贼猫柒柒

java script

**感 谢**

**此笔记观看了**

**我要自学网（吉延鹏）老师javascript课程、**

**网易云课堂(李炎恢) 老师jqueryui课程**

**特别感谢两位老师以及同学帮助，希望能帮助推广一下**

**内容纯属个人手工整理，有不足和错误之处还望指正！**

**zeimao77@qq.com**

目录

[第一章 函数： - 5 -](#_Toc486526636)

[第1节、 函数的三种声明方式及调用 - 5 -](#_Toc486526637)

[第2节、 函数的内部属性 - 6 -](#_Toc486526638)

[第3节、 函数的属性和方法 - 7 -](#_Toc486526639)

[第4节、 变量和作用域 - 8 -](#_Toc486526640)

[第5节、 全局函数 - 9 -](#_Toc486526641)

[第6节、 练习： - 10 -](#_Toc486526642)

[第二章 Js面向对象 - 11 -](#_Toc486526643)

[第1节、 基本模式 - 11 -](#_Toc486526644)

[第2节、 工厂模式 - 11 -](#_Toc486526645)

[第3节、 构造函数模式 - 11 -](#_Toc486526646)

[第4节、 原型模式 - 12 -](#_Toc486526647)

[第5节、 构造函数与原型组合模式 - 13 -](#_Toc486526648)

[第6节、 动态原型模式 - 13 -](#_Toc486526649)

[第7节、 继承 - 13 -](#_Toc486526650)

[第8节、 练习 - 15 -](#_Toc486526651)

[第三章 匿名函数与闭包 - 16 -](#_Toc486526652)

[第1节、 匿名函数 - 16 -](#_Toc486526653)

[第2节、 闭包 - 16 -](#_Toc486526654)

[第3节、 循环函数中的匿名函数与闭包 - 17 -](#_Toc486526655)

[第4节、 闭包中的this - 17 -](#_Toc486526656)

[第5节、 模仿块级作用域 - 18 -](#_Toc486526657)

[第6节、 私有变量 - 19 -](#_Toc486526658)

[第7节、 静态私有变量 - 19 -](#_Toc486526659)

[第8节、 练习 - 20 -](#_Toc486526660)

[第四章 Cookie - 22 -](#_Toc486526661)

[第1节、 认识Cookie - 22 -](#_Toc486526662)

[第2节、 Cookie的可选参数 - 22 -](#_Toc486526663)

[第五章 JSON - 24 -](#_Toc486526664)

[第1节、 JSON简介 - 24 -](#_Toc486526665)

[第2节、 JSON的解析 - 24 -](#_Toc486526666)

[第3节、 JSON创建对象 - 25 -](#_Toc486526667)

[第4节、 练习 - 26 -](#_Toc486526668)

[第六章 JavaJSON - 27 -](#_Toc486526669)

[第1节、 Java创建json方法 - 27 -](#_Toc486526670)

[第2节、 解析JSON - 27 -](#_Toc486526671)

[第3节、 GSON - 28 -](#_Toc486526672)

[第七章 AJAX - 30 -](#_Toc486526673)

[第1节、 AJAX的基本用法 - 30 -](#_Toc486526674)

[第2节、 Jquery ajax - 35 -](#_Toc486526675)

[第八章 正则表达式 - 38 -](#_Toc486526676)

[第1节、 认识正则表达式 - 38 -](#_Toc486526677)

[第2节、 正则表达式语法 - 38 -](#_Toc486526678)

[第3节、 正则表达式分组和多行匹配 - 40 -](#_Toc486526679)

[第4节、 正则表达式属性及方法 - 40 -](#_Toc486526680)

[第5节、 练习 - 42 -](#_Toc486526681)

[第九章 表格 - 44 -](#_Toc486526682)

[第1节、 Table对象集合 - 44 -](#_Toc486526683)

[第2节、 table对象的方法 - 44 -](#_Toc486526684)

[第3节、 Table对象的常用属性 - 45 -](#_Toc486526685)

[第4节、 练习 - 45 -](#_Toc486526686)

[第十章 表单 - 48 -](#_Toc486526687)

[第1节、 表单的4种访问 - 48 -](#_Toc486526688)

[第2节、 遍历表单的所有控件 - 48 -](#_Toc486526689)

[第3节、 Form对象的属性、方法、事件 - 48 -](#_Toc486526690)

[第4节、 通过name属性访问表单元素 - 50 -](#_Toc486526691)

[第5节、 禁用复制和粘贴 - 50 -](#_Toc486526692)

[第十一章 JQuery基础 - 51 -](#_Toc486526693)

[第1节、 JQuery语法 - 51 -](#_Toc486526694)

[第2节、 Document.ready和window.onload的区别 - 51 -](#_Toc486526695)

[第3节、 JQuery选择器 - 51 -](#_Toc486526696)

[第4节、 Jquery工具函数 - 54 -](#_Toc486526697)

[第十二章 JQuery DOM和CSS操作 - 55 -](#_Toc486526698)

[第1节、 文档内容的获取和设置 - 55 -](#_Toc486526699)

[第2节、 元素属性及相关操作。 - 55 -](#_Toc486526700)

[第3节、 CSS属性及相关操作 - 55 -](#_Toc486526701)

[第4节、 CSS类操作 - 55 -](#_Toc486526702)

[第5节、 元素的宽高 - 56 -](#_Toc486526703)

[第6节、 元素的位置 - 56 -](#_Toc486526704)

[第7节、 DOM操作 - 57 -](#_Toc486526705)

[第8节、 过滤方法 - 59 -](#_Toc486526706)

[第9节、 查找方法 - 59 -](#_Toc486526707)

[第10节、 练习 - 61 -](#_Toc486526708)

[第十三章 JQuery事件 - 61 -](#_Toc486526709)

[第1节、 鼠标事件 - 61 -](#_Toc486526710)

[第2节、 键盘事件 - 62 -](#_Toc486526711)

[第3节、 表单事件 - 62 -](#_Toc486526712)

[第4节、 浏览器事件 - 62 -](#_Toc486526713)

[第5节、 事件对象 - 63 -](#_Toc486526714)

[第6节、 事件处理 - 65 -](#_Toc486526715)

[第7节、 事件冒泡和默认事件 - 67 -](#_Toc486526716)

[第十四章 Jquery动画 - 68 -](#_Toc486526717)

[第1节、 显示、隐藏 - 68 -](#_Toc486526718)

[第2节、 滑动、卷动 - 69 -](#_Toc486526719)

[第3节、 淡入、淡出 - 70 -](#_Toc486526720)

[第4节、 自定义动画 - 71 -](#_Toc486526721)

[第5节、 动画相关 - 72 -](#_Toc486526722)

[第6节、 动画全局属性 - 73 -](#_Toc486526723)

[第十五章 Canvas专题 - 74 -](#_Toc486526724)

[第1节、 Canvas绘制直线 - 74 -](#_Toc486526725)

[第2节、 矩形的绘制： - 74 -](#_Toc486526726)

[第3节、 圆及弧的绘制 - 75 -](#_Toc486526727)

[第4节、 文字的绘制 - 76 -](#_Toc486526728)

[第5节、 渐变色 - 76 -](#_Toc486526729)

[第6节、 线条样式 - 76 -](#_Toc486526730)

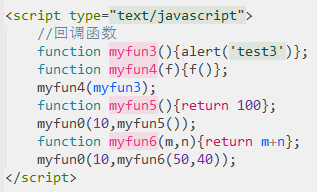
[第十六章 附件： - 78 -](#_Toc486526731)

# 函数：

## 函数的三种声明方式及调用



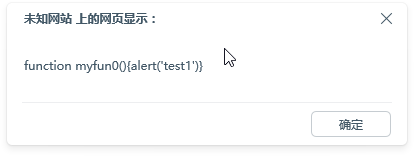
将函数作为参数（回调函数）;



将函数名作为参数将返回函数本身



返回结果如下：



## 函数的内部属性

在函数的内部，有两个特殊的对象，arguments和this

* Arguments

引用属性arguments.length检测函数的参数个数

函数的参数可以以arguments数组方式获取

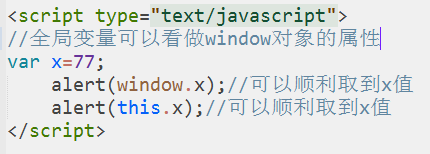
函数运行时不会验证传递给函数的参数个数是否等于定义函数的参数个数。最多可以接受255个参数，而不引发任何错误。任何遗漏的参数都会以undefined传递给函数，多余的函数将忽略。



* this

This是js语言的一个关键字，它代表函数运行时自动生成的一个内部对象，随着函数的使用场合不同，this的值会发生变化，但总是有一个总的原则，那就是this指调用函数的那个对象

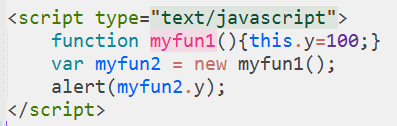
1. 在函数的外部使用this，this就指window对象



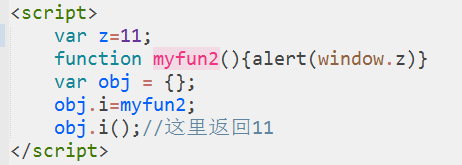
1. 函数内部调用



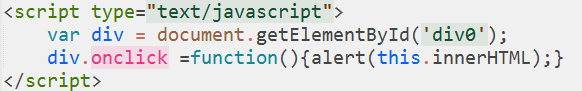
1. 用new来调用



1. 作为某个对象的方法调用，这时this就指这个上级对象



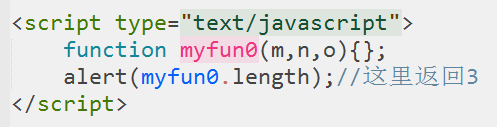
1. 事件监听函数中的this指触发事件的元素；



## 函数的属性和方法

每一个函数都包含两个属性：length和prototype

Length：当前函数希望接受的命名参数的个数；【指形参；与arguments指实际传入的实参】



每一个函数都包含两个非继承而来的方法apply()和call();

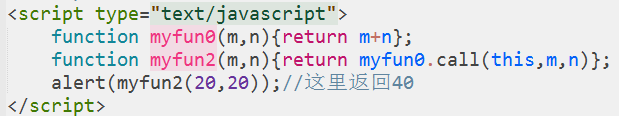
Apply()和call()

这两个方法都是在特定的作用域中调用函数，实际上等于设置函数内this对象的值；

Apply()接受两个参数：一个是函数运行的作用域，另一个参数数组（可以是数组实例也可以是arguments对象）



Call方法等于第一个参数没有变化，变化的是其余的参数都是传递参数，传递给函数的参数需要逐个列举出来



Apply和call通常用来扩充函数运行的作用域而非单纯的传递参数；两者的用法一模一样，只是在传递参数的时候不同



## 变量和作用域

#### 基本类型和引用类型：

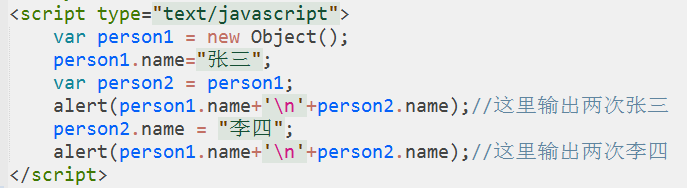
基本类型值有：undefined,NULL,Bollean,Number和String; 这些类型分别在内在中占有固定大小的空间；它们的值保存在栈中。

引用类型

#### 引用类型有：对象、数组、函数

引用类型在内存中占用的空间不固定

在变量赋值的时候，基本数据类型赋值的是值本身，而引用类型是其地址;



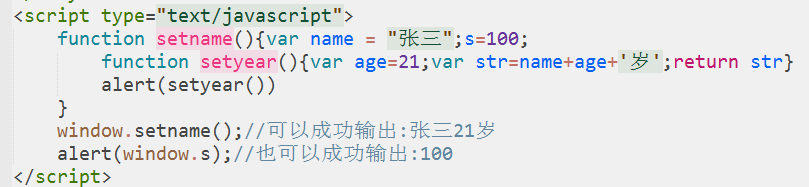
函数的参数都是按值来传递的；



#### 执行环境及作用域

环境是最外围的执行环境，在web浏览器中,全局执行环境是window对象，因此，所有的全局变量的函数都是作为window的属性和方法创建的。

变量没有在函数内声明或者声明的时候没有写var就是全局变量，拥有全局作用域，window对象的所有属性拥有全局全用域；在代码任何地方都可以访问，函数内部声明且以var修饰的变量就是局部变量，只可以在函数体内部。



内部环境可以访问所有外部环境，但外部环境不可以访问内部环境中的任何变量和函数。

在变量的查询中，访问局部变量要比访问全局变量快。

## 全局函数

#### 内存管理

内在的生周期：

1. 分配内存：当我们申明变量、函数、对象的时候，系统会自动为他们分配内存
2. 内存使用：即读写内存，也就是使用变量或函数
3. 内存回收：使用完毕，由垃圾回收自动回收不再使用的内存

垃圾回收算法：核心在于如何判断内存已经不再使用了。

JS的内存注意事项：

1. 避免不必要的定义全局变量

当一个变量定义在全局作域中，默认情况下JS引擎不会回收销毁，如此变量就会一直存在于内存中，直到页面被被关闭。

1. 及时解除不再使用的变量引用，即将其赋值为NULL;

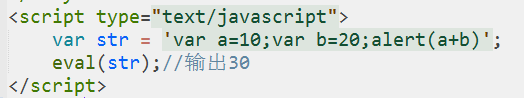
在内在回收周期内，回收器每隔一段时间检查一次，并将其回收

1. 合理使用函数，函数中的局部变量函数执行结束后就会自动释放内存。

#### 全局函数

这里指系统定义好的全局函数

1. ParseInt()函数可以解析一个字符串，并返回一个整数
2. parseFloat()函数可以解析一个字符串，并返回一个浮点数
3. isNaN()函数用于检查其参数是否非数字值
4. String()函数把对象的值转换成字符串
5. Number()把对象的值转换为数字
6. Eval()函数可以计算某个字符串，并执行其中的JS代码



1. Escape()对字符串进行编码

返回值：已编码的string的副本。其中某些字被替换成十六进制的转义序列;该方法不会对ASCII字母和数字进行编码，也不会对下面这些ASCII标点符号进行编码：\* @ \_ - + . / 。其它所有的字符都会被转义序列替换。

1. isFinite()检查某个值是否为有穷大的数

如果number是有限数字，或可转为有限数字，那么返回true,如果number为NaN(非数字)或者正负无穷数，返回false;

1. unescape()对由escape()编码的字符串进行解码。
2. encodeURI()把字符串编码为URI（Uniform Resource Identifier,统一资源标识符）。
3. decodeURI()解码某个编码的URI
4. decodeURIComponent()解码一个编码的URI组件。
5. encodeURIComponent()把字符串编码为URI组件

三种不同编码的区别

Escape不编码字符有69个：\* + - . / @ \_ 0-9 a-z A-Z

主要用于防止特殊字符造成计算错误时候应用

EncodeURI不编码的字符有82个： ! # $ & ‘ ( ) \* + - , / : ；= ？ @ \_ ` 0-9 A-Z a-z

防止特殊字符串造成URI的传递错误，一般用于页面跳转的时候

EncodeURIComponent不编码的字符有71个： ! ’ ( ) \* - . \_ ` 0-9 a-z A-Z

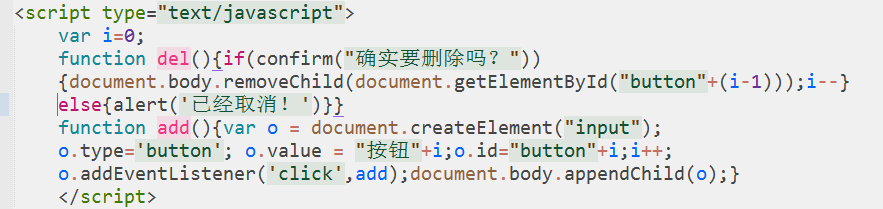
防止URI参数中特殊字符串造成参数读取错误，一般用于传递参数

## 练习：

#### 删除确认提示框，动态添加元素；

知识点：confirm()用于显示一个带有指定消息和OK及取消按钮的对话框

用户点击确定返回true 点击取消返回false



#### 防止用户连击

重点在于判断页面的状态

readState属性返回当前文档的状态，该属性返回以下值：

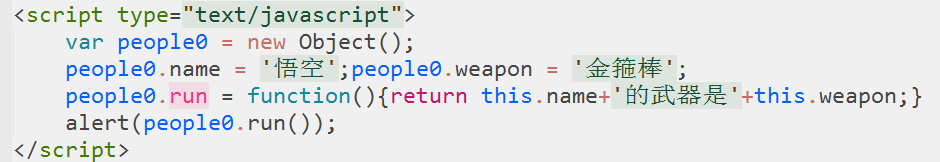
* uninitialized 还未开始载入
* loading载入中
* interactive已加载，文档与用户开始交互
* complete载入完成



