**ES6中箭头函数VS普通函数的this指向**

**很多在撸代码的时候，涉及到this总会出现一些问题，无法得到我们想要的值。大多数时候是我们没有弄清楚this的指向到底是什么，所以在某些情况下，this得到的不是我们想要的值。最近学习了一下函数中this指向的问题。**

**普通函数与ES6中箭头函数里，this指向的问题**

**一、普通函数中this**

**（1）总是代表着它的直接调用者，如obj.fn，fn里的最外层this就是指向obj**

**（2）默认情况下，没有直接调用者，this指向window**

**（3）严格模式下（设置了'use strict'），this为undefined**

**（4）当使用call，apply，bind（ES5新增）绑定的，this指向绑定对象**

**注释：**

**1. call 方法第一个参数是this的指向，后面传入的是一个参数列表。当第一个参数为null、undefined的时候，默认指向window。如：getColor.call(obj, 'yellow', 'blue', 'red')**

**2. apply 方法接受两个参数，第一个参数是this的指向，第二个参数是一个参数数组。当第一个参数为null、undefined的时候，默认指向window。如：getColor.apply(obj, ['yellow', 'blue', 'red'])**

**3. bind 方法和 call 方法很相似，第一个参数是this的指向，从第二个参数开始是接收的参数列表。区别在于bind方法返回值是函数以及bind接收的参数列表的使用。低版本浏览器没有该方法，需要自己手动实现**

**bind()方法创建一个新的函数（称为绑定函数）, 当被调用时，将其this关键字设置为提供的值，在调用新函数时，在任何提供之前提供一个给定的参数序列。**

**语法：**

#### fun.bind(thisArg[, arg1[, arg2[, ...]]])

**参数thisArg表示：当绑定函数被调用时，该参数会作为this的指向。当使用new****操作符调用绑定函数时，该参数无效。**

**参数arg1, arg2, ...表示：当绑定函数被调用时，这些参数将置于实参之前传递给被绑定的方法。**

**先来看个例子：**

**this.name="jack";**

**var demo={**

**name:"rose",**

**getName:function(){**

**return this.name;**

**}**

**}**

**console.log(demo.getName());//输出rose 这里的this指向demo  
  
var another=demo.getName;**

**console.log(another())//输出jack 这里的this指向全局对象**

**//运用bind方法更改this指向**

**var another2=another.bind(demo);**

**console.log(another2());//输出rose 这里this指向了demo对象了**

**bind的应用**

**可以对一个函数预设初始参数：**

**function a(){**

**return Array.prototype.slice.call(arguments);//将类数组转换成真正的数组**

**}**

**var b=a.bind(this,15,20)**

**alert(b());//弹出 15，20**

**var s=b(25,30);**

**alert(s);//弹出 15，20，25，30**

**4. 以上 call，apply，bind 方法是ES5新增.**

**二、ES6箭头函数中this**

**（1）默认指向定义它时，所处上下文的对象的this指向。即ES6箭头函数里this的指向就是上下文里对象this指向，偶尔没有上下文对象，this就指向window**

**（2）即使是call，apply，bind等方法也不能改变箭头函数this的指向**

**一些实例加深印象**

**（1）hello是全局函数，没有直接调用它的对象，也没有使用严格模式，this指向window**

**function hello() {**

**console.log(this);  // window**

**}**

**hello();**

**（2）hello是全局函数，没有直接调用它的对象，但指定了严格模式（'use strict'），this指向undefined**

**function hello() {**

**'use strict';**

**console.log(this);  // undefined**

**}**

**hello();**

**（3）hello直接调用者是obj，第一个this指向obj，setTimeout里匿名函数没有直接调用者，this指向window**

**const obj = {**

**num: 10,**

**hello: function () {**

**console.log(this);    // obj**

**setTimeout(function () {**

**console.log(this);    // window**

**});**

**}**

**}**

**obj.hello();**

**（4）hello直接调用者是obj，第一个this指向obj，setTimeout箭头函数，this指向最近的函数的this指向，即也是obj**

**const obj = {**

**num: 10,**

**hello: function () {**

**console.log(this);    // obj**

**setTimeout(() => {**

**console.log(this);    // obj**

**});**

**}**

**}**

**obj.hello();**

**（5）diameter是普通函数，里面的this指向直接调用它的对象obj。perimeter是箭头函数，this应该指向上下文函数this的指向，这里上下文没有函数对象，就默认为window，而window里面没有radius这个属性，就返回为NaN。**

**const obj = {**

**radius: 10,**

**diameter() {**

**return this.radius \* 2**

**},**

**perimeter: () => 2 \* Math.PI \* this.radius**

**}**

**console.log(obj.diameter())    // 20**

**console.log(obj.perimeter())    // NaN**

**简要介绍：箭头函数中的this，指向与一般function定义的函数不同，比较容易绕晕，箭头函数this的定义：箭头函数中的this是在定义函数的时候绑定，而不是在执行函数的时候绑定。  
1、何为定义时绑定  
我们来看下面这个例子：**

