✅ 二、常用数组方法分类讲解 + 使用场景

🌟 1. 添加和删除元素

✅ `push()`：在数组末尾添加元素

- 用法：`arr.push(item1, item2, ...)`
- 返回：新的长度
- 示例：

```
let arr = [1, 2];
arr.push(3);  // => [1, 2, 3]
```

✅ `pop()`：从末尾删除一个元素

- 用法：`arr.pop()`
- 返回：被删除的元素
- 示例：

```
arr.pop();  // => 删除 3，剩下 [1, 2]
```

✅ `unshift()`：在开头添加元素

- 用法：`arr.unshift(item1, item2, ...)`

- 示例：

```
arr.unshift(0);  // => [0, 1, 2]
```

✅ `shift()`：从开头删除元素

- 示例：

```
arr.shift();  // => 删除 0，剩下 [1, 2]
```

📌 适用场景：

- 添加购物车项（`push()`）
- 模拟队列（`shift()` + `push()`）

🌟 2. 查找与判断元素

✅ `includes()`：判断数组是否包含某个值

- 示例：

```
let colors = ['red', 'green'];
colors.includes('red'); // true
```

✅ `indexOf()`：查找元素的位置（找不到返回 -1）

- 示例：

```
colors.indexOf('green'); // 1
```

📌 **适用场景：**

- 搜索关键词是否在列表中
- 判断用户是否已添加某商品

🌟 3. 修改或提取数组

✅ `slice(start, end)`：不改变原数组，提取部分元素

- 示例：

```
let arr = [1, 2, 3, 4];
let part = arr.slice(1, 3);  // => [2, 3]
```

✅ `splice(start, deleteCount, ...items)`：

改变原数组，删除/插入元素

- 示例：

```
arr.splice(1, 2);  // 删除第1个起2个：  [1, 4]
arr.splice(1, 0, 2, 3); // 插入2和3：  [1, 2, 3, 4]
```

📌 **适用场景：**

- 删除购物车中某项
- 只取前几条数据展示

🌟 4. 遍历和处理数组

✅ `forEach()`：遍历数组，每项都执行一次回调函数

- 示例：

```
let sum = 0;
[1, 2, 3].forEach(item => sum += item); // sum = 6
```

✅ `map()`：返回一个新数组，每项执行函数

- 示例：

```
let doubled = [1, 2, 3].map(x => x * 2); // [2, 4, 6]
```

📌 适用场景：

- 显示列表数据（`map()`）
- 累加评分（`forEach()`）

🌟 5. 筛选和查找特定元素

✅ `filter()`：返回符合条件的新数组

- 示例：

```
let numbers = [1, 2, 3, 4];
let even = numbers.filter(n => n % 2 === 0);  // [2, 4]
```

✅ `find()`：返回第一个满足条件的元素

- 示例：

```
let firstEven = numbers.find(n => n % 2 === 0); // 2
```

📌 **适用场景：**

- 只展示特定条件的项目（如已完成任务）
- 查找某用户对象（`find()`）

🌟 6. 其他实用方法

✅ `sort()`：排序（默认按字符串排序）

- 示例：

```
[3, 1, 4].sort();           // => [1, 3, 4]
['b', 'a', 'c'].sort();     // => ['a', 'b', 'c']
```

✅ `reverse()`：反转数组

- 示例：

```
[1, 2, 3].reverse();  // => [3, 2, 1]
```

✅ `join()`：用字符串连接数组元素

- 示例：

```
['a', 'b', 'c'].join('-'); // => "a-b-c"
```

✅ `reduce()`：累加器，聚合数组为一个值

- 示例：

```
[1, 2, 3].reduce((acc, val) => acc + val, 0); // => 6
```

📌 适用场景：

- 成绩求平均（`reduce()`）
- 格式化输出字符串（`join()`）