# ES6语法**[箭头函数与普通函数的区别](https://www.cnblogs.com/biubiuxixiya/p/8610594.html)**

## 写法上：

箭头函数：

let fun = () => {

console.log('lalalala');

}

普通函数：

function fun() {

console.log('lalla');

}

**箭头函数相当于匿名函数，并且简化了函数定义。箭头函数有几种格式：**

**如果只有一个参数，可以省略():**

**a =>{console.log('lalalala');}**

**如果只包含一个表达式，连{ ... }和return都省略掉：**

**() =>'lalalala' //返回’lalala’，非常简单**

**如果函数体包含多条语句，这时候就不能省略{ ... }和return。**

### **箭头函数是匿名函数，不能作为构造函数，不能使用new**

let FunConstructor = () => {

console.log('lll');

}

let fc = new FunConstructor();

IMG_256

### **箭头函数不绑定arguments，取而代之用rest参数...解决**

**（arguments是函数里的一个关键字，可以代表函数的参数）**

function A(a){

console.log(arguments);

}

A(1,2,3,4,5,8); //  [1, 2, 3, 4, 5, 8, callee: ƒ, Symbol(Symbol.iterator): ƒ]

let B = (b)=>{

console.log(arguments);

}

B(2,92,32,32); // Uncaught ReferenceError: arguments is not defined

解决办法：（和java不定参数相似，用...c）

let C = (...c) => {

console.log(c);

}

C(3,82,32,11323); // [3, 82, 32, 11323]

### **箭头函数不绑定this，会捕获其所在的上下文的this值，作为自己的this值**

var obj = {

a: 10,

b: () => {

console.log(this.a); // undefined

console.log(this); // Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, frames: Window, …} },

c: function() {

console.log(this.a); // 10

console.log(this); // {a: 10, b: ƒ, c: ƒ} }

}

obj.b();

obj.c();

var obj = {

a: 10,

b: function(){

console.log(this.a); //10 },

c: function() {

return ()=>{

console.log(this.a); //10 }

}

}

obj.b();

obj.c()();

### **箭头函数通过 call()  或   apply() 方法调用一个函数时，只相当于传入了一个参数，对 this 并没有影响。（call()和apply()用于改变普通函数中this的指向）**

let obj2 = {

a: 10,

b: function(n) {

let f = (n) => n + this.a;

return f(n);

},

c: function(n) {

let f = (n) => n + this.a;

let m = {

a: 20

};

return f.call(m,n);

}

};

console.log(obj2.b(1)); // 11

console.log(obj2.c(1)); // 11

### **箭头函数没有原型属性**

var a = ()=>{

return 1;

}

function b(){

return 2;

}

console.log(a.prototype); // undefined

console.log(b.prototype); // {constructor: ƒ}

## 总结

* 箭头函数的 this 永远指向其上下文的  this ，任何方法都改变不了其指向，如 call() ,  bind() ,  apply()
* 普通函数的this指向调用它的那个对象