原文地址：<https://segmentfault.com/a/1190000010907053>

**译者按:**人生苦短，我用**ES6**。

原文: [Top 10 ES6 Features Every Busy JavaScript Developer Must Know](https://webapplog.com/es6/)

译者: [Fundebug](https://fundebug.com/" \t "_blank)

**为了保证可读性，本文采用意译而非直译，并且对源代码进行了大量修改。另外，本文版权归原作者所有，翻译仅用于学习。**

**ES6**，正式名称是**ECMAScript2015**，但是**ES6**这个名称更加简洁。**ES6**已经不再是JavaScript最新的标准，但是它已经广泛用于编程实践中。如果你还没用过**ES6**，现在还不算太晚...

下面是10个ES6最佳特性，排名不分先后：

* 函数参数默认值
* 模板字符串
* 多行字符串
* 解构赋值
* 对象属性简写
* 箭头函数
* Promise
* Let与Const
* 类
* 模块化

### 1. 函数参数默认值

#### 不使用****ES6****

为函数的参数设置默认值：

**function** **foo**(height, color)

{

**var** height = height || 50;

**var** color = color || 'red';

//...

}

这样写一般没问题，但是，当参数的布尔值为**false**时，是会出事情的！比如，我们这样调用**foo**函数：

foo(0, "", "")

因为**0**的布尔值为**false**，这样**height**的取值将是**50**。同理**color**的取值为**'red'**。

#### 使用ES6

**function** **foo**(height = 50, color = 'red')

{

// ...

}

### 2. 模板字符串

#### 不使用ES6

使用**+**号将变量拼接为字符串：

**var** name = 'Your name is ' + first + ' ' + last + '.'

#### 使用ES6

将变量放在大括号之中：

**var** name = `Your name is ${first} ${last}.`

**ES6**的写法更加简洁、直观。

### 3. 多行字符串

#### 不使用ES6

使用**"nt"**将多行字符串拼接起来：

**var** roadPoem = 'Then took the other, as just as fair,\n\t'

+ 'And having perhaps the better claim\n\t'

+ 'Because it was grassy and wanted wear,\n\t'

+ 'Though as for that the passing there\n\t'

+ 'Had worn them really about the same,\n\t'

#### 使用ES6

将多行字符串放在反引号**``**之间就好了：

**var** roadPoem = `Then took the other, as just as fair,

And having perhaps the better claim

Because it was grassy and wanted wear,

Though as for that the passing there

Had worn them really about the same,`

### 4. 解构赋值

#### 不使用ES6

当需要获取某个对象的属性值时，需要单独获取：

**var** data = $('body').data(); // data有house和mouse属性

**var** house = data.house;

**var** mouse = data.mouse;

#### 使用ES6

一次性获取对象的子属性：

**var** { house, mouse} = $('body').data()

对于数组也是一样的：

**var** [col1, col2] = $('.column')；

### 5. 对象属性简写

#### 不使用ES6

对象中必须包含属性和值，显得非常多余：

**var** bar = 'bar';

**var** foo = **function** ()

{

// ...

}

**var** baz = {

bar: bar,

foo: foo

};

#### 使用ES6

对象中直接写变量，非常简单：

**var** bar = 'bar';

**var** foo = **function** ()

{

// ...

}

**var** baz = { bar, foo };

### 6. 箭头函数

#### 不使用ES6

普通函数体内的**this**，指向调用时所在的对象。

**function** **foo**()

{

console.log(**this**.id);

}

**var** id = 1;

foo(); // 输出1

foo.call({ id: 2 }); // 输出2

#### 使用ES6

箭头函数体内的**this**，就是定义时所在的对象，而不是调用时所在的对象。

**var** foo = () => {

console.log(**this**.id);

}

**var** id = 1;

foo(); // 输出1

foo.call({ id: 2 }); // 输出1

### 7. Promise

#### 不使用ES6

嵌套两个**setTimeout**回调函数：

setTimeout(**function**()

{

console.log('Hello'); // 1秒后输出"Hello"

setTimeout(**function**()

{

console.log('Fundebug'); // 2秒后输出"Fundebug"

}, 1000);

}, 1000);

#### 使用ES6

使用两个**then**是异步编程串行化，避免了回调地狱：

**var** wait1000 = **new** Promise(**function**(resolve, reject)

{

setTimeout(resolve, 1000);

});

wait1000

.then(**function**()

{

console.log("Hello"); // 1秒后输出"Hello"

**return** wait1000;

})

.then(**function**()

{

console.log("Fundebug"); // 2秒后输出"Fundebug"

});

### 8. Let与Const

#### 使用Var

**var**定义的变量未函数级作用域：

{

**var** a = 10;

}

console.log(a); // 输出10

#### 使用let与const

**let**定义的变量为块级作用域，因此会报错：(如果你希望实时监控JavaScript应用的错误，欢迎免费使用[Fundebug](https://fundebug.com/))

{

**let** a = 10;

}

console.log(a); // 报错“ReferenceError: a is not defined”

**const**与**let**一样，也是块级作用域。

### 9. 类

#### 不使用ES6

使用构造函数创建对象：

**function** **Point**(x, y)

{

**this**.x = x;

**this**.y = y;

**this**.add = **function**()

{

**return** **this**.x + **this**.y;

};

}

**var** p = **new** Point(1, 2);

console.log(p.add()); // 输出3

#### 使用ES6

使用**Class**定义类，更加规范，且你能够继承：

**class** **Point**

{

**constructor**(x, y)

{

**this**.x = x;

**this**.y = y;

}

add()

{

**return** **this**.x + **this**.y;

}

}

**var** p = **new** Point(1, 2);

console.log(p.add()); // 输出3

### 10. 模块化

JavaScript一直没有官方的模块化解决方案，开发者在实践中主要采用**CommonJS**和**AMD**规范。而**ES6**制定了模块(Module)功能。

#### 不使用ES6

Node.js采用**CommenJS**规范实现了模块化，而前端也可以采用，只是在部署时需要使用[Browserify](http://browserify.org/)等工具打包。这里不妨介绍一下**CommenJS**规范。

**module.js**中使用**module.exports**导出**port**变量和**getAccounts**函数：

module.exports = {

port: 3000,

getAccounts: **function**() {

...

}

}

**main.js**中使用**require**导入**module.js**：

**var** service = require('module.js')

console.log(service.port) // 输出3000

#### 使用ES6

**ES6**中使用**export**与**import**关键词实现模块化。

**module.js**中使用**export**导出**port**变量和**getAccounts**函数：

**export** **var** port = 3000

**export** **function** **getAccounts**(url) {

...

}

**main.js**中使用**import**导入**module.js**，可以指定需要导入的变量：

**import** {port, getAccounts} **from** 'module'

console.log(port) // 输出3000

也可以将全部变量导入：

**import** \* **as** service **from** 'module'

console.log(service.port) // 3000

### 参考链接

* [ES6/ECMAScript2015 Cheatsheet](https://github.com/azat-co/cheatsheets/tree/master/es6)([PDF](https://gum.co/LDwVU/git-1CC81D40))
* [Understanding ECMAScript 6](https://leanpub.com/understandinges6)
* [Exploring ES6](https://leanpub.com/exploring-es6)
* [ECMAScript 6 入门](http://es6.ruanyifeng.com/)
* [Javascript的this用法](http://www.ruanyifeng.com/blog/2010/04/using_this_keyword_in_javascript.html)