Objetos en js

*const* producto = {

    nombre: 'venesita',

    estatura: 1.53,

    cabello: 'negro',

    culo: 'promedio',

    apodo: 'culoncita'

}

Agregar al objeto

producto.apellido='flores';

Eliminar del objeto

*const* producto = {

    nombre: 'venesita',

    estatura: 1.53,

    cabello: 'negro',

    culo: 'promedio',

    apodo: 'culoncita',

    disponible: true

}

delete producto.disponible;

asignar valores de un objeto a una variable.

*const* nombre = producto.nombre;

extraer con destructuring, extrae la variable nombre y le asigna el valor es la nueva forma con ES6

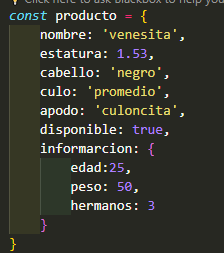
*const* { nombre } = producto;

*const* { nombre,estatura } = producto;

console.log(nombre);

console.log(estatura);

Objeto dentro de otro objeto



Acceder al objeto hijo

producto.informarcion.hermanos

Destructuring al objeto interno

*const* { edad } = producto.informarcion;

Object.freeze( nombre del objeto ) <-> con esto evitamos que se pueda modificar,agregar o eliminar el objeto

Object.isFrozen( nombre del objeto ) <-> con esto podemos ver si el objeto esta o no congelado

Object.seal() <-> no se pueden agregar, ni se pueden eliminar propiedades pero si se pueden modificar las propieddes existentes

console.log(Object.keys(producto)); <-> para obtener las llaves del objeto

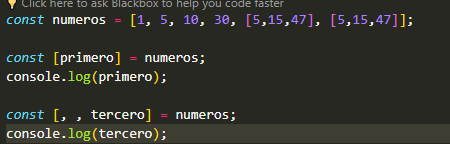
console.log(Object.values(producto)); <-> para obtener los valores del objeto

console.log(Object.entries(producto)); <-> obtenemos los dos valores clave, valor

Arreglos

Destructuring, en el primer caso vemos como se accede al primer elemento, pero para acceder a otro elemento y no sea

Const [primero, segundo, tercero] podemos acceder al tercer elemento como se muestra debajo anteponiendo comilla a los elementos anteriores hasta llegar al elemento requerido.



Otra forma de continuar manteniendo elementos en el arreglo, esta medio raro pero dice que se ocupa mucho en react



Recorrer arreglos

Usando foreach

numeros.forEach((*numero*) *=>* {

    console.log(*numero*);

});

Usando map

numeros.map((*numero*) *=>* {

    console.log(*numero*);

});

.map y foreach entregan lo mismo pero la diferencia es que map puede devolver un nuevo arreglo y foreach solo devuelve el arreglo.

*const* nuevo1 = numeros.forEach((*numero*) *=>* {

    return *numero*;

});

*const* nuevo2 = numeros.map((*numero*) *=>* {

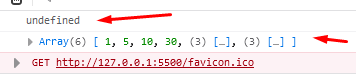
    return *numero*;

});

console.log(nuevo1);

console.log(nuevo2);

Como vemos el resultado el primero retorna undefined que es el del foreach y el otro devuelve un arreglo nuevo con map.



JavaScript Hoisting = es cuando se llama a una funcion antes de definirla, y una funcion asignada a una variable, la llamada a la funcion antes de inicializarla se ejecuta, pero cuando se asigna a una variable no es ejecutada en primer instancia, por eso hay que tener cuidado cuando se crean las funciones

en este ejemplo aqui si se manda a llamar

sumar();

function sumar(){

return 1 + 1;

}

en este ejemplo aqui se manda a llamar pero no tiene nada ya que se asigno a una variable

sumar2();

const sumar2 = function(){

return 1 + 3;

}

const numero1= 25;

const numero2 = '20';

