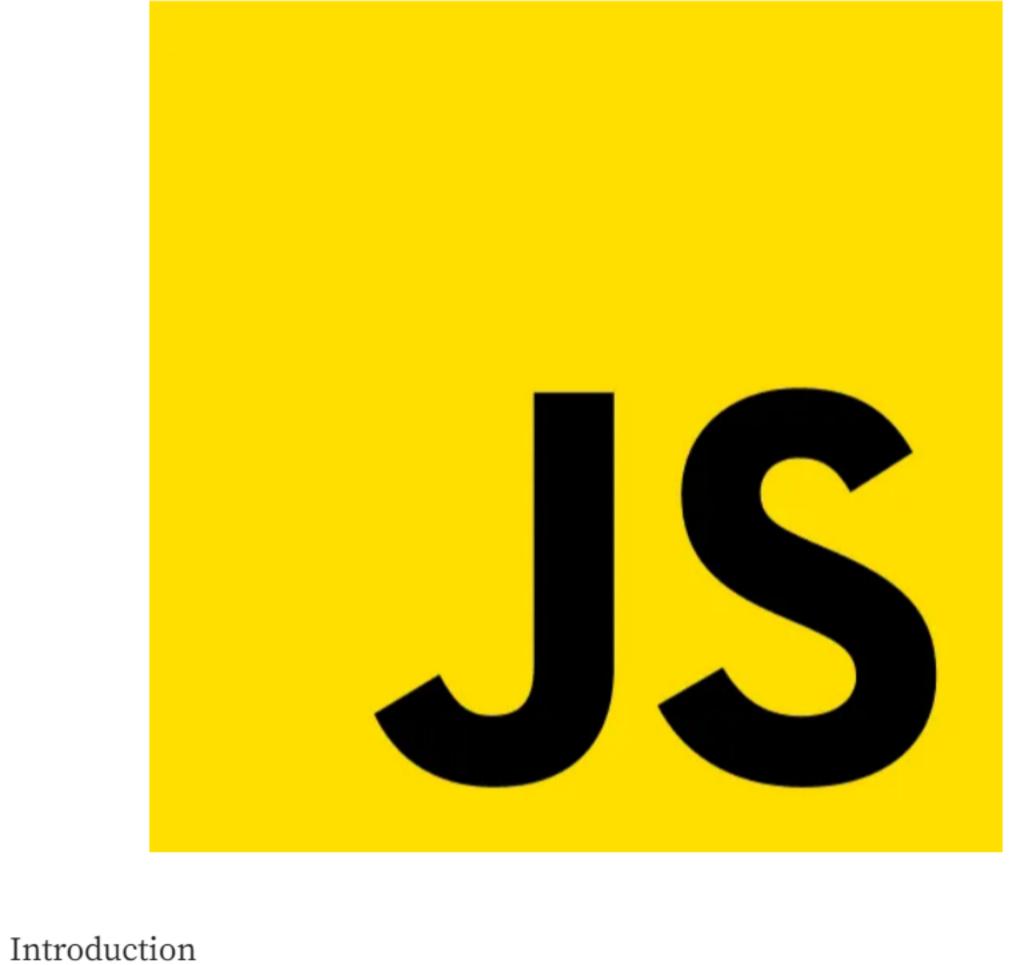


129 Q 1

When working with JavaScript objects and arrays, understanding the concepts of shallow copy and deep copy is crucial. In this article, we'll explore the key differences between these two copy methods and discuss how to achieve them using various techniques, including Lodash's cloneDeep, the structuredClone method, JSON methods (JSON.stringify and JSON.parse), and Object.assign.



