//递归：在函数内部调用自身

function say(){

alert('nong sha lei');

say();

}

say();

1. 递归

在函数的内部调用自身，默认是一个无限循环。

使用：要跳出条件，结合return使用

例

function dg(n){

if (n==1)

{ return 1;

}

n += dg(n-1) //老师直接写得 return n+dg(n-1);

return n; //要返回目前综合值，即n

}

dg(10);

console.log(dg(10));

1. **匿名函数**
2. 创建函数

没有名称的函数 function( ){ }

|  |
| --- |
| 函数表达式  var fn2=function (参数){ 函数体; }  变量名就是函数的名称 |

函数声明与函数表达式区别：

函数声明创建函数存在提升，可以在任意位置创建。函数表达式创建函数不存在提升，必须要先创建。（因为实质是变量）

如果在全局定义变量会造成全局污染，所以变量一般写在函数作用域内。函数名同理， function fn1（）{}也会造成全局污染。所以使用匿名函数来规避。

避免全局污染

把变量放到函数作用域下

用匿名函数来

1. 匿名函数自调用

目的：创建函数作用域包裹起来，防止全局污染

|  |
| --- |
| （function(形参列表){  函数体；//函数体中的变量和函数不能被外部访问  }）（实参列表）；//匿名函数自调用 |

1. 回调函数

把匿名函数以实参的形式传递，意味着形参就是传递的匿名的函数名称。

|  |
| --- |
| function fn(a){  //调用fn的时候匿名函数赋值给参数a，a就是函数名称  //执行传递的匿名函数函数体中的代码  //相当于现在有一个a = function( ){ xxx }  a( ); //调用上面的函数a(){xxx}，就可以执行xxx了  }  fn( function(){xxx} ); |

1. 全局函数

parseInt( )/parseFloat( )/isNaN( ) （Number不是）

encodeURI( ) 对一个网址中的汉字进行编码

decodeURI( ) 对已编码的网址进行解码

isFinite() 判断一个值是否为有限值，是T否F

（特例 console.log（-2/0） → -infinity 无穷 分正负 F）

eval( ) 执行字符串中的表达式

eval（'1+2'） //返回3