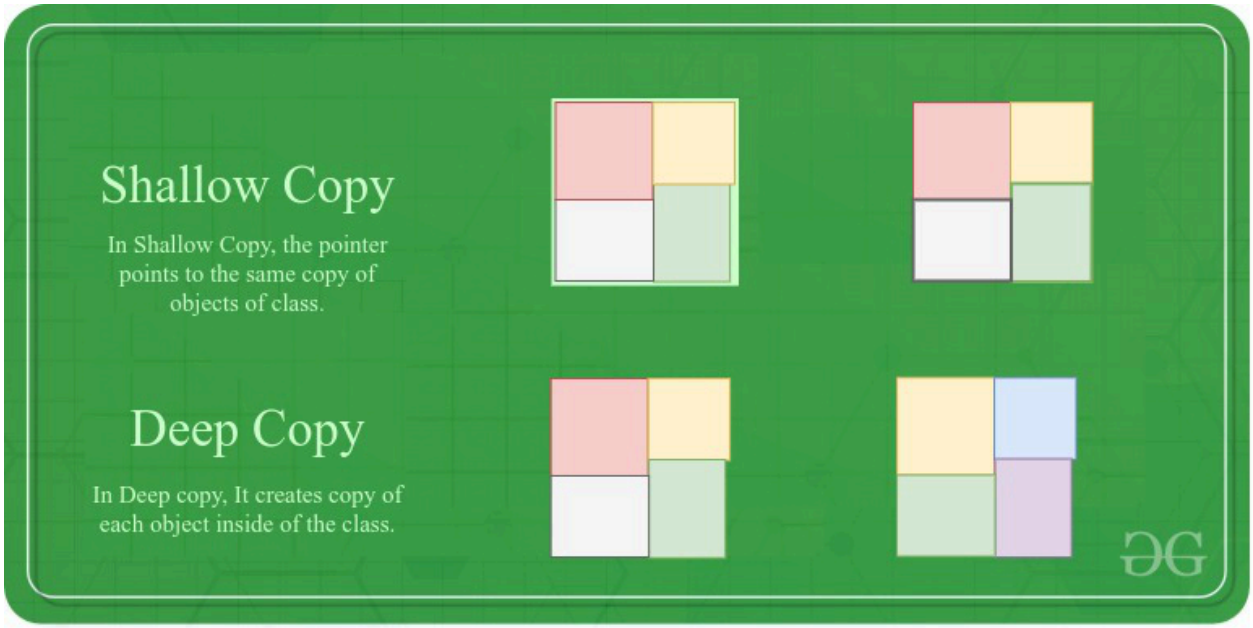
Shallow vs. Deep Copying in JavaScript



JavaScript is a high-level, dynamically typed client-side scripting language. JavaScript adds functionality to static HTML pages. Like most other programming languages JavaScript allows supports the concept of deep copy and shallow copy.

Shallow Copy

Shallow copy method creates a copy where the source and the copied variable reference remain the same. This

means that modifications made in one place would affect both places.

Deep Copy

Deep copy method creates a copy where the source and the copied variable reference are completely different. This means that modifications made in one place would only affect the variable where we are making a change.
In JavaScript, the concepts of shallow copy and deep copy are related to the duplication of objects, arrays, or other complex data structures. Understanding the difference between these two types of copies is crucial for managing data effectively and preventing unintended side effects in your code.

Shallow Copy:

- A shallow copy creates a new object or array and copies the elements of the original one to the new one. However, it does not create copies of nested objects or arrays; instead, it copies references to them.
- Shallow copying is performed using methods like `Object.assign()`, the `spread operator (...)`, or array methods like `slice()` or `concat()`.

- Shallow copies are faster and consume less memory since they only copy references. However, they may lead to unexpected behavior if the original and copied objects share references to nested objects.

Deep Copy:

- A deep copy, on the other hand, creates a completely independent copy of the original object or array, including all nested objects and arrays.
- Achieving a deep copy can be more complex and usually requires custom implementation, such as using recursion to traverse the entire structure and create new instances for each nested element.
- Deep copies provide true isolation between the original and copied data, ensuring that changes to one do not affect the other. However, they can be slower and consume more memory, especially for large and complex data structures.

Pros and Cons

Shallow Copy
Pros: Lightweight, maintains references, suitable for simple structures.

Cons: Changes to nested structures affect both original and copy, may lead to unexpected behavior."
Deep Copy
Pros: Independent duplicates, changes do not affect the original, suitable for complex structures.
Cons: May be computationally expensive, potential issues with circular references."

Best Practices

- Use shallow copy when dealing with simple structures and you want to maintain lightweight references.
- Opt for deep copy when working with complex structures or when you need completely independent duplicates.