Concat in JavaScript

concat

Here is the **simplest, beginner-friendly explanation of `concat` in JavaScript**, exactly as you prefer—clear and to the point.

✅ `concat()` in JavaScript — Very Simple Explanation

`concat()` is a method used to **join two or more arrays** (or strings) **without changing the original ones**.

1️⃣ concat with Arrays

Syntax

```
js

array1.concat(array2, array3, ...)
```

Example

```
js

const arr1 = [1, 2];
const arr2 = [3, 4];

const result = arr1.concat(arr2);
console.log(result);
```

Output

```
csharp

[1, 2, 3, 4]
```

✔️ **`concat()`** creates a **new array**
✔️ **`arr1`** and **`arr2`** are **not changed**

2️⃣ concat with Strings

```
js

const a = "Vipul";
const b = " Malik";
const fullName = a.concat(b);
console.log(fullName);
```

Output

```arduino
"Vipul Malik"
```

3️⃣ Multiple arrays

```js
const a = [1];
const b = [2, 3];
const c = [4, 5];

const result = a.concat(b, c);
console.log(result);
```

Output:

```csharp
[1, 2, 3, 4, 5]
```

4️⃣ concat Never Modifies Original Array

```js
const arr = [1, 2];
const result = arr.concat([3]);

console.log(arr);      // [1, 2]
console.log(result);   // [1, 2, 3]
```

5️⃣ concat vs spread (`...`)

Today, developers use spread more often:

Using concat

```js
const final = arr1.concat(arr2);
```

Using spread (modern)

```js
const final = [...arr1, ...arr2];
```

Both give same result.

Want concat interview questions also?

Say **"concat questions"** and I'll give Easy + Medium + Hard with solutions.

> is there any difference between spread and concat if than what if not than why these two thins are ther

Great question Vipul — **yes, there is a difference**, but **also no**, depending on how you look at it.
Let me explain in a *super simple* and *interview-friendly* way.

✅ Are `concat` and spread (`...`) different?

👉 Short answer

- **Functionally same** for joining arrays
  (both create a new array and don't mutate original)
- **Difference exists** in:
  - readability
  - merging deep/nested arrays
  - merging complex data
  - adding middle elements
  - spreading only iterables
  - merging objects (spread can, concat cannot)

✅ 1. Basic Use — No Difference

Using `concat`

```js
const a = [1, 2];
const b = [3, 4];

const result = a.concat(b);
console.log(result);
```

Using spread

```js
const a = [1, 2];
const b = [3, 4];

const result = [...a, ...b];
console.log(result);
```

Output in both cases:

```csharp
[1, 2, 3, 4]
```

✔️ Both create a new array
✔️ Both do not modify the originals
✔️ Both give same result

✅ 2. But differences exist

🔹 Difference 1: Spread can be used with objects, concat cannot

```js
const obj1 = {a: 1};
const obj2 = {b: 2};

const merged = {...obj1, ...obj2};
console.log(merged);
```

✔️ Works

`concat` works **only** on arrays
You cannot do:

```js
obj1.concat(obj2); ❌
```

🔹 Difference 2: Spread allows inserting items anywhere

```js
const result = [1, ...[2, 3], 4];
console.log(result);
```

Output:

```csharp
[1, 2, 3, 4]
```

With **`concat`**, this becomes longer:

```js
[1].concat([2,3]).concat([4]);
```

◆ Difference 3: Spread is more readable for modern JS

That’s why developers prefer:

```js
const arr = [...a, ...b];
```

Instead of:

```js
const arr = a.concat(b);
```

◆ Difference 4: concat is safer for non-iterables

Example:

```js
const result = [1, 2].concat(3);
console.log(result);
```

Output:

```csharp
[1, 2, 3]
```

But spread fails:

```js
const result = [...[1,2], 3];  // works (3 is fine)
```

But:

```
js

...[3]          // ok
...3            // ❌ TypeError (3 is not iterable)
```

`concat` can merge numbers, strings, arrays.
Spread only expands **iterables**.

🔹 Difference 5: concat flattens only one level, spread also one level

Both behave same:

```
js

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

console.log([1, [2, 3]].concat());
// [1, [2,3]]
```

Neither deeply flattens.
(need `.flat()` for that)

🔹 Difference 6: concat is older — pre-ES6

Before ES6, there was **no spread operator**, so we used:

```
js

const arr = a.concat(b);
```

Now spread exists, but concat still works for backward compatibility.

⭐ FINAL SUMMARY (Easy to remember)

Feature	Spread `...`	Concat
Merge arrays	✅ Yes	✅ Yes
Modifies original	❌ No	❌ No
Merge objects	✅ Yes	❌ No
Insert elements in between	✅ Easy	❌ Hard
Works with non-iterables	❌ No	✅ Yes
Modern syntax	⭐ Yes	Old
Why both exist?	Spread is new	Concat kept for backward compatibility

---

⭐ One line answer for interview

> *Spread is a modern, more flexible and cleaner way to merge arrays/objects, while* **`concat`** *is the older method only for arrays. Both give same result for normal array merging, but spread supports objects and custom placement, whereas concat is safer for non-iterables.*

If you want, I can give
“Spread vs Concat — 5 interview questions with answers”.