数组

在本模块的最后一篇文章中，我们将了解数组——一种将数据项列表存储在单个变量名下的巧妙方法。在这里，我们将了解为什么这是有用的，然后探索如何创建数组、检索、添加和删除存储在数组中的项目，等等。

先决条件：	基本的计算机知识，对 HTML 和 CSS 的基本理解，对 JavaScript 是什么的理解。
客观的：	了解数组是什么以及如何在 JavaScript 中操作它们。

什么是数组?

数组通常被描述为“类列表对象”；它们基本上是包含存储在列表中的多个值的单个对象。数组对象可以存储在变量中并以与任何其他类型的值几乎相同的方式处理，不同之处在于我们可以单独访问列表中的每个值，并对列表做非常有用和高效的事情，比如循环它并对每个值做同样的事情。也许我们有一系列产品项目及其价格存储在一个数组中，我们想要循环遍历它们并在发票上打印出来，同时将所有价格加在一起并在底部打印出总价。

如果我们没有数组，我们必须将每个项目存储在一个单独的变量中，然后调用为每个项目单独打印和添加的代码。这会写出更长的时间，效率更低，并且更容易出错。如果我们有 10 项要添加到发票中，那已经很烦人了，但是 100 项或 1000 项呢？我们稍后将在本文中返回此示例。

与之前的文章一样，让我们通过在浏览器开发人员控制台中输入一些示例来了解数组的真正基础知识。

创建数组

数组由方括号和以逗号分隔的项目组成。

1. 假设我们要将购物清单存储在数组中。将以下代码粘贴到控制台中：

```
const shopping = ['bread', 'milk', 'cheese', 'hummus',
'noodles'];
console.log(shopping);
```

2. 在上面的示例中，每个项目都是一个字符串，但在数组中我们可以存储各种数据类型——字符串、数字、对象，甚至其他数组。我们也可以在一个数组中混合数据类型——我们不必限制自己在一个数组中只存储数字，而在另一个数组中只存储字符串。例如：

```
const sequence = [1, 1, 2, 3, 5, 8, 13];
const random = ['tree', 795, [0, 1, 2]];
```

3. 在继续之前，创建一些示例数组。

查找数组的长度

您可以使用与查找字符串长度（以字符为单位）完全相同的方式来查找数组的长度（其中有多少项）——通过使用属性 `length` 。尝试以下操作：

```
const shopping = ['bread', 'milk', 'cheese', 'hummus',
'noodles'];
console.log(shopping.length);  // 5
```

访问和修改数组项

数组中的项目从零开始编号。这个数字称为项目的*索引*。所以第一项的索引为 0，第二项的索引为 1，依此类推。您可以使用方括号表示法并提供项目的索引来访问数组中的各个项目，就像访问字符串中的字母一样。

1. 在您的控制台中输入以下内容：

```
const shopping = ['bread', 'milk', 'cheese', 'hummus',
'noodles'];
console.log(shopping[0]);
// returns "bread"
```

2. 您还可以通过为单个数组项赋予新值来修改数组中的项。尝试这个：

```
const shopping = ['bread', 'milk', 'cheese', 'hummus',
'noodles'];
shopping[0] = 'tahini';
console.log(shopping);
// shopping will now return [ "tahini", "milk", "cheese",
"hummus", "noodles" ]
```

> **注意：**我们之前已经说过，但只是提醒一下——计算机从 0 开始计数！

3. 请注意，数组中的数组称为多维数组。通过将两组方括号链接在一起，您可以访问一个数组中的项目，该数组本身位于另一个数组中。例如，要访问数组中的一项，即数组中的第三项 `random` （请参阅上一节），我们可以这样做：

```
const random = ['tree', 795, [0, 1, 2]];
random[2][2];
```

4. 在继续之前尝试对数组示例进行更多修改。试一试，看看哪些有效，哪些无效。

查找数组中项目的索引

如果您不知道某个项目的索引，则可以使用该 `indexOf()` 方法。该 `indexOf()` 方法将一个项目作为参数，并将返回该项目的索引，或者 `-1` 如果该项目不在数组中：

```
const birds = ['Parrot', 'Falcon', 'Owl'];
console.log(birds.indexOf('Owl'));   //  2
console.log(birds.indexOf('Rabbit')); // -1
```

添加项目

要将一项或多项添加到数组的末尾，我们可以使用 `push()` . 请注意，您需要包含一个或多个要添加到数组末尾的项目。

```
const cities = ['Manchester', 'Liverpool'];
cities.push('Cardiff');
```

```
console.log(cities);      // [ "Manchester", "Liverpool",
"Cardiff" ]
cities.push('Bradford', 'Brighton');
console.log(cities);      // [ "Manchester", "Liverpool",
"Cardiff", "Bradford", "Brighton" ]
```

方法调用完成时返回数组的新长度。如果你想将新的数组长度存储在一个变量中，你可以这样做：

```
const cities = ['Manchester', 'Liverpool'];
const newLength = cities.push('Bristol');
console.log(cities);     // [ "Manchester", "Liverpool",
"Bristol" ]
console.log(newLength);  // 3
```

