

## JavaScript中的普通函数和箭头函数

最近被问到了一个问題:

“JavaScript 中的箭头函数 ( => ) 和普通函数 ( function ) 有什么区别?”

我当时想的就是: 这个问题很简单啊~ (flag), 然后做出了错误的回答.....

“箭头函数中的 this 和调用时的上下文无关, 而是取决于定义时的上下文”

这并不是很正确的答案.....虽然也不是完全错误

### 箭头函数中的 this

首先说我的回答中没有错误的部分: 箭头函数中的 this 确实和调用时的上下文无关

```
function make () {
  return ()=>{
    console.log(this);
  }
}

const testFunc = make.call({ name:'foo' });

testFunc(); //=> { name:'foo' }
testFunc.call({ name:'bar' }); //=> { name:'foo' }
```

这个例子可以看到, 确实箭头函数在定义之后, this 就不会发生改变, 无论用什么样的方式调用它, this 都不会改变;

但严格来说, 这并不是“取决于定义时的上下文”, 因为箭头函数根本就没有绑定自己的 this, 在箭头函数中调用 this 时, 仅仅是简单的沿着作用域链向上寻找, 找到最近的一个 this 拿来使用罢了;

从效果上看, 这和我之前的理解并没有多大偏差, 但它们的本质却是截然不同, 箭头函数并不是普通函数新增了 this 不受调用时上下文影响的特性, 而是减少了很多特性;

### 箭头函数其实是更简单的函数

实际上箭头函数中并不只是 this 和普通函数有所不同, 箭头函数中没有任何像 this 这样自动绑定的局部变量, 包括: this, arguments, super(ES6), new.target(ES6).....

借用别人的一个例子:

```
function foo() {
  setTimeout( () => {
    console.log("args:", arguments);
  },100);
}

foo( 2, 4, 6, 8 );
// args: [2, 4, 6, 8]
```

在普通函数中, 会自动绑定上的各种局部变量, 箭头函数都是十分单纯的沿着作用域链向上寻找.....

箭头函数就是这么个简单、纯粹的东西;

所以我认为箭头函数更适合函数式编程, 除了它更短以外, 使用箭头函数也更容易被那些没有显示声明的变量影响, 导致你产生意料之外的计算结果;

### 那么普通函数能否实现和箭头函数一样的效果呢?

如果是像当初的我一样简单的考虑固定住 this 这个易变的家伙.....那倒是很简单, 有些常用的方法, 比如这样:

```
function make () {
  var self = this;
  return function () {
    console.log(self);
  }
}
```

或者

#### 公告

昵称: 不带汽的可乐  
园龄: 1年5个月  
粉丝: 5  
关注: 2  
[+加关注](#)

< 2018年3月 >

日	一	二	三	四	五	六
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

#### 搜索

   
 

#### 常用链接

- [我的随笔](#)
- [我的评论](#)
- [我的参与](#)
- [最新评论](#)
- [我的标签](#)

#### 我的标签

- [javascript\(7\)](#)
- [node\(6\)](#)
- [nodejs\(6\)](#)
- [php\(3\)](#)
- [vue\(3\)](#)
- [vue.js\(3\)](#)
- [前端\(3\)](#)
- [爬虫\(2\)](#)
- [laravel\(2\)](#)
- [lumen\(2\)](#)
- [更多](#)

#### 随笔分类

- [javascript\(6\)](#)
- [koa2\(1\)](#)
- [Laravel / Lumen\(2\)](#)
- [Node.js\(6\)](#)
- [PHP\(3\)](#)
- [vue.js\(3\)](#)
- [web前端\(4\)](#)

#### 随笔档案

- [2017年8月 \(1\)](#)
- [2017年4月 \(3\)](#)
- [2017年3月 \(5\)](#)
- [2017年2月 \(1\)](#)
- [2016年12月 \(1\)](#)
- [2016年10月 \(1\)](#)

#### 最新评论

- [1. Re:vue实现一个移动端屏蔽滑动的遮罩层](#)  
已解决~自问自答一波~在楼主的基础上做了一些修改, 不用修饰符, 改为基础的绑定事件:  
@touchmove='scroll\_prevent'方法:

```
function make () {
  return function () {
    console.log(this);
  }.bind(this);
}
```