# Js面向对象

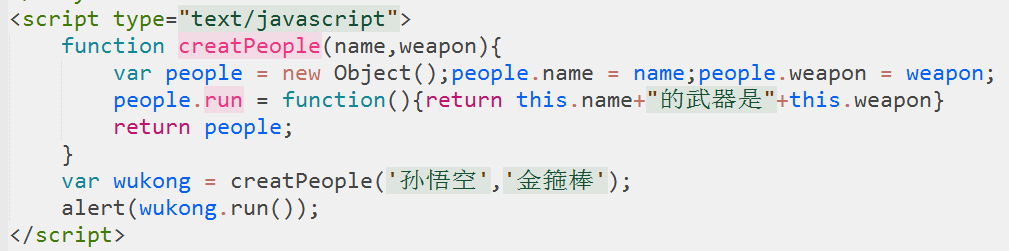
## 基本模式

最普通的创建对象方法,缺陷：如果创建多个对象会比较繁琐，效率低；实例与原型之间没有任何关系。



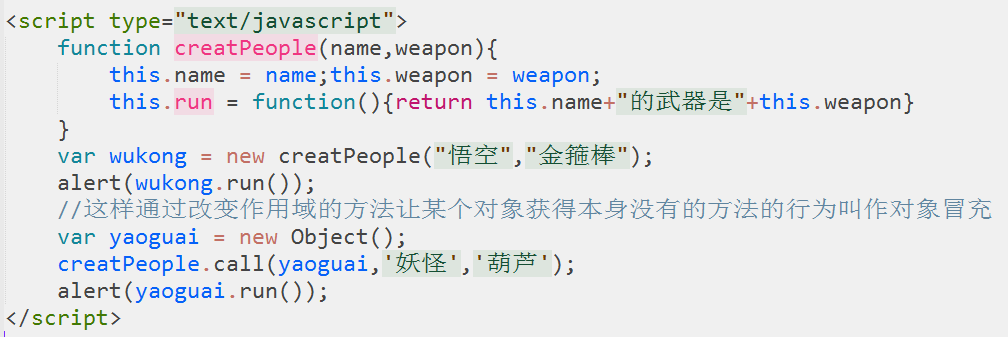
## 工厂模式

使用创建并返回特定类型的对象的工厂函数；虽然一定程度上解决了创建多个对象的麻烦，但创建出的实例之间没有内在的联系，不能反映出它们是同一个原型对象的实例，创建的时候不使用new关键字，每生成一个实例，都会增加一个重复的内容，会多占用一些内存资源。



## 构造函数模式

1. New调用的函数为构造函数，构造函数和普通函数区别仅仅在于是否使用了new来调用
2. 所谓构造函数，就是专门用来生成”对象”的函数，它提供了模板，作为对象的基本结构
3. 构造函数内部使用this变量。对构造函数使用new运算符，就能生成实例，并且this变量会绑定在实例对象上。
4. Instanceof验证原型对象与实例之间的关系
5. Call和apply方法的使用
6. 使用构造函数每生成一个对象实例，都会增加一个重复的内容，占用内存，这样缺乏效率，也造成了内存浪费。

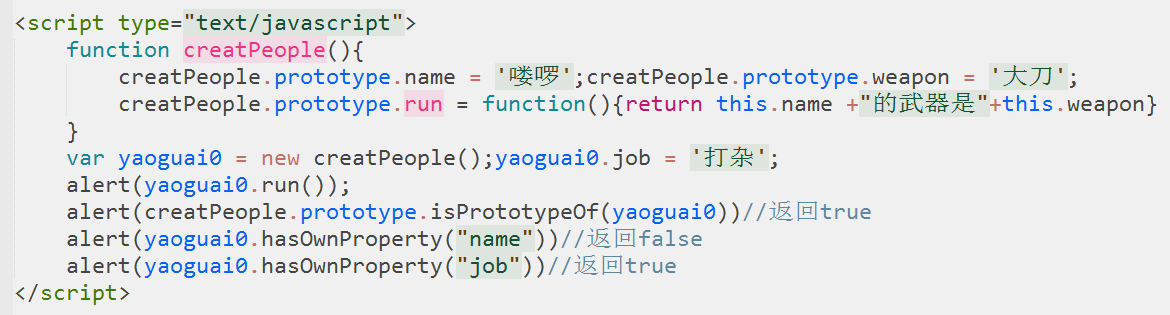


## 原型模式

JS规定，每一个构造函数都有一个prototype属性，指向另一个对象。这个对象的所有属性和方法都会被构造函数的实例所继承，可以把那些不变的属性和方法，直接定义在prototype对象上

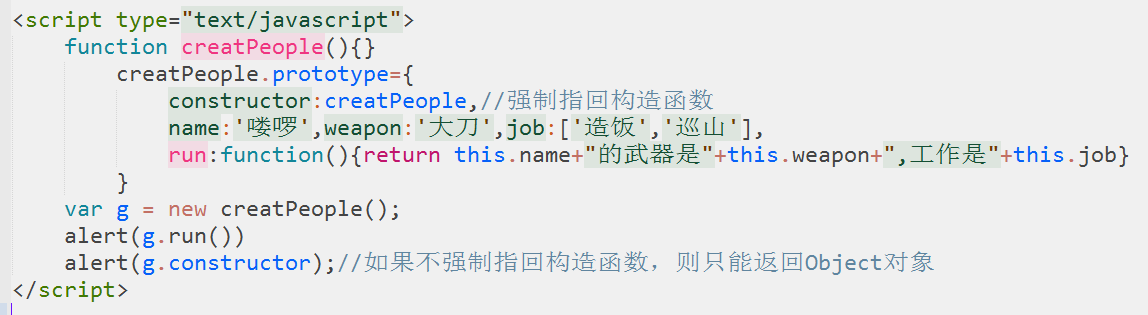
#### 重要方法

1. Prototype方式定义的方式，函数不会拷贝到每一个实例中，所有的实例共享protoytpe中的定义，节省了内存
2. isPrototypeOf()这个方法用来判断某个proptotype对象和某个实例之间的关系
3. hasOwnProperty()每一个实例对象都有一个hasOwnProperty()方法用来判断某一个属性到底是本地属性还是继承prototype对象的属性，本地属性返回true;





1. in运算符可以用来判断某个实例是否含有某个属性，不管是不是本地属性。In运算符还可以用来遍历某个对象的所有属性
2. 对象的constructor属性用于返回创建该对象的构造函数

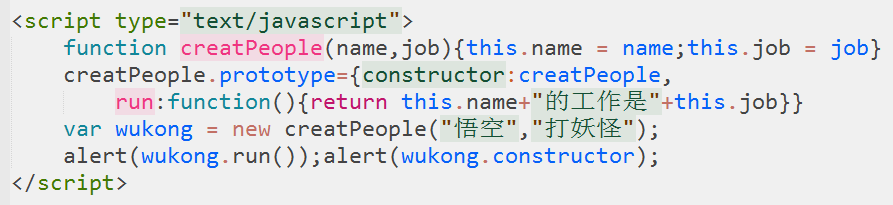


1. 原型模式构造函数没有参数，使用原型方式，不能通过给构造函数传递参数来初始化实例对象。
2. 属性指向的是对象，而不是函数时，函数共享不会造成问题，但对象却很少被多个实例共享，如果共享的是对象就会造成问题（引用类型）;

## 构造函数与原型组合模式

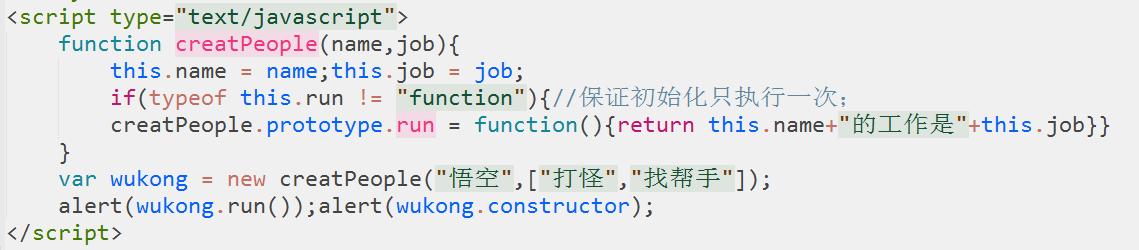
是目前最为常用的创建对象的方式

这种概念非常简单，就是用构造函数定义对象的所有非函数属性，用原型方式定义对象的函数属性（方法）。结果是，所有函数都只创建一次，而每个对象都具有自己的对象属性实例。它支持构造函数传递参数，JQuery类型的封装就是使用组合模式来实现的。



## 动态原型模式

解决了组合模式的封装问题



## 继承

如果两者一旦确实了继承关系，就包含以下三个意思：

1. 子类的实例可以共享父类的方法
2. 子类可以覆盖或扩展父类的方法
3. 子类和父类都是子类实例的类型

#### 继承的实现：

实现继承的方式不止一种，这是因为JS中的继承机制并不是明确规定的，而是通过模仿实现的，以下介绍常用的几种实现方式;

* 对象冒充（构造函数绑定）

原理：使用对象冒充（call或apply方法）(实质上是改变了this的指针的指向)来继承基类。



* 原型链

这种方法更常见，使用prototype属性

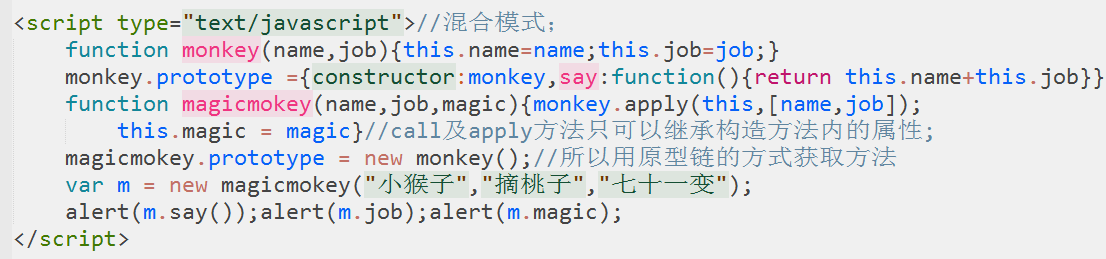
Prototype对象是一个模板，要实例化的对象都以这个模板为基础，总而言之，prototype对象任何属性和方法都被传递给那个类的所有实例，原型链利用这种功能来实现继承机制。

原型链的弊端在于它不支持多重继承，原型链会用另一类型的对象重写类prototype属性。

子类的所有属性和方法都必须在prototype属性被赋值后，因为在它之前的所有方法都会被删除。因为prototype属性被替换成了新的对象，添加了新的方法原始方法将会被销毁。

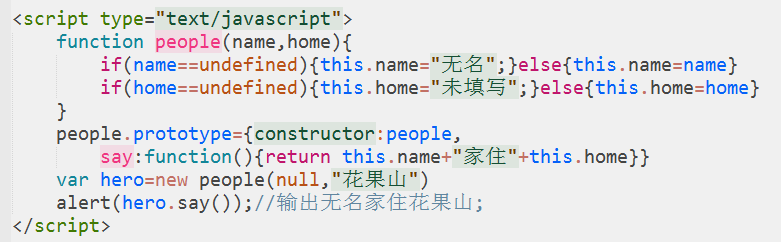


* 混合模式

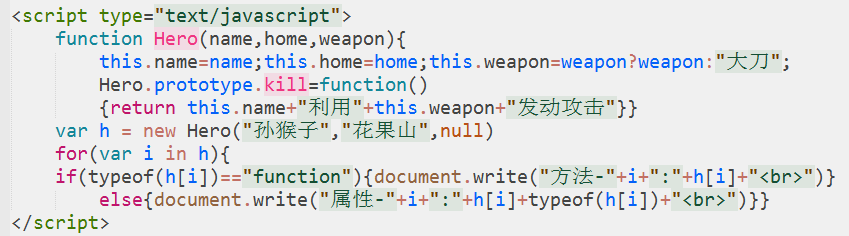


## 练习

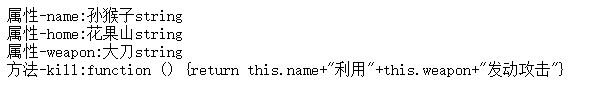
#### 具有默认值的构造函数



#### 遍历对象的属性和方法



执行结果：



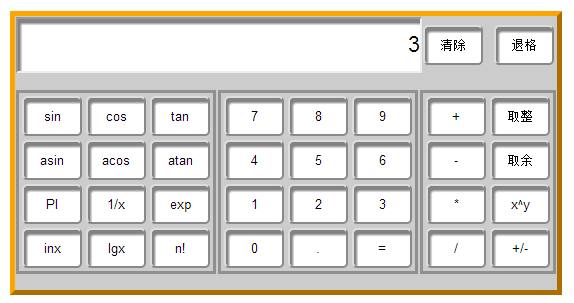
#### 属性及方法的添加与删除

Delete user.name 可以删除user的name属性

Delete user.skill 可以删除user的skill方法

User.set=’男’; 可以为元素添加属性

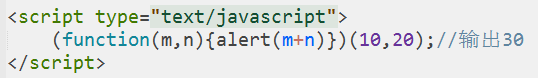
#### 计算器



# 匿名函数与闭包

## 匿名函数

* 单独的匿名函数是无法运行和调用的
* 可以把匿名函数赋值给变量
* 通过表达式自我执行：语法：(匿名函数名)()
* 匿名函数传参语法：(匿名函数名)(参数列表)



## 闭包

闭包相关概念：

闭包（closure）是指有权访问另一个函数作用域中变量的函数

本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。内层的函数可以使用外层函数的所有变量，即使外层函数已经执行完毕。

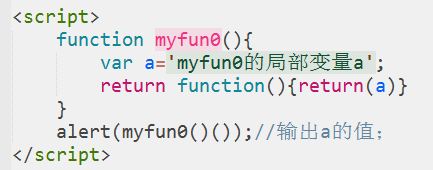
这是JS中非常重要的部分知识，因为使用毕包可以大大减少我们的代码量，使我们的代码看上去更清晰。

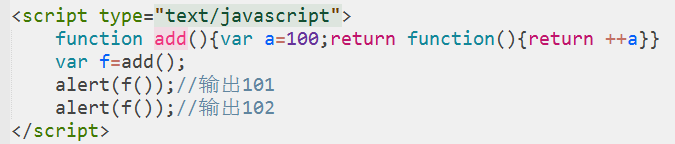
闭包相关知识点：

1. 常见的方式是在函数的内部创建另一个函数
2. 闭包的第一个用途：通过闭包可以访问局部变量
3. 闭包的第二个用途：可以让这些变量的值始终保持在内在中
4. 优点：可以把局部变量驻留在内存中，可以避免使用全局变量

