在ES5中this存在绑定问题（this的指代和其调用的位置有关），而ES6中无this的绑定（this只和其定义的位置有关，和调用的位置无关）

this的一个宗旨：this指代永远都是当前对象

1、this在普通函数调用，this指代全局变量

```
// 全局变量
var name = '元歌'
function demo(){
 // 局部变量
 var name = '孙尚香';

 console.log(name);//孙尚香 优先使用局部变量
 console.log(this.name);//元歌
}
demo()
```

2、this在对象内调用时，this指代当前对象(重点)

```
//方式一：
var obj = {
 name:'翠花',
 age:18,
 say:function(){
 // console.log(this==obj)//true
 // console.log(this===obj)//true
 // console.log(obj.name+'今年芳龄：'+obj.age)//不推荐
 console.log(this.name+'今年芳龄：'+this.age)//推荐
  }
}
obj.say();

//方式二
<div id="box" onclick="changeColor(this)"></div>
function changeColor(obj){
 console.log(obj)
 obj.style.background = 'pink'
```

```
}

//方式三（重点）
var oLi = document.getElementsByTagName('li');
for(var i=0;i<oLi.length;i++){
// 函数会优先加载,但是只有在js侦测到点击时再触发
 oLi[i].onclick = function(){
 console.log(i)//当事件触发时,for循环已经执行完毕,i会得到oLi.length
 // oLi[i].style.background = 'pink'//报错 oLi[i]不存在

 // console.log(this) this可以获取到当前事件操作的对象
 this.style.background = 'orange'
 }
}

//方式四（常错的地方：定时器的影响）
var h2 = document.getElementsByTagName('h2')[0];
h2.onclick = function(){
 // 将this赋值给一个变量或者一个对象的属性,使其能够在普通函数内使用
 // (行业习惯使用that _this )
 var that = this;

 // 使用定时器,过1秒改变h2的内容
 //小心在时间内函数的调用，比如：setTimeout setInterval forEach
 setTimeout(function(){
 console.log(this)//this当前指代的是window,因为再普通函数中调用
 console.log(that)
 that.innerHTML = '待阳光明媚，我们出门可好';
 },1000)
}
```

3、this在构造函数中调用，this指代的是实例化的对象

```
// 自定义构造函数
function test(){
 this.name = '张三';
 this.age = 18;
 this.play = function(){
 console.log(this.name+'说：中午不睡，下午受罪！')
 }
}

```

```
// this在构造函数中被调用时,this指代的是实例化的对象obj
var message = new test()

console.log(message)//{name: "张三", age: 18}
console.log(message.name);//张三
message.play();//张三说：中午不睡，下午受罪
```

4、this在apply()、call()、bind()方法中调用

```
var person = {
 name:'静静',
 sex:'女',
 job:'班主任'
}
// 全局变量 属于window对象
var name = '小明';
var obj = {
 name:'小红',
  age:18,
  play:function(){
  console.log(this.name)
  },
  say:function(a,b,c){
  console.log(this.name+'想吃'+a+'和'+b+'以及'+c)
  }
}

obj.play();//小红 this指代当前对象obj

// apply(obj)切换对象
obj.play.apply(person);//静静 当前this被切换成person对象
obj.play.apply();//小明 apply未传入切换的对象(第一个参数),则默认指向window
// call(obj)切换对象
obj.play.call(person);//静静 当前this被切换成person对象


//apply() call() bind()的区别
obj.say('泡椒凤爪','烧仙草','凤梨')

// apply(obj,arr)的第二个参数必须为数组
obj.say.apply(person,['泡椒凤爪','哈根达斯','香梨'])//静静想吃泡椒凤爪和哈根达斯以及香梨
```

```
33
34 // call(obj,arg1,arg2,arg3...)
35 obj.say.call(person,'泡椒凤爪','哈根达斯','香梨')//静静想吃泡椒凤爪和哈根达斯以及香梨
36
37 // bind(obj,arg1,arg2,arg3...)返回一个函数 其他和call用法相似
38 console.log(obj.say.bind(person,'泡椒凤爪','哈根达斯','香梨'))
39 obj.say.bind(person,'泡椒凤爪','哈根达斯','香梨')()//静静想吃泡椒凤爪和哈根达斯以及香梨
```

apply、call、bind的区别

```
1 A.fun.apply(B,arr)将A对象切换为B对象，采用数组的方式传入参数
2
3 A.fun.call(B,arg1,arg2,arg3...)将A对象切换为B对象，需要一个一个单独传入参数
4
5 A.fun.bind(B,arg1,arg2,arg3...)将A对象切换为B对象，返回一个函数
```