# Objetos en JavaScript: Guía de Partes Importantes

## 1. Cómo crear un objeto

**Explicación:** Un objeto se define con llaves {} y contiene pares clave-valor.

**Ejemplo:**

const persona = {  
 nombre: "Lucía",  
 edad: 30  
};

## 2. Cómo acceder a las propiedades de un objeto

**Explicación:** Se puede acceder usando notación de punto o corchetes.

**Ejemplo:**

console.log(persona.nombre); // "Lucía"  
console.log(persona["edad"]); // 30

## 3. Cómo modificar propiedades de un objeto

**Explicación:** Simplemente se asigna un nuevo valor a la propiedad.

**Ejemplo:**

persona.edad = 31;

## 4. Cómo agregar propiedades a un objeto

**Explicación:** Puedes crear nuevas propiedades asignándolas.

**Ejemplo:**

persona.apellido = "Gómez";

## 5. Cómo eliminar propiedades de un objeto

**Explicación:** Se usa la palabra clave delete.

**Ejemplo:**

delete persona.edad;

## 6. Métodos (funciones dentro de objetos)

**Explicación:** Un método es una función como propiedad de un objeto.

**Ejemplo:**

const auto = {  
 marca: "Toyota",  
 arrancar: function() {  
 console.log("El auto arranca");  
 }  
};  
auto.arrancar();

## 7. Notación de punto y notación de corchetes

**Explicación:** Dos formas de acceder a las propiedades.

**Ejemplo:**

console.log(auto.marca); // Notación de punto  
console.log(auto["marca"]); // Notación de corchetes

## 8. Iterar sobre propiedades de un objeto (for…in)

**Explicación:** Permite recorrer todas las propiedades enumerables.

**Ejemplo:**

for (let clave in persona) {  
 console.log(clave, persona[clave]);  
}

## 9. Uso de this en métodos de objetos

**Explicación:** this se refiere al propio objeto dentro de un método.

**Ejemplo:**

const perro = {  
 nombre: "Max",  
 ladrar: function() {  
 console.log(this.nombre + " está ladrando");  
 }  
};  
perro.ladrar();

## 10. Objetos anidados (objetos dentro de objetos)

**Explicación:** Un objeto puede tener propiedades que son otros objetos.

**Ejemplo:**

const usuario = {  
 nombre: "Ana",  
 direccion: {  
 calle: "Av. Siempre Viva",  
 numero: 742  
 }  
};  
console.log(usuario.direccion.calle);

## 11. Objetos dentro de arrays

**Explicación:** Un array puede almacenar objetos.

**Ejemplo:**

const productos = [  
 { nombre: "Laptop", precio: 1500 },  
 { nombre: "Mouse", precio: 20 }  
];

## 12. Arrays dentro de objetos

**Explicación:** Un objeto puede tener propiedades que sean arrays.

**Ejemplo:**

const alumno = {  
 nombre: "Luis",  
 calificaciones: [8, 9, 10]  
};

## 13. Comparar objetos

**Explicación:** Dos objetos distintos nunca son iguales aunque tengan los mismos valores (salvo que sean el mismo objeto en memoria).

**Ejemplo:**

const obj1 = {a:1};  
const obj2 = {a:1};  
console.log(obj1 === obj2); // false

## 14. Clonar objetos (shallow copy y deep copy)

**Explicación:** Copia superficial: solo copia el nivel superior. Copia profunda: copia todos los niveles anidados.

**Ejemplo:**

const original = {a:1, b:{c:2}};  
const copiaSuperficial = {...original};  
const copiaProfunda = JSON.parse(JSON.stringify(original));

## 15. Métodos para obtener llaves, valores y entradas

* Object.keys(obj) devuelve array de llaves
* Object.values(obj) devuelve array de valores
* Object.entries(obj) devuelve array de [llave, valor]

**Ejemplo:**

Object.keys(persona); // ["nombre", "apellido"]  
Object.values(persona); // ["Lucía", "Gómez"]  
Object.entries(persona); // [["nombre","Lucía"],["apellido","Gómez"]]

## 16. Propiedades dinámicas en objetos

**Explicación:** Puedes crear propiedades usando variables como nombre.

**Ejemplo:**

const prop = "edad";  
persona[prop] = 35;

## 17. Definir getters y setters

**Explicación:** Permiten definir funciones para obtener o establecer valores.

**Ejemplo:**

const persona2 = {  
 nombre: "Sofía",  
 \_edad: 25,  
 get edad() {  
 return this.\_edad;  
 },  
 set edad(valor) {  
 if (valor > 0) this.\_edad = valor;  
 }  
};  
persona2.edad = 28;  
console.log(persona2.edad);