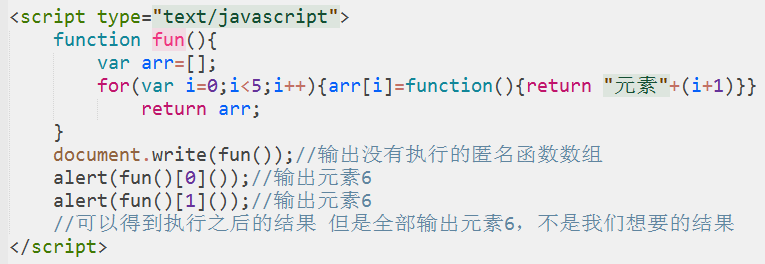
[全局变量在复杂的程序中会造成许多麻烦（比如命名冲突，垃圾回收等）,所以推荐使用私有的，封闭的局部变量。而闭包可以实现这一点]

1. 缺点：由于闭包作用域返回的局部变量资源不会被立刻销毁回收，所以可能会占用更多的内存；所以过渡使用闭包会导致性能下降。



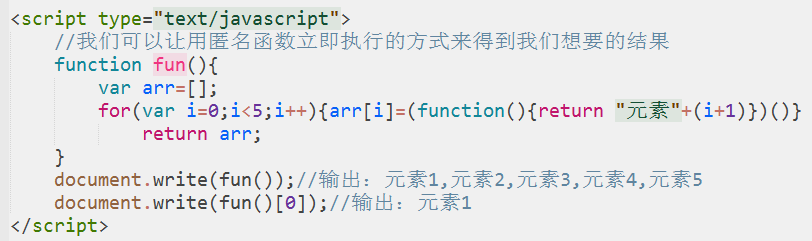


## 循环函数中的匿名函数与闭包



返回一个包函五个匿名函数的数组；结果如下

function (){return "元素"+(i+1)},function (){return "元素"+(i+1)},function (){return "元素"+(i+1)},function (){return "元素"+(i+1)},function (){return "元素"+(i+1)}



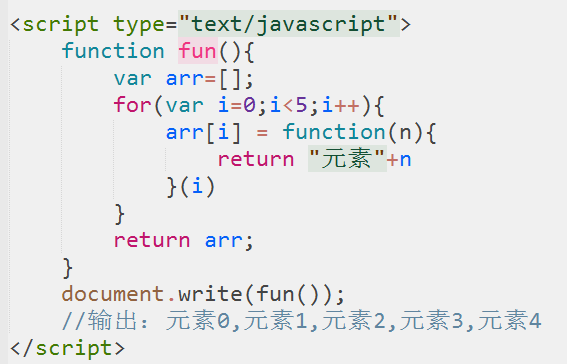
这里的匿名函数有一个参数n，也就是最终将返回的结果数值；

在调用每个匿名函数的时候传入变量i

变量i的当前值会赋值给n

匿名函数内部创建并返回了一个访问的闭包

如此数组arr中的每一个函数中都有了自己的N变量的一个副本（闭包将局部变量贮存在内存中）；



## 闭包中的this

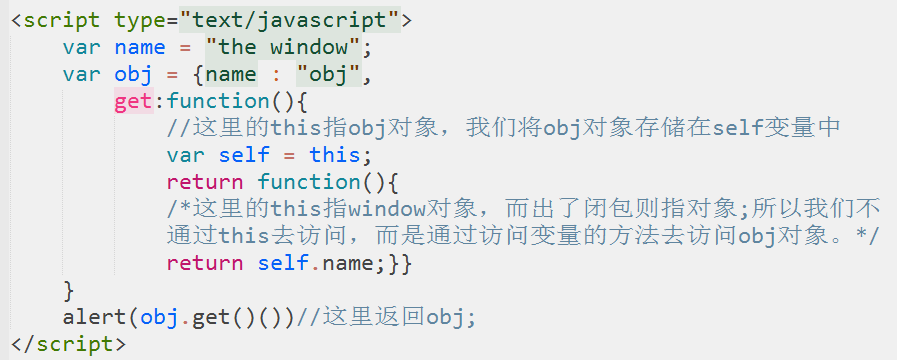
全局函数中的this就是window，而当函数作为某个对象的方法调用时,this就是指的那个对象。

匿名函数的执行环境具有全局性，this通常是指向window的。



可以使用对象冒充强制改变this的指向

将this赋值给一个变量，闭包访问这个变量



## 模仿块级作用域

块级作用域又叫作私有作用域，但是JS没有块级作用域的概念，这意味着在块语句（比如for语句）中定义的变量，不会因为离开了for块就失效了。

使用了块级作用域之后，匿名函数中定义的任何变量，都会在执行结束时被销毁。

一般来说，我们都应该尽量可能少向全作用域中添加变量和函数；过多的全局变量和函数很容易导致命名冲突。

使用块级使用域，每一个开发者既可以使用自己的变量，又不必担心搞乱了全局作用域。





## 私有变量

JS没有没有私有属性；所有的属性都是公有的；

私有变量的概念：在任何函数中定义的变量，都是私有变量，因为不能在函数外部访问这些变量；

私有变量：包括函数的参数/局部变量和在函数内部定义的其它函数；

特权方法：内部创建一个闭包，闭包可以访问私有变量；因此创建用于访问私有变量的公用方法，称为特权方法

这种方法的缺点是会为每一个实例创建一组新的方法，不能实现共享。

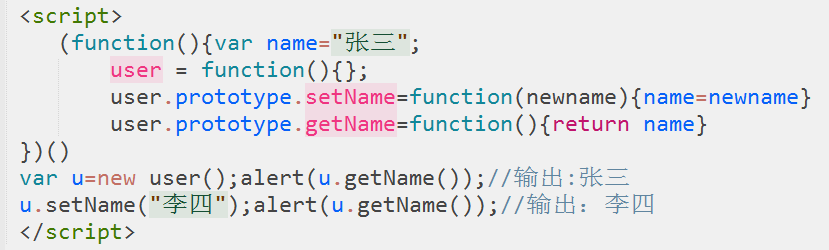




## 静态私有变量

通过块级作用域（私有作用域）中定义私有变量或函数，创建对外公共的特权方法；

* 首先创建私有作用域
* 定义私有变量或函数
* 定义构造函数和特权方法
* 这种方式创建的私有变量因为使用原型而实现共享
* 同时由于共享，实例也就没有自己的变有变量



## 练习

#### 实例1：让浏览器弹出新窗口，并关闭

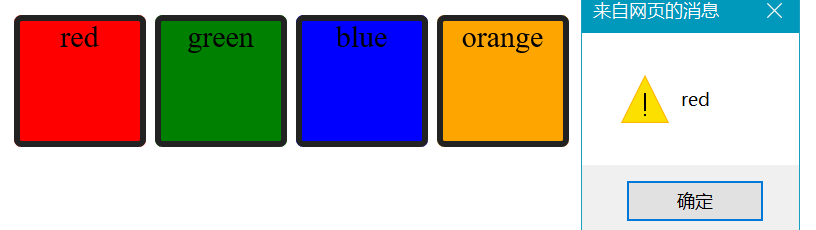
Window.open(URL,name,features,replace)方法

SetTimeout的使用



#### 实例2：绑定数组弹出颜色

掌握匿名函数与闭包的相关写法；



#### 实例3:统计各个按钮的点击次数

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<style>

input{

height: 30px;

width: 60px;

border-radius: 5px

}

</style>

<body>

<input type="button" onclick="countA()" value="a">

<input type="button" onclick="countB()" value="b">

<input type="button" onclick="countC()" value="c">

</body>

<script type="text/javascript">

function count(){

var count =0;

function add(){

count++;

alert("您共点击我"+count+"次！")

}

return add;

}

var countA =new count();

var countC =new count();

var countB =new count();

</script>

</html>

# Cookie

## 认识Cookie

#### 什么是Cookie：

Cookie是存储于访问者的计算机中的变量，每当同一台计算机通过浏览器请求某个页面时，就会发送这个cookie。当用户下次访问同一个页面时，服务器会先查看有没有上传留下的cookie资料，如果有就根据cookie里的资料判断访问者，改善特定的页面内容。

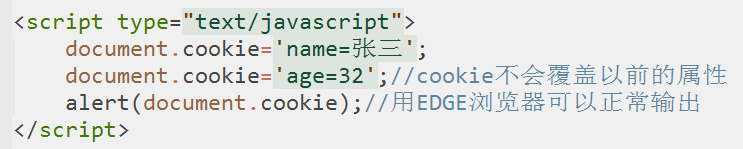
应用场景：自动登录，记住用户名；

最初是用来在客户端和服务器进行通信使用的，所以原则上应该在服务器运行环境下进行。目前大部分浏览器可以在客户端生成和读取cookie数据

#### 创建cookie

* 格式

将document下的cookie属性设置为如下格式的字符串：name=value



#### Cookie的编码与解码

使用encodeURIComponent()进行编码

使用decodeURIComponent()进行解码

Cookie值不能含有分号，逗号和空白符号



## Cookie的可选参数

* Expires=时间：过期时间

默认值为浏览器关闭后过期（即会话结束后）

如果将expires设置为过去的时间可以删除cookie



* path

他指定了cookie关联在一起的网页。默认值是和当前网页同一目录的网页中有效。如果设置path为“/”，那么它对该网站的所有网页可见。

* domain

设定cookie的有效域名，一般使用默认值，即绑定当前域名，本地测试无效。

* Secure

指定了网络上如何传输cookie。默认为普通http协议传输；若设置为安全的，将只能通过https安全协议才可以传输。

语法：document.cookie='name=张三;secure";

# JSON

## JSON简介

#### 什么是JSON？

JSON是存储和交换文本信息的语法，类似XML

JSON比XML更小、更快、更易解析。

JSON是轻量级的文本数据交换格式

JSON独立于语言

JSON使用JAVASCRIPT语法来描述数据对象，但JSON仍独立于语言和平台。

JSON具有自我描述性，更易理解

JSON是在AJAX中代码XML交换数据的最佳方案

#### JSON语法

JSON语法是JS对象表示语法的子集

JSON语法规则：

数据在名称/值对中

数据由逗号分隔

花括号保存对象

方括号保存数组

JSON的值可以是：

数字（整数或浮点）

字符串（在双引号中）

逻辑值（true或false）

数组（在方括号中）

对象（在花括号中）

Null



## JSON的解析

一般情况下，我们的JSON数据都是从服务器端获取到的。获取的JSON数据是以字符串的形式返回的，这个字符串是JSON格式的，但是不能被直接使用，我们必须将该字符串转化为一个对象才能正常解析它

JavaScript函数eval()可以用于将JSON文本转换为JS对象。

Eval()函数可编译并执行JS代码，这隐藏了一个潜在的安全问题（如果JSON包含恶意代码也将被执行）。

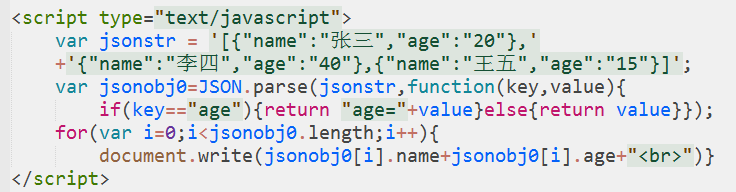
使用JSON解析器将JSON转换为JS对象是更安全的做法。JSON解析器只会识别JSON文本，而不会执行。

JSON的解析：JSON数据转换成JS对象。

语法：var myobject = JSON.parse(MYJSONTEXT,reviver);

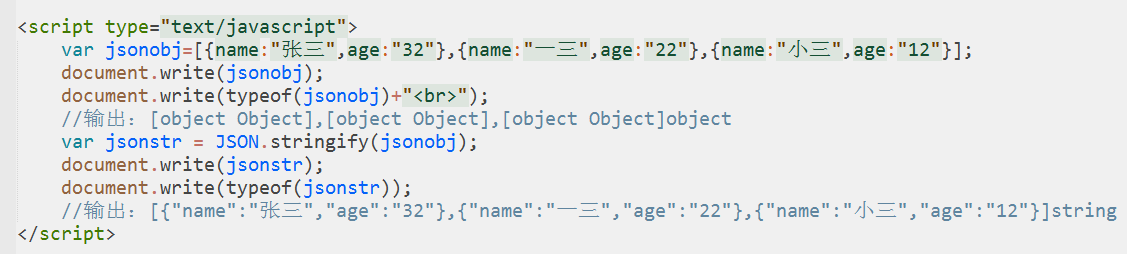


第二个参数



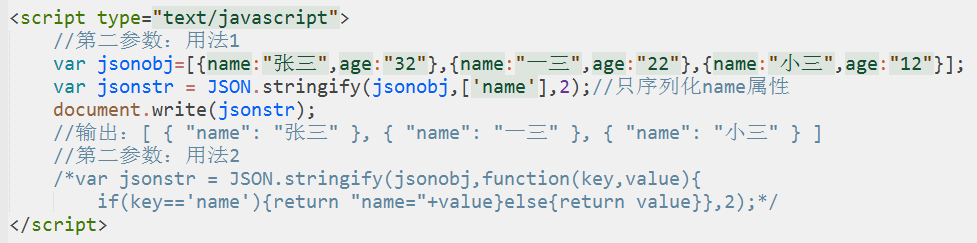
JSON的序列化：将JS对象转换成JSON数据

语法：var MYJSONTEXT = JSON.stringify(myObject,replacer)



