



ES6的解构赋值



jCodeLife



0.241

2020.09.01 06:51:40 字数 1,187 阅读 31

[编辑文章](#)

ES6的解构赋值本质上是属于一种“模式匹配”、“映射关系”：只要等号两边的模式相同，一一对应，左边的变量就会被赋予右边对应的值。其作用可以让代码变得更简洁、可读性变得更高

```
1 var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9];
2 // 没有解构赋值前
3 var a = arr[0];
4 var b = arr[1];
5 var c = arr[2];
6 var d = arr[3];
7 var e = arr[4];
8 var f = arr[5];
9 var g = arr[6];
10 var h = arr[7];
11 var i = arr[8];
12
13 // 使用解构赋值
14 let [a, b, c, d, e, f, g, h, i] = arr;
```

可以看到这是非常高效的。

解构的过程相当于就是分解、匹配的过程。解构赋值的过程就是按照一定的模式从数组或对象中取值，再赋值给对应变量的过程。

那我们先来看数组的解构

数组的解构赋值

1. 一维数组的解构

根据数组数据的下标，有次序的赋值：

```
1 let [a, b, c] = [1, 2, 3]; // es6
2 console.log(a); // 1
3 console.log(b); // 2
4 console.log(c); // 3
```

2. 对嵌套多维数组的解构

```
1 let [a, [[b], c], d] = [1, [[2], 3], 4];
2 console.log(a); // 1
3 console.log(b); // 2
4 console.log(c); // 3
5 console.log(d); // 4
```

3. 不需要匹配的位置可以置空

```
1 let [, c] = [1, 2, 3];
2 console.log(c); // 3
```

4. 使用...扩展运算符，匹配余下的所以值，形成一个数组



jCodeLife

总资产22 (约1.80元)

[一篇深入理解闭包](#)

阅读 198

[深入探究ES6中的Proxy](#)

阅读 14

[深入Object.defineProperty](#)

阅读 19

推荐阅读

[Es6基础语法](#)

阅读 796

[web前端达到什么水平，才能找到工作？](#)

阅读 300

[我：CSS垂直居中还有什么另类方法？求职者：不太了解了](#)

阅读 407

[面试了几个前端，给爷整哭了！](#)

阅读 85,009

[如何理解JavaScript中的对象？](#)

阅读 70



5. 左边数组中没有找到右边数组中对应值，其值为undefined

```
1 let [x,y,z] = [1,2];
2 console.log(x); // 1
3 console.log(y); // 2
4 console.log(z); // undefined
```

以上几种情况可以证实，本质上只要等号两边模式一致，左边变量即可获取右边对应位置的值。这真的是太方便了。

除了数组的解构，对象也能解构

对象的解构赋值

1. 特点

数组的解构是按次序排列的，变量的取值由它的位置决定；而对象的解构必须是变量名与属性名一致，才能取到正确的值

```
1 let { a, b } = { a : 1, c : 2 };
2 console.log(a); // 1
3 console.log(b); // undefined
```

2. 对任意深度的嵌套对象进行解构

```
1 let obj= {
2   arr: [1,{ a: 2 } ]
3 };
4 let { arr: [x, { y}] } = obj;
5 console.log(x); // 1
6 console.log(y); // 2
```

3. 可自定义属性名称(取别名)

上面说到，对象的解构变量名必须要属性名一致才能取到对应的值，有的时候我不想用对象里面的值作为变量，那我又得let一个变量赋值过去吗？其实不用！ES6也想到这样的问题，所以可以在解构的变量名后面自定义属性名称如：

```
1 var {name, id: ID} = { name: 'jack', id: 1 };
2 console.log(ID) // 1
3 console.log(id) // Uncaught ReferenceError: id is not defined
```

说白了其内部机制，是先找到同名属性，然后再赋给对应的变量；真正被赋值的是后者。

可以利用解构对象的特点，简写经常用于代码

```
1 let {log} = console;
2 log("hello word"); //console.log === log
```

除了可以解构对象和数组，对于其他类型能解构吗？

一起来试试

先来看字符串



要注意的是，不能以为是经过了包装类，而将字符串作为对象来解构。那是找不到值的。

```
1 let {a} = "alice";
2 console.log(a); //undefined
```

想想也是，包装类身上也跟本没有这个a属性

但是如果想要字符串包装类上的属性和方法，那是可以的

```
1 let {length,indexOf:index,slice} = "alice"
2 console.log(length); //5
3 console.log(index); //indexOf() { [native code] }
4 console.log(slice); //slice() { [native code] }
```

小结：字符串也能使用解构赋值，如果解构字符串的字符，可以类似像数组那样解构，如果要使用String.prototype的属性可以像解构对象那样，根据变量名去找属性名所对应的值。

看完字符串的解构赋值，其实我们大概能猜到，其实number类型的数字以及boolean的true和false也应该可以解构，至少可以像解构对象那样，解构其包装类上对应的方法，到底能不能呢？

先来测试number数值型的解构赋值吧

发现1

```
1 let [a,...b] = 1234;
2 console.log(a,b); //Uncaught TypeError: 1234 is not iterable
```

发现2

```
1 let {toFixed:toF} = 1234;
2 console.log(toF(2)); //TypeError
```

原来数字类型不能像字符串像对象那样解构，也不能像字符串像数组那样解构，提示 not iterable，不是可迭代对象

布尔值也是一样的提示

```
1 let [a,...b] = true; //Uncaught TypeError: true is not iterable
2 let [x,...y] = false; //Uncaught TypeError: false is not iterable
```

也就是说，只要是不能迭代的東西就不能用于解构，一解构就会报错。因此null、undefined也不能用于解构！

解构的一些用途

1. 交换变量

```
1 let a = 1;
2 let b = 2;
3 [a,b] = [b,a];
```

2. 函数传参中的解构



```
7 // 参数是一组无次序的值
8 function foo2({width,height,left,right}){
9   console.log(width,height,left,right)
10 }
11 foo2({left:300, width:100, right:300, height:200})
```

3. 将函数返回的结果直接解构

```
1 // 返回一个数组
2 function foo() {
3   return [1, 2, 3];
4 }
5 let [a, b, c] = foo();
6
7 // 返回一个对象
8 function foo2() {
9   return {
10     a: 1,
11     b: 2
12   };
13 }
14 let { a, b } = foo2();
```

4. 引入模块的指定方法

加载模块时，往往需要指定输入那些方法

```
1 import { xxx } from 'xxxx'
2 xxx();
```

可以看到解构在工作中还是非常好用的！

有个注意点：

关于括号的用法

如果在解构之前就已经定义了变量，大括号`{}`位于行首时，JS引擎就会认为`{name}`是一个代码块，所以等号就出问题了。

```
1 let name;
2 {name}={name:'Alice'}; //报错
```

解决方式是在包裹一层括号`()`

```
1 let name;
2 ({name}={name:'Alice'});
```

小括号的出现，让整个解构赋值的结构被看做一个代码块，这样内部的`{name}`模式则可以正常匹配到

以上就是解构相关的全部内容！



2人点赞 >



笔记 (2)



"小礼物走一走，来简书关注我"


写下你的评论...

评论 0

赞 2

...



jCodeLife  书山有路勤为径，学海无涯苦作舟

总资产22 (约1.80元) 共写了13.5W字 获得295个赞 共23个粉丝



写下你的评论...

全部评论 0

只看作者

关闭评论

按时间倒序

按时间正序

被以下专题收入，发现更多相似内容

 投稿管理

+ 收入我的专题

 前段开发那些事儿