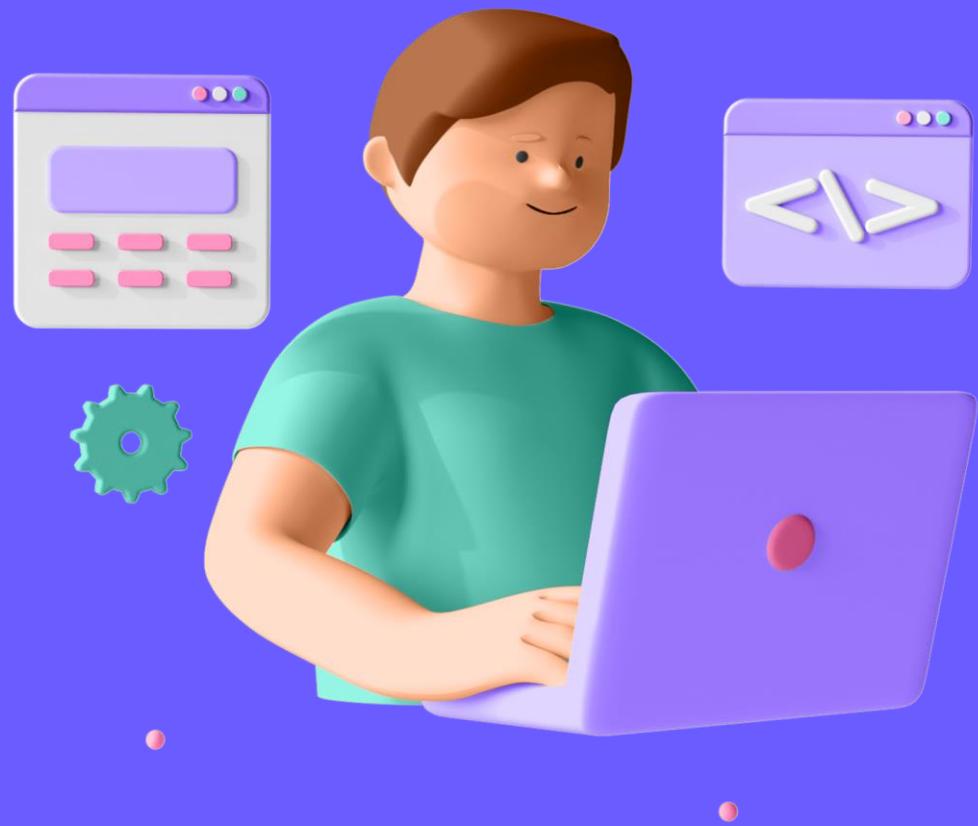




Somos un **ecosistema** de desarrolladores de software

Java Script



```
<!--  
n">  
--> BEGIN NAVIGATION  
" >Home</a></li>  
.html">Home Events</a></li>  
nu.html">Multiple Column Men  
<a href="#" class="current":  
button-header.html">Tall But  
ogo.html">Image Logo</a></  
mber="tall-logo.html">Ta  
<a href="#">Carousels</a>  
ith-slider.html">Variat  
ider.html">Testimoni
```

Objetos

Propiedades

En JavaScript, los objetos son estructuras de datos fundamentales y versátiles que permiten almacenar y organizar información de manera más compleja que los tipos de datos simples, como números o cadenas. Un objeto en JavaScript es una colección de pares clave-valor, donde cada clave (también llamada propiedad) está asociada a un valor. Estos valores pueden ser de cualquier tipo, incluyendo números, cadenas, funciones, otros objetos, etc.



```
//Los literales de los objetos son las llaves {}  
const objeto = {};
```

Objetos

Sintaxis y propiedades

— □ ×

```
let persona = {  
    nombre: 'Juan',  
    edad: 25,  
    ocupacion: 'programador',  
    saludar: function() {  
        console.log(`Hola, soy ${this.nombre} y tengo  
        ${this.edad} años.`);  
    }  
};
```

Objeto

Sintaxis y propiedades

En este ejemplo, persona es un objeto que tiene tres propiedades (nombre, edad, y ocupacion) y un método (saludar). Puedes acceder a las propiedades y métodos de un objeto utilizando la notación de punto o la notación de corchetes:

- □ ×

```
console.log(persona.nombre); // Acceder a la propiedad 'nombre'  
console.log(persona['edad']); // Acceder a la propiedad 'edad'  
persona.saludar(); // Llamar al método 'saludar'
```

Objeto

Añadiendo propiedades

- □ ×

```
//agregar propiedad
let persona = {
    nombre: 'Juan',
    edad: 25
};

// Añadir una nueva propiedad
persona.ocupacion =
'programador';

console.log(persona);
```

Objeto

Añadiendo propiedades

- □ ×

```
//agregar propiedad
let persona = {
    nombre: 'Juan',
    edad: 25
};

let nuevaPropiedad = 'ocupacion';

// Añadir una nueva propiedad usando notación de corchetes
persona[nuevaPropiedad] = 'programador';
console.log(persona);
```

Objeto

Object.defineProperty

- □ ×

```
//agregar propiedad
let persona = {
  nombre: 'Juan',
  edad: 25
};

// Añadir una nueva propiedad con Object.defineProperty
Object.defineProperty(persona, 'ocupacion', {
  value: 'programador',
  writable: true, // La propiedad es modificable
  enumerable: true, // La propiedad aparecerá en bucles for...in
  configurable: true // La propiedad puede ser eliminada y sus
                     atributos pueden ser modificados
});

console.log(persona);
```

Objeto

Object.assign

- □ ×

```
//agregar propiedad
let persona = {
    nombre: 'Juan',
    edad: 25
};

// Añadir propiedades usando Object.assign
Object.assign(persona, { ocupacion: 'programador', ciudad: 'Barcelona' });

console.log(persona);
```

Objeto

Métodos de un objeto

- □ ×

```
// Método
const user = {
    name: "Juan",
    talk: function() { return "Hola"; }
};

user.name;      // Es una variable
(propiedad), devuelve "Juan"
user.talk();    // Es una función (método),
se ejecuta y devuelve "Hola"
```

Objeto

Método .toString()

- □ ×

```
//Metodo .toString()
let persona = {
    nombre: 'Juan',
    edad: 30,
    ocupacion: 'programador',
    toString: function() {
        return `${this.nombre}, ${this.edad} años,
${this.ocupacion}`;
    }
};

console.log(persona.toString());
```

Objeto

Método .toString()

- □ ×

```
//Metodo .toString()
let persona = {
    nombre: 'Juan',
    edad: 30,
    ocupacion: 'programador',
    toString: function() {
        return `${this.nombre}, ${this.edad} años,
${this.ocupacion}`;
    }
};

console.log(persona.toString());
```

Objeto

Metodo to.String()

- □ ×

```
//Metodo .toString()  
const number = 42;          // Tipo Number  
number.toString();          // Devuelve "42"  
  
const booleano = true;      // Tipo Boolean  
booleano.toString();        // Devuelve "true"  
  
const regexp = /.+/.        // Tipo RegExp  
regexp.toString();          // Devuelve "/.+/"
```

Objetos

Métodos



```
const number = 42.5;
number.toString();           // Devuelve "42.5" (Método de variables de tipo Object)
number.toLocaleString();    // Devuelve "42,5" (Método de variables de tipo Object)
number.toFixed(3);          // Devuelve "42.500" (Método de variables de tipo Number)
```

Al crear una variable de un determinado tipo de dato, la variable será siempre también de tipo , ya que todas las variables heredan de este tipo de dato. Por lo tanto, nuestra variable tendrá:

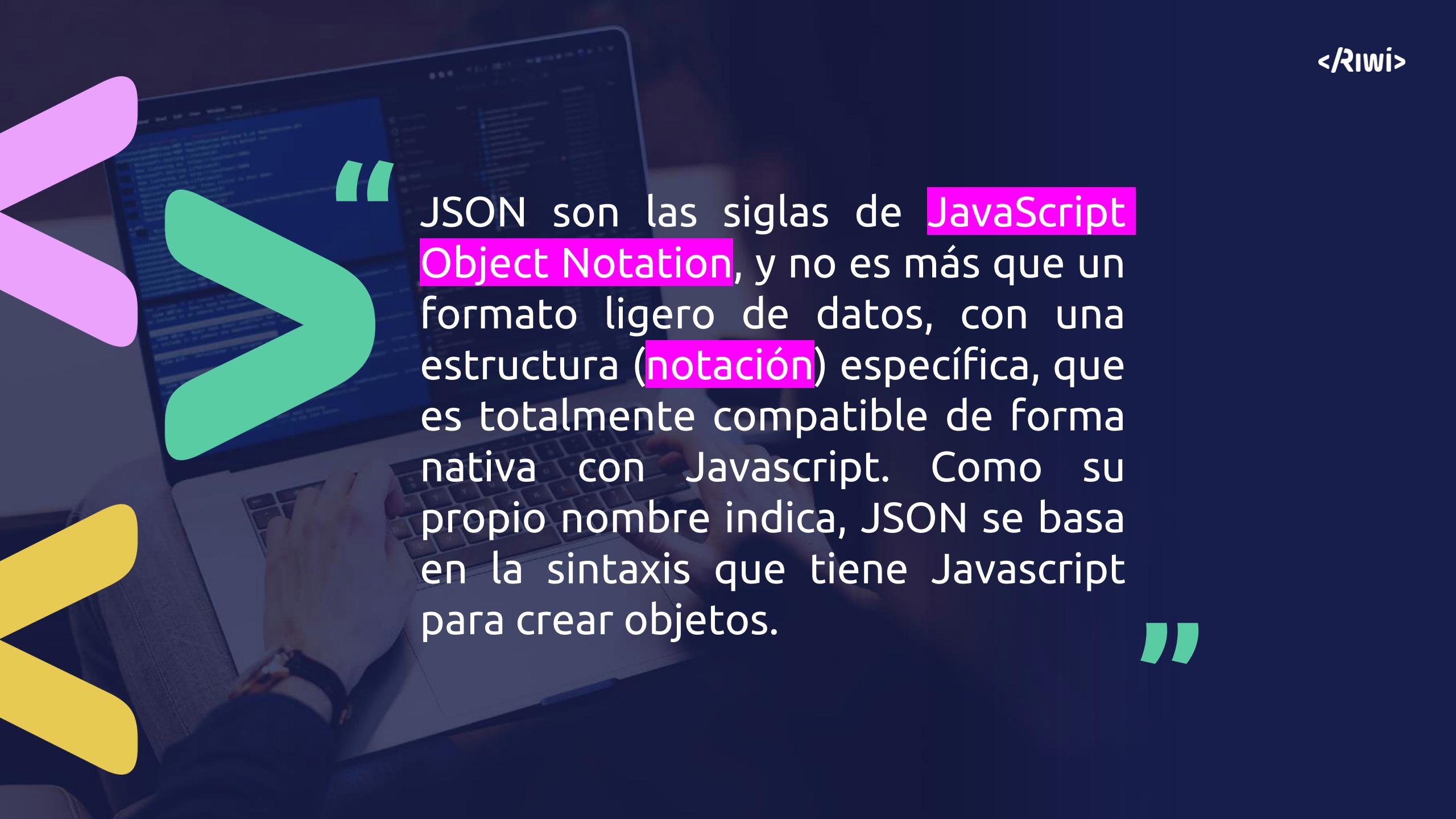
- Los métodos que implementemos nosotros personalmente
- Los métodos heredados de su propio tipo de dato
- Los métodos heredados del tipo

Objeto

Creando un String

- □ ×

```
const player = {  
    name: "Manz",           // Nombre del jugador  
    life: 4,                // Cantidad de vida actual  
    totalLife: 6,           // Máximo de vida posible  
    toString: function() {  
        return `${this.name} (${this.life}/${this.totalLife})`;  
    }  
};  
console.log(player);
```



JSON son las siglas de **JavaScript Object Notation**, y no es más que un formato ligero de datos, con una estructura (**notación**) específica, que es totalmente compatible de forma nativa con Javascript. Como su propio nombre indica, JSON se basa en la sintaxis que tiene Javascript para crear objetos.