**var a =11;**

**var obj={**

**a:22,**

**say:function(){**

**console.log(this.a)**

**}**

**}**

**obj.say();**

**//console.log输出的是22**

**一般的定义函数跟我们的理解是一样的，运行的时候决定this的指向，我们可以知道当运行obj.say()时候，this指向的是obj这个对象。  
（2）**

**var b=11;**

**var obj={**

**b:22,**

**say:()=>{**

**console.log(this.b);**

**}**

**}**

**obj.say();//输出的值为11**

**如果按照之前function定义应该输出的是22,而此时输出了11，那么如何解释ES6中箭头函数中的this是定义时的绑定呢？**

**所谓的定义时候绑定，就是this是继承自父执行上下文！！中的this，比如这里的箭头函数中的this.b，箭头函数本身与say平级以key:value的形式，也就是箭头函数本身所在的对象为obj，而obj的父执行上下文就是window，因此这里的this.b实际上表示的是window.b，因此输出的是11。(this只有在函数被调用，或者通过构造函数new Object()的形式才会有this)**

**类似的还有：**

**（3）**

**var c=11**

**function test1(){**

**this.c=22;**

**let d=function(){**

**console.log(this.c);**

**};**

**d();**

**}**

**var x=new test1();//输出11**

**箭头函数情况：**

**var a=11;**

**function test2(){**

**this.a=22;**

**let b=()=>{console.log(this.a)}**

**b();**

**}**

**var x=new test2(); /输出22**

**个人理解：ES6中定义的时候绑定this的具体含义，应该继承的是父执行上下文里面的this，切忌是父执行上下文！！！这样就很多箭头函数中的指向不明确就迎刃而解了。**

**注意：简单对象（非函数）是没有执行上下文的！  
箭头函数中，this指向的固定化，并不是因为箭头函数内部有绑定this的机制，实际原因是箭头函数根本没有自己的this，导致内部的this就是外层代码块的this。正是因为它没有this，所以也就不能用作构造函数。**

**模拟ES5中的箭头函数转化：**

# ES6：

**function foo() {**

**setTimeout(() => {**

**console.log('id:', this.id);**

**}, 100);**

**}**

# ES5：

**function foo() {**

**var \_this = this;**

**setTimeout(function () {**

**console.log('id:', \_this.id);**

**}, 100);**

**}**

**所以在定义对象的时候，定义对象属性，里面的this指向的一般是全局，或者这个对象所在的那个环境中的this。**

**在window中定义的方法，es5和es6的this都一样指向window**

**<script type="text/javascript">**

**// es5**

**function sum(){**

**console.log(this)**

**}**

**sum()**

**// Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, parent: Window, …}**

**// es6箭头函数**

**var sum1 = ()=>{**

**console.log(this)**

**}**

**sum1()**

**//Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, parent: Window, …}**

**</script>**

* **在事件处理中的this指向**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>es6-let</title>**

**</head>**

**<body>**

**<burron id="app">点我</burron>**

**<script type="text/javascript">**

**var app = document.getElementById('app');**

**// es5**

**// app.onclick = function(){**

**// console.log(this) //指向触发事件的当前元素对象**

**// }**

**// es6箭头函数**

**app.onclick = ()=>{**

**console.log(this)**

**// Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, parent: Window, …}**

**}**

**// 总结： 作为事件处理函数，es5的this指向触发当前事件的对象元素，es6箭头函数指向window**

**</script>**

**</body>**

**</html>**

**深入理解ES6箭头函数中的this**

* **箭头函数看上去是匿名函数的一种简写，但实际上，箭头函数和匿名函数有个明显的区别：**
  + **箭头函数内部的this是词法作用域，由上下文确定。**
  + **箭头函数中的this是在定义函数的时候绑定，而不是在执行函数的时候绑定。**
* **由于this在箭头函数中已经按照词法作用域绑定了，所以，用call()或者apply()、bind()调用箭头函数时，无法对this进行绑定（无法使用call()或者apply()、bind()改变this的指向），即传入的第一个参数被忽略。内部的this指向父执行上下文里面的this。**
* **箭头函数中，this指向的固定化，并不是因为箭头函数内部有绑定this的机制，实际原因是箭头函数根本没有自己的this，导致内部的this就是外层代码块的this。正是因为它没有this，所以也就不能用作构造函数。**
* **箭头函数中的this是在定义函数的时候绑定，看一个例子：**

**var x = 11;**

**var obj = {**

**x: 22,**

**say: ()=>{**

**console.log(this.x);**

**}**

**}**

**obj.say(); // 输出的值为11**

* **所谓的定义时候绑定，就是this是继承自父执行上下文中的this，比如这里的箭头函数中的this.x，箭头函数本身与say平级以key:value的形式，也就是箭头函数本身所在的对象为obj，而obj的父执行上下文就是window，因此这里的this.x实际上表示的是window.x，因此输出的是11。**
* **而对于下面的例子：**

**var obj = {**

**birth: 1983,**

**getAge: function () {**

**var b = this.birth; // 1983**

**var fn = () => new Date().getFullYear() - this.birth; // this指向obj对象**

**return fn();**

**}**

**};**

**obj.getAge(); // 37**

* **例子中箭头函数本身是在getAge方法中定义的，因此，getAge方法的父执行上下文是obj，因此这里的this指向则为obj对象。**