var、let、const 区别?

麻辣小隔壁 关注
3 2016.09.13 15:42:55 字数 1,038 阅读 77,475

随着ES6规范的到来，Js中定义变量的方法已经由单一的 var 方式发展到了 var、let、const 三种之多。var 众所周知，可那俩新来的货到底有啥新特性呢？到底该啥时候用呢？

我们先来絮叨絮叨 var 方式定义变量有啥 bug？

- 1. **Js没有块级作用域**
  请看这样一条规则：**在JS函数中的var声明，其作用域是函数体的全部。**

```
    for(var i=0;i<10;i++){
          var a = 'a';
    }

    console.log(a);
```

明明已经跳出 for 循环了，却还可以访问到 for 循环内定义的变量 a ，甚至连 i 都可以被放访问到，尴尬~

- 2.** 循环内变量过度共享 **
  你可以猜一下当执行以下这段代码时会发生什么

```
for (var i = 0; i < 3; i++) {
      setTimeout(function () {
        console.log(i)
      }, 1000);
 }
```

在浏览器里运行一下，看看和你预想的结果是否相同?
没错，控制台输出了3个3，而不是预想的 0、1、2。

事实上，这个问题的答案是，循环本身及三次 timeout 回调均*共享唯一的变量 ** i 。当循环结束执行时，i* 的值为3。所以当第一个 timeout 执行时，调用的 *i* 当让也为 3 了。

话说到这儿，想必客官已经猜到 let 是干嘛用的。
你没猜错，就是解决这两个bug的。你尽可以把上述的两个例子中的 var 替代成 let 再运行一次。
注意：必须声明 'use strict' 后才能使用let声明变量，否则浏览并不能显示结果

let是更完美的var

- **let声明的变量拥有块级作用域。** 也就是说用let声明的变量的作用域只是外层块，而不是整个外层函数。let 声明仍然保留了提升特性，但不会盲目提升，在示例一中，通过将var替换为let可以快速修复问题，如果你处处使用let进行声明，就不会遇到类似的bug。

- **let声明的全局变量不是全局对象的属性。**这就意味着，你不可以通过window.变量名的方式访问这些变量。它们只存在于一个不可见的块的作用域中，这个块理论上是Web页面中运行的所有JS代码的外层块。

- **形如for (let x...)的循环在每次迭代时都为x创建新的绑定。**
  这是一个非常微妙的区别，拿示例二来说，如果一个for (let...)循环执行多次并且循环保持了

一个闭包，那么每个闭包将捕捉一个循环变量的不同值作为副本，而不是所有闭包都捕捉循环变量的同一个值。
所以示例二中，也以通过将var替换为let修复bug。
这种情况适用于现有的三种循环方式：for-of、for-in、以及传统的用分号分隔的类C循环。

- **用let重定义变量会抛出一个语法错误（SyntaxError）。**
  这个很好理解，用代码说话

```
let a = 'a';
let a = 'b';
```

上述写法是不允许的，浏览器会报错，因为重复定义了。

** 在这些不同之外，let和var几乎很相似了。举个例子，它们都支持使用逗号分隔声明多重变量，它们也都支持解构特性。 **

const

- ES6引入的第三个声明类关键词：const
- 一句话说明白，const 就是用来定义常量的！任何脑洞(fei)大(zhu)开(liu)的写法都是非法的
  比如这样：

```
//只声明变量不赋值
const a
```

这样：

```
//重复声明变量
const a = 'a';
const a = 'b';
```

还有这样：

```
//给变量重新赋值
const a = 'a';
a = 'b'
```

再来个黑科技：

```
//不过不推荐这么干，实在没啥意思，常量常量，不变的才叫常量嘛~
const a = {a:'a'};
//重新赋值当然是行不通的了
a = {a:'b'};
//嘿嘿嘿科技
a.a = 'b'
```

- const 确实没啥说的，普通用户使用完全没问题，高(dou)端(bi)用户咋用都是问题。

以上就是作者对 var、let、const 用法的一些总结，有不当之处，还请大大们指出。