Arguments对象作用深度研究

github.com/tsrot Arguments

地址：http://blog.xieliqun.com/2016/08/14/arguments/

> 每个函数都会有一个Arguments对象实例arguments，它引用着函数的实参，可以用数组下标的方式"[]"引用arguments的元素。arguments.length为函数实参个数，arguments.callee引用函数自身。

Arguments对象介绍

Arguments对象是一个伪数组对象，它有length属性，可以arguments[i]来访问对象中的元素，但它不能用数组的一些方法，例如push，pop，slice等。

Arguments的length属性

Arguments的length属性，表示function函数实际所传参数的个数。函数名点length可以获取函数期望的传参个数。

```
function argTest(a,b,c){
    var t = arguments.length; //实际传参个数
    var e = argTest.length;   //期望传参个数

    console.log(t);
    console.log(e);
}

argTest(11,12);       //t=2,e=3
argTest(11,12,13);    //t=3,e=3
argTest(11,12,13,14); //t=4,e=3
```

Arguments的参数访问

Arguments对象的参数访问可以用arguments[i]来访问函数所传的参数。

```
function argTest(a,b,c){
    var arg = [];
    for(var i=0;i<arguments.length;i++){
        arg.push(arguments[i]);
    }
    console.log(arg);
}

argTest(11,12);       //[11, 12]
argTest(11,12,13);    //[11, 12, 13]
argTest(11,12,13,14); //[11, 12, 13, 14]
```

Arguments的callee调用

Arguments的callee属性可以调用函数本身，当函数正在执行时才可调用，可以实现方法的递归调用。

```
function argTest(a,b,c){
    var e = arguments.callee.toString();
    console.log(e);
}

argTest(); //打印出函数本身
```

Function对象caller属性

Function对象的caller属性可以指向当前函数的调用者，当调用者函数正在执行时才可调用，

```
function callerTest(){

    if(callerTest.caller){
        var caller = callerTest.caller.toString();
        console.log(caller);
    }else{
        console.log("no caller")
    }
}

function handler(){
    callerTest();
}

function handlerToHandler(){
    handler();
}

callerTest();         //no caller
handler();            //返回调用者handler函数
handlerToHandler();   //返回调用者handler函数
```

Arguments的作用

方法重载

方法重载是指在一个类中定义多个同名的方法，但要求每个方法具有不同的参数的类型或参数的个数。
Javascript并没有重载函数的功能，但是Arguments对象能够模拟重载。

```
//普通方法实现方法重载

function test(a,b,c){
    if(a && b && c){
        console.log(a + b + c);
    }else if(a && b){
        console.log(a + b);
    }else{
        console.log(a);
    }
}

test();           //undefined
test(11,12);      //23
test(11,12,13)    //36
```

```
//Arguments对象实现方法重载

function test(){
    var sum = 0;
    for(var i=0;i<arguments.length;i++){
        sum += arguments[i];
    }
    console.log(sum);
}

test();          //0
test(11,12);     //23
test(11,12,13);  //36
```

```
//ES6实现方法重载

function test(...nums){
    var sum = 0;
    for(var i=0;i<nums.length;i++){
        sum += nums[i];
    }
    console.log(sum);
}

test();          //0
test(11,12);     //23
test(11,12,13);  //36
```

递归调用

这样的好处就是可以实现匿名函数的递归调用。

```
//实现一个阶乘函数

function factorial(n){
    if(n == 1){
        return 1;
    }else{
        n * arguments.callee(n-1);
    }
}

factorial(1); //1
factorial(5); //120
```

不定参问题

比如说，我想判断你传给我的一些数字的大小，取出最大的那个

```
function max(){
    var maxNum = Number.NEGATIVE_INFINITY;;
    for(var i=0;i<arguments.length;i++){
        if(arguments[i]> maxNum){
            maxNum = arguments[i];
        }
    }
    return maxNum;
}

max(1,2,3,11,4,10); //11
max(2,-10,22,11);   //22
```