[译]JavaScript ES6解构赋值指南

菜菜蔡伟 发布于 2015-06-19

前言

让我们来仔细地看看ES6所带来的更清晰的变量声明与赋值语法。现今的变量声明语法十分的直接：左边是一个变量名，右边可以是一个数组：`[]`的表达式或一个对象：`{}`的表达式，等等。解构赋值允许我们将右边的表达式看起来也像变量声明一般，然后在左边将值一一提取。听起来有点迷糊？让我们一起看看下面的例子就好。

数组的解构赋值

现在假设我们有一个`value`变量，其值为`[1, 2, 3, 4, 5]`。然后我们想给数组的前三个元素分别声明一个变量。传统的做法是单独声明和赋值每一个变量：

```
var value = [1, 2, 3, 4, 5];
var el1 = value[0];
var el2 = value[1];
var el3 = value[0];
```

有了这几个新变量，我们原本的`value`现在可以被表示为`[el1, el2, el3, 4, 5]`，因为我们现在并不关心后两个元素，所以也可以说被表示为`[el1, el2, el3]`。那么现在，ES6允许我们在左边使用这个表达式来达到和上一个代码块一样的效果：

```
var value = [1, 2, 3, 4, 5];
var [el1, el2, el3] = value;
```

右边不必一定是变量名：

```
var [el1, el2, el3] = [1, 2, 3, 4, 5];
```

左边也不必一定是声明：

```
var el1, el2, el3;
[el1, el2, el3] = [1, 2, 3, 4, 5];
```

这使我们通过仅仅使用两个变量名，就可以交换两个变量的值，这是从前的JavaScript不可能办到的事情：

```
[el1, el2] = [el2, el1];
```

解构赋值也是可嵌套的：

```
var value = [1, 2, [3, 4, 5]];
var [el1, el2, [el3, el4]] = value;
```

ES6中，返回一个数组的函数更像一个头等公民了：

```
function tuple() {
  return [1, 2];
}

var [first, second] = tuple();
```

同样可以通过简单地在指定位置省略变量来忽略数组中的某个元素：

```
var value = [1, 2, 3, 4, 5];
var [el1, , el3, , el5] = value;
```

这使得从正则表达式里取出匹配的分组的过程十分得简洁：

```
var [, firstName, lastName] = "John Doe".match(/^(w+) (w+)$/);
```

更进一步，默认值同样也可以被指定：

```
var [firstName = "John", lastName = "Doe"] = [];
```

需要注意的是默认值只会在对`undefined`值起作用，下面的例子中`firstName`和`lastName`都将是`null`：

```
var [firstName = "John", lastName = "Doe"] = [null, null];
```

rest参数（...变量名）让事情变得更有趣，它使你可以得到数组中“剩余”的元素。下面这个例子中，`tail`变量将接收数组中”剩余“的元素，为[4, 5]：

```
var value = [1, 2, 3, 4, 5];
var [el1, el2, el3, ...tail] = value;
```

不幸的是，ES6中rest参数的实现非常原始，rest参数之后不能再有其他参数（即只能是最后一个参数）。所以下面例子中的这些非常有用模式，在ES6中是不可能的（会报错）：

```
var value = [1, 2, 3, 4, 5];
var [...rest, lastElement] = value;
var [firstElement, ...rest, lastElement] = value;
```

对象的解构赋值

现在，你已经对数组的解构赋值有了清晰的认识，让我们来看看对象的解构赋值。它们几乎以同样的方式工作，仅仅是从数组变成了对象：

```
var person = {firstName: "John", lastName: "Doe"};
var {firstName, lastName} = person;
```

ES6允许变量名与对应的属性名不一致。下面的例子中，`name`变量将会被声明为`person.firstName`的值：

```
var person = {firstName: "John", lastName: "Doe"};
var {firstName: name, lastName} = person;
```

深层嵌套的对象也不会有问题：

```
var person = {name: {firstName: "John", lastName: "Doe"}};
var {name: {firstName, lastName}} = person;
```

你还可以嵌套些数组在里面：

```
var person = {dateOfBirth: [1, 1, 1980]};
var {dateOfBirth: [day, month, year]} = person;
```

或者一些其他的：

```
var person = [{dateOfBirth: [1, 1, 1980]}];
var [{dateOfBirth}] = person;
```

和数组解构赋值一样，对象解构赋值也可以使用默认值：

```
var {firstName = "John", lastName: userLastName = "Doe"} = {};
```

默认值同样也只会在对`undefined`值起作用，下面的例子中`firstName`和`lastName`也都将是`null`：

```
var {firstName = "John", lastName = "Doe"} = {firstName: null, lastName: null};
```

函数参数的解构赋值

ES6中，函数的参数也支持解构赋值。这对于有复杂配置参数的函数十分有用。你可以结合使用数组和对象的解构赋值。

```
function findUser(userId, options) {
  if (options.includeProfile) ...
  if (options.includeHistory) ...
}
```

通过ES6，这会看上去更清晰简洁：

```
function findUser(userId, {includeProfile, includeHistory}) {
  if (includeProfile) ...
  if (includeHistory) ...
}
```

最后

ES6的解构赋值给JavaScript的语法带来了更多的现代化。它在减少了代码量的同时，增加了代码的可读性和表现力。

原文地址

https://strongloop.com/strongblog/getting-started-with-javascript-es6-...

node.js javascript es6

阅读 22k • 发布于 2015-06-19

菜菜蔡伟

保持平常心。

4k 声望 **280** 粉丝