# 概述

函数声明、函数表达式、匿名函数

# 函数定义

## 函数声明

function fnName () {…};

使用function关键字声明一个函数，再指定一个函数名，叫函数声明。

## 函数表达式

var fnName = function () {…};

使用function关键字声明一个函数，但未给函数命名，最后将匿名函数赋予一个变量，叫函数表达式，这是最常见的函数表达式语法形式。

## 匿名函数

function () {};

使用function关键字声明一个函数，但未给函数命名，所以叫匿名函数，匿名函数属于函数表达式，匿名函数有很多作用，赋予一个变量则创建函数，赋予一个事件则成为事件处理程序或创建闭包等等。

# 函数声明和函数表达式区别

函数声明和函数表达式不同之处在于：

1. Javascript引擎在解析javascript代码时会‘函数声明提升’（Function declaration Hoisting）当前执行环境（作用域）上的函数声明，而函数表达式必须等到Javascirtp引擎执行到它所在行时，才会从上而下一行一行地解析函数表达式。
2. 函数表达式后面可以加括号立即调用该函数，函数声明不可以，只能以fnName()形式调用 。

## 实例:

fnName();function fnName(){

    ...

}//正常，因为‘提升’了函数声明，函数调用可在函数声明之前

fnName();var fnName=function(){

    ...

}//报错，变量fnName还未保存对函数的引用，函数调用必须在函数表达式之后

var fnName=function(){

    alert('Hello World');

}();//函数表达式后面加括号，当javascript引擎解析到此处时能立即调用函数

function fnName(){

    alert('Hello World');

}();//不会报错，但是javascript引擎只解析函数声明，忽略后面的括号，函数声明不会被调用

function(){

    console.log('Hello World');

}();//语法错误，虽然匿名函数属于函数表达式，但是未进行赋值操作，//所以javascript引擎将开头的function关键字当做函数声明，报错：要求需要一个函数名

# (function(){…})()和(function(){…}())

( function(){…} )()和( function (){…} () )这两种立即执行函数的写法，最初我以为是一个括号包裹匿名函数，并后面加个括号立即调用函数，当时不知道为什么要加括号，后来明白，**要在函数体后面加括号就能立即调用，则这个函数必须是函数表达式，不能是函数声明**。

## 实例:

(function(a){

    console.log(a);   //firebug输出123,使用（）运算符

})(123);

(function(a){

    console.log(a);   //firebug输出1234，使用（）运算符

}(1234));

!function(a){

    console.log(a);   //firebug输出12345,使用！运算符

}(12345);

+function(a){

    console.log(a);   //firebug输出123456,使用+运算符

}(123456);

-function(a){

    console.log(a);   //firebug输出1234567,使用-运算符

}(1234567);

 var fn=function(a){

    console.log(a);   //firebug输出12345678，使用=运算符

}(12345678)

可以看到输出结果，在function前面加！、+、 -甚至是逗号等到都可以起到函数定义后立即执行的效果，而（）、！、+、-、=等运算符，都将函数声明转换成函数表达式，消除了javascript引擎识别函数表达式和函数声明的歧义，告诉javascript引擎这是一个函数表达式，不是函数声明，可以在后面加括号，并立即执行函数的代码。

加括号是最安全的做法，因为！、+、-等运算符还会和函数的返回值进行运算，有时造成不必要的麻烦。

不过这样的写法有什么用呢？

javascript中没用私有作用域的概念，如果在多人开发的项目上，你在全局或局部作用域中声明了一些变量，可能会被其他人不小心用同名的变量给覆盖掉，根据javascript函数作用域链的特性，可以使用这种技术可以模仿一个私有作用域，用匿名函数作为一个“容器”，“容器”内部可以访问外部的变量，而外部环境不能访问“容器”内部的变量，所以( function(){…} )()内部定义的变量不会和外部的变量发生冲突，俗称“匿名包裹器”或“命名空间”。

JQuery使用的就是这种方法，将JQuery代码包裹在( function (window,undefined){…jquery代码…} (window)中，在全局作用域中调用JQuery代码时，可以达到保护JQuery内部变量的作用。