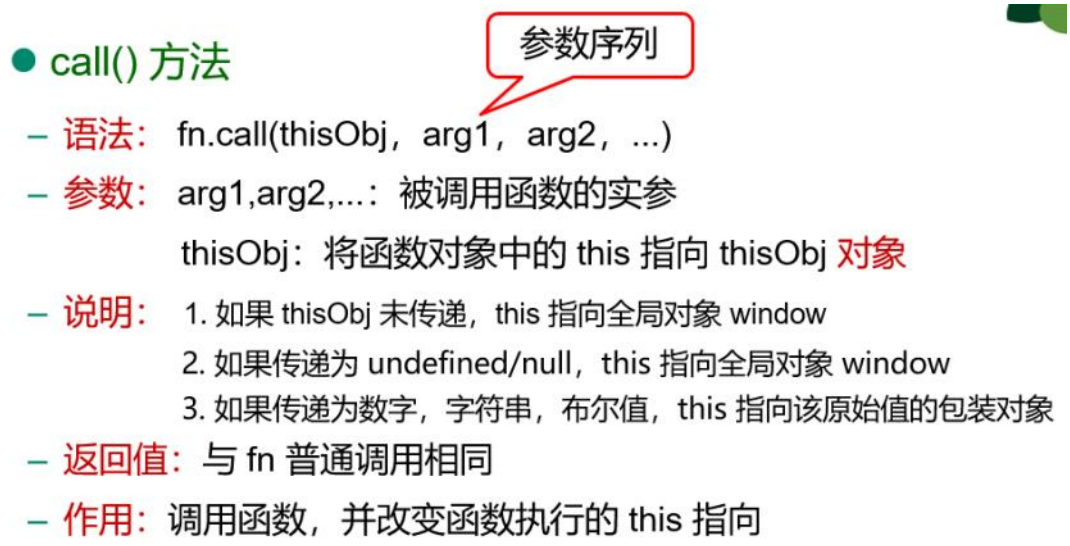
call/apply/bind方法

2020年03月04日, 星期三 14:55

1.this关键字

- 在 function 内部被创建
- 指向调用时所在函数所绑定的对象
- this 不能被赋值，this 的值取决于函数被调用的方式

2.call() 方法




```
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(); //10  this指向的是window对象

fn1.call(); //10  this指向的是window对象

fn1.call(null); //10  this指向的是window对象

fn1.call(undefined); //10  this指向的是window对象
```

```
> var length = 10;

  function fn1(a) {
    console.log(this);
    console.log(this.length);
  }
< undefined
> var str = "123";
  fn1.call(str);
▼ String {"123"}
    0: "1"
    1: "2"
    2: "3"
    length: 3
  ▶ __proto__: String
    [[PrimitiveValue]]: "123"
  3
< undefined

> var length = 10;

  function fn1(a) {
    console.log(this);
    console.log(this.length);
  }

  function fn2(a, b) {
    console.log(this);
    console.log(this.length);
  }
< undefined
> fn1.call(fn2);
ƒ fn2(a, b) {
    console.log(this);
    console.log(this.length);
  }
  2
< undefined
> fn2.call(fn1);
ƒ fn1(a) {
    console.log(this);
    console.log(this.length);
  }
  1
< undefined
```

```
//通过call调用函数时  如何传参
function add(a, b) {
  //需要两个参数
  return a + b;
}
add(1, 2); //直接调用
add.call(window, 1, 2); //call调用
```

3.apply()方法

- apply () 方法

参数数组

  - 语法：fn.apply(thisObj，[arg1，arg2，...])
  - 参数：arg1,arg2,...：被调用函数的实参
    thisObj：将函数对象中的 this 指向 thisObj 对象
  - 说明：1. 如果 thisObj 未传递，this 指向全局对象 window
    2. 如果传递为 undefined/null，this 指向全局对象 window
    3. 如果传递为数字，字符串，布尔值，this 指向该原始值的包装对象
  - 返回值：与 fn 普通调用相同
  - 作用：调用函数，并改变函数执行的 this 指向

注意:call()与apply()使用起来完全一样，唯一的不同就是传参的形式。

call/apply方法练习1：

```
var x = 100;
var obj = {
    x: 50
};
var foo = {
    x: 0,
    getX: function () {
        return this.x;
    }
};
```

```
//通过对象调用方法
console.log(foo.getX());//0

//call()、apply()改变调用方法中this的指向为指定对象
console.log(foo.getX.call(obj));//50
console.log(foo.getX.apply(obj));//50

//call()、apply()没有指定对象时 默认指向全局对象（window）
console.log(foo.getX.call());//100
console.log(foo.getX.apply());//100
```

call()、apply()使用仍然是执行原来对象的方法里面的代码，
只是代码中的this指向改变了
如果调用的对象方法里面没有this，
那么使用call()和apply()没有任何改变，也没有意义

call/apply方法练习2：

```
function add(a,b){
    this(a,b);
    console.log(a+b);
}
function sub(a,b){
    console.log(a-b);
}
add(3,1);
add.call(sub,3,1);//参数列表
add.apply(sub,[3,1]);//参数数组
```

