educación IT
CODE YOUR FUTURE

JavaScript desde cero

Módulo 3

Objetos

Objetos y su definición

En el mundo de la programación existe, desde hace poco más de treinta años, una rama denominada ***Programación Orientada a Objetos (P.O.O.)***. A través de esta, se busca desarrollar aplicaciones de *software* basadas en el paradigma de objetos.

Cuando hablamos de ***objetos***, nos referimos a **entidades que representan conceptos reales o abstractos en un formato digital**.

Si bien el paradigma de la P.O.O. nació a entre los 50's y 60's, recién comenzó a verse reflejado en lenguajes de programación, hacia fines de los 70's.

Cada objeto tiene un estado y un comportamiento:

- el **estado** es definido por sus **atributos** mientras que,
- el **comportamiento**, se define por sus **métodos**.

Los objetos se relacionan entre sí a través de sus mensajes y sus métodos y, su uso en el desarrollo de *software*, permite clarificar de forma efectiva aquellas tareas que definimos en cada línea de código que le da vida a nuestra aplicación.

Atributos y métodos

Cada objeto tiene un estado y un comportamiento:

- el **estado** es definido por sus ***atributos*** mientras que,
- el **comportamiento**, se define por sus ***métodos***.

Los objetos se relacionan entre sí a través de sus mensajes y sus métodos y, su uso en el desarrollo de *software*, permite clarificar de forma efectiva aquellas tareas que definimos en cada línea de código que le da vida a nuestra aplicación.

Propiedades y métodos

Entonces:

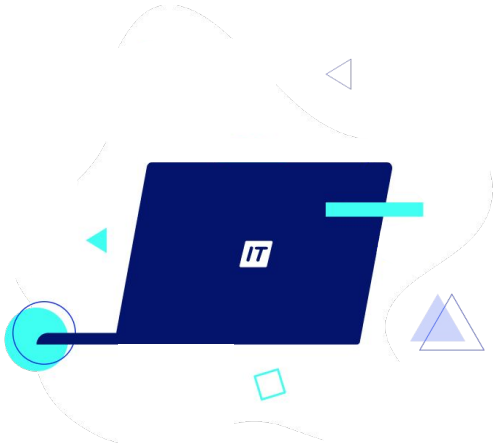
- Cada **característica** (o atributo) que representa a estos objetos dentro del ecosistema de *software*, suele llamarse ***propiedad***.
- Cada **comportamiento** de estos objetos, es representado por un ***método***.

Entidad abstracta

Se puede pensar en, por ejemplo, una ***computadora***. Si bien es algo físico, suele tratarse como una entidad abstracta por sus limitaciones o nicho específico.

Sus características pueden ser:

- Marca.
- Modelo.
- Posee *display.*
- Posee teclado.
- Posee *mouse pad* o puntero.
- Sistema operativo.
- Batería.



El objeto literal

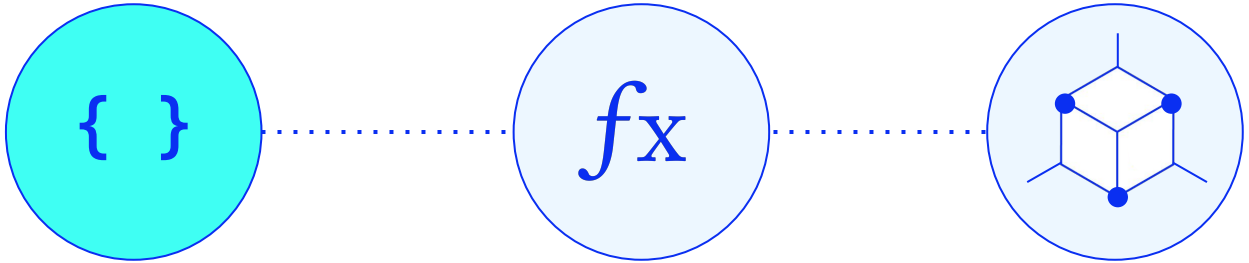
El objeto literal

JS es un lenguaje que evoluciona año tras año, desde 2015 hasta la actualidad. Previo a esto, su evolución se daba en períodos más espaciados (entre 3 y 5 años para que saliera una nueva versión). Y durante el período que lleva este lenguaje de programación, el mismo tuvo una evolución importante en la forma de trabajar con objetos.

Pero de toda su evolución, nos concentraremos solamente en el ***objeto literal***, dado que necesitamos entender su base para luego poder hablar de *array* de objetos.

El objeto literal en JS, **es el más simple de todos, dado que lo declaramos y comenzamos a utilizarlo inmediatamente**.