要将项目添加到数组的开头，请使用 unshift()：

```
const cities = ['Manchester', 'Liverpool'];
cities.unshift('Edinburgh');
console.log(cities);     // [ "Edinburgh", "Manchester",
"Liverpool" ]
```

删除项目

要从数组中删除最后一项，请使用 pop()。

```
const cities = ['Manchester', 'Liverpool'];
cities.pop();
console.log(cities);     // [ "Manchester" ]
```

该 pop() 方法返回已删除的项目。要将该项目保存在新变量中，您可以这样做：

```
const cities = ['Manchester', 'Liverpool'];
const removedCity = cities.pop();
console.log(removedCity);   // "Liverpool"
```

要从数组中删除第一项，请使用 shift()：

```
const cities = ['Manchester', 'Liverpool'];
cities.shift();
console.log(cities);     // [ "Liverpool" ]
```

如果您知道某个项目的索引，则可以使用以下方法将其从数组中删除 splice()：

```
const cities = ['Manchester', 'Liverpool', 'Edinburgh',
'Carlisle'];
const index = cities.indexOf('Liverpool');
if (index !== -1) {
  cities.splice(index, 1);
}
console.log(cities);     // [ "Manchester", "Edinburgh",
"Carlisle" ]
```

在这个对 的调用中 splice()，第一个参数说明从哪里开始删除项目，第二个参数说明应该删除多少项目。所以你可以删除多个项目：

```
const cities = ['Manchester', 'Liverpool', 'Edinburgh',
'Carlisle'];
const index = cities.indexOf('Liverpool');
if (index !== -1) {
  cities.splice(index, 2);
}
console.log(cities);     // [ "Manchester", "Carlisle" ]
```

访问每个项目

很多时候你会想访问数组中的每一项。您可以使用以下 for...of 语句执行此操作：

```
const birds = ['Parrot', 'Falcon', 'Owl'];

for (const bird of birds) {
  console.log(bird);
}
```

有时你会想对数组中的每一项做同样的事情，留下一个包含改变的项目的数组。您可以使用 map() . 下面的代码采用数字数组并将每个数字加倍：

```
function double(number) {
  return number * 2;
}
const numbers = [5, 2, 7, 6];
const doubled = numbers.map(double);
console.log(doubled);  // [ 10, 4, 14, 12 ]
```

我们给 一个函数 map()， 并 map() 为数组中的每个项目调用一次该函数，传入该项目。然后它将每个函数调用的返回值添加到一个新数组中，最后返回新数组。

有时您会希望创建一个新数组，其中仅包含原始数组中与某些测试匹配的项目。您可以使用 filter() . 下面的代码接受一个字符串数组并返回一个仅包含长度大于 8 个字符的字符串的数组：

```
function isLong(city) {
  return city.length > 8;
}
const cities = ['London', 'Liverpool', 'Totnes', 'Edinburgh'];
const longer = cities.filter(isLong);
console.log(longer);  // [ "Liverpool", "Edinburgh" ]
```

就像 map()， 我们给方法一个函数 filter()， 并 filter() 为数组中的每个项目调用这个函数，传入项目。如果函数返回 true ， 则该项目被添加到一个新数组中。最后它返回新数组。

字符串和数组之间的转换

通常，您会看到一些包含在大长字符串中的原始数据，您可能希望将有用的项目分离成更有用的形式，然后对它们进行操作，例如将它们显示在数据表中。为此，我们可以使用该 split() 方法。在其最简单的形式中，它采用单个参数，即要分隔字符串的字符，并将分隔符之间的子字符串作为数组中的项返回。

> **注意：**好的，这在技术上是一个字符串方法，而不是数组方法，但我们已经将它与数组放在一起，因为它在这里运行良好。

1. 让我们试一试，看看它是如何工作的。首先，在您的控制台中创建一个字符串：

   ```
   const data =
   'Manchester,London,Liverpool,Birmingham,Leeds,Carlisle';
   ```

2. 现在让我们在每个逗号处拆分它：

   ```
   const cities = data.split(',');
   cities;
   ```

3. 最后，尝试找到新数组的长度，并从中检索一些项目：

   ```
   cities.length;
   cities[0]; // the first item in the array
   cities[1]; // the second item in the array
   cities[cities.length - 1]; // the last item in the array
   ```

4. 您也可以使用该方法进行相反的操作 `join()` 。尝试以下操作：

   ```
   const commaSeparated = cities.join(',');
   commaSeparated;
   ```

5. 将数组转换为字符串的另一种方法是使用 `toString()` 方法。`toString()` 可以说比 `join()` 它不带参数更简单，但限制更多。您 `join()` 可以指定不同的分隔符，而 `toString()` 始终使用逗号。（尝试使用不同于逗号的字符运行第 4 步。）

   ```
   const dogNames = ['Rocket','Flash','Bella','Slugger'];
   dogNames.toString(); // Rocket,Flash,Bella,Slugger
   ```