Objetos

JSON

```
{  
  "nombre": "Juan",  
  "edad": 25,  
  "ocupacion": "programador",  
  "hobbies": ["leer", "programar", "viajar"],  
  "direccion": {  
    "calle": "123 Calle Principal",  
    "ciudad": "Ciudad Ejemplo",  
    "codigoPostal": "12345"  
  }  
}
```

Objetos

JSON

Método	Descripción
Parseo (De string a objeto)	
OBJECT JSON.parse(str)	Convierte el texto str (si es un JSON válido) a un objeto y lo devuelve.
AB Convertir a texto (De objeto a string)	
STRING JSON.stringify(obj)	Convierte un objeto obj a su representación JSON y la devuelve.
STRING JSON.stringify(obj, props)	Idem al anterior, pero filtra y mantiene solo las propiedades del ARRAY props.
STRING JSON.stringify(obj, props, spaces)	Idem al anterior, pero indenta el JSON a NUMBER spaces espacios.

Objetos

Convertir JSON a objeto

```
const json = `{
  "name": "Manz",
  "life": 99
}`;

const user = JSON.parse(json);

user.name; // "Manz"
user.life; // 99
```

Objetos

Convertir Objeto a JSON

```
const user = {  
    name: "Manz",  
    life: 99,  
    talk: function () {  
        return "Hola!";  
    },  
};  
  
JSON.stringify(user);
```

Objeto

.stringify al convertir siempre será a String lo que devuelva

```
const user = {  
    name: "Manz",  
    life: 99,  
    power: 10,  
};  
  
JSON.stringify(user, ["life"])          // '{"life":99}'  
JSON.stringify(user, ["name", "power"])  //  
'{"name":"Manz", "power":10}'  
JSON.stringify(user, [])                // '{}'  
JSON.stringify(user, null)             // '{"name":"Manz", "life":99, "power":10}'
```

Objeto

Desestructuración de objetos



```
const user = {  
    name: "Manz",  
    role: "streamer",  
    life: 99  
}  
  
const { name, role, life } = user;  
  
console.log(name);  
console.log(role, life);
```

Objeto

Reestructurando nuevos objetos

- □ ×

```
let persona = {  
    nombre: "Juan",  
    edad: 30,  
    ocupacion: "programador"  
};
```

// Desestructuración para crear nuevas variables a partir del objeto

```
let { nombre, edad, ocupacion } = persona;  
console.log(nombre); // "Juan"  
console.log(edad); // 30  
console.log(ocupacion); // "programador"
```

Objeto

Haciendo copias de objetos

```
const user = {  
    name: "Manz",  
    role: "streamer",  
    life: 99,  
    features: ["learn", "code", "paint"]  
}  
  
const fullUser = {  
    ... user,  
    power: 25,  
    life: 50  
}
```

Objeto

Haciendo copias de objetos

- □ ×

```
console.log(user.features);      // ["learn", "code", "paint"]
console.log(fullUser.features);  // ["learn", "code", "paint"]

fullUser.features[0] = "program";
console.log(fullUser.features);  // ["program", "code", "paint"]
console.log(user.features);     // ["program", "code", "paint"]
```

Objeto

Haciendo copias de objetos

```
const user = {  
    name: "Manz",  
    role: "streamer",  
    life: 99,  
    features: ["learn", "code", "paint"]  
}  
  
const fullUser = {  
    ... structuredClone(user),  
    power: 25,  
    life: 50  
}
```

Objeto

Haciendo copias de objetos

- □ ×

```
console.log(user.features);      // ["learn", "code", "paint"]
console.log(fullUser.features);  // ["learn", "code", "paint"]

fullUser.features[0] = "program";
console.log(fullUser.features);  // ["program", "code", "paint"]
console.log(user.features);     // ["learn", "code", "paint"]
```

Objeto

Estructuras anidadas

```
const user = {  
    name: "Manz",  
    role: "streamer",  
    attributes: {  
        height: 183,  
        favColor: "blueviolet",  
        hairColor: "black"  
    }  
}
```

Objeto

Estructuras anidadas extrayendo con attributes

- □ ×

```
// Extraemos propiedad attributes (objeto con 3 propiedades)
const { attributes } = user;
console.log(attributes);

// Extraemos propiedad height (number)
const { attributes: { height } } = user;
console.log(height);
// Extraemos propiedad height (number) y la cambiamos por nombre size
const { attributes: { height: size } } = user;
console.log(size);
```

Objeto

Desestructuración (rest)

```
const user = {  
  name: "Manz",  
  role: "streamer",  
  life: 99  
}
```

```
const { name, ...rest } = user;
```