第二参数与解析使用一样；

还有第三参数为一个数组，指定哪些项需要被序列化；

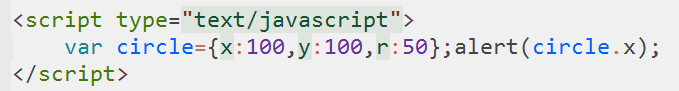


## JSON创建对象

优点：语法简单

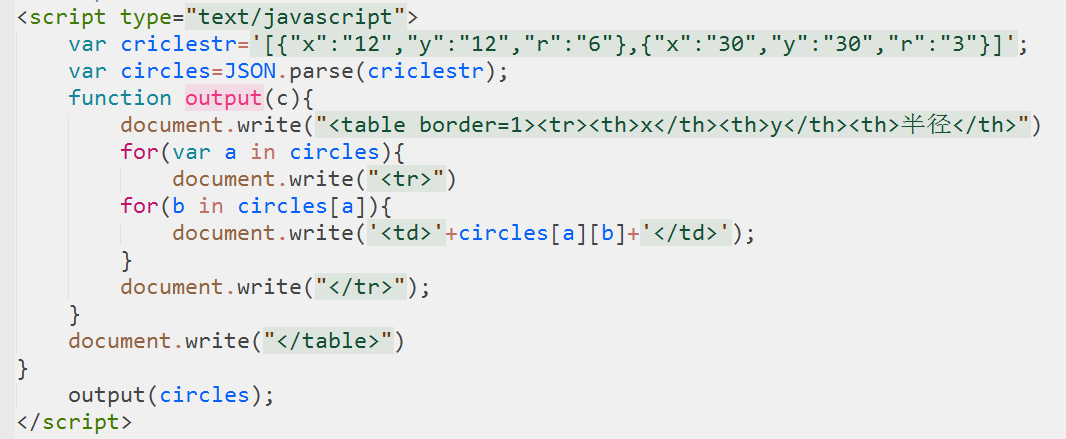
缺点：不适用多个对象创建、

命名空间：如果多个人开发同一个项目，可能发生命名冲突。



## 练习

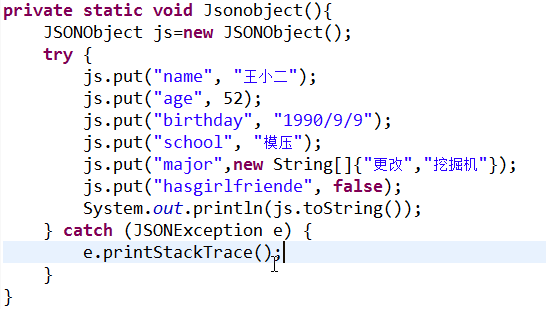
#### 数据读取，输出到页面

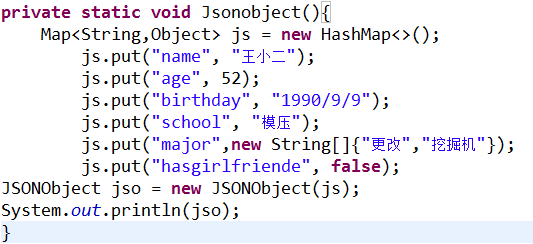


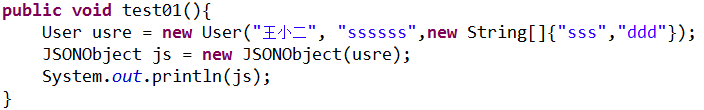
# JavaJSON

JSON是一种与开发语言无关的、轻量级的数据格式。全称JavaScript Object Notation

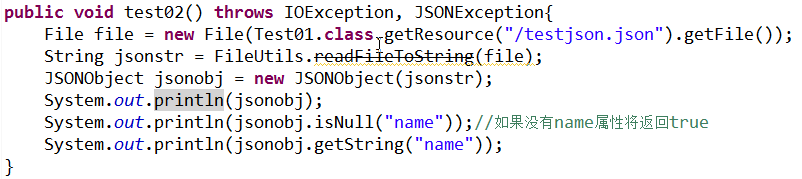
## Java创建json方法



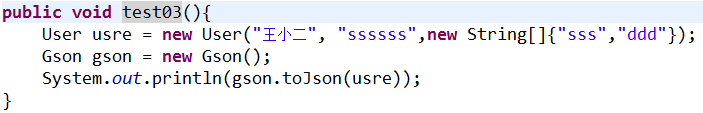




## 解析JSON



## GSON



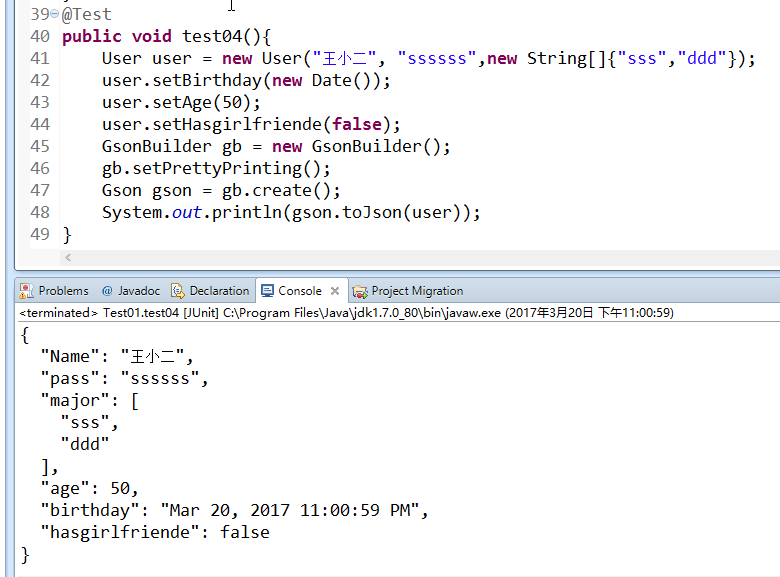
@SerializedName(“别名”)

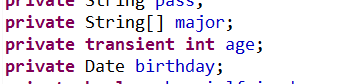
此注解可以将对象转JSON对象的时候改变其属性名

此注解在反转的时候也起着相同的作用



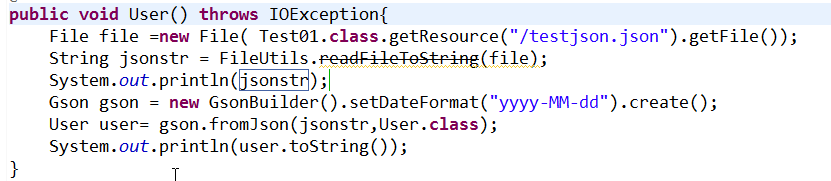
Builder格式化JSON打印样式

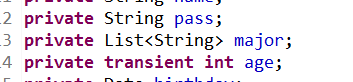




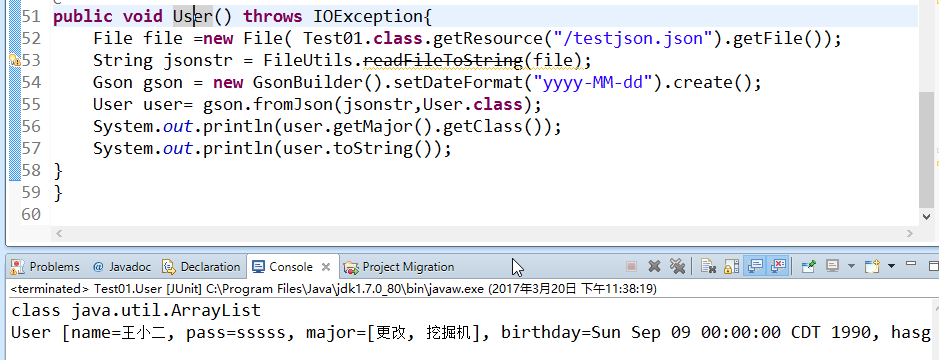
用transient修饰某个属性可以让gson在转json时不显示此属性

#### GSON中的日期转换特别方便





如果我们将属性中的对象改为LIST、set等集合类GSON也可以自动地帮我们转换成我们需要的类型



# AJAX

## AJAX的基本用法

#### 什么是AJAX

AJAX=Asynchronous JavaScript and XML(异步的JavaScript 和XML).

AJAX可以通过在后台与服务器进行少量数据交换，使网页实现异步更新，这意味着可以在不重新加载整个网页的情况下，对网页进行某部分更新。

#### AJAX的基本用法

* 创建XMLHttpRequest对象

语法：var myAjax = new XMLHttpRequest();

* 向服务器发送请求：使用open()和send()方法

Open(method,url,async)：规定请求的类型、URL以及是否异步处理请求。

Method：请求的类型：get 或post;

[url:文件在服务器上的位置](file:///F:\bj\文件在服务器上的位置)

sync:true(异步)或false(同步)

send(string)：string:仅用于post请求；

* 服务器响应

如需获得来自服务器的响应，请使用XMLHttpRequest对象的responseText或者responseXML属性。

responseText属性：responseText属性返回字符串形式的响应。

responseXML属性：如果来自服务器的响应是XML，而且需要作为XML对象进行解析，请使用responseXML属性。

* Onreadystatechange事件

当请求被被发送到服务器时，我们需要执行一些基于响应的任务。每当readyState改变时，就会触发onreadystatechange事件。

XMLHttpRequest对象的三个重要属性:

* Onreadystatechange：存储函数（或函数名），每当readyState属性改变时，就会调用该函数。
* readyState：存在XMLHttpRequest的状态。从1到4发生变化。

0：请求未初始化。

1：服务器连接已建立。

2：请求已接收

3：请求处理中

4：请求已完成，且响应已就绪。

* Status:200：”OK”/404:未找到页面
* Ajax xmlHttpRequest的status的值的含义  
  1xx - 信息提示   
  100 - 初始的请求已经接受，客户应当继续发送请求的其余部分。（HTTP 1.1新）   
  101 - 服务器将遵从客户的请求转换到另外一种协议。（HTTP 1.1新）   
  2xx - 成功   
  200 - OK 一切正常，对GET和POST请求的应答文档跟在后面。   
  201 - Created 服务器已经创建了文档，Location头给出了它的URL。   
  202 - Accepted 已经接受请求，但处理尚未完成。   
  203 - Non-Authoritative Information 文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝，非权威性信息（HTTP 1.1新）。   
  204 - No Content 没有新文档，浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的。   
  205 - Reset Content 没有新的内容，但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容（HTTP 1.1新）。   
  206 - Partial Content 客户发送了一个带有Range头的GET请求，服务器完成了它（HTTP 1.1新）。   
  3xx - 重定向   
  300 - Multiple Choices 客户请求的文档可以在多个位置找到，这些位置已经在返回的文档内列出。如果服务器要提出优先选择，则应该在Location应答头指明。   
  301 - Moved Permanently 客户请求的文档在其他地方，新的URL在Location头中给出，浏览器应该自动地访问新的URL。   
  302 - Found 类似于301，但新的URL应该被视为临时性的替代，而不是永久性的。注意，在HTTP1.0中对应的状态信息是“Moved Temporatily”。出现该状态代码时，浏览器能够自动访问新的URL，因此它是一个很有用的状态代码。注意这个状态代码有时候可以和301替换使用。例如，如果浏览器错误地请求 http://host/~user （缺少了后面的斜杠），有的服务器返回301，有的则返回302。严格地说，我们只能假定只有当原来的请求是GET时浏览器才会自动重定向。请参见307。   
  303 - See Other 类似于301/302，不同之处在于，如果原来的请求是POST，Location头指定的重定向目标文档应该通过GET提取（HTTP 1.1新）。   
  304 - Not Modified 客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。   
  305 - Use Proxy 客户请求的文档应该通过Location头所指明的代理服务器提取（HTTP 1.1新）。   
  307 - Temporary Redirect 和302（Found）相同。许多浏览器会错误地响应302应答进行重定向，即使原来的请求是POST，即使它实际上只能在POST请求的应答是303时才能重定向。由于这个原因，HTTP 1.1新增了307，以便更加清除地区分几个状态代码：当出现303应答时，浏览器可以跟随重定向的GET和POST请求；如果是307应答，则浏览器只能跟随对GET请求的重定向。（HTTP 1.1新）   
  4xx - 客户端错误   
  400 - Bad Request 请求出现语法错误。   
  401 - Unauthorized 访问被拒绝，客户试图未经授权访问受密码保护的页面。应答中会包含一个WWW-Authenticate头，浏览器据此显示用户名字/密码对话框，然后在填写合适的Authorization头后再次发出请求。   
  404 - Not Found 无法找到指定位置的资源。这也是一个常用的应答。   
  405 - Method Not Allowed 请求方法（GET、POST、HEAD、DELETE、PUT、TRACE等）对指定的资源不适用，用来访问本页面的 HTTP 谓词不被允许（方法不被允许）（HTTP 1.1新）   
  406 - Not Acceptable 指定的资源已经找到，但它的MIME类型和客户在Accpet头中所指定的不兼容，客户端浏览器不接受所请求页面的 MIME 类型（HTTP 1.1新）。   
  407 - Proxy Authentication Required 要求进行代理身份验证，类似于401，表示客户必须先经过代理服务器的授权。（HTTP 1.1新）   
  408 - Request Timeout 在服务器许可的等待时间内，客户一直没有发出任何请求。客户可以在以后重复同一请求。（HTTP 1.1新）   
  409 - Conflict 通常和PUT请求有关。由于请求和资源的当前状态相冲突，因此请求不能成功。（HTTP 1.1新）   
  410 - Gone 所请求的文档已经不再可用，而且服务器不知道应该重定向到哪一个地址。它和404的不同在于，返回407表示文档永久地离开了指定的位置，而404表示由于未知的原因文档不可用。（HTTP 1.1新）   
  411 - Length Required 服务器不能处理请求，除非客户发送一个Content-Length头。（HTTP 1.1新）   
  412 - Precondition Failed 请求头中指定的一些前提条件失败（HTTP 1.1新）。   
  413 – Request Entity Too Large 目标文档的大小超过服务器当前愿意处理的大小。如果服务器认为自己能够稍后再处理该请求，则应该提供一个Retry-After头（HTTP 1.1新）。   
  414 - Request URI Too Long URI太长（HTTP 1.1新）。   
  415 – 不支持的媒体类型。   
  416 – Requested Range Not Satisfiable 服务器不能满足客户在请求中指定的Range头。（HTTP 1.1新）   
  417 – 执行失败。   
  423 – 锁定的错误。   
  5xx - 服务器错误   
  500 - Internal Server Error 服务器遇到了意料不到的情况，不能完成客户的请求。   
  501 - Not Implemented 服务器不支持实现请求所需要的功能，页眉值指定了未实现的配置。   
  502 - Bad Gateway 服务器作为网关或者代理时，为了完成请求访问下一个服务器，但该服务器返回了非法的应答。 亦说Web 服务器用作网关或代理服务器时收到了无效响应。   
  503 - Service Unavailable 服务不可用，服务器由于维护或者负载过重未能应答。例如，Servlet可能在数据库连接池已满的情况下返回503。服务器返回503时可以提供一个Retry-After头。这个错误代码为 IIS 6.0 所专用。   
  504 - Gateway Timeout 网关超时，由作为代理或网关的服务器使用，表示不能及时地从远程服务器获得应答。（HTTP 1.1新） 。   
  505 - HTTP Version Not Supported 服务器不支持请求中所指明的HTTP版本。（HTTP 1.1新）。

流程：



#### 练习：访问实例：

Test文本内容：

[{"x":30,"y":30,"z":10},{"x":40,"y":40,"z":20},{"x":50,"y":50,"z":10},{"x":60,"y":60,"z":10}]

ajax.js AJAX函数封装

function myjson(url,funsuss,funFaild) {

var xhr = new XMLHttpRequest();

url = url+'?'+new Date().getTime();

xhr.open('get', url, true);

xhr.send(null);

xhr.onreadystatechange = function() {

if (xhr.readyState == 4) {

if (xhr.status == 200) {

jsonstr = xhr.responseText;

funsuss(xhr.responseText);

} else {

if (funFaild != null) {

funFaild(xhr.statusText)

} else {

alert(xhr.statusText);

}

}

}

}

}

HTML调用：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<body>

</body>

<script type="text/javascript" src="ajax.js"></script>

<script type="text/javascript">

myjson("test.txt",output,null)

function output(str){

document.write("<table border=1><tr><th>横坐标</th><th>纵坐标</th><th>半径</th></tr>")

var jsonobj = JSON.parse(str);

for(var a in jsonobj){

document.write("<tr>")

for(var b in jsonobj[a]){

document.write("<td>"+jsonobj[a][b]+"</td>")

}

document.write("</tr>")

}

document.write("</table>")

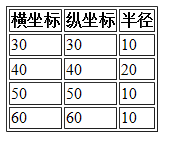
}

</script>

}

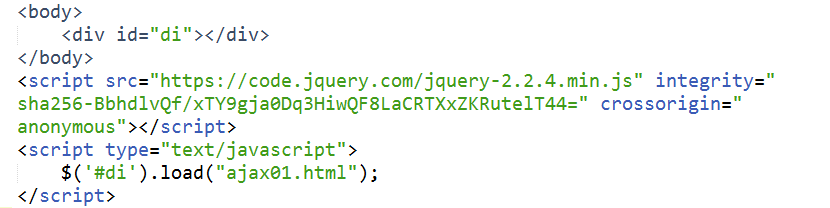
</html>

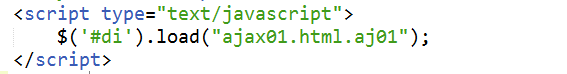
输出：



## Jquery ajax

#### 局部方法：Load()

可以为地址添加参数与过滤请求内容。例：



有三个参数可选，其中URL是必需，如果POST请求可以接对象，第三个参数可以写回调函数。Load方法一般用来请求静态文件



#### 全局方法

$.post()与$.get()方法



可以传递参数：

第一个参数为URL

第二个参数为参数，一般用JSON对象的形式

第三个参数为回调函数

第四个参数为预期返回类型，如果返回类型不能正确匹配，则不能正确获取数据。

如果默认请求已经是xml格式，强行设置为html，则会连同标签也一起返回

除此之外，jquery还提供了两个异步加载的方法：$.getScript和$.getJSON();专门用于加载js文件和json文件。有时候我们希望能够特定的情况下才加载js文件，而不是在一开始的时候就把所有js文件都加载进来，这时这两个方法就可以发挥作用了。一般只需要一个url参数就可以了。