### In JavaScript, objects and arrays are reference types, which means that copying them is not as straightforward as it may seem. Let's begin by

understanding the difference between shallow copy and deep copy. Shallow Copy vs. Deep Copy

Shallow Copy

A shallow copy creates a new object or array and copies references to the elements from the original object or array. This means that if the elements

objects.

const originalArray = [1, 2, [3, 4]]; const shallowCopy = Object.assign({}, originalArray); shallowCopy[2][0] = 99;console.log(originalArray); // [1, 2, [99, 4]]

themselves are objects or arrays, the copy still references those nested

```
Deep Copy
A deep copy creates a completely new object or array, including copies of all
```

## nested elements. Changes made to the deep copy won't affect the original object.

const originalArray = [1, 2, [3, 4]]; const deepCopy = JSON.parse(JSON.stringify(originalArray)); deepCopy[2][0] = 99;console.log(originalArray); // [1, 2, [3, 4]]

```
Copying with Lodash's cloneDeep
Lodash provides a cloneDeep method that easily creates a deep copy of an
object or array. It's a reliable choice for deep copying complex data
structures.
```

#### const deepCopy = \_.cloneDeep(originalArray); deepCopy[2][0] = 99;console.log(originalArray); // [1, 2, [3, 4]]

const \_ = require('lodash');

const originalArray = [1, 2, [3, 4]];

Using structuredClone The structuredClone method, available in modern browsers, is designed for deep cloning structured data, including objects and arrays. It's particularly

useful for cloning objects with non-JSON-safe values like functions.

```
const deepCopy = structuredClone(originalObject);
    deepCopy.name = 'Alice';
    deepCopy.sayHello(); // Outputs "Hello!" even for the deep copy
JSON.stringify and JSON.parse
```

For cases where the data is JSON-safe (does not include functions or non-

JSON data types), you can use JSON.stringify to serialize the object and then

const originalObject = { name: 'John', sayHello: function() { console.log('Hello

## JSON.parse to deserialize it, effectively creating a deep copy:

console.log(originalObject.name); // John

const originalObject = { name: 'John', age: 30 }; const deepCopy = JSON.parse(JSON.stringify(originalObject)); deepCopy.name = 'Alice';

```
Using Object.assign
Object.assign can create a shallow copy of an object. It's simple and effective
for creating a copy with one level of depth.
```

## const originalObject = { name: 'John', age: 30 };

const shallowCopy = Object.assign({}, originalObject); shallowCopy.name = 'Alice'; console.log(originalObject.name); // John

```
Conclusion
Understanding the distinction between shallow copy and deep copy in
JavaScript is crucial when dealing with complex data structures. Shallow
copies merely clone references, while deep copies create entirely
```

independent copies. Techniques like Lodash's cloneDeep, structuredClone,

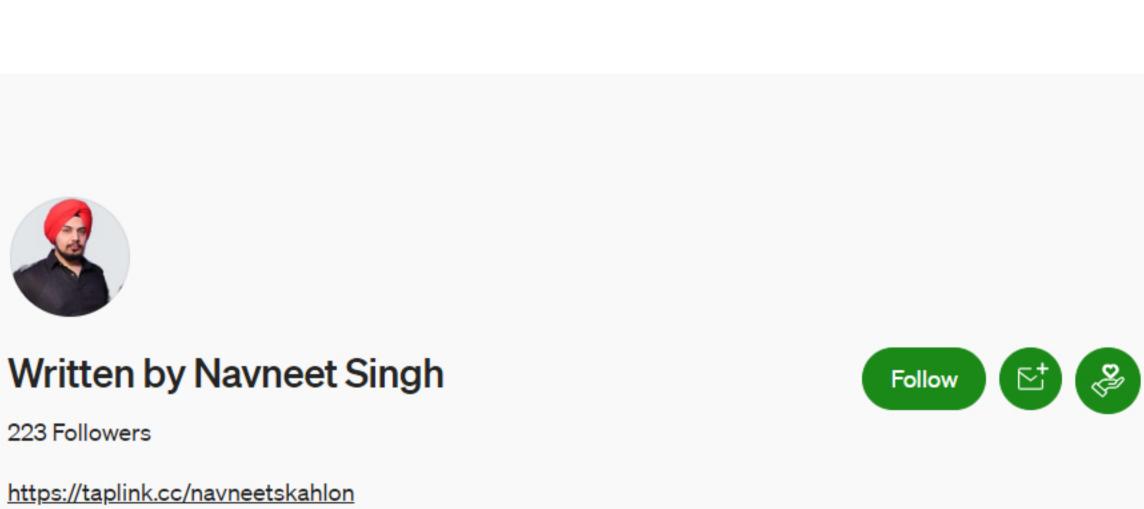
JSON methods (JSON.stringify and JSON.parse), and Object.assign offer

and limitations. Depending on your specific use case, choose the method

various ways to achieve deep and shallow copies, each with its own strengths

that best suits your needs to ensure proper data copying and manipulation in

your JavaScript projects. Buy me a coffee JavaScript Shallow Copy Deep Copy Objects



