

# Rompiendo referencias en JavaScript: Copy y Deep copy

## Comparativa

Tipo de copia	Método	Profundidad	Notas rápidas
Shallow copy	<code>Object.assign</code> , <code>{...obj}</code>	1 nivel	Referencias internas compartidas
Deep copy (JSON)	<code>JSON.parse(JSON.stringify(obj))</code>	Total (limitada)	Pierde funciones y fechas
Deep copy (nativo)	<code>structuredClone(obj)</code>	Total	Más fiable, moderno
Deep copy (librería)	<code>lodash.cloneDeep(obj)</code>	Total	Más robusto para todo tipo de estructuras

## Ejemplo con arrays anidados

```
const original = [[1, 2], [3, 4]];
const copiaShallow = [...original];

copiaShallow[0][0] = 99;

console.log("original:", original); // [[99, 2], [3, 4]] ❌
console.log("copiaShallow:", copiaShallow); // [[99, 2], [3, 4]]
```

## Explicación:

- `[...original]` copia **el array exterior**, pero **no los subarrays internos**.
- `copiaShallow[0]` y `original[0]` apuntan a la **misma dirección de memoria**.
- Al modificar uno, se cambian ambos.

## Ejemplo con objeto complejo

```
const persona = {
  nombre: "Carlos",
  direccion: {
    ciudad: "Madrid",
    coordenadas: { lat: 40.4, lng: -3.7 }
  }
};

const copiaShallow = { ...persona };

copiaShallow.direccion.ciudad = "Sevilla";

console.log("persona:", persona.direccion.ciudad); // "Sevilla" ❌
```

### Explicación:

- `{ ...persona }` hace una **copia superficial**.
- `direccion` sigue siendo una **referencia compartida**.

## Deep copy con JSON.parse(JSON.stringify(...))

```
const persona = {
  nombre: "Carlos",
  direccion: {
    ciudad: "Madrid",
    coordenadas: { lat: 40.4, lng: -3.7 }
  }
};

const copiaProfunda = JSON.parse(JSON.stringify(persona));

copiaProfunda.direccion.ciudad = "Valencia";

console.log("persona:", persona.direccion.ciudad);      // "Madrid" ✓
console.log("copiaProfunda:", copiaProfunda.direccion.ciudad); // "Valencia"
```

### Explicación:

- **Rompe todas las referencias internas.**
- **No funciona** si el objeto tiene funciones, fechas, undefined, Map, Set, etc.



## Deep copy con structuredClone

```
const persona = {
  nombre: "Carlos",
  direccion: {
    ciudad: "Madrid",
    coordenadas: { lat: 40.4, lng: -3.7 }
  },
  fechaNacimiento: new Date("1990-01-01")
};

const copiaProfunda = structuredClone(persona);

copiaProfunda.direccion.ciudad = "Zaragoza";
copiaProfunda.fechaNacimiento.setFullYear(2000);

console.log("original:", persona.fechaNacimiento.getFullYear()); // 1990 ✓
console.log("copia:", copiaProfunda.fechaNacimiento.getFullYear()); // 2000
```



### Explicación:

- structuredClone es una función nativa moderna que maneja:
  - Fechas ✓
  - Arrays ✓
  - Objetos anidados ✓
  - Map, Set, Blob...



# Deep copy con lodash.cloneDeep

## 1) Instalación lodash

```
npm install lodash
```

## 2) Utilizando \_.cloneDeep

```
// Importar cloneDeep desde lodash
const _ = require('lodash');




const original = {
  nombre: "Carlos",
  direccion: {
    ciudad: "Madrid",
    coordenadas: { lat: 40.4, lng: -3.7 }
  },
  hobbies: ["leer", "nadar"],
  fechaNacimiento: new Date("1990-01-01"),
  saludo: function () {
    console.log("Hola");
  }
};

const copiaProfunda = _.cloneDeep(original);

// Cambios en la copia
copiaProfunda.direccion.ciudad = "Barcelona";
copiaProfunda.hobbies[0] = "correr";
copiaProfunda.fechaNacimiento.setFullYear(2000);

console.log("original:", original);
console.log("copiaProfunda:", copiaProfunda);
```

## ¿Qué hace `_.cloneDeep`?

- Copia **recursivamente** todo el contenido.
- Separa completamente referencias internas.
- Mantiene:
  - Funciones 
  - Fechas 
  - Arrays anidados 
  - Objetos anidados 