$.ajax方法

这个方法是所有ajax方法中最底层的方法，所有的其它方法都是基于$.ajax()方法的封装。这个方法只有一个参数，接收各个功能键值的对象。

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| url | String | 发送请求的地址 |
| Type | String | 请求方式，默认get |
| Timeout | Number | 设置请求超时时间（毫秒） |
| Data | Object或String | 发送请求时传递的参数 |
| DataType | String | 预期的返回类型 |
| BeforeSend | Function | 发送请求前可以修改XMLHttpRequest对象函数 |
| Complete | Function | 请求完成后的回调函数 |
| Success | Function | 请求成功后的回调函数 |
| Error | Function | 请求失败后的回调函数 |
| Global | Boolean | 是否触发全局ajax,默认为true |
| Cache | Boolean | 设置浏览器缓存响应，默认为true |
| Content | DOM | 指定某个元素为与这个请求相关的所有回调函数的上下文 |
| ContType | String | 指定请求内容的类型 |
| Async | Boolean | 是否异步处理，默认为true |
| ProcessData | boolean | 数据被处理为url编码格式，默认为true |
| DataFilter | Function | 用来筛选响应数据的回调函数 |
| IfModified | Boolean | 不进行头检测，默认为false |
| Jsonp | String | 指定一个查询参数名称来覆盖默认的jsonp回调参数名 |
| Username | String | 在http认证请求中使用的用户名 |
| Password | String | 在http认证请求中使用的密码 |
| ScriptCharset | String | 当远程和本地内容使用不同的字符集时，用来设置script和json请求所使用的字符集 |
| Xhr | Function | 用来提供SHR实例自定义实现的回调函数 |
| Traditional | Boolean | 不使用传统风格参数序列化，默认为false |

Data表单序列化方法

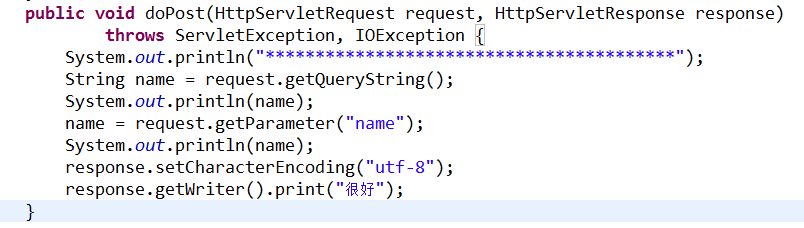
Data:$(‘form’).serialize()

#### Ajax与Servlet之间的请求与响应：

AJAX请求部分部分



服务器响应部分



# 正则表达式

## 认识正则表达式

#### 什么是正则表达式：

Regexp对象表示正则表达式，它对字符串执行模式匹配的强大工具。

正则表达式简洁且功能强大，通常用来匹配字符串，比如在表单验证中检验数据是否合法。它并不仅仅在JavaScript中可以使用，在众多的高级编程语言中都支持正则表达式。

#### 关于字符串的常用操作

Search()方法用于检索字符串，或检索与正则表达式相匹配的子字符串

该参数可以是需要在stringObject中检索的子串，也可以是需要检索的Regexp对象。

match()方法可以在字符串内检索指定的值，或找到一个或多个正则表达式的匹配

该方法类似indexof()和lastIndexOf()，但是它返回指定的值而不是字符串的位置。

Replace()方法用于在字符串中用一些字符串去替换另一些字符，或替换一个与正则表达式相匹配的子串。

Split()方法用于把一个字符串分割成字符串数组。



## 正则表达式语法

#### 创建正则表达式的两种方法

* new RegExp(pattern,attributes);
* /pattern/attributes;



#### 修饰符

|  |  |
| --- | --- |
| 修饰符 | 描述 |
| i | 执行对大小写不敏感的匹配 |
| g | 执行全局匹配 |
| m | 执行多行匹配 |



#### 字符类

|  |  |
| --- | --- |
| [abc] | a、b 或 c（简单类） |
| [^abc] | 任何字符，除了 a、b 或 c（否定） |
| [a-zA-Z] | a 到 z 或 A 到 Z，两头的字母包括在内（范围） |
| [a-d[m-p]] | a 到 d 或 m 到 p：[a-dm-p]（并集） |
| [0-9] | 0-9 |
| [red|blue|orange] | 多值选一 |

#### 元字符

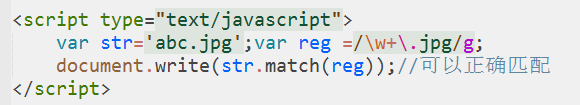
|  |  |
| --- | --- |
| . | 任何字符（换行符和结束符不匹配） |
| \d | 数字：[0-9] |
| \D | 非数字： [^0-9] |
| \s | 空白字符：[ \t\n\x0B\f\r] |
| \S | 非空白字符：[^\s] |
| \w | 单词字符：[a-zA-Z\_0-9] |
| \W | 非单词字符：[^\w] |

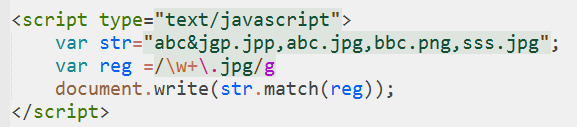
#### 边界匹配器

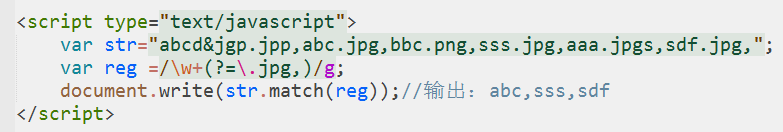
|  |  |
| --- | --- |
| ^ | 行的开头 |
| $ | 行的结尾 |
| \b | 单词边界 |
| \B | 非单词边界 |
| \A | 输入的开头 |
| \G | 上一个匹配的结尾 |
| \Z | 输入的结尾，仅用于最后的结束符（如果有的话） |
| \z | 输入的结尾 |
| ？=n | 匹配任何其后紧接指定字符串n的字符串 |
| ？!n | 匹配任何其后没有紧接指定字符串n的字符串 |

#### 量词

|  |  |
| --- | --- |
| X? | X，一次或一次也没有 |
| X\* | X，零次或多次 |
| X+ | X，一次或多次 |
| X{n} | X，恰好 n 次 |
| X{n,} | X，至少 n 次 |
| X{n,m} | X，至少 n 次，但是不超过 m 次 |







## 正则表达式分组和多行匹配



## 正则表达式属性及方法

#### 属性

* Global RegExp对象是否具有标志g。

语法：RegExpObject.global;

如果g标志被设置，则该属性值为true,否则为false;

* ignoreCase RegExp对象是否具有标志i

语法：RegExpObject.ignoreCase

如果i标志被设置，则该属性值为true，否则为false;

* multiline RegExp对象是否具有标志m

语法：RegExpObject.multiline

如果m标志被设置，则该属性值为true，否则为false;

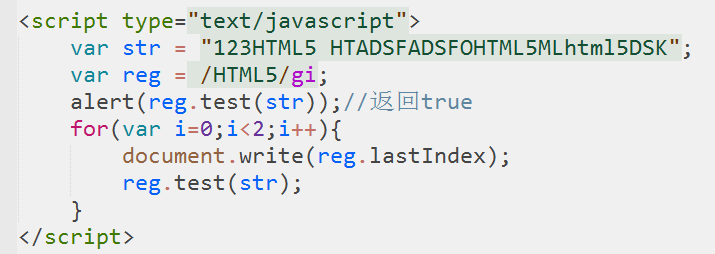


* lastIndex一个整数。标示开始下一次匹配的字符位置。

语法：RegExpObject.lastIndex

该属性存在一个整数，它声明的是上一次匹配文本之后的第一个字符的位置，多用于在一个字符串中进行多次匹配，不具有标志g和不表示全局模式的RegExp不能使用lastIndex属性。

上次匹配的结果由方法RegExp.exec()和RegExp.test()找到的，它们都以lastIndex属性所指的位置作为下次检索的起始点，这样，就可以通过反复调用这两个方法来遍历一个字符串中所有匹配的文本。



* Sourse正则表达式的源文本。

RegExpObject.source

Source属性用于返回模式匹配所用的文件

该文本不包括正则表达式直接使用的定界符,也不包括标志g、i、m。

#### 方法

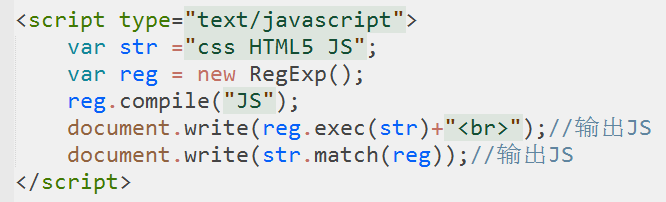
* Test()检索字符串中指定的值，返回true或false;

语法：RegExpObject.test(String)

如果字符串string中含有与RegExpObject匹配的文件，则返回true,否则返回false;

* Compile编译正则表达式

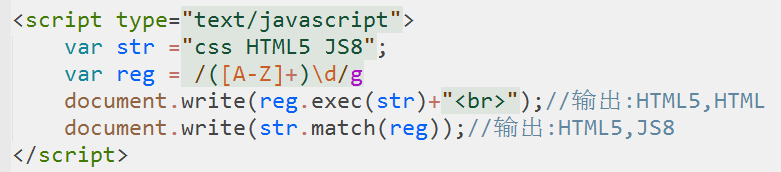
Compile方法将正则表达式转换成为内部的格式，从而执行得更快，例如：这允许在循环中更有效地使用正则表达式，当我们重复使用相同的正则表达式时，编译过的正则表达式时，编译过的正则表达式使执行加速。



Exec()检索字符串中指定的值。返回找到的值，并确定其位置

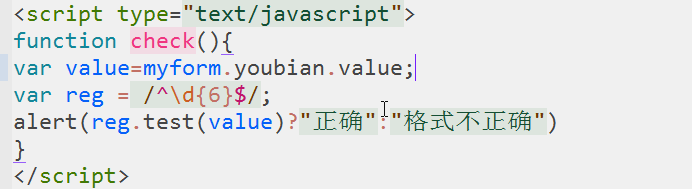
如果exec()方法没有找到匹配将返回null,如果找到匹配项，则exec方法返回一个数组。

数组元素0包含了完整的匹配项，而元素1到N包含的是匹配项中出现的任意一个子匹配项。除了数组元素和length属性之外，exec()方法还返回两个属性，index属性声明的是匹配文本的第一个字符的位置。属性则存放的是被检索的字符串string。在调用无子表达式的非全局的RegExp对象的exec()方法时，返回的数组与调用方法String.match()返回的数组是相同的。当正则表达式无子表达式，并且定义为全局匹配时，exec和match执行，若存在多处匹配内容，则match返回的是多个元素数组。当正则表达式有子表达式，并且定义为全局匹配，exec和match执行的结果不一样，此时match将忽略子表达式，只查找全匹配正则表达式并返回所有内容。



## 练习

#### 验证邮政编码：



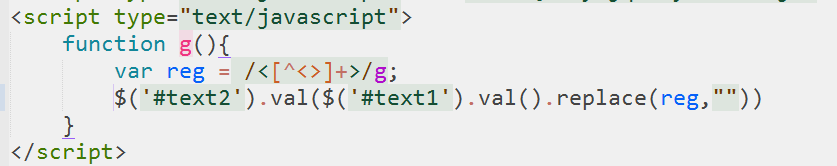
#### 验证身份证号



#### 验证电话号码



#### 过滤HTML标签



# 表格

## Table对象集合

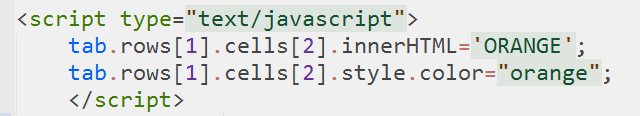
* Cells[]返回包含表格中所有单元格的一个数组。

语法：tableObject.cell[];

* Rows[]返回包含表格中所有行的一个数组。

Rows集合返回表格中所有行的一个数组，该集合包括thead,tfoot和tbody中定义的所有行。

* tBodies[]返回包含表格中所有tbody的一个数组



## table对象的方法

createCaption()为表格创建一个caption元素。

createTfoot()在表格中创建一个空的tFoot元素。

createThead()在表格中创建一个空的tHead元素。

deleteCaption()从表格中删除caption元素以及其内容。

deleteRow()从表格中删除一行。

deleteTFoot()从表格中删除Tfoot元素及其内容。

deleteTHead()从表格中删除Thead元素及其内容。

InsertRow()在表格中插入一个新行





## Table对象的常用属性

Frame设置或返回表格的外部边框。

Rules设置或返回表格的内部边框

Caption对表格的caption元素的引用。

CellPadding设置或返回单元格内容和单元格边框边间的空间量

CellSpacing设置或返回在表格中的单元格之间的空间量

Summary设置或返回对表格的描述

Tfoot/thead返回表格的tfoot/thead对象。如果在存在该元素，则返回NULL;

Border/width/id……

## 练习

#### 案例1：单击单元格后改变其背景色

单击单元格后改变其背景色，再次单击则恢复这前样式：

重要知识点：

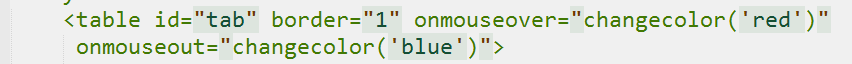
通过table的rows属性，遍历表格所有行，然后通过cells属性，遍历每一行中的单元格。

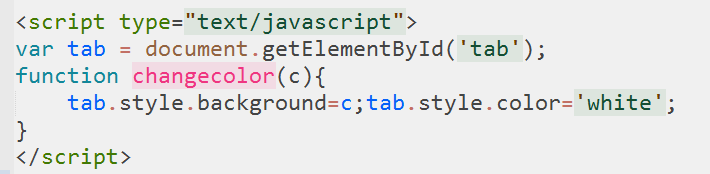
遍历的过程中，动态的为每一个单元格定义单击事件，改变单元格的背景色。



#### 案例2：鼠标移入移出改变颜色

首先此功能可以完全由CSS完成，但这里主要学习onmouseover(),和onmouseout()





#### 案例3：克隆表格

知识点：

innerHTML只可以复制表格的内容

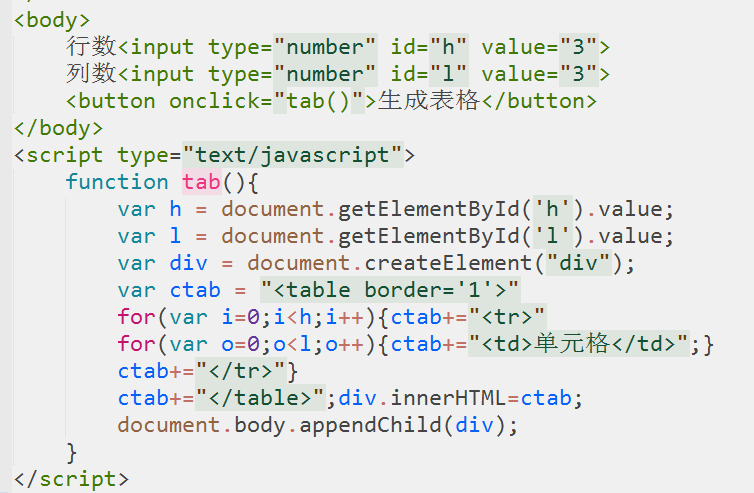
表格属性的运用



#### 案例4：表格动态显示



#### 案例5：动态创建表格





# 表单

## 表单的4种访问



## 遍历表单的所有控件

Form对象集合：

Elements[]包含表单中所有元素的数组。



## Form对象的属性、方法、事件

Action设置或返回表单的action属性

Length返回表单中的元素的数量。

Id/name/target/method………..

