# 箭头函数

在前端项目开发中，我们对函数的使用还是比较多的，但是在使用函数进行开发的时候有些问题一直让我们很是烦恼，而且出现问题之后还不好查找，这样导致我们的工作进度容易之后，或者打断我们的思路使我们效率下降。相信很多人多被函数对象中的this 作用域困扰已久，什么时间使用的this是当前函数对象的，什么时间使用的this是全局函数对象的，需不需要使用call或者apply进行调用，改变当前的this对象等等诸多问题，那我们如何来解决诸如此类的问题?

在ES6（ES2015）版本中这类的问题得到了改善，其中有一个箭头函数可以极大的解决我们对this 作用域问题的处理，虽然箭头函数存在一定的局限性，但是完全可以满足我们的日常开发，而且可以不用过多的去估计this作用域问题。那在ES6中箭头函数是如何使用，在我们的Vue项目中，箭头函数的使用频率到底有多高？

接下来我们就来学习一下箭头函数的使用和注意的一些问题，首先需要了解一下箭头函数的定义：ES6允许使用“=>”来定义函数。

## 1.箭头函数的基础用法

let fn=p=>p;

同等于：

var fn = function(p){ return p;}

上面这种方式是使用箭头函数最简单的一种，如果不需要携带参数的话需要使用”( )”圆括号来代替，如果不写直接使用会报错，因此不带参数的箭头函数的写法如下：

let fn = ()=> 10;

在使用箭头函数时，如果涉及到多条数据操作或者返回一个对象的话需要给箭头函数增加对应的大括号或者小括号示例如下：

多行代码：

let fn=(a,b)=> let c=0; return c+b+c; // 错误

let fn=(a,b)=> { let c=0; return c+b+c;}// 正确

单个对象：

let fn= (a,b)=>{ a:a,b:b};// 错误

let fn= (a,b)=>( { a:a,b:b});// 错误

## 2.箭头函数高级应用

综上我们对箭头函数的基本用法已经掌握了，但是对于箭头函数来说，最主要的还是其this作用域的使用，接下来我们来学习一下，箭头函数的特点，主要有以下四个特点：

（1）函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。

（2）不可以当作构造函数，换而言之，不可以使用new操作符，否则会抛出一个错误。

（3）不可以使用arguments对象，该对象在函数体内不存在，但可以用 rest 参数代替。

（4）不可以使用yield命令，因此箭头函数不能用作 Generator 函数

以上四点我们就简单的针对1、2两点进行讲解，第3、4点大家记住就可以了。这里重点进行讲解第1、2点。

### 1.1箭头函数this对象

在使用箭头函数时，在一个函数体内，this对象是定义时所在函数体内的对象，例如下面的这个例子：

function consoleThis() {

this.id=0;

let sef = this;

let a = () => {

console.log(sef)

console.log(this)

console.log("sef==this: " + (sef == this));

};

a();

}

new consoleThis();

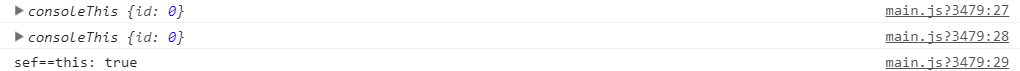


图1

通过上面的例子我们在浏览器的控制台内可以打印出 “sef==this: true” 这么一句话（如图1），也就是表明consoleThis函数中的this跟a函数中的this是同一个对象。也就是说a对象中的this是a函数定义时所在函数体的对象。这个地方是有点绕的，等大家理解了就好了，但是需要先记住的。其次是此this对象是可以继承给其子箭头函数对象的，也就是说在使用箭头函数嵌套开发时，箭头函数的this是在定义时所在的函数体内的this决定的。我们可以通过下面的例子来看对应的实现：

function consoleThis() {

let sef = this;

this.id=0;

let a = () => {

let a\_sef = this;

let b = () => {

console.log(sef.id);

console.log(a\_sef.id);

console.log(this.id);

console.log("sef==this: " + (a\_sef == this));

}

b()

}

a();

}

new consoleThis();

浏览器打印信息如图2：

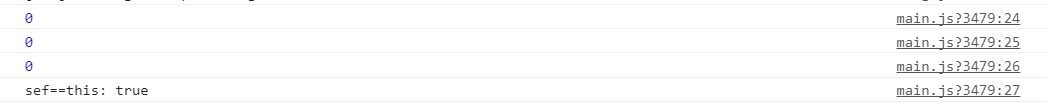


图2

通过上面的信息我们能看出，在箭头函数中的this对象，创建时是取决于他定义时所在的对象，跟我们使用ES5定义function是不一样的。这样我们可以直接使用父节点定义的一些属性不需要再去设定一个全局变量，来调用父节点或者父辈节点的方法或者属性。而且这样维护起来也很方便，减少了大量变量的调度。整洁了代码空间，减少变量的污染。

### 1.2 箭头函数rest参数

在使用箭头函数后，对于参数的传递是非常重要的，有的时候有些参数是不需要传递的，有些时候有些参数是需要传递进去的，在ES5中我们使用arguments来解决参数传递的问题，在ES6中我们需要使用到rest进行参数的传递。大家还要注意的是在箭头函数中除了this，以下三个（包括arguments）指向外层函数的对应变量在箭头函数之中也是不存在的：arguments、super、new.target。

那我们如何使用箭头函数中的rest来解决我们的问题呢，首先我们先来看一下，rest在箭头函数中的应用：

const array = (...nums) => nums;

console.log(array(1,2,3,4,5,6))



图3

上述代码打印出来的数据如图三所示，我们能看出来，我们得到的数据是对应的参数的一个数组信息，也就是说我无论传多少个信息都能通过参数的方式传递到我的函数体内，针对于多参数的操作我们可以看一下代码：

const array = (arg,...nums) => [arg,nums];

console.log(array(1,2,3,4,5,6))



图4

如图4，我们可以将后面的参数作为是一个扩展参数，这样便于我们对不确定参数的处理。

箭头函数除了以上功能外还有一个功能是非常实用的，就是参数的解构，首先我们来看这样的一段代码：

const object = ({head,body}) => (head+"\_"+body);

console.log(object({head:'head',body:'body'}))



图5

通过上面的示例我们能看出来，我们在调用函数的时候就不需要使用param.head或者param.body 来获取对应的参数信息了。这样一来是不是很方便。

关于箭头函数还有很多方法这里就不一一列举了，其他的知识希望大家在下面系统的学习一下。