***Curso JS ES6***

**Usando forEach:**

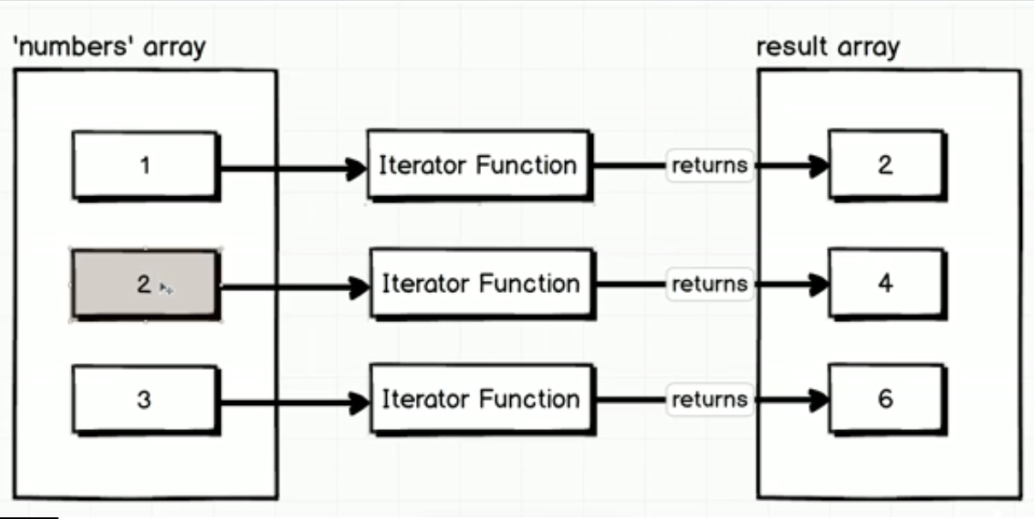
Cuando utilizamos el método forEach llamamos a una función anónima que se ejecuta una vez por cada elemento del array, y el elemento pasa como un argumento.

* forEach lee todas las líneas y no devuelve algo ,en cambio map si muestra una respuesta con el Return

**Usando map:**

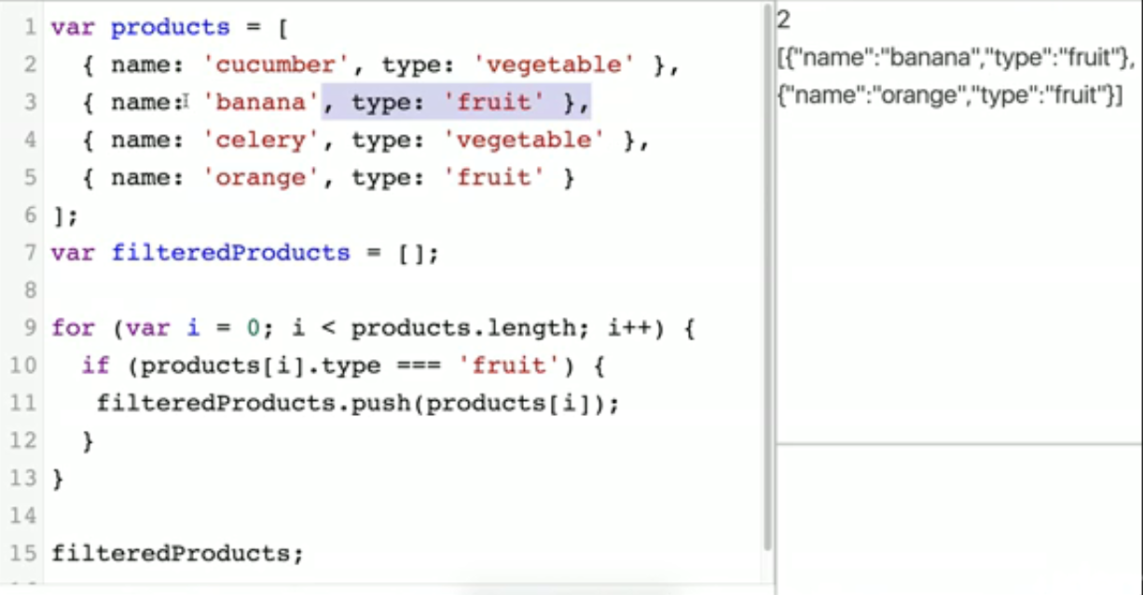
Cuando se utiliza map no se cambia el array existente sino que se crea un nuevo array.