#### Form对象方法

Reset()把表单的所有输入元素重置为它们的默认值。

Submit()提交表单。

Form对象事件句柄。

Onreset在重置表单之前调用

Onsubmit在提交表单之前调用

#### Text对象属性

accessKey属性可设置或返回访问文本域的快捷键。

Alt设置或返回当浏览器不支持文本域时提供显示的替代文本。

DefaultValue属性可设置或返回文本域的默认值。

Disabled设置或返回文本文本域的默认值。

Form返回一个对包含文本域是否应被禁用。

MaxLength设置或返回文本域中最大字符数。

readOnly设置或返回文本域是否应是只读的。

tabIndex设置或返回文本域的tab键控制次序。

#### Text对象方法

Blur()从文本上移开焦点。

Focus()从文本域上设置焦点。

Select()选取文本域中的内容。

#### Select对象属性

Length返回下拉列表的选项数目

Multiple设置或返回是否选择多个项目

SelectedIndex设置或返回下拉列表中被选项目索引号

Size设置或返回下拉列表中的可见行数

#### Select对象方法

Add()向下拉列表添加一个选项



要运用其返回参数，得加上return;

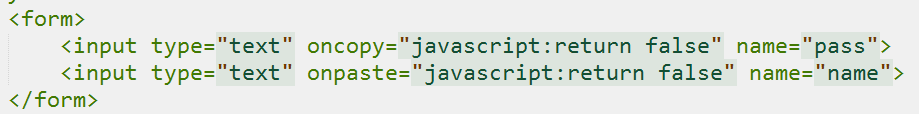
## 通过name属性访问表单元素



## 禁用复制和粘贴

要点：onpaste事件在用户向元素中粘贴文本时触发。

Oncopy事件在用户复制元素上的内容时触发。



# JQuery基础

JQUERY简介。

JQuery是一个强大的JS库。极大的简化了JavaScript编程。

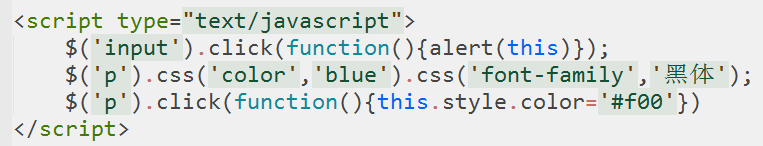
## JQuery语法

基础语法：$(selector).action()

美元符号定义JQuery对象，是JQuery中选取元素的符号，也是JQuery中功能的缀

选择符：选择或查找HTML元素

JQuery的action()执行对应元素的方法。



## Document.ready和window.onload的区别

JQuery中$(document).ready()的作用类似于传统JS中的window.onload方法，不过与window.onload方法还是有一些区别的。

$(function(){})和$(document).ready(function(){})这两个方法的效果是一样的，都是在dom文档树加载完成之后执行一个函数。

Window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。

$(document).ready()是dom结构加载完毕后就执行。

Window.onload不可以同时编写多个，如果有多个window.onload方法，只会执行一个，而$(document).ready()可以同时编写多个，并且都可以得到执行。

## JQuery选择器

#### 基本选择器

Id选择器：$(‘#id’);

元素选择器：$(“元素名“);

类选择器：$(‘.类名‘);

群选择器：$(“选择器1选择器2，……，选择器n”);

\*选择器$(‘\*’);

#### 层级选择器

层级选择器：就是通过元素之间的层次关系来选择元素，常见的层次关系包括：父子、兄弟、相邻、后代。

$(“M N”)后代选择器，选择M元素内部后代元素N

$(“M>N”)子代选择器，选择M元素内部子代元素N

$(“M~N”)兄弟选择器，选择M元素的所有同级元素N

$(“M+N”)相邻选择器，选择M元素相邻的下一个元素N

#### 属性选择器

HTML元素通常包括很多属性，JQUERY的属性选择就是把各种属性作为选择器来选择对象。

$(“selector[attr]”)选择包含给定属性的元素

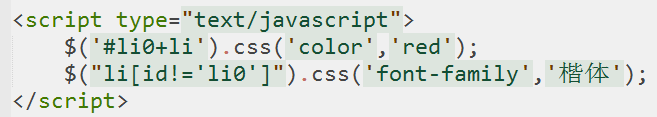
$(“selector[attr=’value’]”)返回给定的属性是某个特定值的元素

$(“selector[attr!=’value’]”)返回所有含有指定属性，但属性值不等于value的元素

$(“selector[attr\*=’value’]”)返回给定的属性是以包含某些值的元素

$(“selector[attr^=’value’]”)返回给定的属性是以某些值开始的元素

$(“selector[attr$=’value’]”)返回给定的属性是以某些值结尾的元素



#### 伪类选择器

伪类选择器也称为过滤选择器。

JQuery伪类选择器都是以英文冒号开关，和CSS中的伪类选择器用法相似，使得我们可以快速地选择我们想要获取的元素

* 简单的伪类选择器

Not(selector)选择除了某个选择器以外的所有元素。

First或first()选择某元素的第一个元素（非子元素）

Last或last()选择某元素的最后一个元素（非子元素

Odd选择某个元素的索引值为奇数的元素

Even选择某个元素索引值为偶数的元素

Eq(index)选择给定值索引值的元素，索引值index是一个整数，以0开始

Lt(index)选择所有小于索引值的元素

Gt(index)选择所有大于索引值的元素

Header选择h1-h6的标题元素

Focus选择当前具有焦点的元素

Root选择页面的根元素

Animated选择所有正在执行动画效果的元素



* 子元素伪类选择器

First-child选择父元素的第一个子元素

Last-child选择父元素的最后一个子元素

Nth-child(n)选择父元素下的每N个子元素或奇偶元素，N的值为整数odd|even

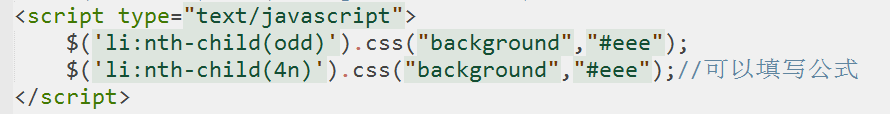
Only-child选择父元素中唯一的子元素（该元素只有一个子元素）

First-of-type选择同元素类型的每一个同级兄弟元素。

Last-of-type选择同元素类型的最后一个同级兄弟元素。

Nth-of-type选择同元素类型的每N个同级兄弟元素，N值是可以整数|odd|even.

Only-of-type匹配父元素中特定类型的唯一子元素（但父元素可以有多个）。



* 可见性伪类选择器

Hidden选取所有一可见元素。

“：hidden”选择器选择的不仅包括样式为display:none所有元素，而且还包括属性type=”hidden”和样式为visibility:hidden的所有元素。

使用visibility隐藏元素会占用空间。

Visible选择所有可见元素。

* 内容伪类选择器

内容伪类选择器，就是根据元素的文字内容或所包含的子元素特征来选择元素，其文字内容可以模糊或绝对匹配来进行元素定位。

Contains(text)选择包含给定文本内容的元素。

Has(selector)选择含有选择器所匹配元素的元素

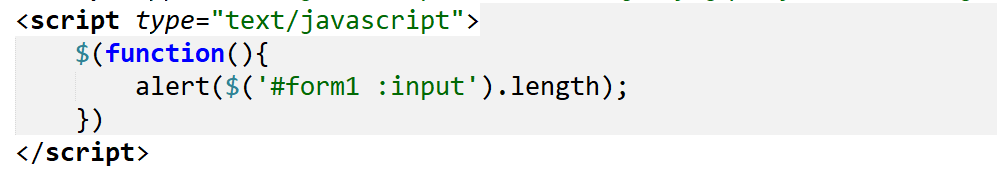
Empty选择所有不包含子元素或者不包含文本的元素

Parent选择含有子元素或者文本的元素（与empty相反）。



* 表单伪类选择器。

Input选择所有input元素



Button选择所有type=”button”的input元素；button元素也可以被选中；

Submit选择所有type=”submit”的input元素

Reset选择所有tyep=”reset”的input元素

Text选择所有单行文本框

Textarea选择所有多行文本框

Password选择所有密码文本框

Radio选择所有单选按钮

Checkbox选择所有复选框

Image选择所有图像域

Hidden选择所有隐藏域

File选择所有文件域

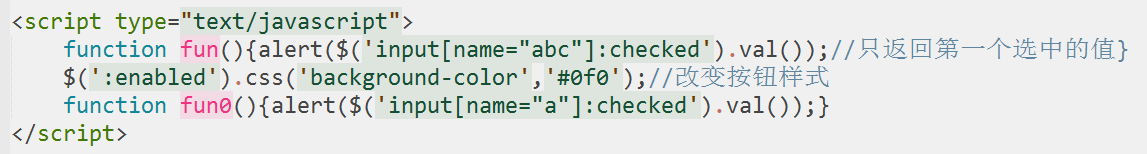
* 表单属性伪类选择器

Enabled选择所有可用input元素

Disabled所有禁用的input元素

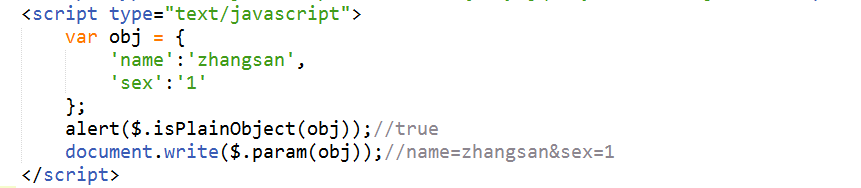
Selected选择所有被 选择中的option元素

Checked选择所有被选中的表单元素，一般用于radio和checkbox.



## Jquery工具函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| $.isArray(obj) | 判断是否为数组对象 |
| $.isFunction(obj) | 判断是否为函数 |
| $.isEmptyObject(obj) | 判断是否为空对象 |
| $.isPlainObject(obj) | 判断是否为纯粹对象 |
| $.contains(obj) | 判断DOM节点是否包含DOM节点 |
| $.type(Data) | 判断数据类型 |
| $.isNumeric(Data) | 判断数据是否为数值 |
| $.isWindow(Data) | 判断数据是否为window对象 |
| $.param(obj) | 由对象返回字符串 |
|  |  |



# JQuery DOM和CSS操作

## 文档内容的获取和设置

页面内容操作的方法：

Text()方法：获取或设置某个元素的文本内容

Html()方法：获取或设置某个元素的HTML内容

Val()方法：获取或设置“表单元素”的值。



## 元素属性及相关操作。

Attr()设置或返回匹配元素的属性和值。

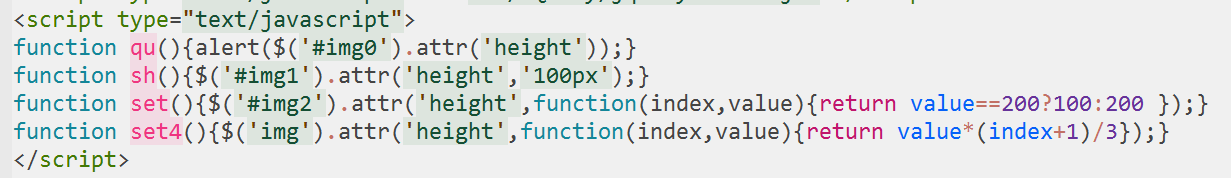
返回被选元素的属性值。语法：$(selector).attr(attribute).

设置被选元素的属性和值：语法$(selector).attr(attribute,value).

使用函数来设置属性/值，语法$(selector).attr(attribute,function(index,oldvalue){}).

removeAttr()从元素中移除指定的属性，语法：

$(selector).removeAttr(attribute).



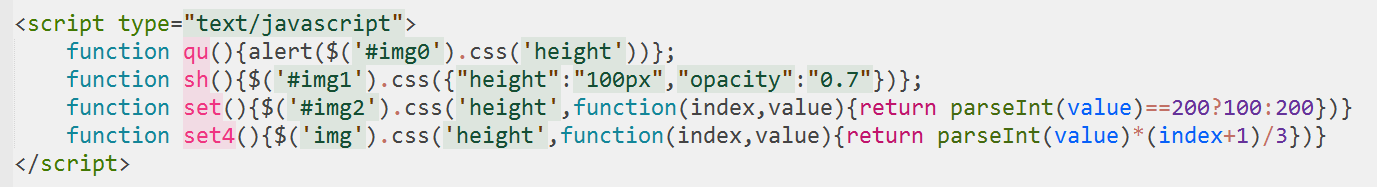
## CSS属性及相关操作

CSS()方法

$().css(“属性”)获取CSS属性值

$().css(“属性”,”属性值”)设置单个CSS属性

$().css({“属性0”:“属性值0”,”属性1”:”属性值1”,…….}同时设置多个属性。



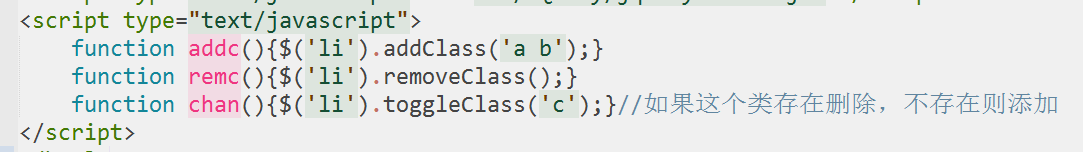
## CSS类操作

AddClass()向被选元素添加一个或多个类

RemoveClass()从被选元素删除一个或多个类

ToggleClass()对被选元素进行添加/删除类的切换操作

ToggleClass(classname,boolean)第二个布尔值如果为true则添加，false为删除



## 元素的宽高

Width()方法设置或返回元素的宽度（不包括内边距、边框、外边距）;

InnerWidth()方法返回元素的宽度（包括内边距）；

OuterWidth()方法返回元素的宽度（包括内边距和边框）;

Height()、innerHeight()、outerHeight()与宽度类似。

Width(function(index,value){return })

## 元素的位置

Offset()方法返回或设置匹配元素相对于文档的偏移（位置），

该方法返回一个对象，包含两个整形属性：top,left以像素计，此方法只对可见元素有效。

设置偏移坐标$(selector).offset(Value),

可以使用函数来设置所有匹配元素的偏移坐标：

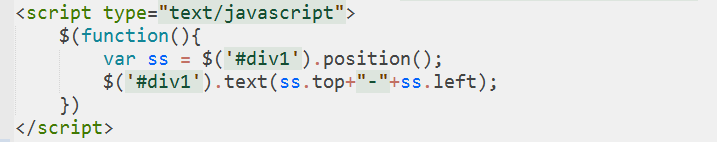
$(selector).offset(function(index,oldoffset){})

Index-可选。接受选择器的index位置

Oldvalue-可选，接受选择器选择元素的当前坐标



Position()获取匹配元素相对父元素的偏移。

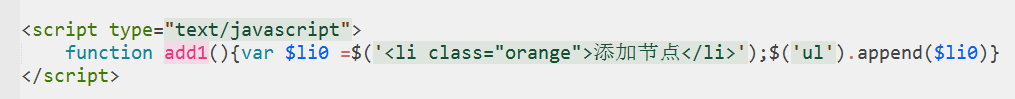
返回的对象包含两个整型属性：top 和 left。为精确计算结果，请在补白、边框和填充属性上使用像素单位。此方法只对可见元素有效。

## DOM操作

DOM操作是Jquery核心内容之一，当前端开发时，我们往往需要在页面某个地方添加一个或者删除一个元素，这种添加删除元素的操作就是通过DOM操作来实现的。

#### 创建节点

创建节点：$(html)如:$(<li></li>”);



#### 插入节点

Append()与appendTo()

Append()方法在被选元素的结尾插入指定内容。

语法1：$(selector).append(content)

语法2：$(selector).append(function(index,html){})

appendTo()跟append()类似，作用都是在所选元素内部的末尾插入内容。

语法：$(content).appendTo(selector)

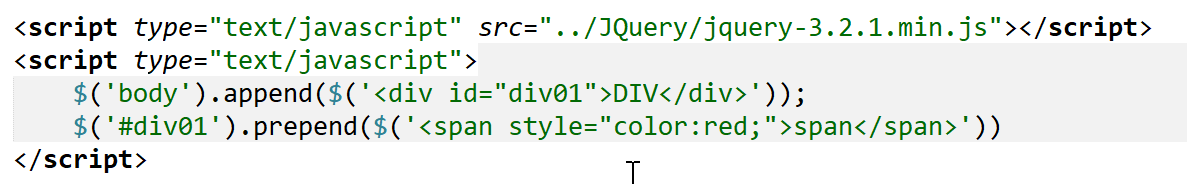
Append()和appendTo()方法执行的任务相同，不同之处在于：内容和选择器的位置，以及append能够使用函数来附加内容。





