## 对this对象的理解

##### 方法调用模式

var name = "window";

var obj = {

    name: "kxy",

    sayName: function() {

        console.log(this.name);

    }

};

obj.sayName();  //kxy

此时this指向的是obj对象

##### 函数调用模式

var name = "window";

function sayName() {

    console.log(this.name);

}

sayName();

此时this指向的window全局对象

##### 构造函数模式

function Obj() {

    this.name = "kxy";

}

var person = new Obj();

console.log(person.name);   //kxy

此时this指向的是person实例对象

##### apply调用模式

var name = "window";

var person = {

    name: "kxy"

};

function sayName() {

    console.log(this.name);

}

sayName();    //window

sayName.apply(person);   //kxy

sayName.apply();    //window

此时this指向的是apply传入的第一个参数对象 改变this此时的指向

## 理解apply() call() bind()

在javascript中 call apply bind 都是function对象自带的三个方法，这三个方法的主要作用是改变函数中的this指向

##### Call()

定义：调用一个对象的一个方法，用call改变this的指向 不知道该调用对象的方法，而是指向另一个对象，这样this访问的属性就是另一个对象里的属性值

语法：call([thisObj[,arg1[, arg2[, [,.argN]]]]])

thisObj的取值有以下4中情况：

1. 不传 或者传null undefined 此时函数中的this指向window对象
2. 传递另一个函数的函数名，函数中的this指向这个函数的引用
3. 传递字符串 数值 或者布尔类型等基础类型，函数中的this指向其 对应的包装对象 String Number Boolean
4. 传递一个对象，函数中的this指向这个对象

例子1：

function class1(){

this.name=function(){

console.log("我是class1内的方法");

}

}

function class2(){

// 此行代码执行后，当前的this指向了class1（也可以说class2继承了class1）

// 可以看作是由class2调用class1函数

class1.call(this);

}

var f=new class2();

f.name(); //调用的是class1内的方法，将class1的name方法交给class2使用

例子2

function eat(x,y){

console.log(x+y);

}

function drink(x,y){

console.log(x-y);

}

// 相当于在drink中调用eat函数

eat.call(drink,3,2);

输出：5

例子3

function Animal(){

this.name="animal";

this.showName=function(){

console.log(this.name);

}

}

function Dog(){

this.name="dog";

} var animal=new Animal(); var dog=new Dog();

animal.showName.call(dog);

输出：dog

##### Apply()

定义：应用某一对象的一个方法，用另一个对象替代当前对象

语法：apply([thisObj[,argArray]])

说明：对于apply call 二者而言，作用完全一样，只是接收参数的方式不太一样

例子：

function class1(args1,args2){

this.name=function(){

console.log(args,args);

}

}

function class2(){

var args1="1";

var args2="2";

class1.call(this,args1,args2);

/\*或\*/

class1.apply(this,[args1,args2]);

}

var c=new class2();

c.name();

输出：1 2

call 需要把参数按顺序传递进去，而 apply 则是把参数放在数组里。

在JavaScript 中，某个函数的参数数量是不固定的，因此要说适用条件的话，当你的参数是明确知道数量时用 call ；而不确定的时候用 apply，然后把参数 push 进数组传递进去。当参数数量不确定时，函数内部也可以通过 arguments 这个数组来遍历所有的参数。

##### Bind()

说明：Bind() 方法会创建一个新的函数 成为绑定函数，当调用这个绑定 函数时， 绑定函数会以创建它时传入的bind方法的第一个参数作 为this，传入bind() 方法的第二个以及以后的参数加上绑定函 数运行时本身的参照按照顺序作 为原函数的参数来调用原函数

注意：bind方法的返回值是函数

###### 面试题：定义一个log方法 让它可以代理console.log方法

function log(msg)　{

  console.log(msg);

}

log(1);    //1

log(1,2);    //1

这是一个有缺陷的方法，当传入参数不唯一时，会出现错误

正确方法：

function log(){

  console.log.apply(console, arguments);

};

log(1);    //1

log(1,2);    //1 2

接下来的要求是给每一个 log 消息添加一个"(app)"的前辍

log("hello world");    //(app)hello world

function log(){

  var args = Array.prototype.slice.call(arguments);

// unshift是一个数组的方法 但是arguments是一个伪数组对象

// 用call方法让arguments对象能够调用数组的方法

  args.unshift('(app)');

  console.log.apply(console, args);};

总结：

1. apply call bind 三者都是用来改变函数的this对象的指向的
2. apply call bind 三者第一个参数都是this要指向的对象，也就是想指 定的上下文
3. apply call bind 三者都可以利用后续参数进行传参
4. Bind是返回对应函数，便于稍后调用 apply call 则是立刻调用