Se utiliza return



**Usando filter:**

usando for:



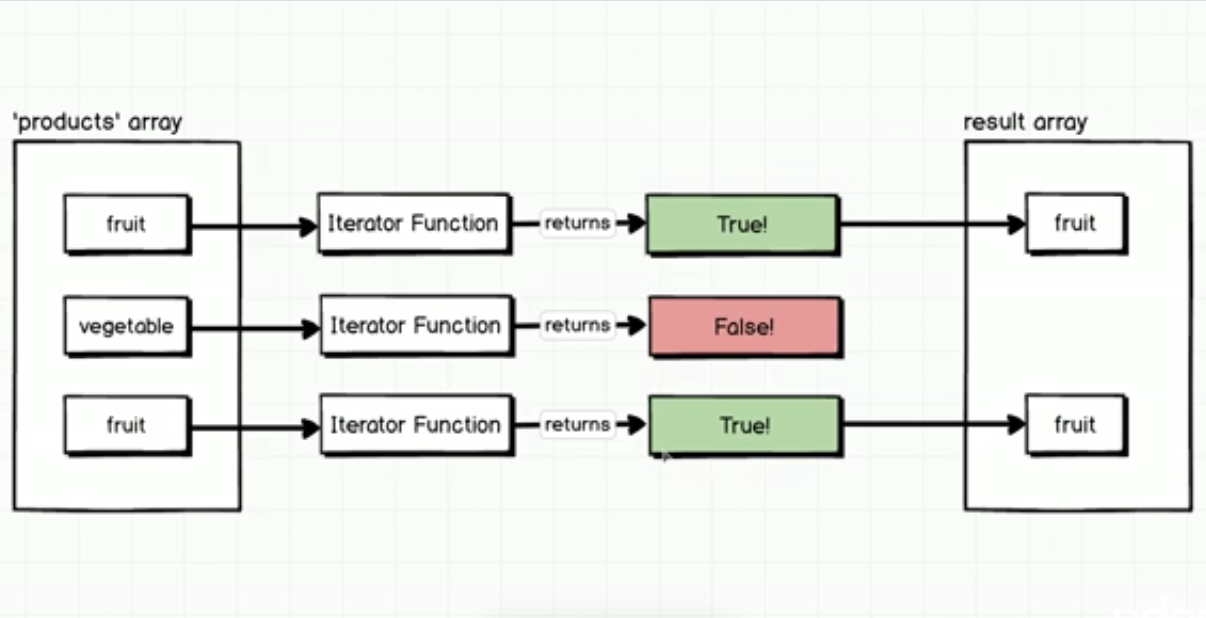
filter:



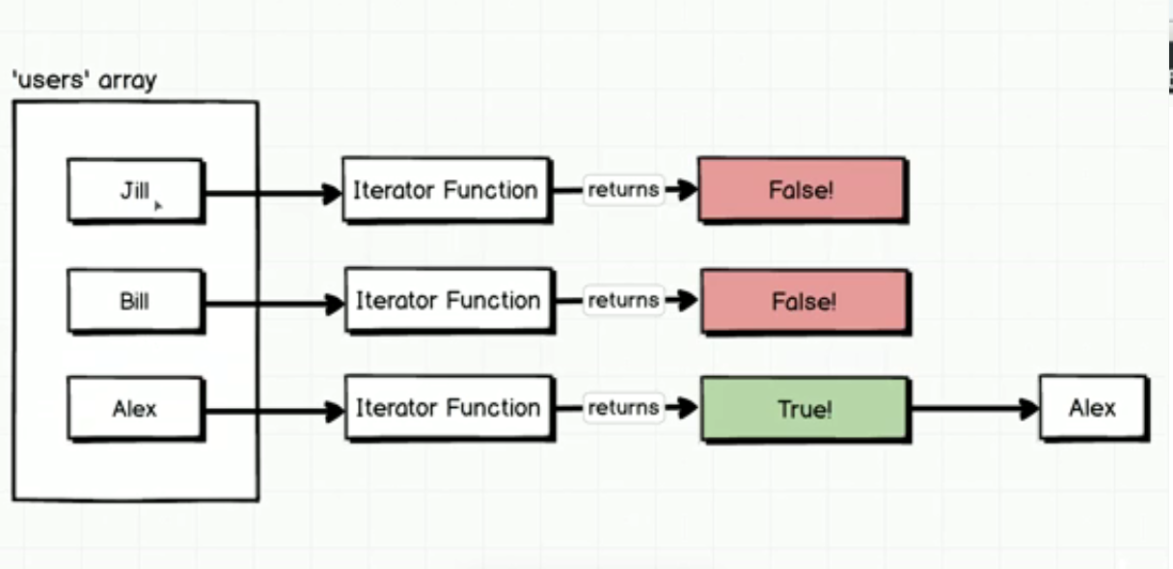
Cuando usamos filter lo que sucede es q el elemento pasa al iterador si cumple con las condiciones entonces devuelve true y se coloca en el nuevo array

Es un booleano devuelve true o false

Se utiliza return



**Usando find:**



Devuelve true o false.

El iterador se repite hasta que devuelve un true, ahí sale de la iteración. Es decir que si hay dos valores que dan true, solo devuelve el primero

Se utiliza return.

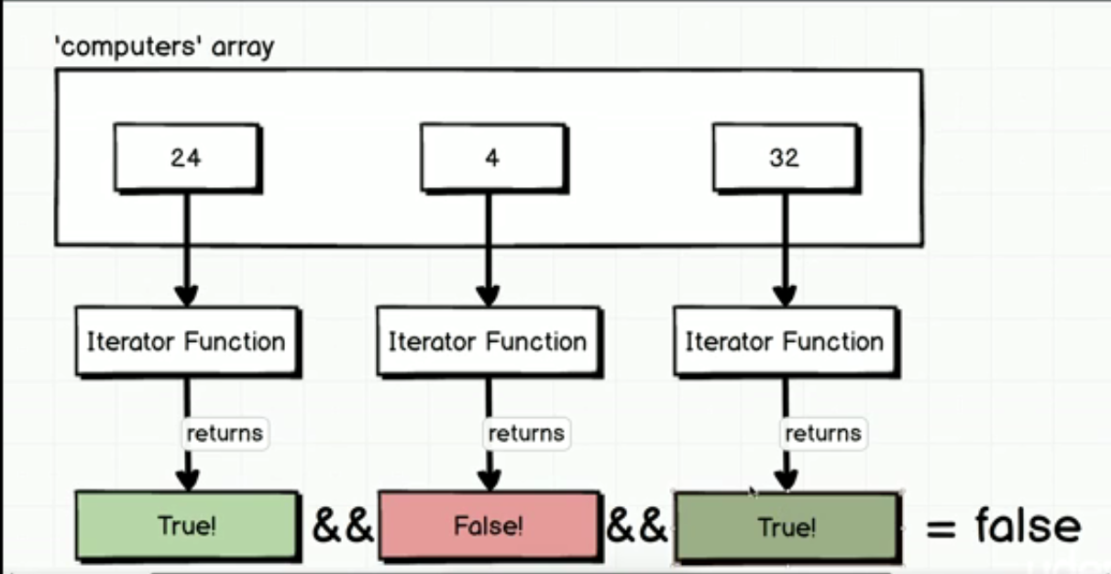
Si tengo dos objetos con name Alex solo devuelve el primero que encuentra.

