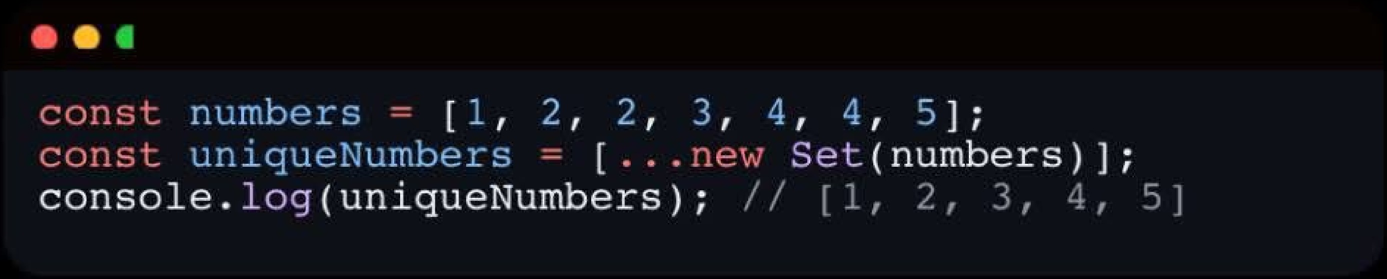
Remove Duplicates from a JavaScript Array

SWIPE FOR MORE

Why Duplicates?

When working with arrays in JavaScript, you'll often run into duplicate values. Removing them efficiently not only keeps your data clean but also makes your code more optimized. Let's look at the most common ways to do it.

1. Using Set (ES6+)



```javascript
const numbers = [1, 2, 2, 3, 4, 4, 5];
const uniqueNumbers = [...new Set(numbers)];
console.log(uniqueNumbers); // [1, 2, 3, 4, 5]
```

✅ Simple and efficient

✅ Best for modern JavaScript

2. Using filter() + indexOf()

```javascript
const numbers = [1, 2, 2, 3, 4, 4, 5];
const uniqueNumbers = numbers.filter((value, index) =>
  numbers.indexOf(value) === index
);
console.log(uniqueNumbers); // [1, 2, 3, 4, 5]
```

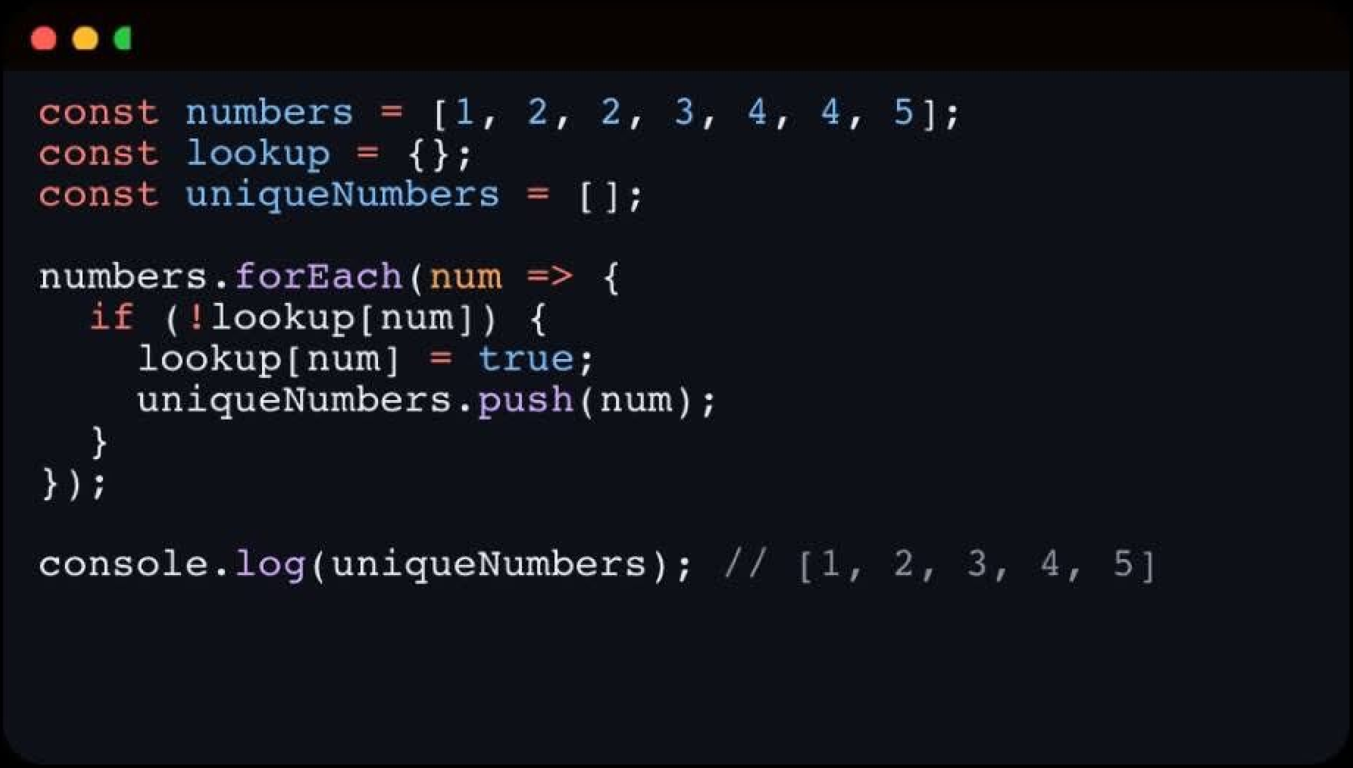
✅ Works everywhere
⚠️ Slightly slower on large arrays

3. Using reduce() + includes()

```javascript
const numbers = [1, 2, 2, 3, 4, 4, 5];
const uniqueNumbers = numbers.reduce((acc, value) => {
  if (!acc.includes(value)) acc.push(value);
  return acc;
}, []);
console.log(uniqueNumbers); // [1, 2, 3, 4, 5]
```

✅ Great for learning array methods

4. Using forEach() + Object Lookup



```
const numbers = [1, 2, 2, 3, 4, 4, 5];
const lookup = {};
const uniqueNumbers = [];

numbers.forEach(num => {
  if (!lookup[num]) {
    lookup[num] = true;
    uniqueNumbers.push(num);
  }
});

console.log(uniqueNumbers); // [1, 2, 3, 4, 5]
```

✅ Very fast for larger datasets

What You Should Know

The *Set* method is the easiest and most modern solution, but depending on the scenario (browser support, performance needs), other approaches can still be useful.

Summary

Removing duplicates in JavaScript arrays is a common task with many solutions. The most modern and efficient method is using *Set*, while traditional methods like *filter()* + *indexOf()*, *reduce()* + *includes()*, or a lookup object offer alternatives depending on performance needs and project requirements.

The key takeaway: JavaScript gives you multiple tools to keep your data clean—pick the one that best fits your use case.