this：指向调用当前函数的那个对象。

JS中的this指向并不是在函数定义的时候确定的，而是在调用的时候确定（调用函数后创建一个新的执行上下文环境）。**函数的调用方式决定了this的指向。**

JS中普通的函数调用方式有三种：

直接调用：通过函数名(…)方式调用，this指向全局对象window

function Foo(){

this.name = "hello";

this.year = "2017";

console.log(this); //this指向window

console.log(this.name);

}

Foo(); //相当于window.Foo() window调用Foo()函数

方法调用：函数作为对象的一个属性时，当用对象调用函数时this指向对象本身

var obj = {

x:10,

fn:function(){

console.log(this); //指向obj

console.log(this.x);

}

};

obj.fn(); //window.obj.fn(); 是obj对象调用fn()方法

特别注意：

var obj = {

x:10,

fn:function(){

console.log(this);

console.log(this.x);

}

};

var fn1 = obj.fn; //fn1是obj的方法fn赋值的，fn1只是指向对象的指针，他属于window

fn1(); //相当于直接调用函数方式 //指向window

特别注意：

var obj = {

x:10,

fn:function(){

function f(){

console.log(this);

console.log(this.x);

}

console.log(this.x); //这里指向obj对象，调用fn是通过obj.fn()执行的

f(); //f()虽然是在obj.fn()内部定义，但它是闭包，this还是指向window

}

};

obj.fn();

new调用：

构造函数就是用来new对象的函数，构造函数的函数名第一个字母大写（规则约定）

function Foo(){

this.name = "hello";

this.year = "2017";

this.SayYear = function(){console.log(this.year);};

console.log(this); //使用new方式调用函数，this指向函数本身

}

var f1 = new Foo(); // Foo {name: "hello", year: "2017"}

console.log(f1.name);

console.log(f1.year);

f1.SayYear()

function Foo(){

this.name = "hello";

return {};

}

var f1 = new Foo();

console.log(f1.name); //==>undefined

function Foo(){

this.name = "hello";

return function(){};

}

var f1 = new Foo();

console.log(f1.name); //==>undefined

function Foo(){

this.name = "hello";

return 1;

//return undefined;

//return null;

}

var f1 = new Foo();

console.log(f1.name); //==>hello

如果返回值是一个对象，this指向的是那个返回的对象，如果不是返回一个对象，this指向函数的实例（才能访问函数内部的属性、方法）

bind()将函数绑定到对象之后再调用

var obj = {

x:10,

fn:function(){

console.log(this);

console.log(this.x);

}

};

var fn1 = obj.fn;

var fn2 = fn1.bind(obj); //返回一个修改过的函数

fn2(); //this指向对象本身obj

color = “red”;

var obj = {color:”blue”};

function saycolor(){

console.log(this.color);

}

var callcolor = saycolor.bind(obj);

callcolor();

通过call()/apply()调用

call()

var obj = {

x:10,

fn:function(){

console.log(this);

console.log(this.x);

}

};

var fn1 = obj.fn;

fn1.call(obj); //使用call方法 this指向函数本身fn

call(参数1，参数2…..参数n)

var obj = {

x:10,

fn:function(){

console.log(this);

console.log(this.x);

}

};

var fn1 = obj.fn;

fn1.apply(obj,[1,2,3]); //使用apply方法this指向函数本身fn

apply(参数1，[参数2，….参数n])第二个参数必须是数组

call(null)/apply(null); //参数1=null，this指向window