Hay dos tipos de objeto:

-literal que se crea ahí mismo entre llaves, no tiene padre

-y el otro que se crea a partir de una clase que sería el padre. Utiliza un molde y un constructor

**Usando Every:**



“Y”

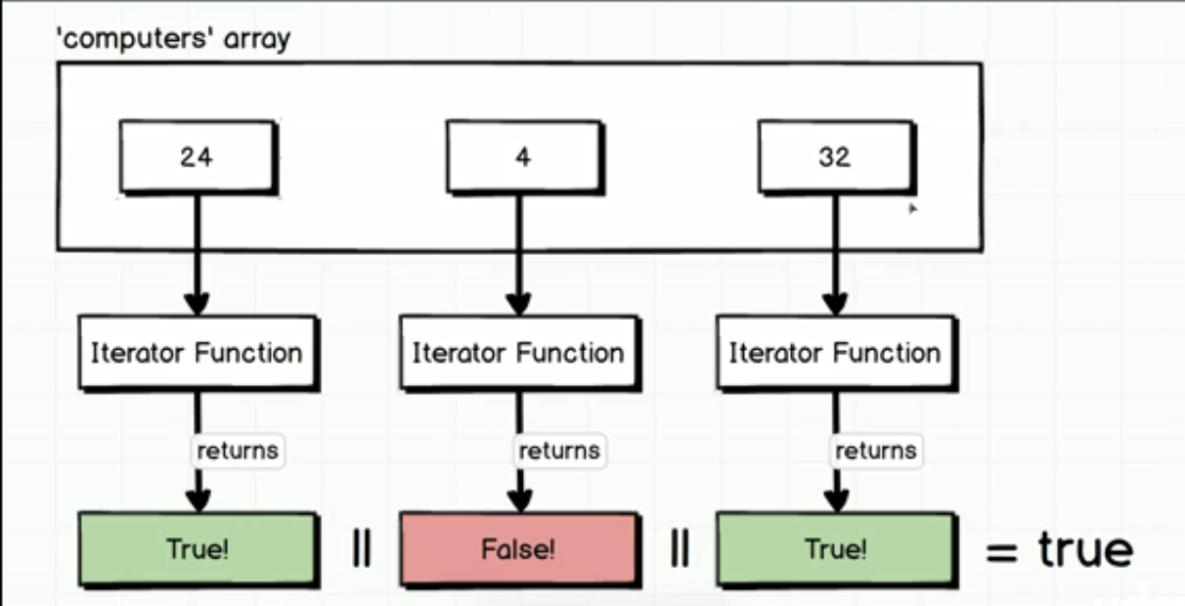
Cada uno de los elementos pasa por el iterador y devuelve true o false.

Si algún iterador devuelve falso, entonces devuelve falso al final.

Se utiliza return

**Usando Some:**

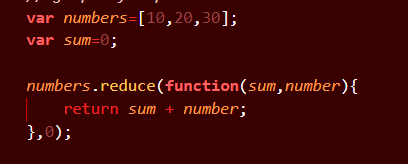
Se utiliza igual que el Every, pero en este caso si al menos uno de los iteradores devuelve true entonces, al final la función devuelve true

