# 函数

函数:函数其实就是对一段代码进行封装，被封装的代码总是完成某项功能，函数不会主动的运行，需要在程序中手动的调用函数（比如：用户点击的时候），并且可以多次调用这个函数；

* 1. 函数的定义
     1. 为什么要使用函数呢?

因为函数可以封装一个功能,该功能可以被多次使用从而达到代码重复使用.

函数是代码封装和代码复用的体现.

* + 1. 最简单的定义语法

方式一：函数声明，必须指定函数名

function 函数的名字(){

//功能代码

}

通过函数的名字调用执行,如果先多次执行,那么就多次调用:

函数的名字();

方式二：函数表达式，函数名可以选

var 变量 = function(){

//功能代码

};

//函数其实也是一种数据类型，所有把函数赋值给一个变量是可行的，与var a =123与这种赋值一样。

* + 1. 函数的调用

函数封装(打包)了一段代码，被函数（打包）封装的代码不会自动运行，它需要我们在合适的时候调用这个函数。

调用函数：其实就是执行函数中封装的代码。

调用函数语法： 函数名();

* + 1. 函数定义举例:





* 1. 函数参数(形式参数)
     1. 为什么用形式参数(形参)

函数是用来完成一个功能,但是有时候后完成此功能时需要数据的参与,而这些数据只有在使用该功能时能够确定.

调用函数时可以把函数中需要的数据传入到函数中，那么就需要使用参数。

比如：函数中，计算两个数的和，但是这个两个数可以是任何的数据；

* + 1. 形式参数定义

写在函数定义中,出现在函数名的()里面

function 函数名字(形式参数1,形式参数2){

功能代码…

}

* + 1. 参数举例说明



使用函数是传入的参数数据叫做实际参数(实参)

执行过程: 实际参数分别赋值后形式参数后,方法才执行.

* 1. 函数返回值
     1. 为什么用使用返回值

函数功能代码执行结果可能被其他的程序使用.

* + 1. 函数返回值定义

如果函数执行参结果的数据需要被其他程序所使用. 可以通过return关键字标记函数执行的结果(返回结果).

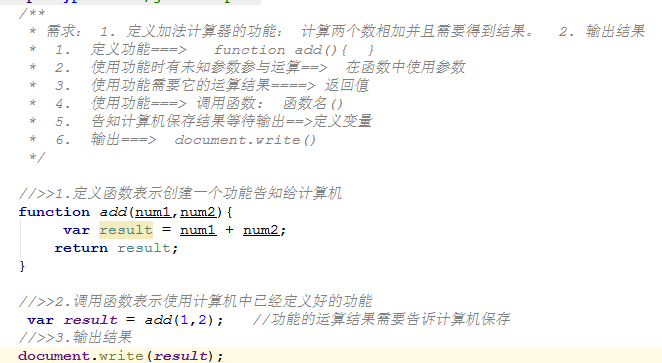
function 函数名(形参1,形参2){

//功能代码.

**return 功能代码运算结果;**

}

* + 1. 返回值举例



执行过程:将1赋值为num1, 2赋值给num2后在执行函数.

传入实际参数时注意:

* + - 1. 实际参数列表的顺序和形式参数的顺序一一对应
      2. 实际参数列表的类型和形式参数的期望类型一一对应
  1. 定义函数三部曲

function 函数名(参数，参数){

//功能代码 （函数体）

return 返回值；

}

var result = 函数名(实参，实参);

分析思路：

从大局入手🡺🡺🡺🡺根据需求分析语义🡺🡺🡺🡺函数的名字 printHello

从功能内部🡺🡺🡺🡺未知数据是否参与运算🡺🡺🡺🡺参数个数 num1,num2

功能外部🡺🡺🡺🡺功能使用后的结果🡺🡺🡺🡺 返回值 return;



函数是一种代码封装的体现：

* 1. 练习
     + 1. 设计比较两个数的最大值功能
       2. 设计制作任意行数和列数表格的功能

<tr>

<td></td>

</tr>

1. 变量的作用域
   1. 什么是变量作用域

变量作用域指的是一个变量在哪个范围内可以使用，变量作用域被函数分为两种

全局变量：函数外定义的变量，整个程序中都可以使用。

局部变量：在某个特定的函数中定义的变量，称之为局部变量，只能够在函数内部使用

全局变量的特殊形式：

在函数内定义但是没有加var，也被作为全局变量使用。

* 1. 为什么要用变量作用域

可以在不同的范围内(函数外和函数内)使用相同的变量名字，引用不同的值。

* 1. 如何选择变量的作用域

如果变量的数据只在函数内部使用，请使用局部变量。

如果变量的数据函数内部和外部都要使用，请使用全局变量。

1. 系统函数
   1. 什么是系统函数

在浏览器已经定义好并且告知给计算机的函数（功能）， 我们无需定义直接使用即可。

* 1. 有哪些函数

String(值)：将值转换为字符串

Number( 值)：将值转换为数字类型

Boolean(值)： 将非布尔类型值转为布尔类型 0 “” null转换为false 其余转换 为true

parseInt (值)转为整数

parseFloat(值)：转为小数类型

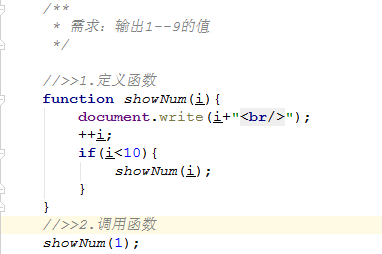
isNaN(值)：判定 值 如果不是数字 返回为true 是数字返回为false

eval(字符串): 将字符串转换为js代码进行执行； 通常用其制作js模板。

1. 递归函数
   1. 什么是递归函数

在函数体内对函数自身调用！

通过递归的函数打印出1—9的数值:



* 1. 递归函数的前提
     + 1. 完成用户需要必须是一个函数
       2. 函数必须有出口
  2. 为什么要使用递归函数

大问题拆解成小问题，小问题与大问题的解决思路一致！它往往使函数的描述简洁而且易于理解。

* 1. 递归函数在哪里使用

在工作中最常用的地方是解决树状结构类的问题. 例如读取文件目录.以及货品分类树, 快速排序算法等等…

* 1. 递归调用的特点

递归算法所体现的“重复”一般有三个要求：

一是每次调用在规模上都有所缩小(通常是减半)；

二是相邻两次重复之间有紧密的联系，前一次要为后一次做准备(通常前一次的输出就作为后一次的输入)；

三是在问题的规模极小时必须用直接给出解答而不再进行递归调用，因而每次递归调用都是有条件的(以规模未达到直接解答的大小为条件)，无条件递归调用将会成为死循环而不能正常结束。

总结:

**递归点: 什么时候再次调用当前函数**

**递归出口:什么时候不再调用当前函数**

* 1. 递归实例

使用递归算法打印1-num的累加

递归点: num+(num-1)

递归出口: num==1 || num==0;

/\*

\* 使用递归算法打印1-N的累加

\*/

function f(num){

if(num==1 || num==0){

return num;

}

return num+f(num-1);

}

console.log(f(100));