Navneet Singh

More from Navneet Singh

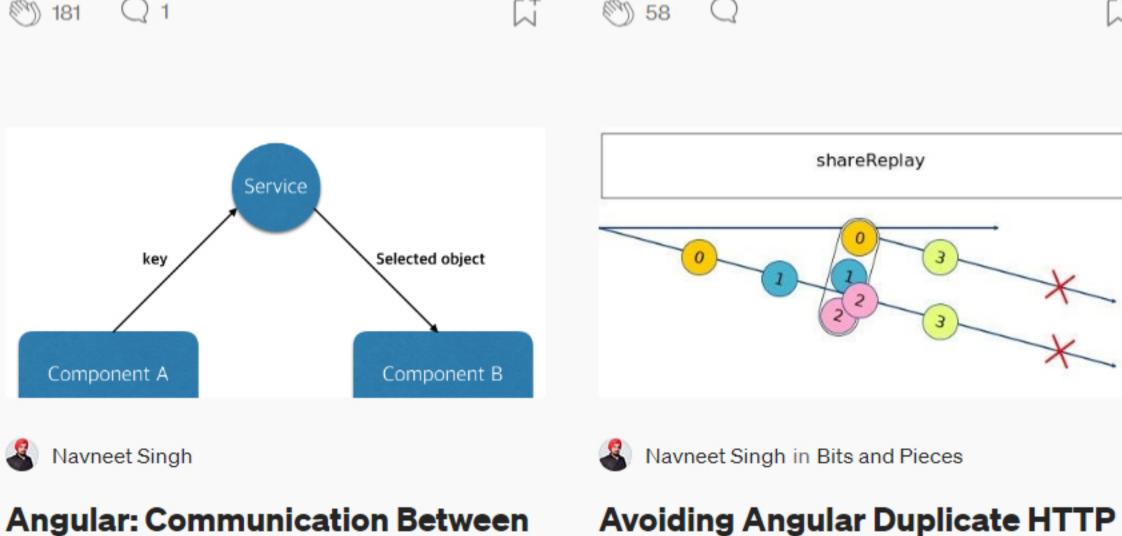
129 Q 1

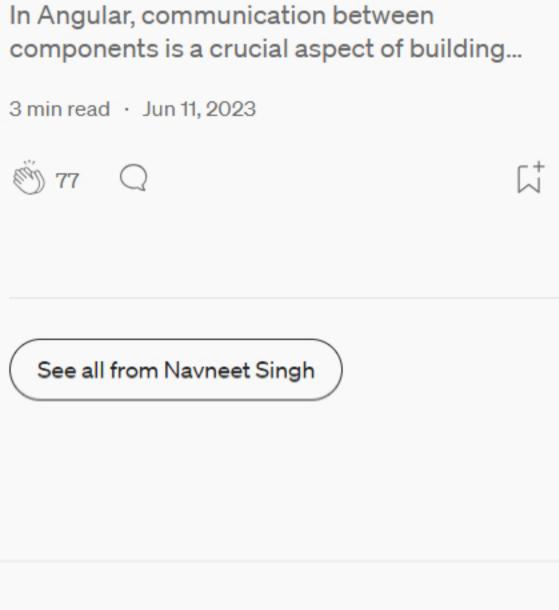
**CORS Errors Using proxy.conf.json** Cross-Origin Resource Sharing (CORS) errors can be a common challenge when developin... 3 min read · Jun 14, 2023 181 Q 1

