


JAVASCRIPT

Objetos

A logo consisting of the letters 'JS' in a bold, dark blue, sans-serif font, centered within a light yellow square. The square is positioned on the right side of the image, against a solid yellow background.

JS

Objetos

Es un **tipo de dato** que nos permite crear colecciones de “variables” pero que a **diferencia de los arrays**, estas se encuentran **identificadas** mediante una **“clave”** en lugar de un índice.

Partes de un Objeto

Nuestros objetos poseen propiedades.

Una **propiedad** está definida mediante un par de **clave/valor** y separada de otra propiedad mediante una **coma**.

Las **propiedades** de un objeto se encuentran **encerradas en un par de llaves** que definen sus límites y la **declaración** se puede asignar a una variable tradicional.

```
const superheroe = {  
  alias: 'Superman',  
  nombre: 'Clark',  
  apellido: 'Kent',  
  universo: 'DC'  
};
```

Atributos de las propiedades

Cada propiedad de un objeto posee 4 atributos:

value: valor de la propiedad en cuestión.

configurable: nos permite definir si los atributos de la propiedad van a poder ser **modificados**.

enumerable: controla si la propiedad va a ser mostrada cuando se enumeren las propiedades del objeto.

writable: nos permite definir si el valor de una propiedad va a poder ser modificado o no.

Para acceder a los atributos usamos:

Object.getOwnPropertyDescriptor(target, propiedad):
donde target es el atributo que deseamos ver de esa propiedad.

Object.defineProperty(myObj, propiedad, {atributos}):
lo usamos para redefinir una propiedad en específico.

Para acceder a las propiedades usamos:

Object.keys(): Devuelve un arreglo que contiene todos los nombres de las propiedades.

Object.values(): Devuelve un arreglo que contiene todos los valores correspondientes a las propiedades.

Leer propiedades

También es posible acceder a las **propiedades** de un objeto a través del punto o los corchetes.

Dado el siguiente objeto:

```
const mascota = {  
  nombre: 'Firulaïs',  
  familia: 'Perro',  
  raza: 'Caniche',  
  peso: 3000,  
  edad: '7 meses'  
};
```

Podemos acceder a sus propiedades de la siguiente manera:

```
console.log(mascota.nombre); // Firulaïs  
console.log(mascota.peso); // 3000  
  
console.log(mascota['familia']); // Perro  
console.log(mascota['edad']); // 7 meses
```

Añadir propiedades

También podemos agregar propiedades a un objeto existente, para ello hacemos simplemente lo siguiente:

```
mascota.color = 'blanco';
```

Si ahora mostramos nuestro objeto por consola, tendremos el siguiente resultado:

```
console.log(mascota);  
  
▼ {nombre: 'Firulais', familia:  
  color: "blanco"  
  edad: "7 meses"  
  familia: "Perro"  
  nombre: "Firulais"  
  peso: 3000  
  raza: "Caniche"  
  ► [[Prototype]]: Object
```

Recorrer un objeto

Si bien con un poco de ingenio podemos utilizar el bucle for, javascript nos provee de una estructura más sencilla para poder iterar sobre nuestros objetos.

Esta estructura se llama **For ... In**

```
var obj = {a: 1, b: 2, c: 3};

for (const prop in obj) {
  console.log(`obj.${prop} = ${obj[prop]}`);
}

// Produce:
// "obj.a = 1"
// "obj.b = 2"
// "obj.c = 3"
```

En el ejemplo vemos cómo podemos acceder a cada clave del objeto y con ella **acceder a los respectivos valores** asignados a esa clave.

Métodos

Se conoce con este nombre a las **funciones** que declaremos **dentro de un objeto** y si bien **ya hemos utilizado** otros métodos a lo largo del curso **ahora veremos** cómo se declaran en un objeto propio.

En este caso, **declaramos** una propiedad llamada saludar y le **asignamos** una **función** que imprime por consola el valor de **la propiedad** sonido.

El uso de la palabra reservada **“this”**, hace referencia a que **“sonido”** **es parte del mismo objeto y no un valor externo**.

```
const mascota = {  
  nombre: 'Firulais',  
  familia: 'Perro',  
  raza: 'Caniche',  
  peso: 3000,  
  edad: '7 meses',  
  sonido: 'Guau Guau!',  
  saludar: function() {console.log(this.sonido)}  
};  
  
mascota.saludar();
```