1. arguments传递的是传递的是类数组,不具有数组的方法。
2. 在调用时，实参个数确定，arguments.length确定不会改变。
3. 双向绑定，是对传递的实参个数来说，对于undefined类型的没传递实参的形参，没有这个特点。
4. call（指向，参数1，参数2，...）方法，调用函数并改变函数this的指向。
5. apply（指向，[参数1，参数2，...]）与call方法一样，只有传参形式不一样
6. fn.bind()不会调用fn函数，会返回函数的拷贝(叫做绑定函数)。所以要执行要再调用一次。

var length=10;

function fn1(a){

console.log(this);

console.log(this.length);

}

function fn2(a,b){

console.log(this);

console.log(this.length);

}

fn1();//fn1的this指向window对象，所以window.length=10

fn1.call();//call方法的第一个参数是改变的this指向，没有传参时，仍默认是window对象，则window.length=10

fn1.call(null);//第一个参数是undefined与null时，仍默认是window对象，指向全局对象，则window.length=10

var str="123";//如果传递参数是数字，字符串，布尔值，this指向该原始值的包装对象

fn1.call(str);//this指向str，this.length=3

fn1.call(fn2);//执行fn1，this指向fn2，则fn1的console.log(this.length);=fn2.length=2

//arguments.length是实参个数，fn.length是形参个数

fn2.call(fn1);

fn2.call(window,1,2);//call（指向，参数1，参数2，...）方法的调用

function add(a,b,c,d){

console.log(Array.prototype.slice(array,3));//arguments传递的是传递的是类数组，不具有数组的方法,apply可以借用已存在的方法.

}

add(1,2,3,4);

var fn1Bound=fn1.bind(fn2,3);//返回绑定函数，fn1Bound绑定了this=fn2,绑定了参数3，传几个参数绑定几个参数

//虽然fn1Bound是返回了fn1的拷贝，但fn1Bound与fn1是两块不同内存

var div=document.getElementById("div");

div.onclick=function(){

function change(){

}

}

//call与apply会立即调用函数执行，bind不会立即调用函数

1. 函数本质就是一个子程序，由函数体的一系列语句组成。
2. 定义函数的方式：1.通过函数声明的形式2.通过函数表达式的形式（function前有东西就是函数表达式，什么都没有就是函数声明）3.通过Function构造函数实例化的形式（可以传入任意数量的实参，最后一个实参是函数体.new Function()创建的是一个匿名函数）
3. 匿名函数：单独的匿名函数是无法运行的。可以把匿名函数赋值给变量或立即执行。
4. 具名函数的优点是可以快速定位错误
5. length属性指明函数定义的形参的个数
6. name属性指明函数实例的名字。当有函数代理名时，name是函数代理名f2，当没有函数代理名时，name是函数名f1.
7. //函数声明的形式

function max(a, b) {

return a > b ? a : b;

};

//通过函数表达式的形式

var max = function(a, b) {

console.log(a > b ? a : b);

};

max(1, 2);

//通过Function构造函数实例化的形式 var max = new Function("a", "b", " console.log(a>b?a:b);")

var f1=function f2(){}

f1();//正确。

f2();//报错。代理函数名的作用域只能在函数的主体内部，相当于局部变量（代理函数名可有可无）

//在这里f1===f2,只是作用域不同