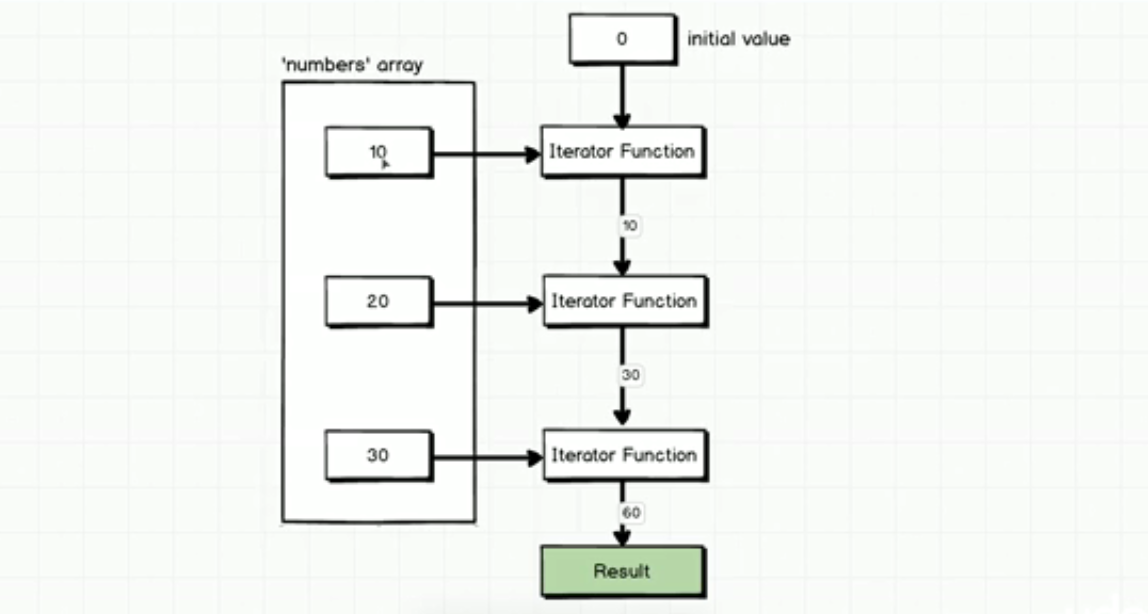


“O”

**Usando Reduce:**

La fórmula seria array.reduce(function,0)

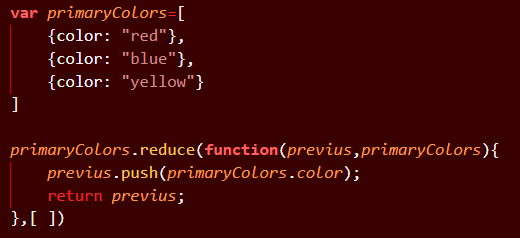




Al iterador ingresa el valor inicial y el primer número del array, y devuelve 10. En la segunda vuelta sum tiene valor 10 y el argumento number (array) tiene valor 20.

En la función al final se coloca el valor inicial para el primer argumento.

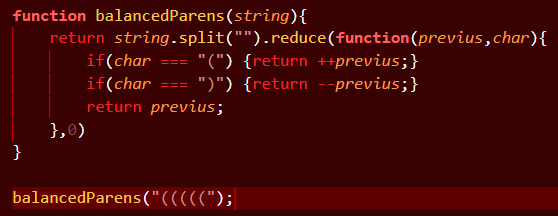
Otro ejemplo:



En este, previus es un array entonces por eso el objeto inicial es un array vacío.

Previus devuelve un array conformado por los valores del array primaryColors.

Otro ejemplo:



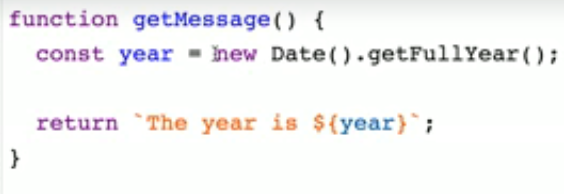
Se llama a la función dándole un string

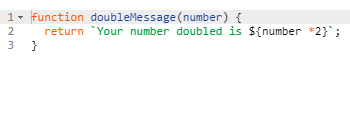
Se separa en un array y se le aplica reduce, previus tiene valor 0 si el elemento del array es ( se suma 1 a previus y si es ) se resta uno.

Si esta balanceado el valor de previus es 0

Para obtener un falso o verdadero en vez de un numero se coloca un! Después de return y antes de string.

**Template string:** Cuando se llama a una variable dentro de un string se coloca todo el texto entre comillas ` ` y antes de la variable se coloca $ y {} dentro de las llaves se coloca la variable y se pueden hacer operaciones. Si se escriben dos variables separadas no hace falta el +.





