Array creation and initialization

📚 Array Creation and Initialization in JavaScript — In Detail

Arrays in JavaScript are ordered, dynamic, and versatile collections that store multiple values in a single variable. Let's explore **how to create and initialize** arrays in various ways.

✅ 1. Array Literal Syntax

The most common and preferred way:

```
const fruits = ["apple", "banana", "cherry"];
```

- Creates an array with 3 string elements
- Indexes: `0`, `1`, `2`

```
console.log(fruits[0]); // "apple"
console.log(fruits.length); // 3
```

✅ 2. Using the `Array` Constructor

```
const numbers = new Array(1, 2, 3, 4);
console.log(numbers); // [1, 2, 3, 4]
```

⚠️ Special Case:

When passed a **single numeric value**, it creates an empty array of that length:

```
const emptyArr = new Array(5);
console.log(emptyArr); // [ <5 empty items> ]
```

> ❗ Use with caution — it creates sparse arrays (holes) instead of initialized values.

✅ 3. Array.of()

Always creates an array from arguments, **even if there's one number**:

```
const arr = Array.of(5);
console.log(arr); // [5]
```

> Unlike new Array(5), this creates a normal array with 5 as a value, not length.

✅ 4. Array.from()

Creates an array from:

- array-like objects (like NodeLists, strings)
- iterable structures
- mapping functions

```
const str = "hello";
const letters = Array.from(str);
console.log(letters); // ['h', 'e', 'l', 'l', 'o']

const numbers = Array.from({ length: 5 }, (_, i) => i + 1);
console.log(numbers); // [1, 2, 3, 4, 5]
```

✅ 5. Empty Array and Pushing Later

You can create an empty array and populate it dynamically:

```
const data = [];
data.push(10);
data.push(20);
console.log(data); // [10, 20]
```

✅ 6. Using Spread Operator (Cloning or Filling)

```
const original = [1, 2, 3];
const copy = [...original]; // shallow copy
console.log(copy); // [1, 2, 3]
```

✅ 7. Array Fill Method

You can initialize an array of fixed length with default values:

```
const filled = new Array(5).fill(0);
console.log(filled); // [0, 0, 0, 0, 0]
```

🧠 Summary Table

Method	Description
`[]`	Literal creation
`new Array()`	Constructor (caution with single numbers)
`Array.of()`	Always creates an array with values
`Array.from()`	Converts iterable/array-like into array
`[].push()`	Adds elements dynamically
`new Array(n).fill(x)`	Initializes array of `n` length with value `x`
`[...arr]`	Clones or expands arrays

📌 Best Practices

- Prefer `[]`, `Array.of()`, or `Array.from()` over `new Array()` for predictability
- Use `Array.from()` or `fill()` to initialize arrays with values

- Avoid sparse arrays unless you have a specific use case

Would you like to explore array initialization in **loops**, **matrix-style**, or related array methods next?