ES6-- 扩展运算符与剩余操作符...

作者：oagnahz
原文链接：https://ld246.com/article/1539350981967
来源网站：链滴

扩展运算符与剩余操作符都是以三点开头的操作符，二者长的很像，只是在用法上有些差别。它们已被 ES6 数组支持，能解决很多之前 arguments 解决起来很麻烦的问题。

扩展运算符

因为在React里面经常碰到，自己就做了很简单的几个归纳。主要几种在以下几个方面。

1. 改变函数的调用

将数组作为函数的参数传递进去的时候，通常使用的方法是Function.prototype.apply，例如下面的子

```
var students = ['Abby','Andy'];
var otherStudents = ['Jane','Tom'];
Array.prototype.push.apply(students,otherStudents);
console.log(students); // Abby Andy Jane Tom
```

这样写起来就不太方便，现在有了延展操作符，它可以将数组，更确切的说是（可遍历对象）进行展然后作为函数入参进行调用。现在只需要像下面简单地直接调用push方法就行了。

```
var students = ['Abby','Andy'];
var otherStudents = ['Jane','Tom'];
students.push(...otherStudents);
console.log(students); // Abby Andy Jane Tom
```

2. 数组构造

在延展操作符之前，我们如果想让一个数组成为另一个数组的一部分，不可避免地就要用到push, splie, concat等等。现在，延展操作符允许我们简单地操作就能实现，如下面代码

```
var parts = ['shoulders', 'knees'];
var lyrics = ['head', ...parts, 'and', 'toes'];
console.log(lyrics)// ['head', 'shoulders, 'knees', 'and', 'toes']
```

除此之外，我们还可以使用它来快速复制一个数组。

```
var arr = [1, 2, 3];
var arr2 = [...arr]; // like arr.slice()
arr2.push(4);
// arr2 becomes [1, 2, 3, 4]
// arr remains unaffected
```

3. 数组解构

除了可以快速构造数组之外，还可以用来实现数组的解构。可以用来提取数组中的部分值，并且提取结果是一个数组对象。注意延展操作符总是应该位于解构赋值的最后一个位置。举个栗子。

```
var students = ['Abby', 'Andy', 'Jane', 'Tom'];
```

```
var somestudents, otherStudents;
[somestudents, ...otherStudents] = students ;
console.log(somestudents); // Abby
console.log(otherStudents); // Andy Jane Tom
```

4. 将类数组对象转换成数组

扩展运算符可以将一个类数组对象中索引范围在[0,length)之间的所有属性的值添加到一个数组中，样可以得到一个真正的数组：

```
var nodeList = document.querySelectorAll('div');
var array = [...nodeList];
```

5. React里面的延展属性（Spread Attributes）

React里面，延展操作符可以在JSX中，{...props}可以将ReactElement的props属性展开。

```
var props = {};
props.foo = x;
props.bar = y;
var component = <Component {...props} />;
```

传入对象的属性会被复制到组件内。

它能被多次使用，也可以和其它属性一起用。注意顺序很重要，后面的会覆盖掉前面的。

```
var props = { foo: 'default' };
var component = <Component {...props} foo={'override'} />;
console.log(component.props.foo); // 'override'
```

剩余操作符

剩余参数（rest parameter）允许长度不确定的实参表示为一个数组。如果一个函数的最后一个形参以 ... 为前缀的，则在函数被调用时,该形参会成为一个数组,数组中的元素都是传递给该函数的多出来实参的值。

1. 获取参数

剩余操作符可以用来方便地获取传进来的参数。

```
function fn(a, b, ...args) {
   return args.length;
}
fn(1, 2, 3, 4, 5 ); // 传进来的参数的个数 3
```

其中第一个形参a对应的是1,第二个形参b对应的2,...args表示的就是[3, 4, 5]。

2. 与arguments的不同

上面args可以有length属性，也说明了args是一个数组而不是像arguments一样是一个伪数组，因此在在它上面直接使用所有的数组方法，比如 sort，map，forEach，pop。

arguments想要变成数组，可以通过Array.prototype.slice.call方法，使用剩余操作符可以避免将arguments转为数组的麻烦。这种用法和上面扩展运算符的用法还有点异曲同工之妙。

```
// 下面的代码模拟了剩余数组
function fn1(a, b){
  var args = Array.prototype.slice.call(arguments, fn1.length);
  console.log(args.length);
}

// 现在代码可以简化为这样了
function fn2(a, b, ...args) {
   console.log(args.length);
}
fn1(1,2,3); // 1
fn2(1,2,3); // 1
```

扩展运算符与剩余操作符的区别

关于扩展运算符与剩余操作符之间的区别，简单地说，在某种程度上，剩余操作符和扩展运算符相反扩展运算符会“展开”数组变成多个元素，剩余操作符会收集多个元素和“压缩”成一个单一的元素。