javascript 函数

args 不是array的实例

可以args[]

可以args.length

```
// 作为返回值
function fn() {
        return function () {
                console.log(1);
        };
}
// var newFn = fn();
// newFn();
fn()();
```

定义方式

字面量

函数声明 不用加分号 执行写前写后都可以

```
function add(){
}
add()
```

语句 要加分号 执行要写后

```
var add = function fn(argument){
    fn();//按习惯
};
add();
```

构造函数

```
var add = new Function('num1','num2','return num1 + num2;');
add();
```

匿名函数自调用 避免function大头

var = function... ();

(function...)()

一元运算符function()();

递归调用

普通函数 和　构造函数

add()　　new Person()

　　返回对象

常见的构造函数

Object() Array()

间接调用

```
var name='xm';
var person={};
person.name='xh';
person.getName=function(){
    return this.name;
};
console.log(person.getName.call(window));   //xm
console.log(person.getName.apply(window));      //xm

function add(num1,num2){
    return num1+num2;
}
console.log(add(1,2));    //直接调用
var datas=[1,2];
console.log(add.apply(datas));      //间接调用
```

函数的参数

形参 实参

参数的个数

1.实参==形参

2.实参<形参　未必赋值的参数为undefined

```
function pow(base,power){
    power=power||2;
```

```
    return Math.pow(base,power);
}
```

3.实参>形参

```
function add(){
    var sum=0;
    if(arguments.length == 0) return;
    for(var i=0;i<arguments.length;i++){
        sum += arguments[i];
    }
    return sum;
}
```

arguments

类数组对象

arguments.callee

等于函数本身

```
function factorial(num){
    if(num <= 1) return 1;
    return num*arguments.calleew(num-1);
}

function add(num1,num2){
    if(argument.length!=add.length) throw new Error('请传入‘+add.length+’个参数');
    return num1+num2;
}
```

什么可以作为参数