# 1．JS函数

## 1.1函数定义与调用

函数定义分为两种：声明式与引用式。

声明式定义是我们用的最多也最通用的方式，即通过function funName(var1,var2…){方法体内容}。

引用式定义是通过var funName = function(var1,var2…){方法体内容}。

两者的区别体现在运行顺序上，如下例，大家先猜猜输出的结果：

|  |
| --- |
| <script>  function dwn(s){  document.write(s+'</br>');  }  function f1(){ //1  dwn("date");  }  f1();//2  function f1(){//3  dwn("new date");  }  f1();//4  f1 = function(){//5  dwn("new new date");  }  f1();//6  </script> |

通过选择得到结构为：new date;new date;new new date.并没有输出我们可能想像的date;new date;new new date.这是因为声明式定义的函数优先于函数执行。而引用式函数是在运行时动态解析的。因此上述代码运行顺序可以理解为1->3->2->4->5->6.

## 1.2函数的参数

函数分为两种参数：形参与实参。形参指的是声明定义时的参数，如function abc(x,y){}其中的x和y就是形参，实参指调用时传入的实际参数，如：abc(10,10),此中的10，10就是实际参数。

Js中，实参多可以多于或者小于形参，如果小于形参，未传入的将以undefined来表式，如果多于，多出的部分在函数中将得不到。

函数内部有一个特殊的arguments对象，是Arguments对象的一个引用，与形参作用域一样，是一个数据对象，代表实参。如下例：

|  |
| --- |
| function testArguments(x,y){  alert(testArguments.length);//打出形参个数  alert(arguments.length);//打出实参个数  }  testArguments(10);//输出为2,1;  testArguments(1,2,3);//输出为2,3; |

arguments很像数组，但它不是数据，无数据的一些核心方法如join,sort等。