**1. What is an array?**

An array is a data structure that stores a collection of elements, typically of the same type, in a contiguous block of memory. Each element in the array can be accessed by its index, which is a numerical representation of its position in the array.

**2. What types of values can be held in an array?**

In JavaScript, an array can hold values of any type, including numbers, strings, objects, arrays, booleans, null, undefined, functions, and symbols.

**3. What happens if you try to contain values of different data types in an array?**

JavaScript arrays are heterogeneous, meaning they can contain values of different data types. There are no issues or errors when mixing data types within the same array.

javascript

Copy code

let mixedArray = [42, "hello", { key: "value" }, [1, 2, 3], true];

**4. What is the index of the first element in an array?**

The index of the first element in an array is 0.

**5. What is the index of the last element in an array?**

The index of the last element in an array is array.length - 1.

**6. If you control an array, what is the easiest way to determine the number of values?**

The easiest way to determine the number of values in an array is to use the length property.

javascript

Copy code

let array = [1, 2, 3];

console.log(array.length); // 3

**7. When would the above method not work well, and what must you do in that case?**

The length property may not work well if the array has been manipulated in a way that leaves "holes" (i.e., undefined elements) or if elements are deleted without properly adjusting the length. To handle such cases, you might need to iterate through the array and count the actual elements.

javascript

Copy code

let array = [];

array[5] = 'hello';

console.log(array.length); // 6

// Counting actual elements

let count = array.reduce((acc, val) => acc + (val !== undefined ? 1 : 0), 0);

console.log(count); // 1

**8. What are two ways to make an array one element longer?**

1. Using the push method:

javascript

Copy code

let array = [1, 2, 3];

array.push(4);

console.log(array); // [1, 2, 3, 4]

1. Directly setting the value at length index:

javascript

Copy code

let array = [1, 2, 3];

array[array.length] = 4;

console.log(array); // [1, 2, 3, 4]

**9. What are two ways to make an array two elements shorter?**

1. Using the pop method twice:

javascript

Copy code

let array = [1, 2, 3, 4];

array.pop();

array.pop();

console.log(array); // [1, 2]

1. Adjusting the length property:

javascript

Copy code

let array = [1, 2, 3, 4];

array.length -= 2;

console.log(array); // [1, 2]

**10. What is the upper limit on the number of values an array can hold?**

The upper limit on the number of values an array can hold is determined by the maximum safe integer in JavaScript, which is 2^53 - 1 (approximately 9 quadrillion). However, practical limits are much lower and depend on available memory and system constraints.

The number 53 in the context of JavaScript's maximum safe integer comes from the IEEE 754 double-precision floating-point format, which is the standard used by JavaScript for representing numbers.

**IEEE 754 Double-Precision Floating-Point Format**

In this format:

* A number is represented using 64 bits.
* These 64 bits are divided into three parts:
  + 1 bit for the sign (positive or negative).
  + 11 bits for the exponent.
  + 52 bits for the significand (also known as the fraction or mantissa), with an implicit leading bit that effectively makes it 53 bits.

**11. When working with arrays, does working “in-place” really matter? If so, when and why?**

Working “in-place” matters in scenarios where memory efficiency is crucial, such as when dealing with large datasets or in performance-critical applications. In-place operations avoid the overhead of creating and managing additional arrays, thus reducing memory usage and potentially increasing performance.

**12. Which values are “falsy” in JavaScript, and what does “falsy” mean anyway?**

In JavaScript, “falsy” values are values that evaluate to false when converted to a Boolean. The falsy values are:

* false
* 0
* -0
* "" (empty string)
* null
* undefined
* NaN

A falsy value means it will be treated as false in a Boolean context, such as in an if statement.

javascript

Copy code

if (0) {

console.log('This will not run.');

} else {

console.log('0 is falsy.');

}

**13. What does “passing by reference” mean?**

“Passing by reference” means passing a reference to the actual data, rather than a copy of the data. Changes made to the reference will affect the original data. In JavaScript, objects (including arrays) are passed by reference, while primitive types (such as numbers and strings) are passed by value.

javascript

Copy code

let obj = { key: 'value' };

function modify(o) {

o.key = 'new value';

}

modify(obj);

console.log(obj.key); // 'new value'

In the above example, obj is passed by reference to the modify function, so changes to o affect obj.