Si por ejemplo la variable es igual al valor de otra variable y no hay string, se puede escribir el nombre la variable solo.



**Usando arrow:**

**const** *add*=**function** sum(*a*,*b*){

**const** *suma*=*a*+*b*

    return *suma*

    }

*sum*(2,3);

**const** *add*=**function** sum(*a*,*b*){

    return *a*+*b*

    }

*sum*(2,3);

**const** *sum*=(*a*,*b*)**=>** *a*+*b*;

*sum*(2,3);

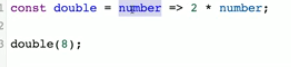
Se ponen los argumentos, el arrow y después lo que se quiere hacer con ellos.

En arrow ya está implícito el return

Si se utilizan las llaves después del arrow, se debe poner el return.!!

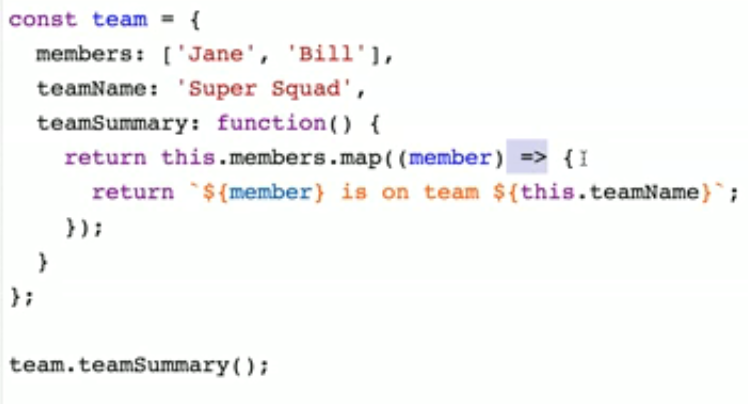
Cuando es una expresión muy larga se utiliza el return y las llaves ya que facilita la lectura y el orden.

Si se tiene un solo argumento se pueden omitir los paréntesis.



En el siguiente ejemplo se arregla un problema con this, utilizando arrow.

This === team

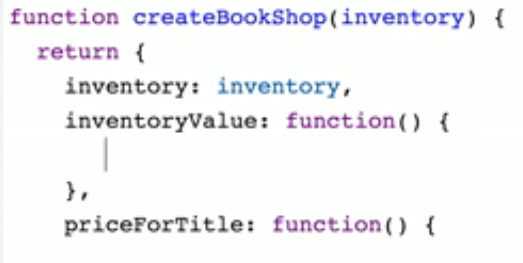


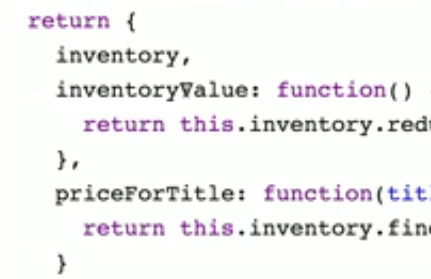
**Enhaced object:**

Se trata de optimizar código, hacerlo más legible.

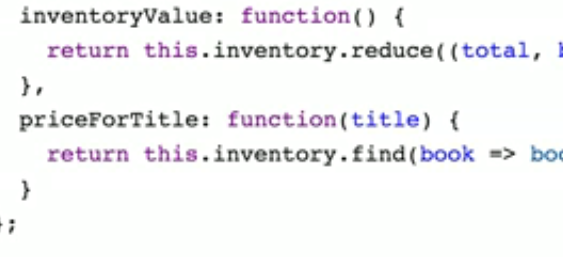
Si en un objeto, key y velue tiene en mismo nombre, se puede borrar el value y los:

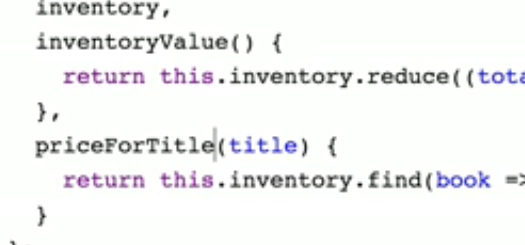
Por ejemplo aquí iventory (key) está llamando a una variable llamada inventory (value)





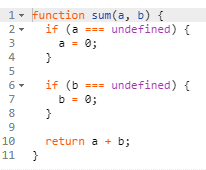
Cuando en el objeto el value es una función, también se puede eliminar la palabra función y los:

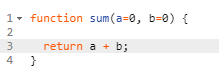




**Default function arguments:**

Cuando se quiere dar el valor al argumento en el caso de undefined se hace en el mismo argumento.

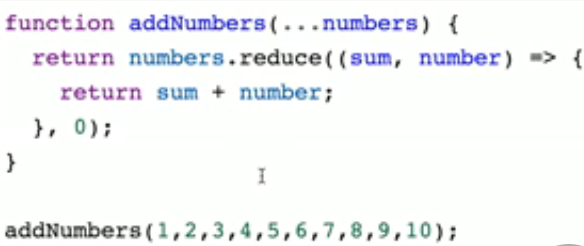




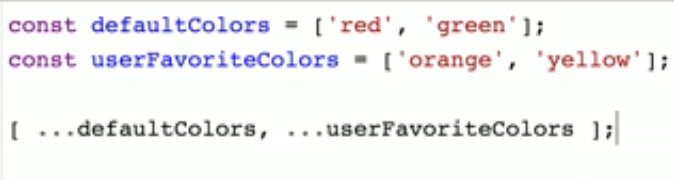
**Rest and Spread Operator:**

Rest Operator: Cuando el argumento de la función toma una cantidad desconocida de datos se pone (…number) y los pone en un array que sería number.

Está relacionado con juntar elementos

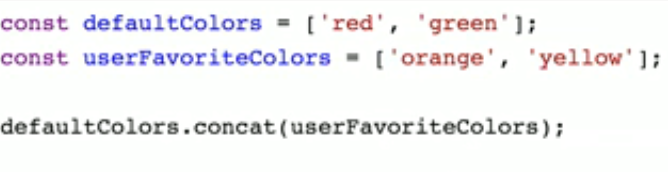
****

Spread Operator: Para concatenar dos array se abren corchetes (se crea un array) y se utiliza … seguido del nombre del array que se quiere concatenar. Se separan por comas.

  
Si se quiere agregar un elemento más al array nuevo:



Método de concatenar arrays:



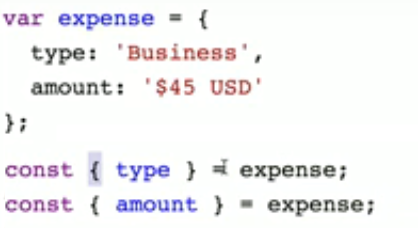
**Destructurin:**

Se trata de hacer referencia a un objeto reduciendo la cantidad de caracteres duplicados.

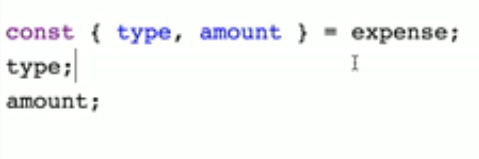
Entonces se coloca variable entre llaves si se llama a la propiedad de objeto y entre corchetes si se llama a la propiedad de un array, igual al nombre del objeto o array llamado.

Lo que significa es crear una variable haciendo referencia a la propiedad de ese objeto.

El nombre de la variable debe ser igual a la propiedad referenciada.



Y reduciendo aún más cantidad de caracteres duplicados.



Otro ejemplo: por medio de file se llama al objeto y luego a las propiedades del objeto.



