ES6中常用的10个新特性讲解


李鸾辉 (/u/c2d0f9fe284a) ＋关注
0.1 2018.06.10 20:59* 字数 1018 阅读 465 评论 0 喜欢 1
(/u/c2d0f9fe284a)

> ECMAScript 6(ES6) 目前基本成为业界标准，它的普及速度比 ES5 要快很多，主要原因是现代浏览器对 ES6 的支持相当迅速，尤其是 Chrome 和 Firefox 浏览器，已经支持 ES6 中绝大多数的特性。

ES6

下面逐一为大家详解常用的ES6新特性：

1.不一样的变量声明：const和let

ES6推荐使用let声明局部变量，相比之前的var（无论声明在何处，都会被视为声明在函数的最顶部）
let和var声明的区别：

```
var x = '全局变量';
{
  let x = '局部变量';
  console.log(x); // 局部变量
}
console.log(x); // 全局变量
```

let表示声明变量，而const表示声明常量，两者都为块级作用域；const 声明的变量都会被认为是常量，意思就是它的值被设置完成后就不能再修改了：

```
const a = 1
a = 0 //报错
```

如果const的是一个对象，对象所包含的值是可以被修改的。抽象一点儿说，就是对象所指向的地址没有变就行：

```
const student = { name: 'cc' }

student.name = 'yy';// 不报错
student  = { name: 'yy' };// 报错
```

有几个点需要注意：

> - let 关键词声明的变量不具备变量提升（hoisting）特性
> - let 和 const 声明只在最靠近的一个块中（花括号内）有效
> - 当使用常量 const 声明时，请使用大写变量，如：CAPITAL_CASING
> - const 在声明时必须被赋值

2.模板字符串

在ES6之前，我们往往这么处理模板字符串：
通过“\”和“+”来构建模板

```
$("body").html("This demonstrates the output of HTML \
content to the page, including student's\
" + name + ", " + seatNumber + ", " + sex + " and so on.");
```

而对ES6来说

1. 基本的字符串格式化。将表达式嵌入字符串中进行拼接。用${}来界定；
2. ES6反引号(``)直接搞定；

```
$("body").html(`This demonstrates the output of HTML content to the page,
including student's ${name}, ${seatNumber}, ${sex} and so on.`);
```

3.箭头函数（Arrow Functions）

> ES6 中，箭头函数就是函数的一种简写形式，使用括号包裹参数，跟随一个 =>，紧接着是函数体；

箭头函数最直观的三个特点。

> - 不需要 function 关键字来创建函数
> - 省略 return 关键字
> - 继承当前上下文的 this 关键字

```
// ES5
var add = function (a, b) {
    return a + b;
};
// 使用箭头函数
var add = (a, b) => a + b;

// ES5
[1,2,3].map((function(x){
    return x + 1;
}).bind(this));

// 使用箭头函数
[1,2,3].map(x => x + 1);
```

> 细节：当你的函数有且仅有一个参数的时候，是可以省略掉括号的。当你函数返回有且仅有一个表达式的时候可以省略{} 和 return；

4. 函数的参数默认值

在ES6之前，我们往往这样定义参数的默认值：

```
// ES6之前，当未传入参数时，text = 'default'；
function printText(text) {
    text = text || 'default';
    console.log(text);
}

// ES6；
function printText(text = 'default') {
    console.log(text);
}

printText('hello'); // hello
printText();// default
```

5.Spread / Rest 操作符

> Spread / Rest 操作符指的是 ...，具体是 Spread 还是 Rest 需要看上下文语境。

当被用于迭代器中时，它是一个 Spread 操作符：

```
function foo(x,y,z) {
  console.log(x,y,z);
}

let arr = [1,2,3];
foo(...arr); // 1 2 3
```

当被用于函数传参时，是一个 Rest 操作符：当被用于函数传参时，是一个 Rest 操作符：

```
function foo(...args) {
  console.log(args);
}
foo( 1, 2, 3, 4, 5); // [1, 2, 3, 4, 5]
```

6.二进制和八进制字面量

> ES6 支持二进制和八进制的字面量，通过在数字前面添加 0o 或者0O 即可将其转换为八进制值：

```
let oValue = 0o10;
console.log(oValue); // 8

let bValue = 0b10; // 二进制使用 `0b` 或者 `0B`
console.log(bValue); // 2
```

7.对象和数组解构

```
// 对象
const student = {
    name: 'Sam',
    age: 22,
    sex: '男'
}
// 数组
// const student = ['Sam', 22, '男'];

// ES5：
const name = student.name;
const age = student.age;
const sex = student.sex;
console.log(name + ' --- ' + age + ' --- ' + sex);

// ES6
const { name, age, sex } = student;
console.log(name + ' --- ' + age + ' --- ' + sex);
```

8.对象超类

ES6 允许在对象中使用 super 方法：

```
var parent = {
  foo() {
    console.log("Hello from the Parent");
  }
}

var child = {
  foo() {
    super.foo();
    console.log("Hello from the Child");
  }
}

Object.setPrototypeOf(child, parent);
child.foo(); // Hello from the Parent
             // Hello from the Child
```

9.for...of 和 for...in

for...of 用于遍历一个迭代器，如数组：

```
let letter = ['a', 'b', 'c'];
letter.size = 3;
for (let letter of letters) {
  console.log(letter);
}
// 结果: a, b, c
```

for...in 用来遍历对象中的属性：

```
let stu = ['Sam', '22', '男'];
stu.size = 3;
for (let stu in stus) {
  console.log(stu);
}
// 结果: Sam, 22, 男
```

10.ES6中的类

ES6 中支持 class 语法，不过，ES6的class不是新的对象继承模型，它只是原型链的语法糖表现形式。

函数中使用 static 关键词定义构造函数的的方法和属性：

```
class Student {
  constructor() {
    console.log("I'm a student.");
  }

  study() {
    console.log('study!');
  }

  static read() {
    console.log("Reading Now.");
  }
}

console.log(typeof Student); // function
let stu = new Student(); // "I'm a student."
stu.study(); // "study!"
stu.read(); // "Reading Now."
```

类中的继承和超集：

```
class Phone {
  constructor() {
    console.log("I'm a phone.");
  }
}

class MI extends Phone {
  constructor() {
    super();
    console.log("I'm a phone designed by xiaomi");
  }
}

let mi8 = new MI();
```

> extends 允许一个子类继承父类，需要注意的是，子类的constructor 函数中需要执行 super() 函数。
> 当然，你也可以在子类方法中调用父类的方法，如super.parentMethodName()。
> 在 这里 阅读更多关于类的介绍。

> 有几点值得注意的是：
>
> - 类的声明不会提升（hoisting），如果你要使用某个 Class，那你必须在使用之前定义它，否则会抛出一个 ReferenceError 的错误
> - 在类中定义函数不需要使用 function 关键词