Prepend()与prependTo()

方法与append()方法使用基本相同，但是prepend会在选定元素内部的开始位置添加指定内容



Before(function(index,html){}、Before()和insertBefore()

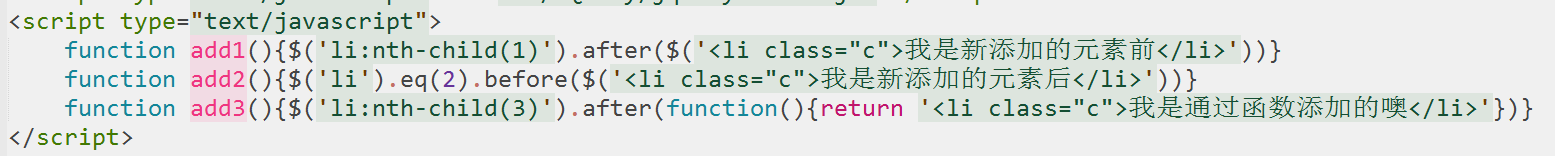
Before(content|fn);在每个匹配的元素之前插入内容

InsertBefore(content)把所有匹配的元素插入到另一个指定的元素、元素集合的前面。

After(function(index,html){}、After()和insertAfter()

After(content|fn);在每个匹配的元素之后插入内容。

InsertAfter(content):把所有匹配的元素插入到另一个指定的元素、元素集合的后面



#### 删除节点



Remove()删除匹配的元素集合或所有的子节点。【绑定事件，附加的数据等都会被移除】

Detach()从DOM中删除所有匹配的元素【与remove不同的是，所有绑定的事件、附加的数据都会被保留下来！】

Empty()：删除匹配的元素集合中所有的子节点。

Remove()和detach()这两个方法删除节点时，会将自身节点以及后代节点一并删除，但empty()方法仅删除后代节点，而会保留自身节点。

#### 复制节点

语法：$(selector).clone(includeEvents)includeEvents:可选，布尔值，规定是否复制元素的所有事件处理。



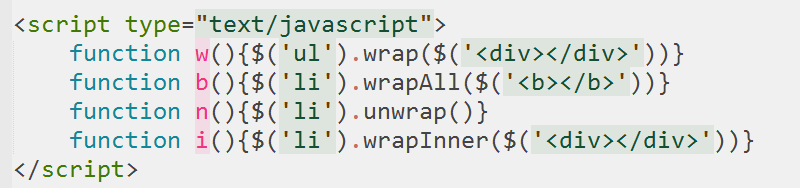
#### 包裹节点：

Wrap()：把每个被选元素放置在指定的HTML内容或元素中。

WrapAll()：将所有匹配的元素用单个元素包裹起来。

Unwrap()：移出元素的父元素。

wrapInner()将每一个匹配的元素的子内容（包容文本节点）用一个HTML结构包裹起来。

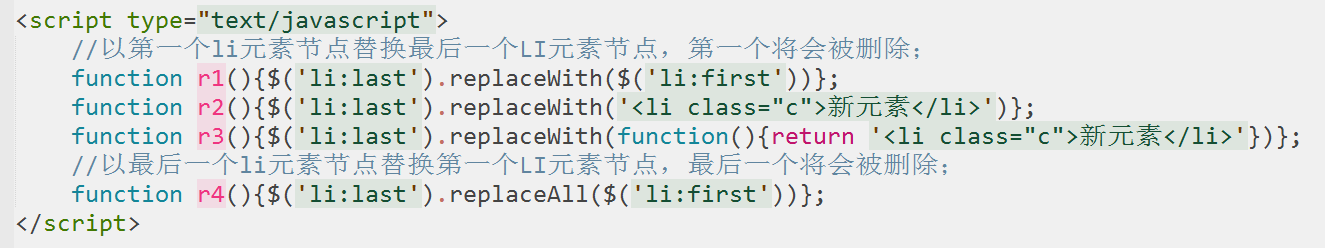


#### 替换节点

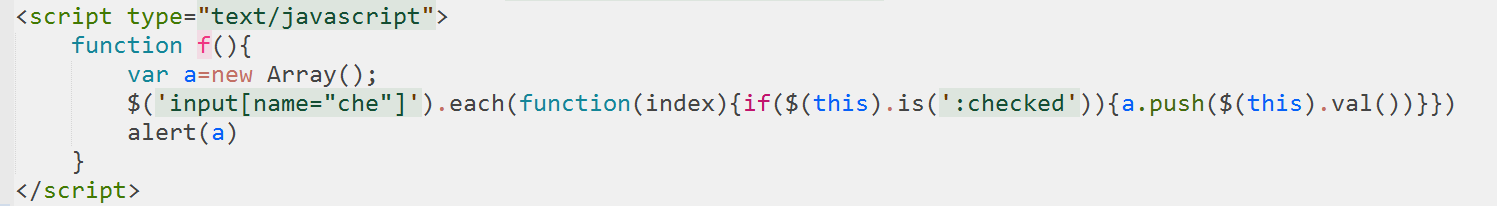
ReplaceWith():用指定的HTML内容或元素替换被选元素

ReplaceAll():用指定的HTML内容或元素替换被选元素。

ReplaceAll()与replaceWith()作用相同。差异在于语法：内容和选择器的位置，以及replaceWith()能够使用函数进行替换。



#### 遍历节点



在Jquery中，我们可以使用each()方法来轻松实现元素的遍历操作。

语法：$(selector).each(function(){});

## 过滤方法

First()/last()方法返回被选元素的首个元素/最后一个元素。

Eq()方法返回被选元素中带有指定索引号的元素。【索引号从0开始】

Filter()方法允许您规定一个标准。不匹配这个标准的元素会被从集合中删除，匹配的元素会被返回。

Not()方法返回不匹配标准的所有元素。

Is()判断所选元素是否有符合某个条件的元素，返回值为true或false;

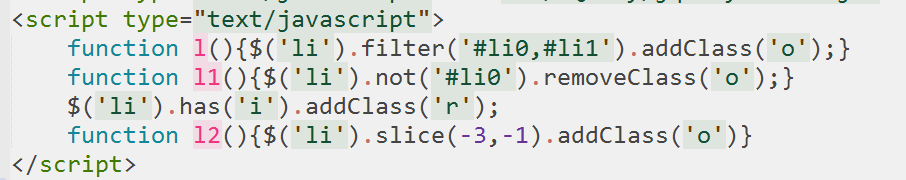
根据选择器、元素或Jquery对象来检测匹配元素集合，如果这些元素到少有一个元素匹配给定的参数则返回true;

Has()使用“自定义表达式”来选择符合条件的元素。

Slice(start,[end])把匹配元素集合缩减为指定的指数范围的子集。

Start：开始选取子集的位置。第一个元素是0，如果是负数，则可以从集合的尾部开始选起。

End：结束选取自己的位置，如果不指定，则就本身的结尾。



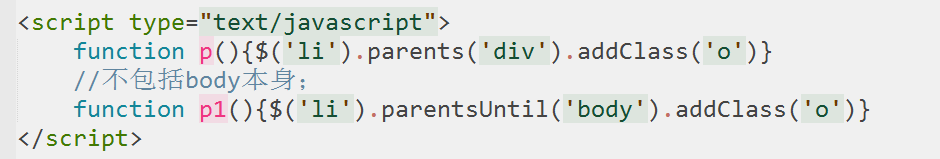
## 查找方法

#### 查找祖先：

Parent()获得当前匹配元素集合中每一个元素的父元素，由选择器筛选（可选）

Parents()获取当前匹配元素集合中每一个元素的祖先元素，由选择器筛选（可选）

parentUntil()获得当前匹配元素集合中每个元素的祖先元素，直到遇到匹配选择器的元素为止。

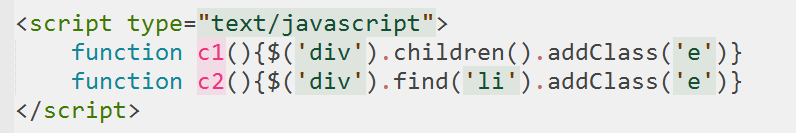


#### 查找后代元素；

Children()获得匹配元素集合中每个元素的所有子元素。

Find()获得当前匹配元素的集合中每个元素的后代，由选择器进行筛选。

Contents()获得匹配元素集合中每个元素的子元素，包括文本和注释节点。

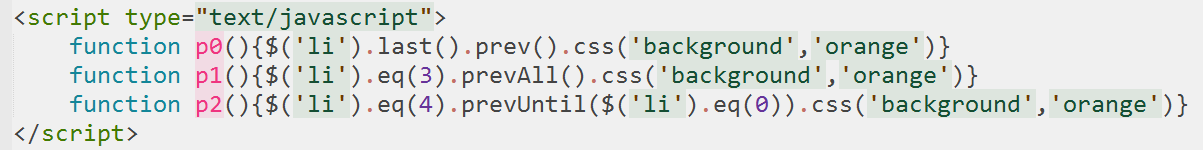


#### 向前查找

Prev()获得匹配元素集合中每个元素紧邻的前一个同辈元素，由选择器筛选（可选）。

prevAll()获得匹配元素集合中每一个元素之前的所有同辈元素，由选择器筛选（可选）

prevUntil()获得每一个元素之前所有的同辈元素，直到遇到匹配选择器的元素为止。



#### 向后查找

Next()获得匹配元素集合中每个元素紧邻的同辈元素

nextAll()获得匹配元素集合中每个元素之后的所有同辈元素，由选择器进行筛选（可选）

nextUntil()获得每个元素之后所有的同辈元素，直到遇到匹配选择器的元素为止。

#### 其它方法

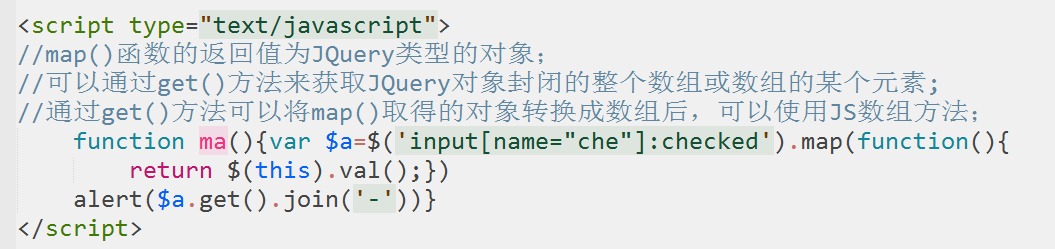
Add()方法将元素添加到匹配元素的集合中

Addback()添加堆栈中元素集合到当前集合，一个选择性的过滤选择器。

End()结束当前链中最近的一次筛选操作，并将匹配元素集合返回到前一次的状态。



Map()把当前匹配集合中的每个元素传递给函数，产生包含返回值的新Jquery对象。



Closest()方法获得匹配选择器的第一个祖先元素，从当前元素开始沿DOM树向上查找。

Closest和parents的主要区别在于：closest从当前元素开始匹配寻找，parents从父元素开始匹配寻找；consest逐级向上查找，直到发现匹配的元素后就停止，parents是一直向上查找直到根元素，然后把这些元素放进一个临时集合中，再用给定的选择器表达式去过滤。

offsetParent()获得用于定位的第一个父元素，即：返回父元素中每一个其position设置为relative或者absolute的元素。此方法仅对可见元素有效。

## 练习

#### 弹出框

#### 2.选项卡

见附件

# JQuery事件

Ready()文档就绪事件（当HTML文档就绪可用时触发）。

在js学习中，常用的事件有：click、dbclick、mousedown、nouseup、mousemove、mouseover、mouseout、change、select、submit、keydown、keypress、keyup、blur、focus、load、resize、scroll、error……

## 鼠标事件

Click()触发、或将函数绑定到指定元素的Click事件

Dblclick()当双击元素时会触发

Mousedown()/mouseup()鼠标按下和松开事件

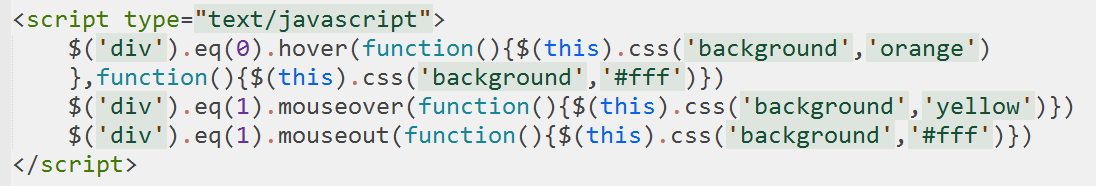
Mouseover()/mouseout()鼠标移入和移出事件

Mouseenter()/mouseleave()鼠标的移入和移出事件。

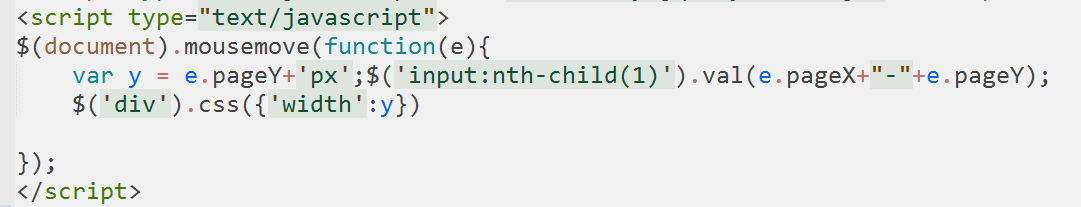
Mouseout()/meouseover()事件在鼠标移入所选元素及其后代子元素时都会触发

Mouseenter()/mouseleave()事件只有鼠标移入所选元素才会触发，如果鼠标仅仅移入所选元素的后代子元素不会触发。

hover()一个模仿鼠标悬停事件



Mousemove()鼠标移动事件；



## 键盘事件

Keydown()当键盘或按键被按下时，发生keydown事件。

Keyup()在键盘的某个键被按下之后松开的瞬间触发的事件。

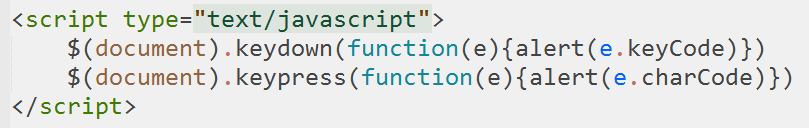
Keypress()在键盘上的某个键被按下到松开‘整个过程’中触发的事件。

Keypress事件只在按下键盘的任一字符键时触发，单独按下功能键不会触发，而keydown无论是字符键还是功能键都会触发。

Keydown()、keyup()返回的是键码，而keypress()返回的是字符编码

在jquery中event.which将event.keycode和event.charcode标准化了，推荐使用event.which.

