一、let/var

1.事实上var的设计可以看成JavaScript语言设计上的错误，但是这种错误多半不能修复和移除，因为需要向后兼容

- 大概十年前，Brendan Eich就决定修复这个问题，于是他添加了一个新的关键字：let
- 我们可以将let看成更完美的var

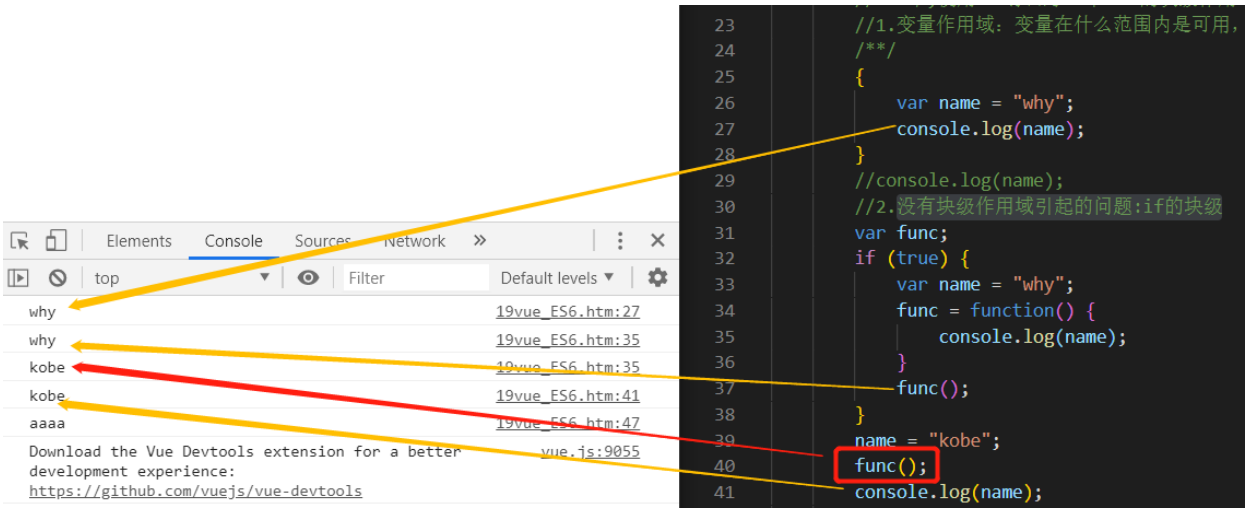
2.块级作用域

- JS中使用var来声明一个变量时，变量的作用域主要和函数的定义有关
- 针对于其他块定义来说是没有作用域的，比如if/for等，这样开发中往往会引起一些问题。
- ES5中的var是没有块级作用域的(if/for)
- ES6中的let是有块级作用的的(if/for)
- ES5之前因为if和for都没有块级作用域的概念,所以在很多时候,我们都必须借助function的作用域来

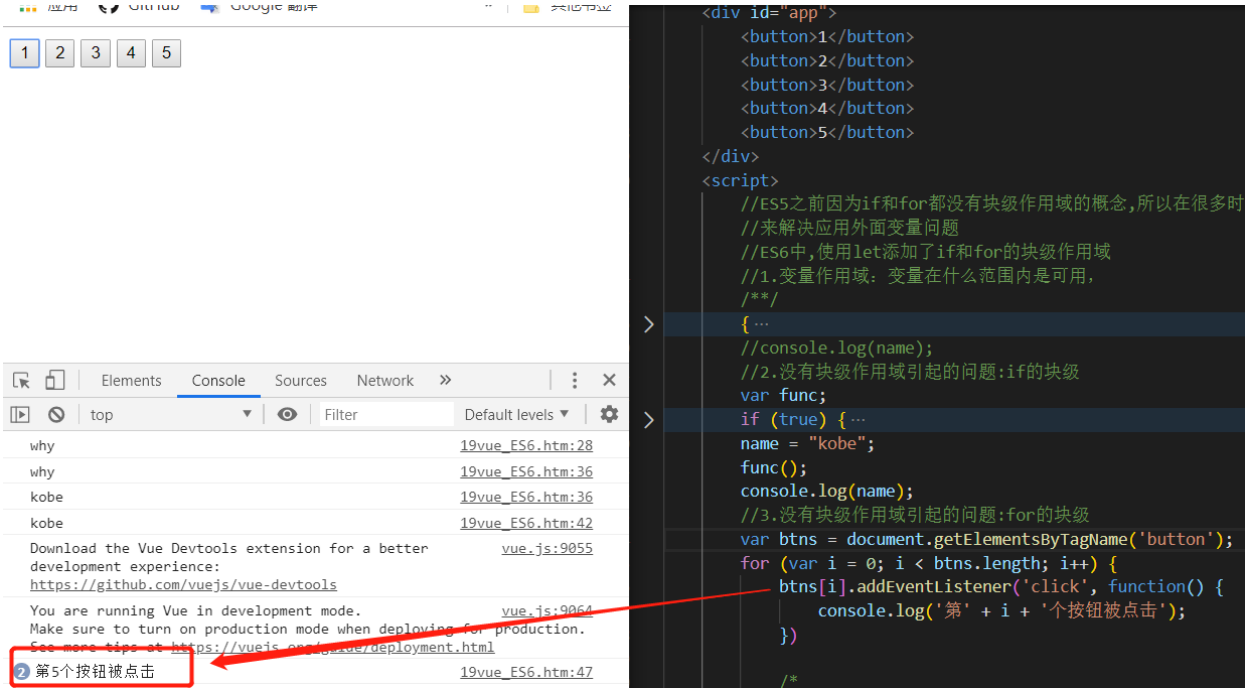
```
// 监听按钮的点击
var btns = document.getElementsByTagName('button');
for (var i = 0; i < btns.length; i++) {
    (function (i) {
     btns[i].onclick = function () {
      alert('点击了' + i + '个')
     }
    })(i)
}
```

```
let btns = document.getElementsByTagName('button');
for (let i = 0; i < btns.length; i++) {
   btns[i].onclick = function () {
   alert('点击了' + i + '个')
 }
}
```