首先，add.call(sub,3,1)执行的是 add 函数
然后，add 函数执行时函数内的 this 指向 sub, 故 this(a,b) 相当于 sub(a,b)
所以，先输出 2，后输出 4。

add（3，1）执行时会报错：因为this这时指向window

apply方法应用技巧：（改变this、借用方法）

```
var arr1 = [1, 2, 3, 4];
var arr2 = [5, 6, 7, 8];
//拼接数组arr1,arr2
arr1.push(5, 6, 7, 8);
arr1.push.apply(null, arr2);
```

拼接数组， 调用时没有使用到this因此在apply方法中的thisObj处写null或者Array

```
var arr3 = [1, 5, 8, 2, 0, -2, 20];
//求数组的最大值和最小值
Math.max.apply(null, arr3);
Math.min.apply(null, arr3);
```

借用方法：借用Math对象的max属性、min属性求最大值最小值。

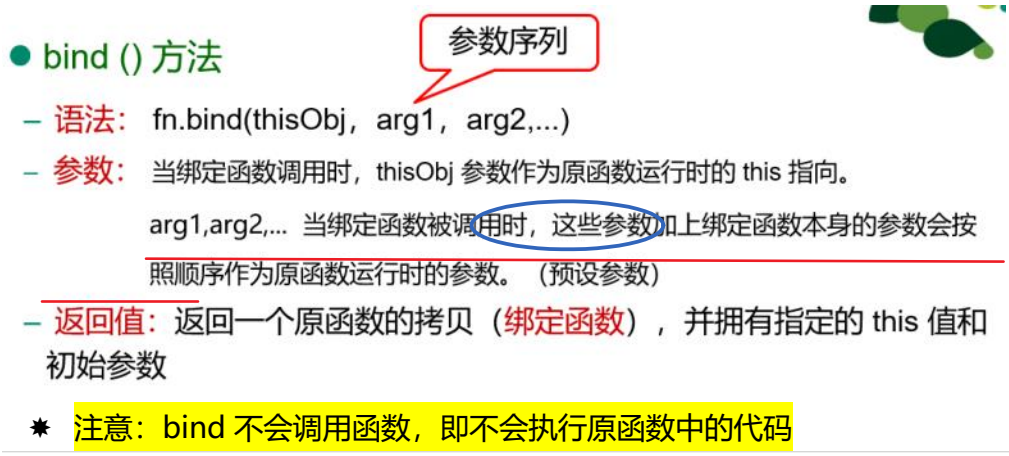
```
function add(a,b,c,d){
    arguments //类数组  没有push slice等这些方法

    console.log(Array.prototype.slice.apply(arguments,[0,3]))
};
add(1,2,3,4)
```

```
▶ (3) [1, 2, 3]                                    VM367:4
undefined
```

//slice方法：切割数组

4.bind()方法




```
//案例1
var length = 10;

function fn1() {
    console.log(this);
    console.log(this.length);
}

function fn2() {
    console.log(this);
    console.log(this.length);
}
var fn1Bound = fn1.bind(fn2);
console.log(fn1Bound == fn1); //false  但是大括号里面的结构是一样的
console.log(fn1Bound);
```

```
false                                              VM373:14
ƒ fn1() {                                          VM373:15
    console.log(this);
    console.log(this.length);
}
undefined
```

```
> //案例2
  function add(a, b) {
      console.log(this);
      console.log(a + b);
  }

  function sub(a, b) {
      console.log(this);
      console.log(a - b);
  }
  var addBound = add.bind(sub, 3);
< undefined
> addBound(1)
  ƒ sub(a, b) {                                          VM507:3
      console.log(this);
      console.log(a - b);
  }
  4                                                      VM507:4
< undefined
> addBound(4)
  ƒ sub(a, b) {                                          VM507:3
      console.log(this);
      console.log(a - b);
  }
  7                                                      VM507:4
< undefined
```

点击完两秒后改变div背景颜色：

```
<div id="div"></div>
<script>
    var div = document.getElementById("div");
    div.onclick = function() {
        function change() {
            this.style.background = "black";
        }
        // setTimeout(change, 2000);//报错，this指向window对象，并没有指向div
        // setTimeout(change.call(div), 2000);//会立即调用函数，定时器失效
        // setTimeout(change.apply(div), 2000);//会立即调用函数，定时器失效
        // setTimeout(change.bind(div), 2000);//不会立即调用函数
        setTimeout(change.bind(this), 2000);//这里的括号中的this就指向div
    };
</script>
```