Los objetos son usados para almacenar colecciones de varios datos y entidades más complejas asociados con un nombre clave.

```
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.

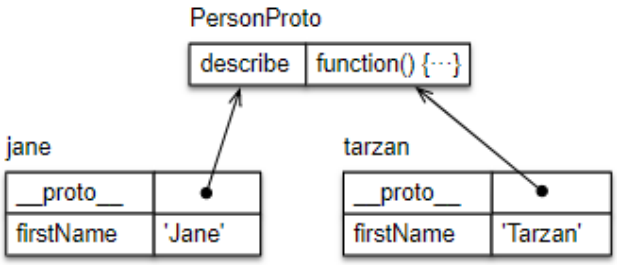
Encadenamiento opcional para obtener propiedades y llamadas a métodos

```
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

Ejercicio 1:

El software que se desarrollará controlará un cajero automático (ATM) a través de una simulación usando el lenguaje de programación JavaScript.

El cajero automático atenderá a un cliente a la vez. Se le pedirá al cliente que inserte su documento de identidad y su pin de 4 dígitos, los cuales se enviarán al banco para su validación como parte de cada transacción. El cliente podrá entonces realizar una o más transacciones. El menú se mostrará en la consola hasta que el cliente indique que no desea realizar más transacciones.

El cajero automático debe ser capaz de proporcionar los siguientes servicios al cliente:

- Un cliente debe poder realizar un retiro de efectivo de cualquier cuenta adecuada vinculada al documento de identidad, en múltiplos de $50000. Se debe obtener la aprobación del banco antes de entregar efectivo.
- Un cliente debe poder realizar un depósito en cualquier cuenta vinculada al documento de identidad, consistente en efectivo y/o cheques. El cliente ingresará el monto del depósito en el cajero automático e indicar si es efectivo o cheque.
- Un cliente debe poder realizar una transferencia de dinero entre dos cuentas cualesquiera vinculadas a al documento de identidad.
- Un cliente debe poder realizar una consulta de saldo de cualquier cuenta vinculada al documento de identidad.
- El cajero automático comunicará al cliente los resultados de cada transacción dependiendo de su tipo. Ejemplo “retiro exitoso, puede tomar x dinero de la bandeja principal”
- Si el banco determina que el PIN del cliente no es válido, se le pedirá al cliente que vuelva a ingresar el PIN antes de que se pueda continuar con la transacción. Si el cliente no puede ingresar correctamente el PIN después de tres intentos saldrá de la aplicación.
- El cajero automático tendrá un panel de operador con un interruptor que permitirá apagar o encender el cajero.

Ejercicio 2:

Desarrollar en JavaScript un programa para la gestión reservas de un hotel, el cual, debe tener las siguientes características y consideraciones:

1. Un cliente puede reservar cualquier tipo de habitación: individual, doble y familiar.

2. Las habitaciones pueden ser para fumadores o no fumadores.
3. Las mascotas solo se aceptan en habitaciones familiares.
4. El hotel cuenta con 3 habitaciones de cada tipo.
5. No se puede exceder el número de personas por habitación: individual 2 personas, 4 personas para doble y 6 personas para familiar.
6. El hotel necesita una estadística de las reservas: nombre de quien reserva, país de origen, número de personas, el periodo de la estadía, número de personas que están ocupando el hotel y si el huésped trajo mascota.