**Angular Proxy: How to Bypass** 

proxy

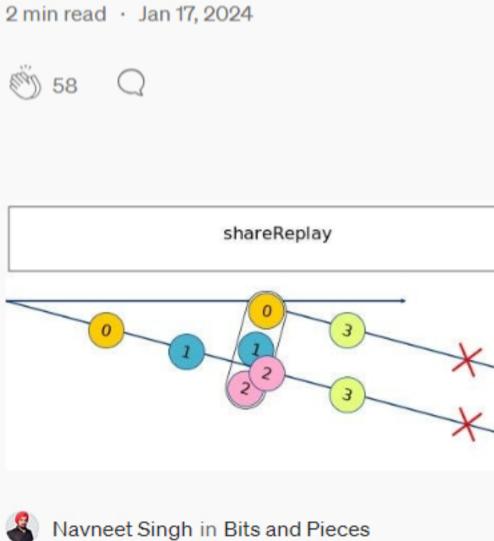






Recommended from Medium

Components using a Shared...



Navneet Singh

A Step-by-Step Guide to

**Upgrading Your Angular App to...** 

Angular 17, the latest iteration of the widely-

used web development framework, brings a...

₩,



Requests with the RxJS...

Explore how the shareReplay operator can

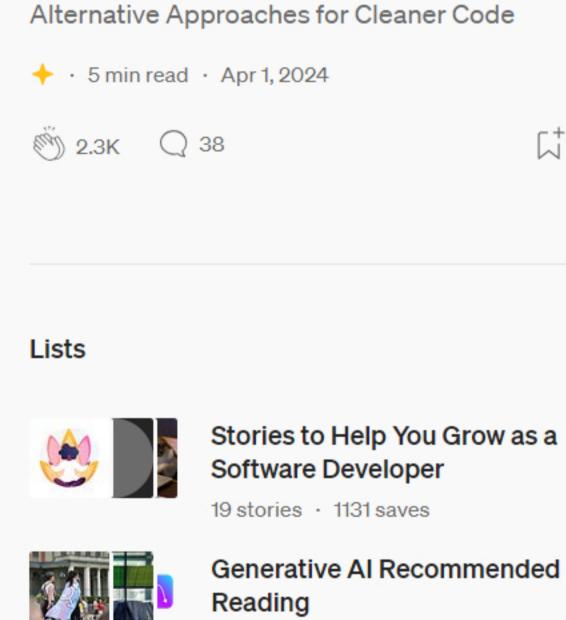
help us avoid duplicate HTTP requests in...

# Enes Talay in CodeX

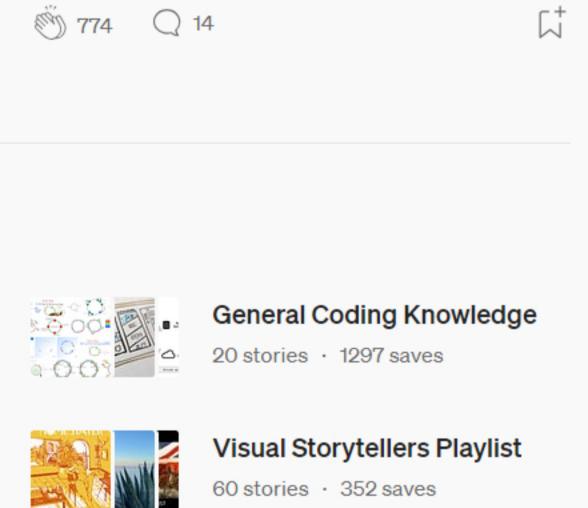
JavaScript

Stop Using find() Method in

Forget the find() Method in JavaScript:



52 stories · 1130 saves



JavaScript

Awwwesssoooome in JavaScript in Plain English

A JavaScript Interview Question

That 90% of People Get Wrong

Let's take a look at the question first:

→ 2 min read · Mar 18, 2024



907







New JavaScript features



See more recommendations

Help Status About Careers Press Blog Privacy Terms Text to speech Teams

9 min read · Jan 18, 2024

552 Q 5