

Agencia de Aprendizaje a lo largo de la vida

Desarrollo Fullstack







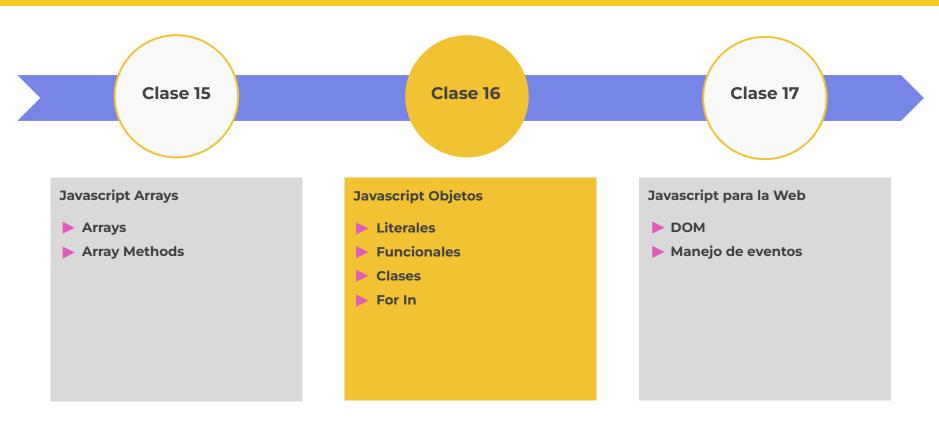
Les damos la bienvenida

Vamos a comenzar a grabar la clase













JAVASCRIPT Objetos



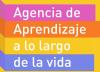
Agencia de Aprendizaje a lo largo de la vida





Objetos

Es un **tipo de dato** que nos permite <u>crear colecciones de</u> <u>"variables"</u> 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 <u>definida</u> mediante un par de <u>clave/valor</u> y <u>separada</u> de otra propiedad mediante una <u>coma</u>.

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

```
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 <u>si la propiedad va a ser</u> <u>mostrada</u> cuando se enumeren las propiedades del objeto.

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

Para acceder a los atributos usamos:

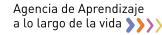
Object.getOwnPropertyDescriptor(target, propiedad): donde target es el atributo que deseamos ver de esa propiedad.

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

Para acceder a las propiedades usamos:

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

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







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: 'Firulais',
   familia: 'Perro',
   raza: 'Caniche',
   peso: 3000,
   edad: '7 meses'
};
```

Podemos acceder a sus propiedades de la siguiente manera:

```
console.log(mascota.nombre); // Firulais
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, <u>tendremos el siguiente resultado</u>:





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.

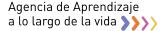
Esta estructura se llama For ... In

```
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 <u>podemos</u> <u>acceder a cada clave del objeto</u> y con ella **acceder a los respectivos valores** asignados a esa clave.







Métodos

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

En este caso, **declaramos** una propiedad llamada <u>saludar</u> y le **asignamos** una <u>función</u> que <u>imprime por consola</u> 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)}
};

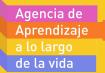
mascota.saludar();
```





Otros objetos

Hasta ahora vimos cómo trabajan los objetos
LITERALES de javascript, pero esta, aunque si es la
más popular, no es la única forma de declarar
nuestros objetos en este lenguaje.







Objetos Funcionales

Se crean a partir de funciones y nos permiten <u>definir "moldes"</u> para luego poder **realizar copias** de nuestros objetos con mismas propiedades pero **distintos valores**.

```
function Mascota(nombre, familia, raza, peso, edad, sonido) {
   this.nombre = nombre;
   this.familia = familia;
   this.raza = raza;
   this.peso = peso;
   this.edad = edad;
   this.sonido = sonido;
   this.saludar = function() {
       console.log(this.sonido);
```





Objetos Funcionales

En el caso anterior **definimos** un objeto llamado Mascota que poseía las <u>mismas</u> <u>propiedades</u> que nuestro <u>objeto literal mascota</u> declarado al principio de la clase, sin embargo, **este objeto no tiene asociado ningún valor**.

Para poder hacerlo, ahora debemos crear una *instancia* de mi <u>objeto Mascota,</u> es decir una copia con valores únicos.

```
const miPerro = new Mascota('Firulais','Perro','Caniche',3000,'7 meses','Guau Guau!');
```

```
miPerro.saludar(); // Guau Guau!
```





Objetos de Clase

En este caso, las clases en javascript son un sugar syntax de la sintaxis utilizada en <u>los lenguajes de Paradigma Orientado a Objetos</u>. El **propósito** de agregar esta forma de declarar objetos era facilitar la escritura de aquellas <u>personas que estaban acostumbradas a dicho</u> paradigma.







Objetos de Clase

Si bien se parecen, a diferencia de los objetos funcionales, en las clases usamos la palabra reservada class y los parámetros del objeto son pasados mediante un **método constructor** dentro de la misma clase.

```
class Mascota {
    constructor(nombre, familia, raza, peso, edad, sonido) {
        this.nombre = nombre;
        this.familia = familia;
        this.raza = raza;
        this.peso = peso;
        this.edad = edad;
        this.sonido = sonido;
    }
    saludar = function() {
        console.log(this.sonido);
    }
}
```





Objetos y Arrays







Objetos y Arrays

Estas estructuras se llevan muy bien y es muy común **utilizarlas en conjunto** para crear colecciones de datos robustas que nos permitan contar con <u>estructuras fácilmente</u> iterables.

En este caso **tenemos un array** con una <u>colección de tareas</u> donde **cada tarea** es <mark>un objeto</mark> con sus **respectivas propiedades**.

¿Cómo podríamos recorrer esta estructura para obtener sólo los nombres de cada tarea?

```
let tasks = [
        id: 1.
        day: 'Lunes',
        task: "Leer un libro",
        state: "Pendiente"
        id: 2,
        day: 'Miércoles',
        task: "Sacar al Perro",
        state: "Pendiente"
        id: 3,
        day: 'Viernes',
        task: "Jugar Videojuegos",
        state: "Pendiente"
```





No te olvides de dar el presente





Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

Todo en el Aula Virtual.





Gracias