Objeto

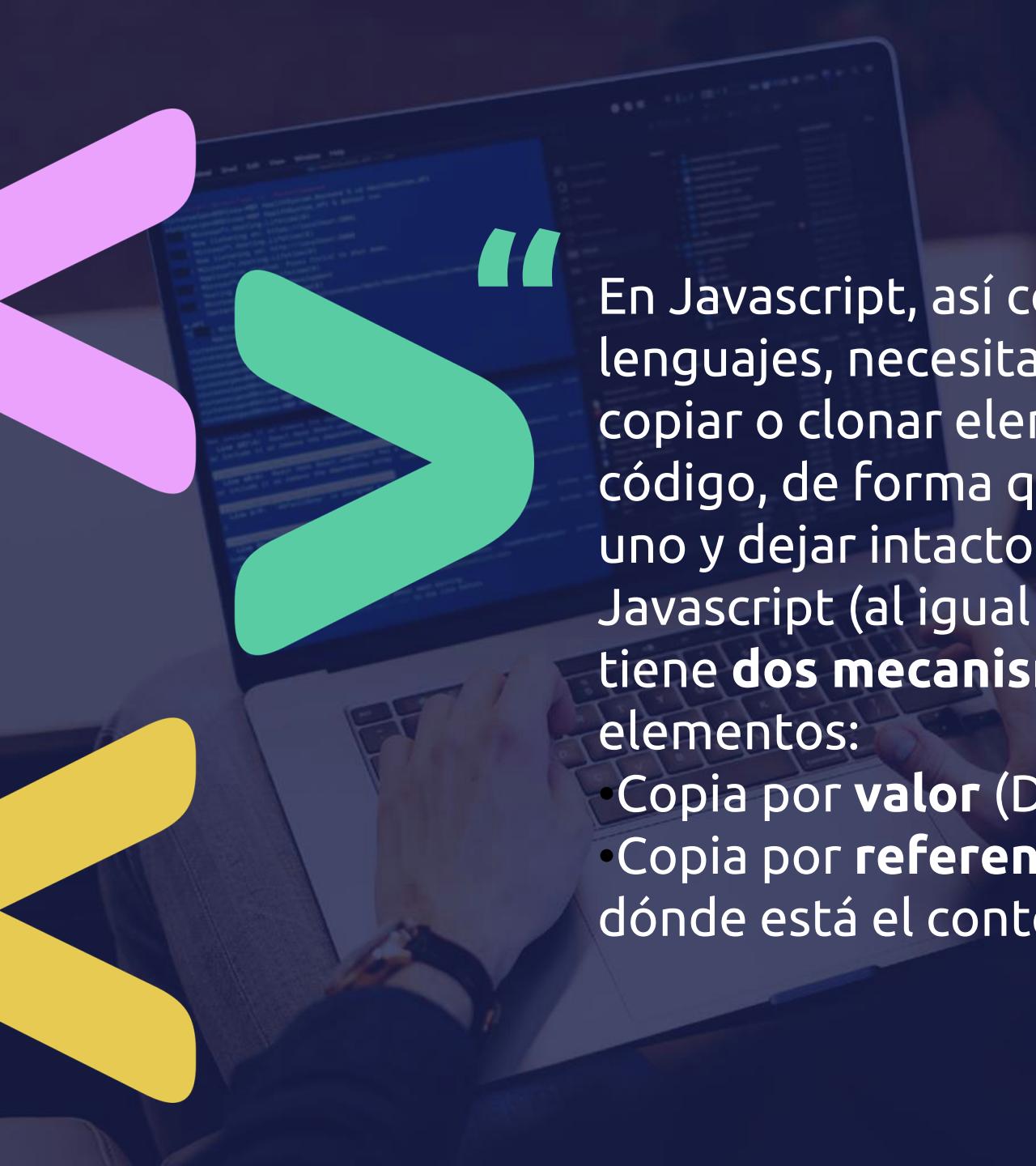
Parámetros desestructurados

La desestructuración de parámetros es una característica de ECMAScript 6 (ES6) que permite extraer valores de un objeto o matriz directamente en los parámetros de una función. Esto puede hacer que tu código sea más limpio y conciso.

```
// Objeto que contiene información sobre una persona
let persona = {
    nombre: "Juan",
    edad: 30,
    ocupacion: "programador"
};

// Función que acepta un objeto y utiliza desestructuración de parámetros
function mostrarInformacion({ nombre, edad, ocupacion }) {
    console.log(`Nombre: ${nombre}`);
    console.log(`Edad: ${edad}`);    console.log(`Ocupación:
${ocupacion}`);
}

// Llamada a la función pasando el objeto 'persona'
mostrarInformacion(persona);
```



