Shallow and Deep Copy in JavaScript

In JavaScript, copying objects or arrays can be done in two primary ways: shallow copy and deep copy. Understanding the difference between them is crucial for preventing unintended side effects in your code.

# Shallow Copy

A shallow copy of an object or array is a new object or array that has the same values as the original. However, if the original object or array contains references to other objects or arrays, only the references are copied, not the actual objects. This means that changes to nested objects or arrays in the shallow copy will affect the original.

Methods to create a shallow copy include:

- Using the spread operator (`...`):  
 ```javascript  
 const original = { a: 1, b: { c: 2 } };  
 const shallowCopy = { ...original };  
 ```

- Using `Object.assign()`:  
 ```javascript  
 const original = { a: 1, b: { c: 2 } };  
 const shallowCopy = Object.assign({}, original);  
 ```

In both examples, the `shallowCopy` is a new object that has the same properties as `original`, but the nested object `b` is still shared between the two.

# Deep Copy

A deep copy, on the other hand, involves copying all elements and nested objects or arrays such that the new object or array is completely independent of the original. Changes made to the deep copy will not affect the original and vice versa.

Methods to create a deep copy include:

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

- Using structured cloning (for modern browsers):  
 ```javascript  
 const original = { a: 1, b: { c: 2 } };  
 const deepCopy = structuredClone(original);  
 ```

While `JSON.parse(JSON.stringify())` is widely used, it has limitations, such as not copying functions, undefined properties, and handling certain special objects like Dates. `structuredClone`, introduced in more recent versions of JavaScript, handles these cases better.

# Summary

To summarize, a shallow copy is faster but might lead to unexpected results if the object contains references to other objects or arrays. A deep copy, while more resource-intensive, ensures that all elements are fully copied and independent of the original.