

# **JS | DATA TYPES - OBJECTS**

## Objetivos:

- ✓ Explicar la relación clave-valor
- ✓ Usar objetos en JavaScript y comprender su importancia
- ✓ Agregar, eliminar y modificar claves y valores en un objeto
- ✓ Acceder a valores en un objeto con anotaciones de punto y paréntesis
- ✓ Listar las propiedades de un objeto



# Los objetos

Los objetos son colecciones de propiedades y cada propiedad se representa con un par clave-valor. La representación de un objeto en JavaScript son llaves {}.

## Par clave-valor

Es un conjunto de dos elementos de datos vinculados.

La clave es un string que identifica una propiedad de un objeto. Por lo general, corresponde al nombre de la propiedad a la que desea acceder.

Las claves son únicas en un objeto; una clave siempre tendrá un solo valor asociado.



## ¿Por qué deberíamos usar objetos?

Los objetos son útiles para agrupar valores que están relacionados. Almacenan relaciones entre variables y propiedades utilizando asociaciones de clave y valor.

El valor de la propiedad en un objeto puede ser de cualquier tipo que necesitemos: strings, numbers, arrays, funciones, o incluso otros objetos.

## Sintaxis de objeto

```
let someObject = {  
  key1: value,  
  key2: value,  
  key3: value  
}
```



En un par de lecciones, aprenderemos qué son los constructores, pero ten en cuenta que también puedes crear objetos utilizando la sintaxis del constructor de objetos

```
let someObject = new Object();
```

PREGUNTA:

Cuál es la clave y cuál es el valor?

```
{  
  athletics100Men: "Justin Gatlin"  
}
```



## Acceder a los valores dentro del objeto:

- puntos
- corchetes

```
let olympicRecords = {  
  athletics100Men: "Justin Gatlin",  
  athleticsLongJumpMen: "Mike Powel"  
}  
  
olympicRecords.athletics100Men;           // => "Justin Gatlin"  
olympicRecords["athleticsLongJumpMen"]; // => "Mike Powel"
```

## Agregar propiedades al objeto

```
let olympicRecords = {  
  athletics100Men: "Justin Gatlin",  
  athleticsLongJumpMen: "Mike Powel",  
}  
  
olympicRecords.swimming200Men = "Michael Phelps";
```

0

```
let olympicRecords = {  
  athletics100Men: "Justin Gatlin",  
  athleticsLongJumpMen: "Mike Powel",  
  swimming200Men: "Michael Phelps"  
}  
  
olympicRecords["swimming400Women"] = "Katie Ledecky";
```



## Prop in object

Verificar si existe cierta propiedad en un objeto. Devuelve un valor booleano dependiendo de si la propiedad existe o no.

Sintaxis:

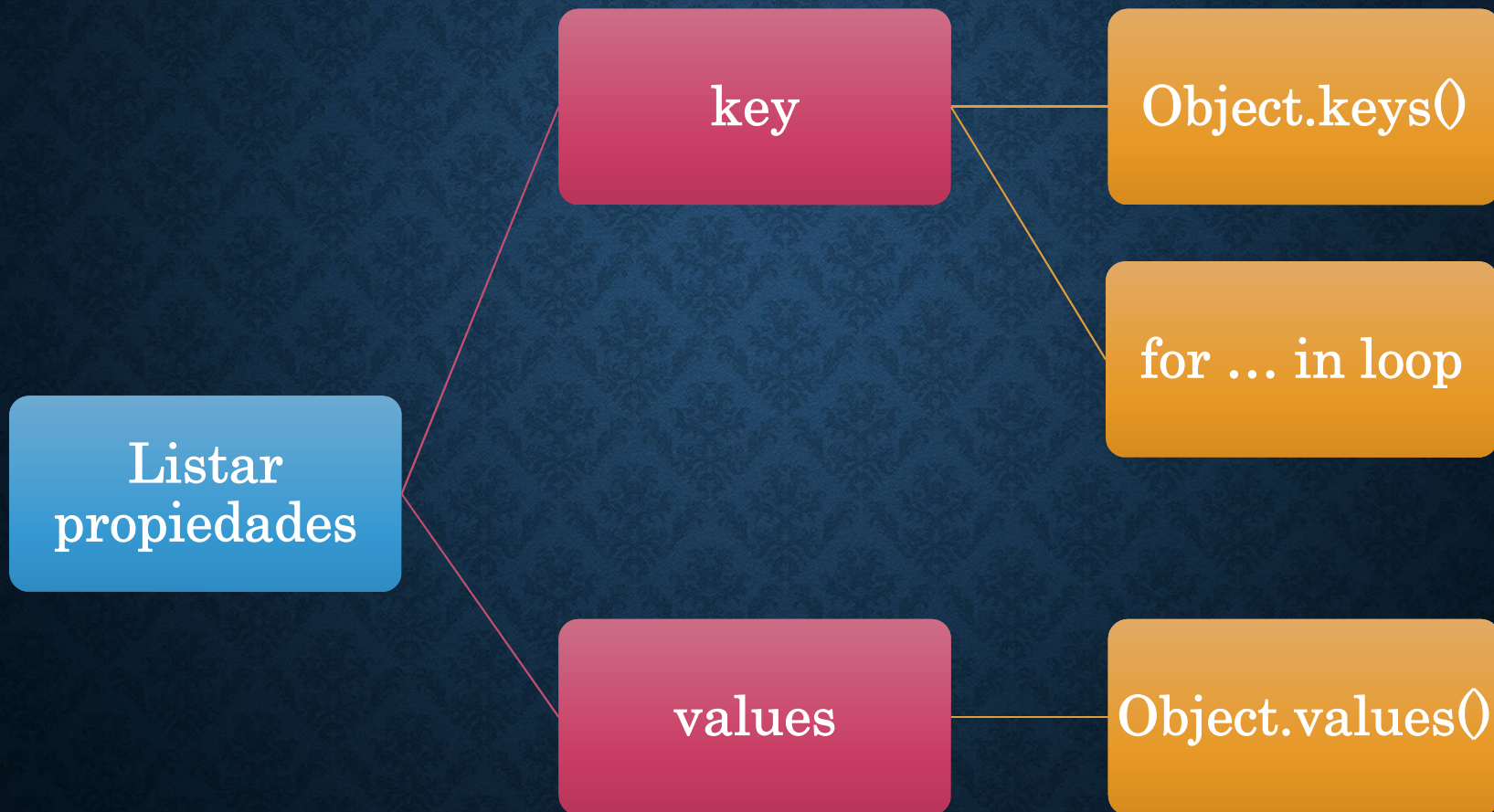
```
prop in objectName
```

## Eliminar propiedades

Tenemos el operador *delete* para eliminar claves de un objeto. Solo tienes que especificar qué clave deseas eliminar:

```
delete olympicRecords.doubleOllie;  
  
// or  
  
delete olympicRecords["doubleOllie"];
```






- KEY

Object.keys()

Recibe, como parámetro, el objeto que desea inspeccionar. Por ejemplo:

```
let olympicRecords = {  
  athletics100Men: "Usain Bolt",  
  athleticsLongJumpMen: "Mike Powel",  
  swimming200Men: "Michael Phelps",  
  swimming400Women: "Katie Ledecky"  
}  
 Object.keys(olympicRecords);  
  
// => ["athletics100Men",  
// "athleticsLongJumpMen",  
// "swimming200Men",  
// "swimming400Women"]
```



- KEY

for ... in loop



```
let olympicRecords = {  
  athletics100Men: "Usain Bolt",  
  athleticsLongJumpMen: "Mike Powel",  
  swimming200Men: "Michael Phelps",  
  swimming400Women: "Katie Ledecky"  
}  
for (let key in olympicRecords) {  
  console.log(key);  
}  
  
// => ["athletics100Men",  
// "athleticsLongJumpMen",  
// "swimming200Men",  
// "swimming400Women"]
```

- VALUE

Object.values()

```
let olympicRecords = {  
  athletics100Men: "Usain Bolt",  
  athleticsLongJumpMen: "Mike Powel",  
  swimming200Men: "Michael Phelps",  
  swimming400Women: "Katie Ledecky"  
}  
  
Object.values(olympicRecords);  
  
// => ["Usain Bolt",  
// "Mike Powel",  
// "Michael Phelps",  
// "Katie Ledecky"]
```



Object.entries() → ES2017 / ES8

Devuelve un array cuyos elementos son arrays correspondientes a los pares de propiedad enumerable [clave, valor] encontrados en el objeto.

```
const habilidades = { strength: 50, health: 100, height: 170 }  
const habilidadesArray = Object.entries(habilidades)  
console.log(habilidadesArray);
```

`Object.fromEntries( )` → ES2019 / ES10

También podremos hacer el camino inverso: pasar de un array de arrays a un Object de JS.

Puedes usarlo nativamente a partir de: Node.js v12.4.0, Chrome 73, Firefox 67 y Safari 12.1

```
const arr = [["strength", 50], ["health", 100], ["height", 170]];
const obj = Object.fromEntries(arr)
console.log(obj) // { strength: 50, health: 100, height: 170 }
```



¿Podemos usar `const` para declarar un objeto?

La respuesta es sí.

En el caso de declarar un objeto usando la keyword ***const***, se pueden agregar nuevas propiedades y valores pero **no se puede reasignar**.

Veamos un ejemplo

# PRÁCTICA



## Resumiendo: Aprendimos...

- ✓ algunos fundamentos de programación como pares clave-valor
- ✓ qué es un Objeto y por qué es una buena práctica usarlos
- ✓ cómo podemos crear objetos e interactuar con sus propiedades y valores
- ✓ un método `Object` para enumerar todas las propiedades de un objeto