数组

Ref Javascript

创建与修改数组

数组 (array) 按顺序存放一列值, 写在方括号里。每一项是一个元素 (element), 第一个索引是 0:

```
const scores = [72, 88, 95];
console.log(scores[0]);        // 72
console.log(scores.length);    // 3
scores[1] = 90;                // change one element
console.log(scores);           // [72,90,95]
```

- const 不允许给 scores 换一个全新的数组, 但仍然可以修改数组里面的内容。
- 读取超出末尾的位置会得到 undefined:scores[3] 是 undefined。

用 push 和 pop 在末尾添加和删除:

```
const queue = ["Ann", "Ben"];
queue.push("Cai");             // add to the end
console.log(queue);            // ["Ann","Ben","Cai"]
const last = queue.pop();      // remove from the end and return it
console.log(last);             // Cai
console.log(queue.length);     // 2
```

数组里还可以放数组, 这样就构成一个网格, 用两个索引读取, 先写行:

```
const grid = [
  [1, 2, 3],
  [4, 5, 6],
];
console.log(grid[1][2]);       // 6  (row 1, column 2)
```

map、filter 与 reduce

这几个方法接收一个函数, 并把它作用到每个元素上。它们返回一个**新**数组 (或一个值), 原数组不变:

- map 把每个元素都改一下, 组成一个新数组。
- filter 只保留让函数返回 true 的元素。
- reduce 把所有元素合成一个值, 比如总和。

exercises.lol exercises.com.cn

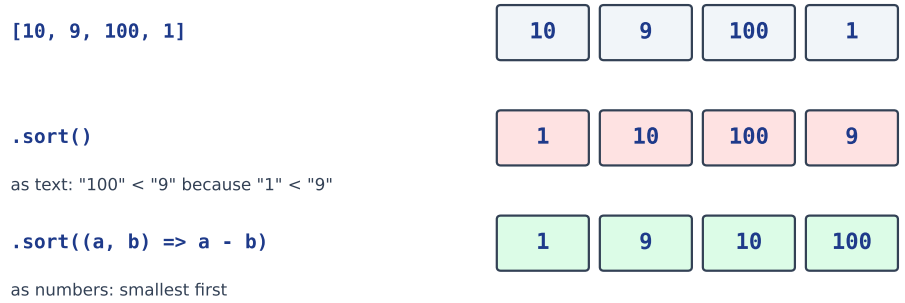
```
const prices = [4, 10, 25];
const withTax = prices.map((p) => p * 1.1);
console.log(withTax.map((p) => p.toFixed(2)));   //
↪  ["4.40","11.00","27.50"]

const cheap = prices.filter((p) => p < 20);
console.log(cheap);                              // [4,10]

const total = prices.reduce((sum, p) => sum + p, 0);
console.log(total);                              // 39
```

在 reduce 里,sum 是累加器。它从最后一个实参 (0) 开始, 每处理一个元素就换成函数的结果。

查找与排序



方法	返回	对 [5, 8, 2] 的例子	结果
.includes(x)	true 或 false	.includes(8)	true
.indexOf(x)	x 第一次出现的索引,没有则为 -1	.indexOf(2)	2
.find(f)	第一个让 f 为真的元素	.find((n) => n > 6)	8
.findIndex(f)	这个元素的索引,没有则为 -1	.findIndex((n) => n > 6)	1

`sort` 把元素排好顺序。不给函数时，它把元素**当作文本**排序，所以数字的顺序会出错：

```
const nums = [10, 9, 100, 1];
console.log([...nums].sort());                    // [1,10,100,9]   (as
↪  text!)
console.log([...nums].sort((a, b) => a - b));     // [1,9,10,100]
console.log([...nums].sort((a, b) => b - a));     // [100,10,9,1]
```

- 比较函数在 `a` 应该排在前面时返回负数，在 `b` 应该排在前面时返回正数。
- `sort` 会改变数组本身。`[...nums]` 先复制一份，所以 `nums` 保持原来的顺序。

常见错误

- 最后一个元素是 `arr[arr.length - 1]`，不是 `arr[arr.length]`。
- `map` 和 `filter` 不会改变数组：要把它们的结果存起来。
- 对数字用 `nums.sort()` 会按文本排序。要传入 `(a, b) => a - b`。