, es decir, supongo que si no estoy mirando mal 43, 310 y 701 a ver si esto es correcto, console-log entré nums y ahí tenemos 310, 701 y 43 pero dijimos que al igual que map no es destructivo, por lo tanto nums debería seguir siendo igual y ahí lo tenemos sigue siendo igual.