En vez de llamar a file.name, file.extension etc…



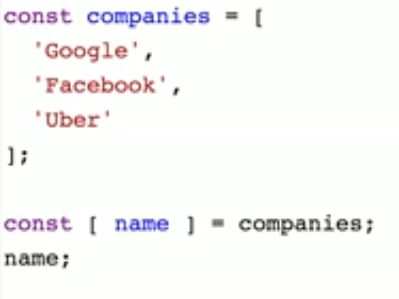
Entonces de esta manera en el argumento de la función llamamos al primer objeto de savesFiled y hacemos referencia a las propiedades.

En un objeto al llamarlo podemos llamar a las propiedades en el orden que deseemos, como en el ejemplo anterior donde llama primero a la propiedad name. En el objeto desestructuramos las propiedades.

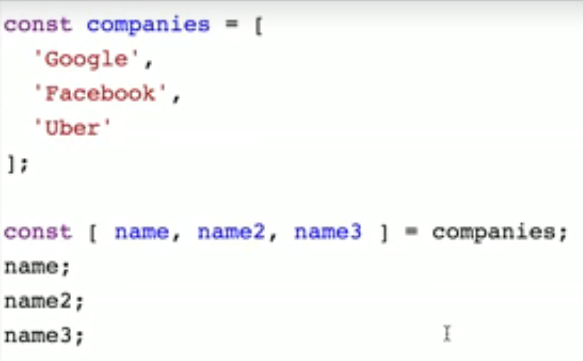
En un array el orden en que ponemos las variables es el orden en que las sacamos del array. En el array desestructuramos los elementos

Otro ejemplo:

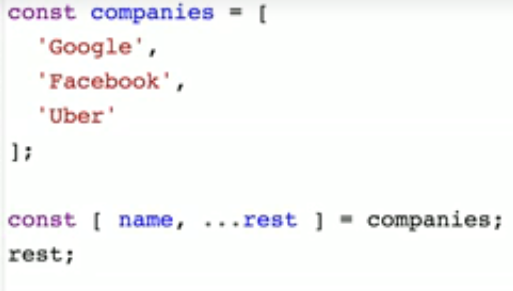
Como es un array la o las variables creadas deben ir entre corchetes.



El primer name se refiere al primer elemento del array y así sigue

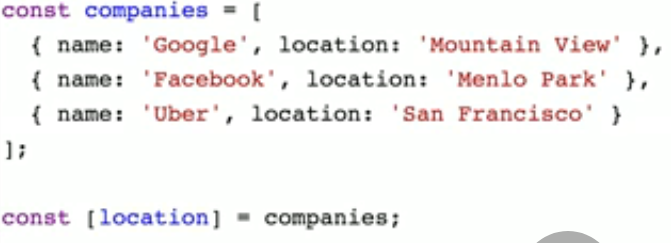


Si utilizamos rest coloca el resto de elementos en un array:

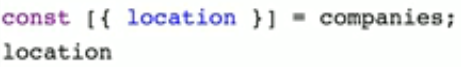


[“Facebook”,”uber”]

Otro ejemplo: buscar una propiedad dentro de un array.

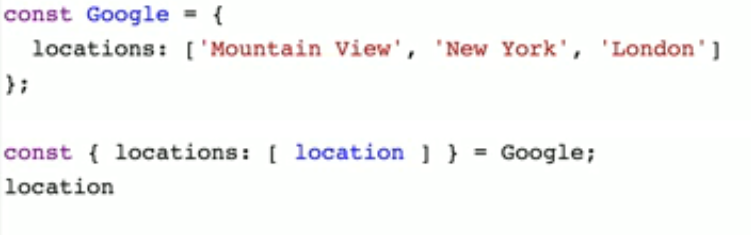


De esta manera buscamos el primer elemento del array.



Y asi se llama a la propiedad location, lo que devuelve “mountain view”

Si se desea llamar a un elemento de un array que es value de una propiedad:



Entonces se coloca entre llaves para ingresar a la propiedad y luego dos puntos y entre corchetes location para llamar al primer elemento de ese array.