3.没有块级作用域引起的问题:if的块级



4.没有块级作用域引起的问题:for的块级

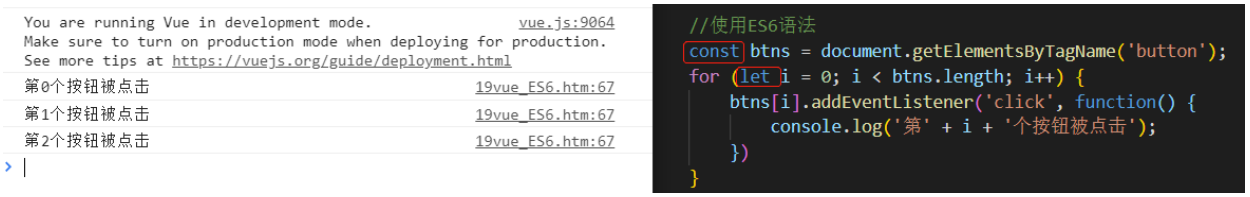


解决方法：

闭包：为什么闭包可以解决问题:函数是一个作用域**

```
//解决方案:闭包
(function(num) {
    btns[i].addEventListener('click', function() {
        console.log('第' + num + '个按钮被点击');
    })
})(i)
```

**ES6



二、const的使用

1.const关键字

- 在很多语言中已经存在，比如C/C++，主要的作用是将某个变量修饰为常量
- 在JavaScript中也是如此，使用const修饰的标识符为常量，不可以再次编辑

2.什么时候使用const呢?

- 当我们修饰的标识符不会被再次赋值时，就可以使用const来保证数据的安全性

建议：在开发中，优先使用const,只有需要改变某一个标识符的时候才使用let

3.const的注意

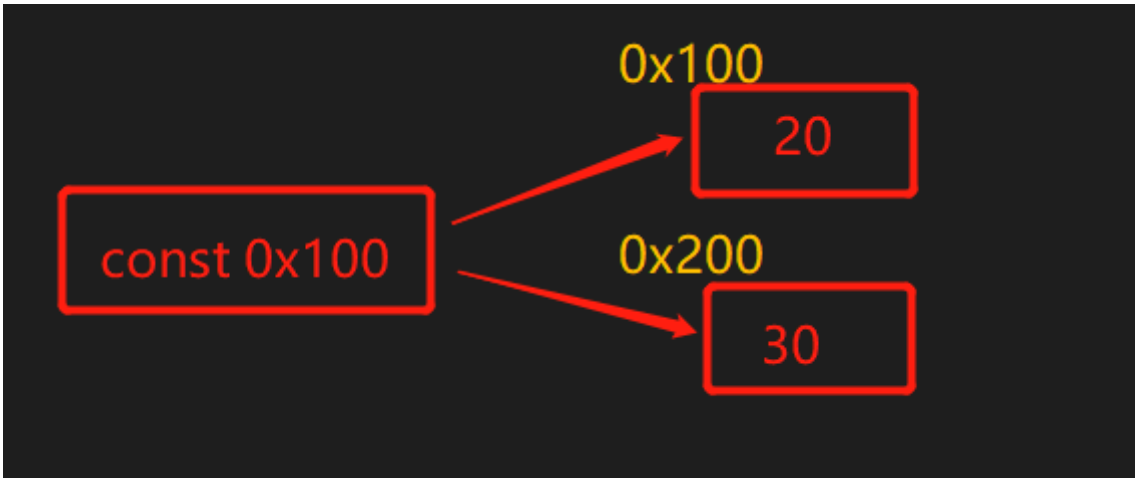
- **const注意一：一旦给const修饰的标识符被赋值之后，不能修改**

```
const a = 20;
a = 30; //错误：不可以修改
```