主动学习：打印那些产品

让我们回到我们之前描述的示例——在发票上打印出产品名称和价格，然后合计价格并将它们打印在底部。在下面的可编辑示例中，有包含数字的注释

——每个注释都标记了一个您必须向代码添加内容的地方。它们如下：

1. 评论下方 `// number 1` 是一些字符串，每个字符串包含产品名称和价格，以冒号分隔。我们希望您将其转换为数组并将其存储在名为 的数组中 `products` 。
2. 在评论下方 `// number 2` ，启动一个 `for...of()` 循环以遍历 `products` 数组中的每个项目。
3. 在注释下方， `// number 3` 我们希望您编写一行代码，将当前数组项 (`name:price`) 拆分为两个单独的项，一个仅包含名称，一个仅包含价格。如果您不确定如何执行此操作，请参阅有用的字符串方法一文以获得帮助，或者更好的方法是查看本文的字符串和数组之间的转换部分。
4. 作为上述代码行的一部分，您还需要将价格从字符串转换为数字。如果您不记得如何执行此操作，请查看第一篇关于字符串的文章。
5. `total` 在代码顶部创建了一个名为 0 的变量。在循环内（下图 `// number 4` ），我们希望您添加一行，在循环的每次迭代中将当前商品价格添加到总计，以便在代码末尾将正确的总计打印到发票上。您可能需要一个赋值运算符来执行此操作。
6. 我们希望您更改正下方的行 `// number 5` ，使 `itemText` 变量等于“当前商品名称 - $ 当前商品价格”，例如在每种情况下都是“鞋子 - $23.99”，这样每个商品的正确信息就会打印在发票。这只是简单的字符串连接，您应该很熟悉。
7. 最后，在 `// number 6` 注释下方，您需要添加一个标记循环 `}` 结束的标记 `for...of()` 。

Live output

- 0

Total: $0.00

Editable code

Press Esc to move focus away from the code area (Tab inserts a tab character).

```
const list =
document.querySelector('.output ul');
const totalBox =
document.querySelector('.output p');
let total = 0;
list.innerHTML = '';
totalBox.textContent = '';
// number 1
                'Underpants:6.99'
                'Socks:5.99'
                'T-shirt:14.99'
                'Trousers:31.99'
                'Shoes:23.99';

// number 2

  // number 3

  // number 4

  // number 5
  let itemText = 0;

  const listItem =
document.createElement('li');
  listItem.textContent = itemText;
  list.appendChild(listItem);
```

Reset Show solution

主动学习：前 5 个搜索

当您在 Web 应用程序中维护当前活动项目的记录时，数组方法（如 `push()` 和 ）的一个很好的用途。 `pop()` 例如，在一个动画场景中，您可能有一个表示当前显示的背景图形的对象数组，出于性能或混乱的原因，您可能希望一次只显示 50 个。当创建新对象并将其添加到数组中时，可以从数组中删除旧对象以保持所需的数量。

在这个例子中，我们将展示一个更简单的用法——在这里我们给你一个假的搜索网站，带有一个搜索框。这个想法是，当在搜索框中输入术语时，列表中会显示前 5 个先前搜索的术语。当词条数超过 5 时，每次在顶部添加新词条时，最后一个词条开始被删除，因此始终显示前 5 个词条。

> **注意：**在真正的搜索应用中，您可能可以点击之前的搜索词返回之前的搜索，它会显示实际的搜索结果！我们现在只是保持简单。

要完成该应用程序，我们需要您：

1. 在注释下方添加一行 `// number 1`，将输入到搜索输入中的当前值添

Live output

Search

Editable code

Press Esc to move focus away from the code area (Tab inserts a tab character).

```
const list = document.querySelector('.output
ul');
const searchInput =
document.querySelector('.output input');
const searchBtn =
document.querySelector('.output button');

list.innerHTML = '';

const myHistory = [];
const MAX_HISTORY = 5;

searchBtn.onclick = () => {
  // we will only allow a term to be entered if
the search input isn't empty
  if (searchInput.value !== '') {
    // number 1

    // empty the list so that we don't display
duplicate entries
    // the display is regenerated every time a
search term is entered.
    list.innerHTML = '';

    // loop through the array, and display all
```

Reset Show solution

测试你的技能！

您已读完本文，但您还记得最重要的信息吗？在继续之前，您可以找到一些进一步的测试来验证您是否保留了这些信息——请参阅测试您的技能：数组。

结论

阅读完本文后，我们相信您会同意数组看起来非常有用；您会看到它们在 JavaScript 中无处不在，通常与循环相关联，以便对数组中的每个项目执

行相同的操作。我们将在下一个模块中教给您有关循环的所有有用的基础知识，但现在您应该给自己鼓掌，好好休息一下；您已经阅读了本模块中的所有文章！

唯一剩下要做的就是完成本模块的评估，这将测试您对之前文章的理解程度。

也可以看看

- 索引集合——数组及其表亲类型数组的高级指南。
- `Array` — `Array` 对象参考页面 — 有关此页面中讨论的功能的详细参考指南，等等。

此页面最后修改于 2023 年 4 月 7 日由MDN 贡献者提供。