Además, su estructura de datos fácil de entender y leer, inspiró la creación del formato de transporte de datos conocido como JSON, y utilizado desde hace más de una década para intercambiar información entre aplicaciones de servidor y aplicaciones *frontend* (web, nativas, móviles).



Objeto literal
1999

Función constructora
2009

Clase JS
2015

Definición

Cuando se define un objeto literal, éste se debe crear a través de la palabra reservada **`const`**.

El **nombre** que le asignamos suele definirse con **su primera inicial en mayúsculas**, y debe ser una palabra (sustantivo), expresado en singular.

Su estructura de datos, se debe encerrar entre las llaves de bloque.

Ejemplos

Aquí tenemos una representación rápida del objeto literal denominado **`Persona`**, con algunos atributos (propiedades), que lo describen.

Cada propiedad llevará un valor asignado. Este valor puede ser expresado en cualquier tipo de dato (*string, boolean, number*).

Para definir múltiples propiedades, cada una con su valor, utilizamos la coma (**`,`**) como separador.

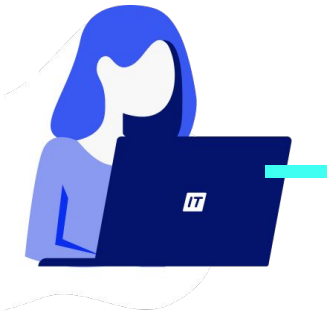
Objeto Literal

```
const Persona = {
    nombreCompleto: '',
    edad: 0,
    profesion: '',
    genero: ''
}
```

Además de los datos del tipo ***string*** y ***number***, **podemos agregar a sus propiedades, otros valores** como ser un *array* de elementos, un objeto interno, entre otros tantos.

Objeto Literal

```
const Persona = {
    nombreCompleto: 'Fer Moon',
    edad: 47,
    profesion: 'Profesor',
    genero: 'Masculino'
}
```



Para leer el valor de alguna de las propiedades, simplemente escribimos **`Objeto.propiedad`**. Así se obtiene el valor almacenado.

Para cambiarlo, se utiliza esta misma estructura y se suma el operador de asignación igual (**=**), más el nuevo valor.

Objeto Literal

```
console.log(Persona.nombreCompleto);

console.log(Persona.profesion);


Persona.edad = 21;
```

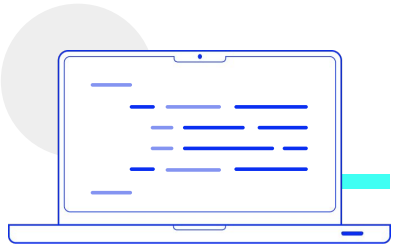
Arreglos de objetos

Arreglos de objetos

En JavaScript, es posible crear *arrays* de objetos literales.

Estas estructuras son las más apropiadas para **representar un conjunto de objetos usualmente del mismo tipo**, y poder trabajar con ellos de forma efectiva.

Este orden permitirá trabajar con los datos **de forma prolija**, aprovechando métodos de búsqueda y transformación.



Ejemplo:

```
const productos = [
  { id: 1, nombre: 'TV 55', precio: 1000, stock: 30, categoria: 'Video'},
  { id: 2, nombre: 'Laptop', precio: 1500, stock: 100, categoria: 'Computación'},
  { id: 3, nombre: 'iPhone 8', precio: 800, stock: 42, categoria: 'Telefonía'},
  { id: 4, nombre: 'Tablet', precio: 500, stock: 71, categoria: 'Computación'},
  { id: 5, nombre: 'Pods', precio: 100, stock: 28, categoria: 'Audio'},
  { id: 6, nombre: 'MP3 player', precio: 200, stock: 11, categoria: 'Audio'},
  { id: 7, nombre: 'Videocámara', precio: 300, stock: 22, categoria: 'Video'},
  { id: 8, nombre: 'Smartwatch', precio: 250, stock: 88, categoria: 'Computación'},
  { id: 9, nombre: 'Impresora', precio: 150, stock: 14, categoria: 'Accesorios'},
  { id: 10, nombre: 'Altavoces', precio: 120, stock: 18, categoria: 'Audio'}
];
```

Si se depura una aplicación de *software*, el método **`console.table()`** permite ver tabulados los objetos de este *array*. Mucho más cómodo y práctico que el clásico método **`.log()`**.

También se aprecia de forma más clara qué índice le corresponde a cada objeto del *array*.

