<codoacodo/>

Agencia de
Aprendizaje
a lo largo
de la vida

# Desarrollo Fullstack


BA

# Les damos la bienvenida

Vamos a comenzar a grabar la clase

## Clase 15

### Javascript Arrays

- Arrays
- Array Methods

## Clase 16

### Javascript Objetos

- Literales
- Funcionales
- Clases
- For In

## Clase 17

### Javascript para la Web

- DOM
- Manejo de eventos

# JAVASCRIPT

## Objetos

# 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: '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, 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();
```

# 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 definir "moldes" 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 mismas propiedades que nuestro objeto literal mascota 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 objeto Mascota, 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 los lenguajes de Paradigma Orientado a Objetos. El **propósito** de agregar esta forma de declarar objetos era facilitar la escritura de aquellas personas que estaban acostumbradas a dicho 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 estructuras fácilmente iterables.

En este caso **tenemos un array** con una colección de tareas donde **cada tarea** es un objeto 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