Guía completa del operador Spread en JavaScript

Esta guía modular cubre todos los usos modernos del operador Spread, con ejemplos claros, advertencias y buenas prácticas. Está organizada en archivos independientes para facilitar el estudio y la consulta.

📌 Introducción

El operador Spread (`...`) permite expandir elementos de arrays, objetos o iterables en lugares donde se esperan múltiples elementos o propiedades. Es una herramienta clave para escribir código limpio, inmutable y expresivo.

📦 Índice navegable

Haz clic en cualquier sección para ir directamente:

🧠 Buenas prácticas

- Usa Spread para mantener la inmutabilidad
- Evita mutar objetos originales
- Prefiere Spread sobre `Object.assign()` por legibilidad
- Para estructuras profundas, usa `_.cloneDeep()` o `structuredClone()`

⚠️ Advertencias comunes

- No puedes usar Spread en valores no iterables (ej. `...123`)
- No copia métodos ni prototipos (ej. `new Date()`)

- No confundas Rest (`...args` en definiciones) con Spread (`...args` en llamadas)

☑ Secciones detalladas

Uso básico

Archivo: `01-uso_basico.js`
Expande arrays y objetos en estructuras nuevas sin mutar los originales.

Convertir estructuras parecidas a arrays

Archivo: `02-convertir_estructuras_parecidas_a_array_en_array.js`
Convierte `arguments`, `NodeList`, `Set`, etc. en arrays reales usando Spread.

Clonación de arrays y objetos

Archivo: `03-clonacion_de_arrays_y_objetos.js`
Copia superficial de arrays y objetos para mantener inmutabilidad.

Spread en funciones

Archivo: `04-spread_en_funciones.js`
Expande arrays como argumentos individuales en llamadas a funciones como `Math.max`.

Diferencia entre Rest y Spread

Archivo: `05-diferencia_entre_rest_y_spread.js`
Rest captura argumentos, Spread los expande. Se usan en contextos opuestos.

Desestructuración inversa

Archivo: `06-destructuracion_inversa.js`
Extrae propiedades específicas y agrupa el resto con Spread.

Limitaciones del Spread

Archivo: `07-limitaciones_del_spread.js`
No clona métodos, prototipos ni estructuras profundas. Solo copia superficial.

Agregar elementos en un array

Archivo: `08-agregar_elementos_en_un_array.js`
Construye nuevos arrays añadiendo elementos sin mutar el original.

Agregar propiedades a un objeto

Archivo: `09-agregar_propiedades_a_un_objeto.js`
Crea objetos nuevos con propiedades adicionales usando Spread.

Fusionar arrays

Archivo: `10-fusionar_arrays.js`
Combina múltiples arrays en uno nuevo, con posibilidad de intercalar elementos.

Fusionar objetos

Archivo: `11-fusionar_objetos.js`
Combina objetos y sobrescribe propiedades según el orden. Comparativa con `Object.assign()` incluida.

Estructuras anidadas

Archivo: `12-estructuras_anidadas.js`
Clonación parcial de objetos anidados con Spread. Requiere cuidado con la profundidad.

Estructuras anidadas con Lodash

Archivo: `13-estructuras_anidadas_con_libreria_lodash.js`
Uso de `_.cloneDeep`, `_.merge`, `_.get` para manipular estructuras complejas.

Propiedades condicionales

Archivo: `14-propiedades_condicionales.js`
Construcción dinámica de objetos usando ternarios, cortocircuito lógico y Spread condicional.

Advertencias comunes

Archivo: `15-advertencias_comunes.js`
Errores típicos al usar Spread: valores no iterables, confusión con Rest, objetos especiales.

Resumen global

Archivo: `spread-resumen.js`
Resumen técnico, tabla de usos, ventajas, limitaciones y recomendación final.

📌 Recomendación final

Si estás trabajando con estructuras complejas, considera usar Lodash o `structuredClone()` para clonación profunda. Esta guía está pensada para ayudarte a dominar el operador Spread con ejemplos claros y aplicables.
