Un objeto en JavaScript es una colección de datos que se compone de propiedades y métodos. Es una de las estructuras de datos más fundamentales y poderosas en el lenguaje.

De manera muy sencilla, un objeto en JavaScript se puede definir como:

- Una entidad que puede contener datos (propiedades) y funcionalidad (métodos).
- Una forma de agrupar datos y funciones relacionadas.
- Una variable que puede

```
let persona : {...} = {
    nombre: "Juan",
    edad: 30,
    saludar: function() : void {
        console.log(`Hola, mi nombre es ${this.nombre}`);
    }
};

persona.saludar()
```

En JavaScript, existen varias formas de crear objetos. La más común es usar la notación de llaves {}:

```
let persona = {
  nombre: "Juan",
  edad: 30,
  profesion: "Ingeniero"
};
```

Otra forma es usar el constructor Object():

```
let persona = new Object();
persona.nombre = "Juan";
persona.edad = 30;
persona.profesion = "Ingeniero";
```

```
let dog = {
  nombre: "Buddy",
  edad: 5,
  raza: "Labrador Retriever"
};
```

```
let user1 = new Object(); // sintaxis de "constructor de objetos"

let user2 = {};  //  Esa declaración se llama objeto literal.

let user3 = {     // un objeto
    name: "John",  // En la clave "name" se almacena el valor "John"
    age: 30        // En la clave "age" se almacena el valor 30
};

user3.isAdmin = true;
delete user3.age;

console.log(user3);
```

Agregar una nueva propiedad

Borrar una propiedad

```
{ name: 'John', isAdmin: true }
```

```
let person = {
    id:2,
    name:"Arle",
    salary:"5000"
}
person["whatsapp cell phone"] = "3137082781";

console.log(person.name); // Arle
console.log(person["whatsapp cell phone"]); // 3137082781

let key = "Email";
person[key] = "arleth64@cue.edu.co";
console.log(person);

let key2 = "favorite";
person[key2 + 'Sport'] = "soccer";
console.log(person.favoriteSport); // soccer
```

objeto

propiedad compuesta

mostrar propiedad compuesta

pasando propiedad por variable

concatenar nombre propiedades

```
let payroll = {
    id: "1109214314",
    name:"Jose Suarez",
    overtime: 4,
    hourValue:20
}

console.log(`The value of hours is ${payroll.hourValue * payroll.overtime}`);

let key = "overtime";
let check = (key in payroll )? `found`: `No found`;
console.log(check);

for(let x in payroll){
    console.log(x, payroll[x]);
}
console.log("--------------------");
Object.entries(payroll).forEach(([key : string , value : number | string ]) => {
    console.log(`la clave es ${key} y su valor ${value}`);
});
```

objeto

cálculos con propiedades del objeto

Búsqueda de propiedades

Iterar objetos

1

2

```
let animal = {
    name: "cat",
    color: "black"
}

let animal2 = animal;
console.log(animal2.name);

let dataAnimal = {
    eyes: "green"
}

let newAnimal = {   // { name: 'cat', color: 'black', eyes: 'green' }

    ...animal,
    ...dataAnimal
}
console.log(newAnimal);
```

crear copia

concatena objetos

```
let user = {
    name: "John",
    age: 30,

    sayHi() {
        return `hola ${this.name} tienes ${this.age}`;
    },
    set val(value) {
        if (value.length < 4) {
            console.log("El nombre es demasiado corto, necesita al menos 4 caracteres");
            return;
        }
        this.name = value;
    },
    get fullData() {
        return `${this.name}  tiene ${this.age} años`;
    }
};
console.log(user.sayHi());
user.val = "Maria";
console.log(user.fullData);
```

método

método set (cargar)

método get (obtener)

llamada a método

asignación de valor y validación.

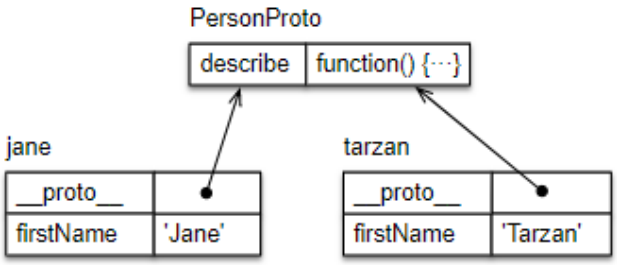
Encadenamiento opcional para obtener propiedades y llamadas a métodos

```
const persons = [
    {   surname: 'Zoe',
        address: {
            street: {
                name: 'Sesame Street',
                number: '123',
            },
        },
    },
    {   surname: 'Mariner',     },
    {
        surname: 'Carmen',
        address: {
        },
    },
];
const streetNames = persons.map(
    p => p.address?.street?.name);

console.log(streetNames);// [ 'Sesame Street', undefined, undefined ]
```

Si el valor antes del signo de interrogación no es ni undefined ni null, realice la operación después del signo de interrogación.
De lo contrario, regresa undefined.

Los prototipos son el único mecanismo de herencia de JavaScript: cada objeto tiene un prototipo que es nullo un objeto.

```
const PersonProto = {
    describe() {
        return 'Person named ' + this.firstName;
    },
};
const jane = {
    __proto__: PersonProto,
    firstName: 'Jane',
};
const tarzan = {
    __proto__: PersonProto,
    firstName: 'Tarzan',
};

console.log(jane.describe()); // Person named Jane
console.log(tarzan.describe()); // Person named Tarzan
```



Ejemplo:

```
let user = {
    name: "John",
    surname: "Smith",

    set fullName(value) {
        [this.name, this.surname] = value.split(" ");
    },

    get fullName() {
        return `${this.name} ${this.surname}`;
    }
};

let admin = {
    __proto__: user,
    isAdmin: true
};

console.log(admin.fullName); // John Smith (*)
// ¡Dispara el setter!
admin.fullName = "Alice Cooper"; // (**)

console.log(admin.fullName); // Alice Cooper , estado de admin modificado
console.log(user.fullName)
```

admin hereda de user y agrega una propiedad adicional.

Material de apoyo

1. Visite los siguientes enlaces:
   https://www.w3resource.com/javascript-exercises/javascript-array-exercises.php
   https://www.w3resource.com/javascript-exercises/javascript-object-exercises.php