然而第二种方法只能固定 this 这一个变量而已，如前文所述，箭头函数中的 arguments 等变量也是从作用域链中寻找的，为了实现类似的效果，我们只有重新定义一个局部变量这一种方式，而 babel 也是使用这种方式对箭头函数进行处理的。

```
function make () {
  return ()=>{
    console.log(this);
    console.log(arguments);
  }
}

//babel it...

function make() {
  var _this = this,
      _arguments = arguments;

  return function () {
    console.log(_this);
    console.log(_arguments);
  };
}
```

## 那么.....如果我想在箭头函数中使用 arguments 该怎么办?

.....我觉得如果你有这个需求，可能还是用普通函数更合适一点.....

但并不是说在箭头函数中无法以类似数组的形式取到所有参数，我们可以利用展开运算符来接收参数，比如这样：

```
const testFunc = (...args)=>{
  console.log(args) //数组形式输出参数
}
```

或许真的有场景需要用到这种写法，但我还是认为，箭头函数更适合那些接受固定的参数，返回一个计算结果的简单情况；

特别感谢 vajoy 所译 [getify](#) 大神的文章，把我从错误的理解上引导回来；

同时也让我明白了，有些时候代码运行的结果和你的预期相同，未必就代表代码运行的原理和你的理解相同

如果这篇文章中有什么错误或不足，欢迎指正

分类: [Node.js](#), [web前端](#), [javascript](#)

标签: [javascript](#), [node](#), [nodejs](#), [前端](#), [function](#), [函数](#), [箭头函数](#)

好文要顶

关注我

收藏该文



不带汽的可乐

关注 - 2

粉丝 - 5



+加关注

« 上一篇: [用递归的方式处理数组 && 把递归方法方法定义到数组的原型上](#) (这是一次脑洞大开的神奇尝试)

» 下一篇: [使用javascript解一道关于会议日程安排的面试题](#)

0 推荐

0 反对

posted @ 2017-03-20 20:22 不带汽的可乐 阅读(839) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

### 最新IT新闻:

- [因为一张黑板Word图画 微软向加纳老师捐赠电脑](#)
- [运营商被曝即将迎来大调整：流量不分省内省外](#)
- [贾跃亭终爆仓 谁要进场接盘被割韭菜](#)
- [OKEX公众号被封，火币网公众号被删名](#)

scroll\_prevent(e){ if(thi.....

--向死而生Voker

### 2. Re:vue实现一个移动端屏蔽滑动的遮罩层

楼主您好，按照您说的方式照做了，非常管用！十分感谢！但是现在我一个需求是禁止滚动的页面还有一个子页面，而它的子页面又需要滚动。。。改如何实现您有思路吗？

--向死而生Voker

### 3. Re:vue实现一个移动端屏蔽滑动的遮罩层

@为了谁我更新了一下文章，PC端的方式也差不多，换个事件罢了... --不带汽的可乐

### 4. Re:vue实现一个移动端屏蔽滑动的遮罩层

@不带汽的可乐划动与滚动不一样，这下明白了。如果在PC上，有了layer后，如何禁用鼠标滚轮？...

--为了谁

### 5. Re:vue实现一个移动端屏蔽滑动的遮罩层

@不带汽的可乐 对，我用的鼠标滚轮的。...

--为了谁

### 阅读排行榜

1. [vue实现一个移动端屏蔽滑动的遮罩层\(11331\)](#)
2. [从vue.js的源码分析，input和textarea上的v-model指令到底做了什么\(4922\)](#)
3. [在Vue中通过自定义指令获取元素\(1664\)](#)
4. [lumen 中的 .env 配置文件简介和适用场景\(1571\)](#)
5. [将node.js程序作为服务，并在windows下开机自动启动（使用forever）\(1151\)](#)

### 评论排行榜

1. [vue实现一个移动端屏蔽滑动的遮罩层\(11\)](#)
2. [用递归的方式处理数组 && 把递归方法方法定义到数组的原型上](#) (这是一次脑洞大开的神奇尝试) (2)
3. [在Vue中通过自定义指令获取元素\(2\)](#)

### 推荐排行榜

1. [将node.js程序作为服务，并在windows下开机自动启动（使用forever）\(2\)](#)
2. [vue实现一个移动端屏蔽滑动的遮罩层\(1\)](#)