(índice)	id	nombre	precio	stock	categoria
0	1	'TV 55'	1000	30	'Video'
1	2	'Laptop'	1500	100	'Computación'
2	3	'iPhone 8'	800	42	'Telefonía'
3	4	'Tablet'	500	71	'Computación'
4	5	'Pods'	100	28	'Audio'
5	6	'MP3 player'	200	11	'Audio'
6	7	'Videocámara'	300	22	'Video'
7	8	'Smartwatch'	250	88	'Computación'
8	9	'Impresora'	150	14	'Accesorios'
9	10	'Altavoces'	120	18	'Audio'

educación IT
CODE YOUR FUTURE

Operaciones básicas con arreglos de objetos

Operaciones básicas con arreglos de objetos

Para **acceder a un objeto específico** del *array*, se utilizan **corchetes ([ ]) junto al índice que le corresponde al objeto**, para poder acceder al mismo y a sus propiedades.

```
const productos = [
  { id: 1, nombre: 'TV 55', precio: 1000, stock: 30, categoria: 'Video'},
  { id: 2, nombre: 'Laptop', precio: 1500, stock: 100, categoria: 'Computación'},
  { id: 3, nombre: 'iPhone 8', precio: 800, stock: 42, categoria: 'Telefonía'},
  { id: 4, nombre: 'Tablet', precio: 500, stock: 71, categoria: 'Computación'},
  { id: 5, nombre: 'Pods', precio: 100, stock: 28, categoria: 'Audio'},
  { id: 6, nombre: 'MP3 player', precio: 200, stock: 11, categoria: 'Audio'},
  { id: 7, nombre: 'Videocámara', precio: 300, stock: 22, categoria: 'Video'},
  { id: 8, nombre: 'Smartwatch', precio: 250, stock: 88, categoria: 'Computación'},
  { id: 9, nombre: 'Impresora', precio: 150, stock: 14, categoria: 'Accesorios'},
  { id: 10, nombre: 'Altavoces', precio: 120, stock: 18, categoria: 'Audio'}
];

console.log(productos[3])
// retornará: { id: 4, nombre: 'Tablet', precio: 500, stock: 71, categoria: 'Computación'}

console.log(productos[3].nombre) // retornará: 'Tablet'
```

Partiendo de un *array* de **productos** existente, **se define una función** para cargar nuevos productos.

Esta función recibe los valores del nuevo producto por parámetro.

- Arma un objeto literal con los valores recibidos.
- Llama al método **`.push()`** y le envía como parámetro el nuevo producto a agregar.

Veamos el código de la siguiente diapositiva.

```
const productos = [
  { id: 1, nombre: 'TV 55', precio: 1000, stock: 30, categoria: 'Video'},
  { id: 2, nombre: 'Laptop', precio: 1500, stock: 100, categoria: 'Computación'},
  { id: 3, nombre: 'iPhone 8', precio: 800, stock: 42, categoria: 'Telefonía'},
  { id: 4, nombre: 'Tablet', precio: 500, stock: 71, categoria: 'Computación'},
  { id: 5, nombre: 'Pods', precio: 100, stock: 28, categoria: 'Audio'},
  { id: 6, nombre: 'MP3 player', precio: 200, stock: 11, categoria: 'Audio'},
  { id: 7, nombre: 'Videocámara', precio: 300, stock: 22, categoria: 'Video'},
  { id: 8, nombre: 'Smartwatch', precio: 250, stock: 88, categoria: 'Computación'},
  { id: 9, nombre: 'Impresora', precio: 150, stock: 14, categoria: 'Accesorios'},
  { id: 10, nombre: 'Altavoces', precio: 120, stock: 18, categoria: 'Audio'}
];

function agregarProducto(nombre, precio, stock, categoria) {
    const nuevoProducto = {
        id: calcularNuevoId(),  //tenemos una función que retorna el ID que corresponde
        nombre: nombre,
        precio: precio,
        stock: stock,
        categoria: categoria
    }
    productos.push(nuevoProducto)
}
```

Objetos, *arrays* de elementos, *arrays* de objetos

Tanto los **objetos**, como **ambos tipos de *arrays***, **se declaran siempre anteponiendo la palabra reservada `const`**, y no `let`.

Esta práctica permite **evitar que se sobre-escriba** (nosotros mismos o algún compañero de equipo) el contenido de un objeto o *array*, en cualquier otra parte de nuestra aplicación, con un valor diferente al que tiene. Es una **forma útil de proteger la estructura de datos dinámica que usa JavaScript para declarar espacios de memoria que almacenen diferentes valores**.

Además, se cumplimenta la buena práctica de cualquier lenguaje de programación que dice:

> **Todo lo que no deba cambiar su valor inicialmente asignado, debe declararse siempre como constante.**

Revisión

- Repasar el concepto de **arreglo más objeto**.
- Mostrar una **lista de empleados y diferentes propiedades**.
- Estudiar los métodos **mostrados en el contenido**.
- Practicar los **métodos** para poder entenderlos de mejor forma.
- Trabajar en el ***Proyecto integrador***.
- Realizar las preguntas necesarias a la/el docente antes de continuar.

¡Sigamos trabajando!