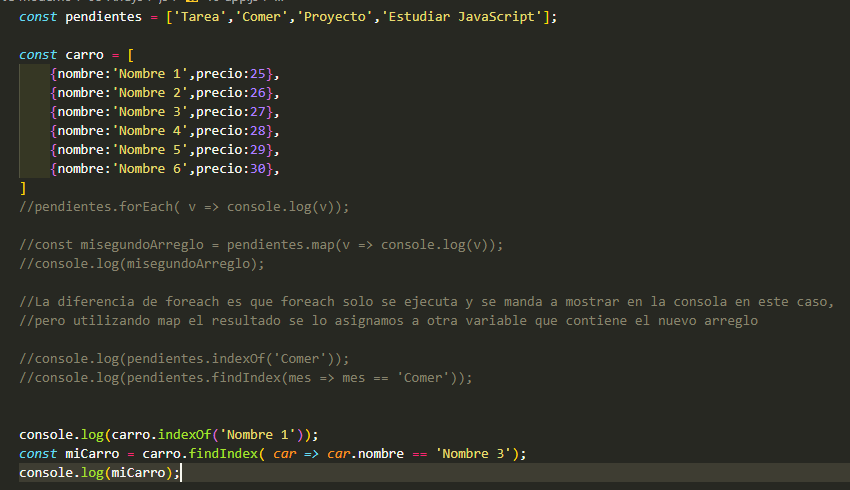
perseInt(numero2) //esto es una funcion

numero1.toString() //esto es un metodo

fineIndex y indexOf son muy semejantes solo que fineIndex es como una funcion y puede acceder a arreglos o a arreglos de objetos.

console.log(pendientes.indexOf('Comer'));

console.log(pendientes.findIndex(mes => mes == 'Comer'));



Se recomienda utilizar indexOf para arreglos y fineIndex para arreglo de objetos

console.log(pendientes.findIndex(*pendiente* *=>* *pendiente* == 'Comer'));

//filter

es como indexOf o fineIndex pero filter busca todas las coincidencias y las asigna a otro array

*const* carro = [

    {nombre: 'Nombre 1', precio: 25},

    {nombre: 'Nombre 2', precio: 26},

    {nombre: 'Nombre 3', precio: 26},

    {nombre: 'Nombre 4', precio: 28},

    {nombre: 'Nombre 5', precio: 29},

    {nombre: 'Nombre 6', precio: 30},

]

*const* car = carro.filter( *producto* *=>* *producto*.precio == 26);

console.log(car);

//find

El metodo fine es como fineIndex pero fine nos genera un nuevo arreglo, pero solo trae la primer coincidencia.

//reduce

El método reduce JavaScript nos permite, como su nombre indica, reducir el array insertado a un solo valor

es como un acumulador como si lo hicieramos con un for he ir acumulando un dato a la variable

<https://keepcoding.io/blog/utilizar-el-metodo-reduce-de-javascript/>

//every

busca sobre todo el arreglo de objetos y comprueba si todos los elementos cumplen una condicion, en caso de que si se cumplea devuelve true, de lo contrario devuelve false

