DigitalHouse >
Coding School

OBJETOS

LITERALES

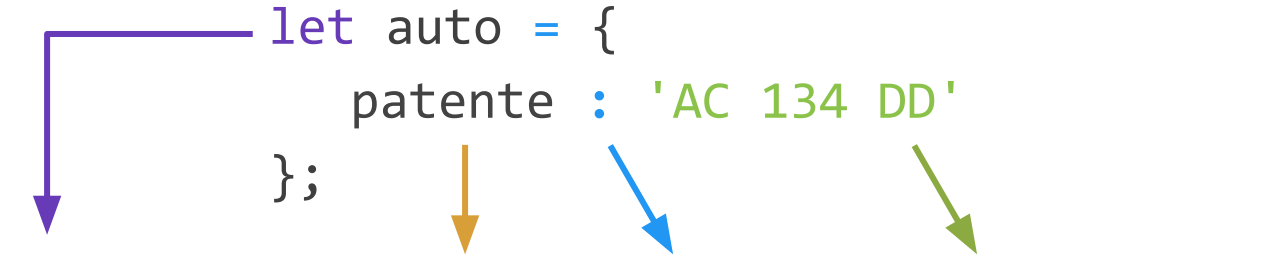
“

Podemos decir que son la **representación** en **código** de un **elemento** de la **vida real**.

ESTRUCTURA DE OBJETO LITERAL

Un **objeto** es una estructura de datos que puede contener **propiedades** y **métodos**.

Para crearlo usamos llave de apertura y de cierre { }.



Objeto

Definimos un nombre para nuestro **objeto**.

Propiedad

Definimos un nombre para la **propiedad** del objeto.

Dos puntos

Asigna el valor de la derecha a la propiedad de la izquierda.

Valor

Puede ser cualquier **tipo de dato** que conocemos.

ESTRUCTURA DE OBJETO LITERAL

Un **objeto** puede tener la cantidad de propiedades que queramos, si hay más de una las separamos con comas `,`.

Con la notación **objeto.propiedad** accedemos al **valor** de cada una de ellas.

```
lettenista = {
    nombre: 'Roger',
    apellido: 'Federer'
};

console.log(tenista.nombre) // Roger
console.log(tenista.apellido) // Federer
```

MÉTODOS DE UN OBJETO

Una propiedad puede **almacenar** cualquier **tipo dato**.

Si una propiedad almacena una función, diremos que es un **método** del objeto.

```
let tenista = {
    nombre: 'Roger',
    apellido: 'Federer',
    edad: 38,
    saludar: function() {
        return '¡Hola! Me llamo Roger';
    }
};
```

MÉTODOS DE UN OBJETO

Para ejecutar el método de un objeto usamos la notación **objeto.metodo()**, los paréntesis del final son los que hacen que el método se ejecute.

```
let tenista = {
    nombre: 'Roger',
    apellido: 'Federer',
    saludar: function() {
        return '¡Hola! Me llamo Roger';
    }
};

console.log(tenista.saludar()) // ¡Hola! Me llamo Roger
```

TRABAJANDO DENTRO DEL OBJETO

La palabra reservada **this** hace referencia al objeto en sí desde el cual estamos invocando la palabra.

Con la notación **this.propiedad** accedemos al valor de **cada propiedad interna** de ese objeto.

```
let tenista = {
    nombre: 'Roger',
    saludar: function() {
        return '¡Hola! Me llamo ' + this.nombre;
    }
};

console.log(tenista.saludar()) // ¡Hola! Me llamo Roger
```

CONSTRUIR UN OBJETO

Javascript nos da una opción más para crear un objeto y es a través del uso de una **función constructora**.

La función constructora nos permite armar un molde y luego crear todos los objetos que necesitemos.

La función recibe un parámetro por cada propiedad que queramos asignarle al objeto.

```
function auto(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

CONSTRUIR UN OBJETO

Objeto

Definimos un **nombre** para la función, que será el nombre de nuestro **objeto**. Por convención, solemos nombrar a los objetos con la primera letra mayúscula.

```
function auto(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

CONSTRUIR UN OBJETO

Parámetros

Definimos la cantidad de parámetros que consideremos necesarios para crear nuestro objeto.

```
function auto(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

CONSTRUIR UN OBJETO

Propiedades

Con la notación **this.propiedad** definimos la propiedad del objeto que estamos creando en ese momento.

Por lo general los valores de las propiedades serán los que vengan por parámetros.

```
function auto(marca, modelo){
    this.marca = marca;
    this.modelo = modelo;
};
```

INSTANCIAR UN OBJETO

La función constructora Auto() espera dos parámetros: *marca* y *modelo*. Para crear un **objeto Auto** debemos usar la palabra reservada **new** y llamar a la función pasándole los parámetros que espera.

```
{}  let miAuto = new Auto('Ford', 'Falcon');
```

Cuando ejecutamos el método **new** para crear un objeto, lo que nos devuelve es una **instancia**. Es decir, en la variable `miAuto` tendremos almacenada una instancia del objeto Auto. Usando la misma función, podemos instanciar cuantos autos queramos.

```
{}  let miOtroAuto = new Auto('Chevrolet', 'Corvette');
```