“En Javascript, así como en muchos otros lenguajes, necesitaremos en ocasiones copiar o clonar elementos de nuestro código, de forma que podamos cambiar uno y dejar intacto el original. Para ello, Javascript (al igual que en otros lenguajes) tiene **dos mecanismos** para copiar elementos:

- Copia por **valor** (Duplica el contenido)
- Copia por **referencia** (Hace referencia a dónde está el contenido)

”

Objeto

Copia por valor

- □ ×

```
let originalValue = 42;

// Creamos una copia del valor de originalValue
let copy = originalValue;

originalValue;      // 42
copy;               // 42

// Alteramos el valor de copy
copy = 55;

originalValue;      // 42
copy;               // 55
```

Objeto

Copia por valor

- □ ×

```
let originalValue = 42;

// Creamos una copia del valor de originalValue
let copy = originalValue;

originalValue;      // 42
copy;               // 42

// Alteramos el valor de copy
copy = 55;

originalValue;      // 42
copy;               // 55
```

Objeto

Clonado de elementos en Javascript

Estrategia	Clonación superficial	Clonación profunda	Tipos de datos avanzados	Nativo	Más info
Asignación =	✗ No	✗ No	✗ No	✓ Sí	
Usar Object.assign()	✓ Sí	✗ No	✗ No	✓ Sí	
Usar spread ...	✓ Sí	✗ No	✗ No	✓ Sí	Copias con spread
Serializar con JSON.parse()	✓ Sí	✓ Sí	⚠ Solo tipos básicos ⚠ No funciones/DOM	✓ Sí	JSON
Usar <code>_.cloneDeep()</code> de Lodash	✓ Sí	✓ Sí	✓ Tipos avanzados ⚠ No DOM	✗ No	cloneDeep
Usar <code>structuredClone()</code>	✓ Sí	✓ Sí	✓ Tipos avanzados ⚠ No funciones/DOM	✓ Sí	

Objeto

Iteradores de objetos

Método	Descripción
ARRAY Object.keys(obj)	Itera el obj y devuelve sus propiedades o keys .
ARRAY Object.values(obj)	Itera el obj y devuelve los valores de sus propiedades.
ARRAY Object.entries(obj)	Itera el obj y devuelve un array con los pares [key, valor].
OBJECT Object.fromEntries(array)	Construye un objeto con un array de pares [key, valor].

Objeto

Convertir un objeto

- □ ×

```
const user = {  
    name: "Manz",  
    life: 99,  
    power: 10,  
    talk: function() {  
        return "Hola!";  
    }  
};  
  
Object.keys(user);      // ["name", "life", "power", "talk"]  
Object.values(user);    // ["Manz", 99, 10, f]  
Object.entries(user);   // [[["name", "Manz"], ["life", 99], ["power", 10],  
  ["talk", f]]]
```

Objeto

Convertir un objeto

Con el método `Object.keys()` obtenemos un ARRAY de las claves (propiedades, índices, keys) del objeto.

Con el método `Object.values()` obtenemos un ARRAY de los valores de las claves anteriores, en el mismo orden.

Con el método `Object.entries()` obtenemos un ARRAY de entradas. Cada entrada es un ARRAY del par clave-valor, es decir, la propiedad del objeto original y su valor correspondiente.

Objeto

Convertir un array a objeto

```
const keys = ["name", "life", "power", "talk"];
const values = ["Manz", 99, 10, function() { return "Hola" }];

const entries = [];
for (let i of Object.keys(keys)) {
  const key = keys[i];
  const value = values[i];
  entries.push([key, value]);
}

const user = Object.fromEntries(entries);      // {name: 'Manz', life: 99,
power: 10, talk: f}
```

</Be a
coder>