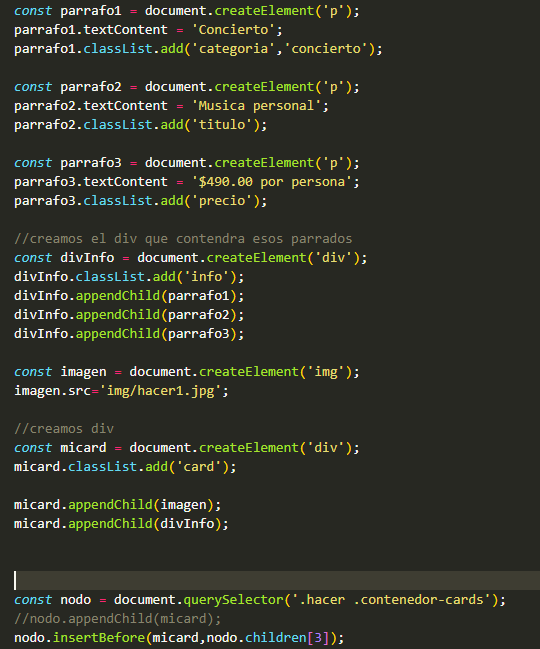
//concat

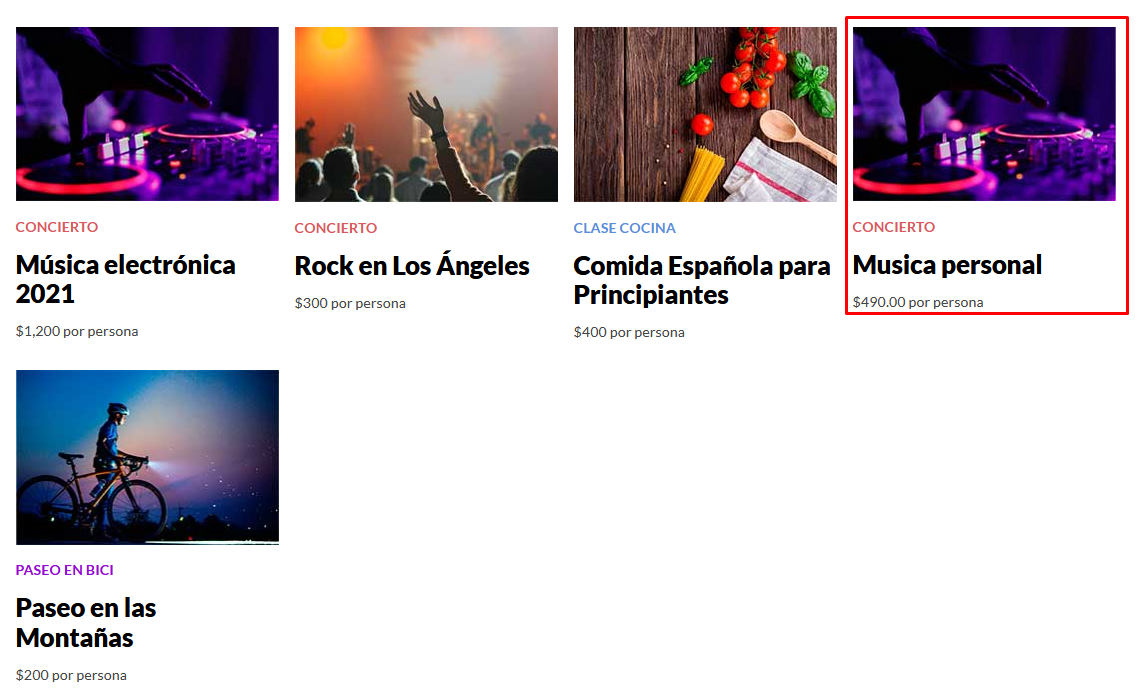
permite unir dos o mas arreglos

//unir con spread operator

const resultado2 = [...meses, ...meses1, ...meses2]

Creando un card ‘tarjeta ‘con puro js





Propagacion de eventos, se da cuando hay elementos que tiene acciones y los elementos están muy juntos, al ejecutar uno se pueden ejecutar los demás y si eso no es lo que queremos podemos evitar la propagación.

*const* cardDiv = document.querySelector('.card');

*const* infoDiv = document.querySelector('.info');

*const* titulo = document.querySelector('.titulo');

cardDiv.addEventListener('click',()*=>*{

    console.log('click en el card');

});

infoDiv.addEventListener('click',()*=>*{

    console.log('click en el info');

});

titulo.addEventListener('click',()*=>*{

    console.log('click en el titulo');

});

Se le da click en la info pero se propagan los demás eventos



Para evitar esto y solo se ejecute la acción correcta existe stopPropagation

*const* cardDiv = document.querySelector('.card');

*const* infoDiv = document.querySelector('.info');

*const* titulo = document.querySelector('.titulo');

cardDiv.addEventListener('click',(*e*)*=>*{

*e*.stopPropagation();

    console.log('click en el card');

});

infoDiv.addEventListener('click',(*e*)*=>*{

*e*.stopPropagation();

    console.log('click en el info');

});

titulo.addEventListener('click',(*e*)*=>*{

*e*.stopPropagation();

    console.log('click en el titulo');

});

Ahora si le damos click en el titulo solo se ejecuta dicha acción



Otra forma de prevenir el Even Bubbling

*const* cardDiv = document.querySelector('.card');

cardDiv.addEventListener('click',(*e*)*=>*{

    if( *e*.target.classList.contains('titulo')){

        console.log('click en el titulo');

    }

    if( *e*.target.classList.contains('info')){

        console.log('click en info');

    }

    if( *e*.target.classList.contains('card')){

        console.log('click en el card');

    }

    if( *e*.target.classList.contains('precio')){

        console.log('click en el precio');

    }

    if( *e*.target.tagName=='IMG'){

        console.log('click en la imagen');

    }

});