“What is an array?”. An array is a special variable, which can hold more than one value at a time. While there are many ways to work with an array, there are two groups we can split them into destructive and non-destructive.

The main difference between these two groups can be summarized by the fact that one is destructive in the manner that it returns a modified array. While the other being non-destructive returns us a new or non-modified array.

"Destructive" and "non-destructive" are terms often used to describe operations on data structures, including arrays. These terms refer to whether an operation modifies the original data structure (destructive) or returns a new data structure without modifying the original (non-destructive). Here's an explanation of each:

Destructive Array Operation:

A destructive array operation is one that modifies the original array.

When you perform a destructive operation on an array, the changes are made directly to the array in memory, and the original array is typically altered or "destroyed" in the process.

Examples of destructive array operations include methods like **push**, **pop**, **splice**, and **sort**, among others.

Here's an example in JavaScript:

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

numbers.pop(); // Destructive operation, modifies the 'numbers' array by removing the last element.

console.log(numbers); // Output: [1, 2, 3]

Non-Destructive Array Operation:

A non-destructive array operation is one that does not modify the original array.

Instead of changing the original array, these operations typically return a new array with the desired changes, leaving the original array intact.

Examples of non-destructive array operations include methods like **concat**, **slice**, **map**, and **filter**, among others.

Here's an example in JavaScript:

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

const newNumbers = numbers.slice(0, 2); // Non-destructive operation, creates a new array 'newNumbers' without modifying 'numbers'.

console.log(numbers); // Output: [1, 2, 3, 4]

console.log(newNumbers); // Output: [1, 2]

whether an operation on an array is destructive or non-destructive depends on whether it modifies the original array or returns a new array without altering the original data. It's important to understand the behavior of these operations when working with arrays, as it can impact the correctness and predictability of your code.