```
// Javascript está diseñado sobre un paradigma simple basado en objetos.
    //Un objeto es una colección de propiedades, y una propiedad es una asociación
    // entre un nombre (o clave) y un valor.

//Los objetos son estructuras donde se puede guardar información y funcionalidades
// La principal diferencia con los arreglos es la forma en que se guarda la información.
// Nos permite ser mas especificos
// Se declaran como cualquier otra  "estructura". Se declaran usando llaves {};
//var miObjeto = {}

// Los objetos tienen una estructura PAR CLAVE-VALOR
//POR UN LADO TENEMOS LA CLAVE QUE ES LA PROPIEDAD Y POR OTRO SU VALOR
//Todos los objetos respetan esta regla. Los Objetos pueden tener todas las propiedades
// que queramos
// VALOR: datos
var deportes = {
    conBalon: ['Futbol', 'Basketball', 'Rugby'],
    sinBalon: ['Boxeo', 'Surf', 'Trekking'],
};
// OBJETO llamado persona con tres propiedades:
var persona = { nombre: 'Lucas', edad: 26, estudios: { esProgramador: true } };
// nombre
// edad
// estudios

//Trabajando con OBJETOS:
// Conceptos básicos:
// Dot notacion y Bracket-Notation [] corchetes

// Estructura con forma Key:value
// Asignar propiedades y valores

// Para acceder a a la propiedad de un objeto, se escribe el nombre del OBJETO seguido de punto(.)
// y el nombre de la propiedad

//console.log(persona.edad);
//Como asignar valores a las propiedades:
// Para cambiar el valor de una propiedad 1) Accedemos a la propiedad,
// o sea escribimos el nombre del  objeto seguido de punto y el nombre de la propiedad
// e igualamos al valor nuevo  (OBJETO.propiedad = nuevo valor);

//***********  NO SE PUEDE CREAR UNA PROPIEDAD VACIA, SIEMPRE TIENE QUE TENER UN VALOR  ************

persona.nombre = 'Martín'; // Aqui queremos cambiar Lucas por "Martin"

//console.log(persona.nombre);
persona.edad = 32;

// CREAR PROPIEDADES

var autos = {}; //creamos un OBJETO vacío llamado autos
//Ahora vamos a crear una propiedad llamada 'marcas'. NOTA: Las propiedades nunca pueden estar vacías.
autos.marcas = ['Ford', 'Audi', 'Ferrari'];
//console.log(autos); // el resultado es: { marcas: [ 'Ford', 'Audi', 'Ferrari' ] }

// Eliminación de propiedades (DELETE: palabra reservada)
// Vamos a eliminar la propiedad 'marcas' del Objeto "autos";
delete autos.marcas;
//console.log(autos);
//Ahora vamos a ver una particularidad que tienen este tipo de datos:
```

```
//Dentro de una propiedad de un objeto podemos guardar una funcion.
var misFunciones = {
    saludar: function () {        // para incluir una función se escribe function() OJO: no hay que definir a
la función
        console.log('hola');
    }
};
misFunciones.saludar();



    // ******  CONCLUSIONES *******
    // 1) Para acceder a la propiedad de un objeto simplemente tenemos que escribir el nombre del objeto segu
punto y el nombre de la propiedad.
    // 2) Para cambiar el valor de una propiedad simplemente tenemos que acceder a ella e igualarla al nuevo
    // 3) Para eliminar propiedades utilizaremos una palabra reservada llamada 'delete'

    // La sintaxis para crear una propiedad es igual a la que se usa para cambiar el valor de una propiedad.
    // La diferencia está en que cuando la propiedad en que cuando usamos esta sintaxis y la propiedad ya exi
a reescribir su valor;
    // pero si no existe vamos a estar creando la propiedad y asignándole su valor.



//  ***********   UTILIZACION DE BRACKET NOTATION    ****************************************
        // Creemos un objeto llamado atuendos y propiedades: manos, pies y piernas

var atuendos = {manos: ['Guantes', 'Anillos'], pies: ['Zapatos', 'Soquetes'] };
//console.log(atuendos.manos);

atuendos["piernas"] = ['Bermudas', 'Pantalones'] // Usamos brackets o corchetes []. La particularidad es que
va entre corchetes.
//console.log(atuendos);

//Recordemos la  información clave de la video clase...

//De la misma manera que utilizamos la Dot-Notation o notación por puntos para acceder o asignar un valor,
// también podemos usar Bracket-Notation, o notación por corchetes. Lo único que cambia es la forma en la que
escribimos.

// Muchas veces nos puede suceder que necesitemos utilizar una variable externa para guardarla como propiedad
// Es importante que en esos casos recordemos utilizar Bracket-Notation sin comillas para que funcione correc

//NOTA: si necesitamos usar una variable externa se usará la bracket-notation solamente y el nombre de la ppr
[dentro de los brackets] irá sin comillas.
// Ej.
var comidas = {}
var diferencisDeNotaciones = function (propUno, propDos) {
    comidas.propUno = ["Frutas", "Vegetales"]
    comidas['propDos'] = ["Hamburguesas", "Papas Fritas"];
    };
    diferencisDeNotaciones ('saludable', 'noSaludable');
    console.log (comidas)

    // y el resultado del console.log es:
 //   propUno: [ 'Frutas', 'Vegetales' ],
 //   propDos: [ 'Hamburguesas', 'Papas Fritas' ] **** No respondió lo que esperabamos***
 //   No respondió con los nombres de las propiedades "saludable", "noSaludable".
        // Para corregir eso cuando se usan variables externas (saludable y noSaludable) se tiene que definir
Bracket-Notation
        // y las propiedades como variables o sea sin comillas
    //comidas[propUno] = ["Frutas", "Vegetales"]
    //comidas[propDos] = ["Hamburguesas", "Papas Fritas"];
    // };
```

```
    //diferencisDeNotaciones ('saludable', 'noSaludable');
    //console.log (comidas)*** Siendo el siguiente resultado:
    //    {
    //    saludable: [ 'Frutas', 'Vegetales' ],
    //    noSaludable: [ 'Hamburguesas', 'Papas Fritas' ]
    //  }

    //
*************************************************************************************************************
```