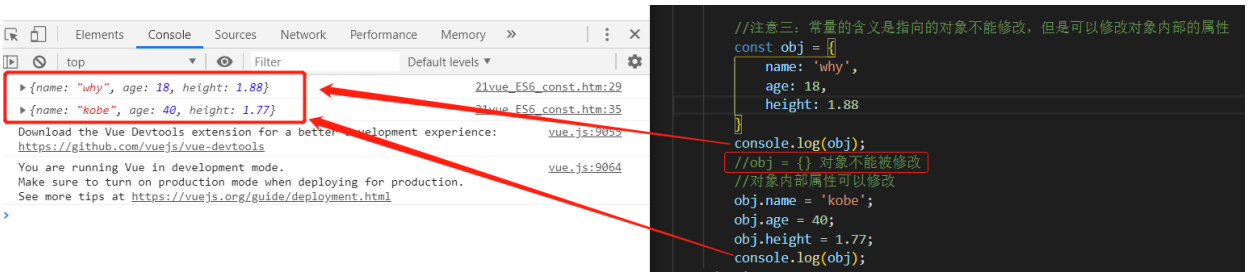
- **const注意二：在使用const定义标识符，必须赋值**

```
const name; //错误：const修饰的标识符必须赋值
```

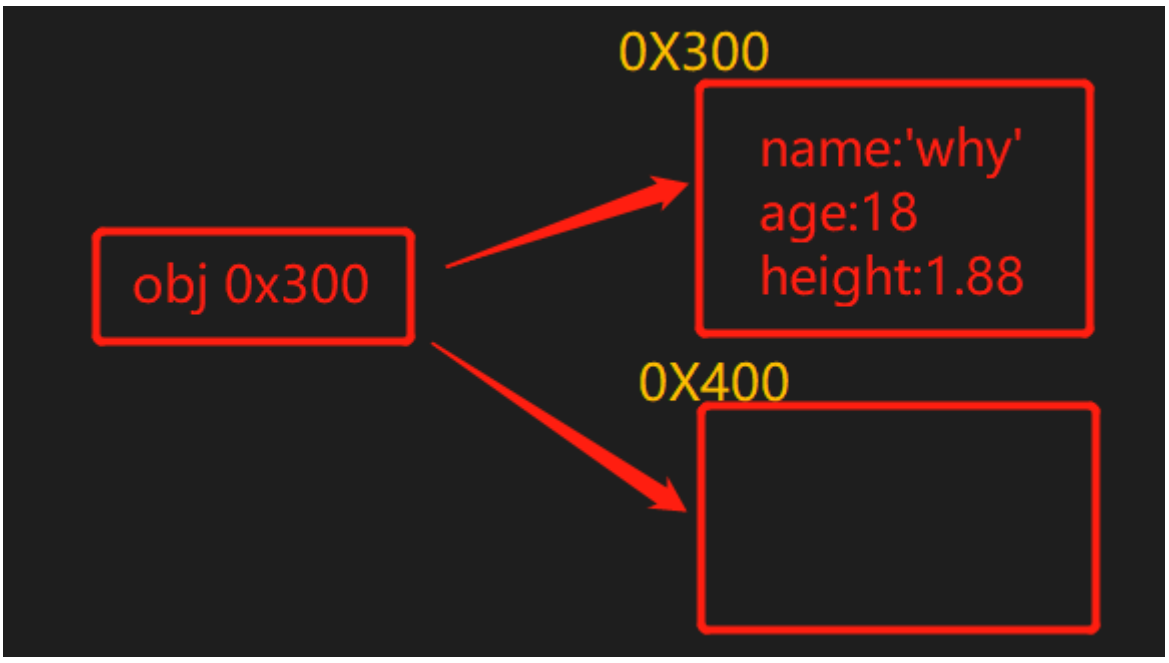
内存图解释：const只记住一个内存地址



- **const注意三：常量的含义是指向的对象不能修改，但是可以修改对象内部的属性**



内存图解释：改变对象属性其内存地址不变



三、对象增强写法

- ES6中，对对象字面量进行了很多增强
- 属性初始化简写和方法的简写

```
// 1.属性的简写
// ES6之前
let name = 'why'
let age = 18
let obj1 = {
   name: name,
    age: age
}
console.log(obj1);
// ES6之后
let obj2 = {
   name, age
}
console.log(obj2);
```

```
// 2.方法的简写
// ES6之前
let obj1 = {
   test: function () {
   console.log('obj1的test函数');
 }
}
obj1.test()

// ES6之后
let obj2 = {
   test () {
      console.log('obj2的test函数')
   }
}
obj2.test()
```

1.属性的增强写法：const obj={name,age,height}

```
//1.属性的增强写法
const name = 'why';
const age = 18;
const height = 1.88

//ES5的写法
const obj1 = {
    name: name,
    age: age,
    height: height
}

//ES6写法
const obj2 = {
    name,
    age,
    height
}
```

2.函数的增强写法:const obj=run(){}

```
//2.函数的增强写法
//ES5的写法
const obj3 = {
    run: function() {

    },
    eat: function() {

    }
}
//ES6写法
const obj4={
    run(){

    },
    eat(){

    }
}
```