perplexity

Shallow Copy vs Deep Copy in JavaScript

Shallow Copy

- **Definition:** A shallow copy duplicates only the top-level properties of an object or array. If those properties are **primitives** (like numbers or strings), the values themselves are copied. However, if properties are objects (like arrays or other objects), only the **reference** is copied—not the actual nested data.
- **Impact:** Any changes made to nested objects/arrays in either the original or the copied object will affect the other, since they both point to the same reference.
- **Examples of Shallow Copy:**
  - `Object.assign({}, obj)`
  - Spread operator: `{ ...obj }` or `[ ...arr ]`

```
const original = { a: 1, b: { c: 2 } };
const shallowCopy = { ...original };
shallowCopy.b.c = 42;
console.log(original.b.c); // Output: 42 (original is changed)
```

- **When to Use:** Suitable for flat objects or arrays when you don't need to protect nested structures from modification.[1] [2] [3]

Deep Copy

- **Definition:** A deep copy creates a completely independent clone, **recursively** copying all levels of nested objects or arrays. Changes made to the deep copy do **not affect** the original object, and vice versa.
- **Impact:** Every layer, including nested objects/arrays, gets a fresh copy, so they have **no shared references** with the original.
- **Examples of Deep Copy:**
  - Using `JSON.parse(JSON.stringify(obj))` (works only for JSON-safe data)
  - `structuredClone(obj)` – a modern, native deep copy method (browser/runtime support needed)
  - Libraries like Lodash: `_.cloneDeep(obj)`

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

```
deepCopy.b.c = 99;
console.log(original.b.c); // Output: 2 (original remains unchanged)
```

- **When to Use:** Recommended when your object or array has nested structures and you need full independence between copies.[4] [2] [1]

Comparison Table

	Shallow Copy	Deep Copy
Copies levels	Only top-level	All levels (recursive)
Reference	Nested refs are shared	No shared references
Methods	`Object.assign`, spread	`JSON.parse(JSON.stringify)`, `structuredClone`, Lodash
Performance	Faster, less memory	Slower, more memory intensive
Use case	Flat/simple structures	Nested/complex structures

Key Takeaways

- **Shallow copy** duplicates only the surface (top-level), so nested objects/arrays remain linked. Use it for simple, flat data.
- **Deep copy** duplicates everything, so the original and copy are fully independent. Use it when you need to modify nested data safely.[2] [1] [4]

1. https://www.geeksforgeeks.org/javascript/what-is-shallow-copy-and-deep-copy-in-javascript/
2. https://www.greatfrontend.com/questions/quiz/explain-the-difference-between-shallow-copy-and-deep-copy
3. https://mayallo.com/deep-copy-vs-shallow-copy-in-javascript/
4. https://dev.to/hkp22/javascript-shallow-copy-vs-deep-copy-examples-and-best-practices-3k0a