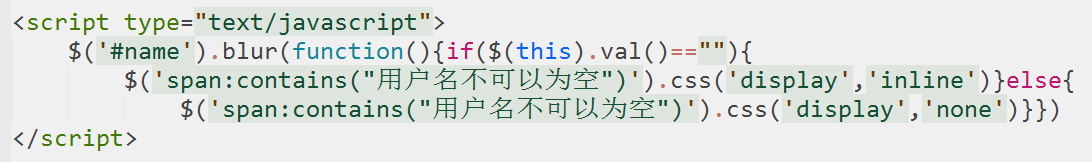
Event,which指示按下了哪个键或按钮。



## 表单事件

Blur()当元素失去焦点时触发blur事件

可以通过返回false来防止触发浏览器的行为。Blur事件会在元素失去焦点的时候触发，即可以是鼠标行为，也可以是tab键离开。



Focus()当元素获得焦点时触发

Focusin()当元素获得焦点时触发。【focusin事件跟Focus事件区别在于，他可以在父元素上检测子元素获得焦点的情况】

Focusout()当元素失去焦点时触发【focusout事件跟blur事件区别在于，他可以在父元素上检测子元素失去焦点的情况】

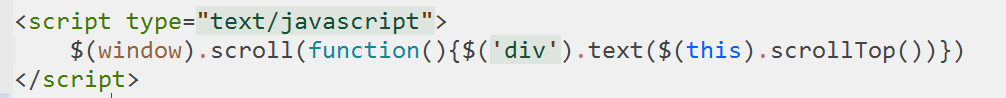
Change()当元素的值发生改变时触发。

该事件仅适用于文本域，以及select元素。当用于select元素时，change事件会在选择某个选项时发生。

## 浏览器事件

Scorll()当用户滚动滚动条时触发。

Resize()当调整浏览器窗口的大小时触发



## 事件对象

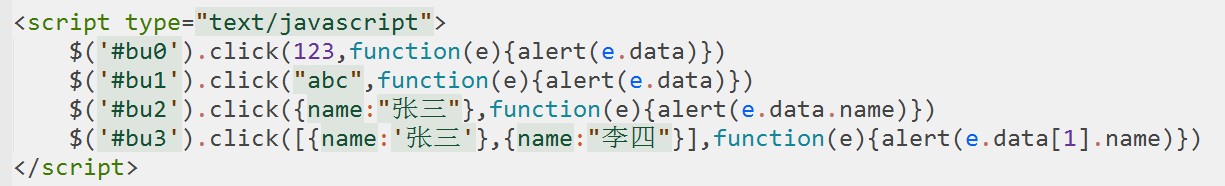
Event.type描述事件的类型

Event.target触发该事件的DOM元素

Event.currentTarget在事件冒泡阶段中的当前DOM元素，等同于this

Event.relatedTarget返回当鼠标移动时进入或退出的那个元素

Event.data事件调用时的额外数据



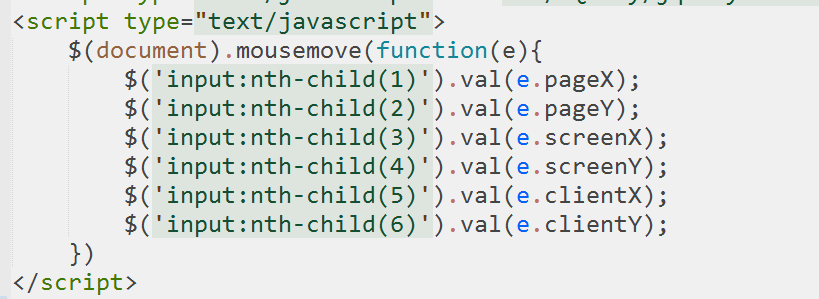
Event.pageX/event.page显示鼠标相对于文件在左侧和顶部边缘的位置

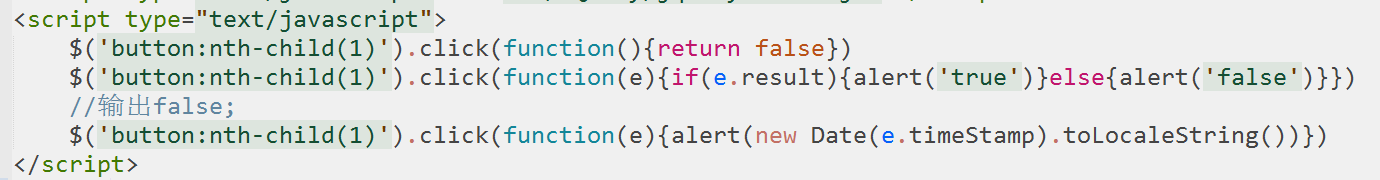
注意区分：screenX/screenY：获取显示器屏幕位置的坐标。

clientX/clientY:获取相对于页面视口的坐标

event.result这个属性包含了当前事件最后触发的那个处理函数的返回值

如果为DOM元素的同一事件类型绑定了多个事件处理函数，你可以使用result属性获取上一个事件处理函数执行的返回值。





Event.which针对键盘和鼠标事件，这个属性能确定你到底按的是哪个键或按钮触发此事件。

鼠标键及返回

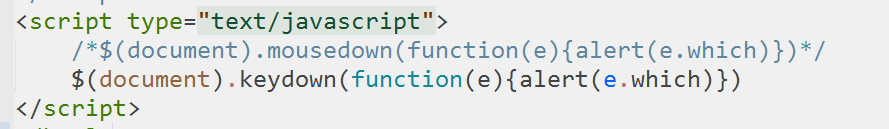
|  |  |
| --- | --- |
| event.which属性值 | 对应的鼠标按钮 |
| 1 | 鼠标左键 |
| 2 | 鼠标中键(滚轮键) |
| 3 | 鼠标右键 |

常用的字符Unicode代码对应表

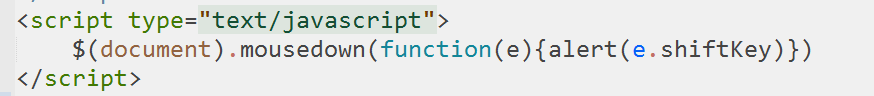
|  |  |
| --- | --- |
| which属性值(或范围) | 对应的输入字符 |
| 48 - 57 | 对应字符 0 - 9 |
| 65 - 90 | 对应字符 A - Z |
| 97 - 122 | 对应字符 a - z |

event.which属性返回的是对应按键的映射代码值(相当于event.keyCode)

|  |  |
| --- | --- |
| which属性值(或范围) | 对应的键盘按键 |
| 8 | Backspace键 |
| 9 | Tab键 |
| 13 | Enter键 |
| 16 | Shift键 |
| 17 | Ctrl键 |
| 20 | Alt键 |
| 20 | Caps Lock键(大小写锁定) |
| 27 | Esc键 |
| 33 - 36 | 对应按键 PageUp、PageDown、End、Home |
| 37 - 40 | 对应按键 左、上、右、下(方向键) |
| 45 - 46 | 对应按键 Insert、Delete |
| 48 - 57 | 对应按键 0 - 9(非小键盘) |
| 65 - 90 | 对应按键 A - Z |
| 91 | Windows键 |
| 96 - 105 | 对应按键 0 - 9(小键盘) |
| 106、107、109、110、111 | 对应按键\*、+、-、.、/(小键盘) |
| 112 - 123 | 对应按键 F1 - F12 |



返回事件发生时是否按下了altKey/shiftKey/ctrlKey



Event.preventDefault()阻止事件的默认动作。

Event.isDefaultPrevented()返回event对象是否调用了event.preventDefault().

Event.stopPropagation()防止事件冒泡

Event.isPropagationStopped()判断是否调用过event.stopPropagation().

Event.stopImmediatePropagation()阻止剩余的事件处理函数执行并且防止事件冒泡

Event.isImmediatePropagationStopped()检测

Event.stopImmediatePropagation()是否被调用过。

|  |  |
| --- | --- |
| 属性 | 描述 |
| Type | 获取事件的事件类型 |
| Target | 获取绑定事件的DOM元素 |
| Data | 获取事件调用时的额外数据 |
| relatedTarget | 获取移入移出目标点离开或进入的那个DOM元素 |
| CurrentTarget | 获取冒泡前触发的DOM元素，等同与this |
| PageX/pageY | 获取相对于页面原点的水平/垂直坐标 |
| screenX/screenY | 获取显示器屏幕位置的水平/垂直坐标 |
| clientX/clientY | 获取相对于视口的水平/垂直坐标 |
| Result | 获取上一个相同事件的返回值 |
| timestamp | 获取事件触发的时间戳 |
| Which | 获取鼠标的左中右键（1,2,3），获取键盘按键 |
| AltKey/shiftKey  CtrlKey/metaKey | 获取是否按下了alt、shift、ctrl或meta键 |

## 事件处理

#### 事件绑定

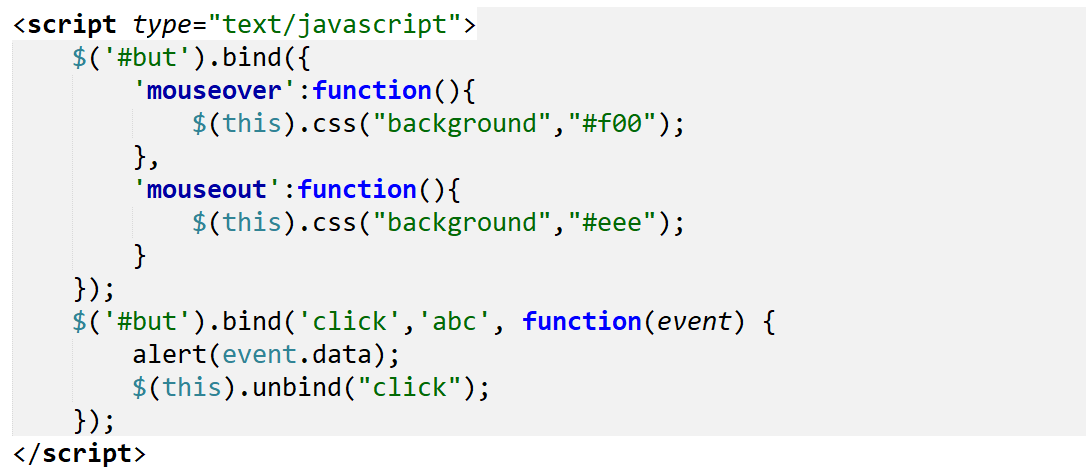
On()为某些元素绑定一个事件或多个事件。

该函数可以为同一元素、同一事件类型绑定多个事件处理函数。触发事件时，JQuery会按照绑定的先后顺序依次执行绑定的事件处理函数。



Bind(事件类型,参数,函数);

相比Onclick方法：bind方法及html元素解耦合性好，删除和添加都比较方便，而且可以同时添加多个事件，



事件绑定简写方法

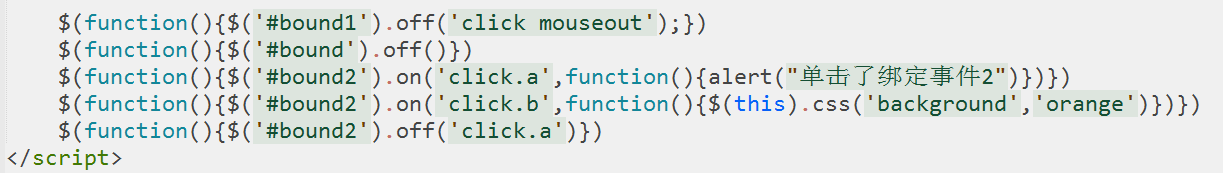
|  |  |  |
| --- | --- | --- |
| 方法 | 触发 | 描述 |
| Click(fn) | 鼠标 | 单击时触发 |
| Dbclick(fn) | 鼠标 | 双击时触发 |
| Mousedown(fn) | 鼠标 | 点击后触发 |
| Mouseup(fn) | 鼠标 | 点击弹起触发 |
| Mouseover(fn) | 鼠标 | 鼠标移入时触发 |
| Mouseout(fn) | 鼠标 | 鼠标移出时触发 |
| Mousemove(fn) | 鼠标 | 鼠标移动时触发 |
| Mouseenter(fn) | 鼠标 | 鼠标穿过时触发 |
| Mouseleave(fn) | 鼠标 | 鼠标穿出时触发 |
| Keydown(fn) | 键盘 | 键盘按下触发 |
| Keyup(fn) | 键盘 | 键盘按下弹起 |
| Keypress(fn) | 键盘 | 键盘按下事件 |
| Unload(fn) | 文档（window） | 当卸载页面时触发 |
| Resize(fn) | 文档 | 文档大小改变时触发 |
| Scroll(fn) | 文档 | 滚动条拖动 |
| Focus(fn) | 表单 | 获取焦点触发 |
| Blur(fn) | 表单 | 丢失焦点触发 |
| Focusin(fn) | 表单 | 焦点激活触发 |
| Focusout(fn) | 表单 | 丢失焦点触发 |
| Select(fn) | 表单 | 文本选定时触发 |
| Change(fn) | 表单 | 值被改变时触发 |
| Submit(fn) | 表单 | 表单提交时触发 |

#### 事件解绑

Off()解除绑定的某一指定的事件或者所有事件。

“绑定”与“解绑”是相反的关系，在JQuery中，我们可以通过off()方法来解除绑定的某一指定的事件或者所有事件。

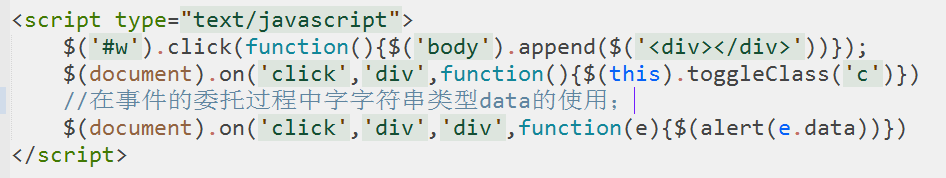
对同一元素绑定的多个同一事件进行解绑，可以使用命名空间解决，例如：click,…

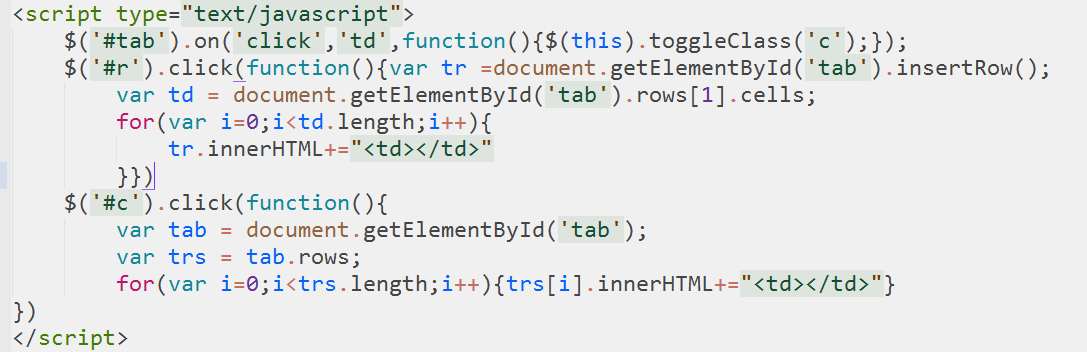


#### 事件委托

事件委托是通过事件冒泡，让子元素绑定的事件冒泡到父元素（或祖先元素）上，然后再进行事件处理。

通过事件委托动态为元素添加事件



One()为每一个匹配的特定事件绑定一个一次性的事件处理函数。

Trigger()在每一个匹配的元素上触发某类事件

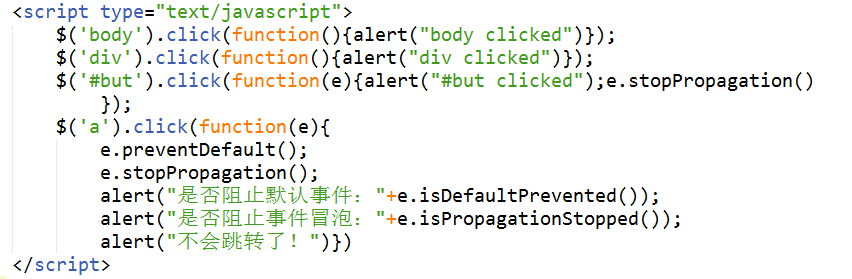




Jquery还提供了一个另外的一个模拟用户行为的方法：tiggerHandler():这个方法的使用和tigger()方法一样。在常规的使用情况下，两者几乎没有区别，都是模拟用户行为，也可以传递额外的参数，但是在某些特殊情况下，就产生了差异。

1. TiggerHandler()方法并不会触发事件的默认行为，而trigger()会
2. TiggerHandler()方法只会影响第一个匹配到的元素，而tigger()会影响所有，也就是tiggerHandler只会执行一次。
3. TiggerHandler()方法会返回当前事件执行的返回值，如果没有返回值，则返回undefined; 而trigger()则返回当前包含事件触发元素的jquery对象，方便链式连缀调用。
4. Trigger()在创建事件的时候，会冒泡。但这种冒泡是自定义事件才能体现出来，是jquery扩展于dom的机制，并非dom特性。而triggerHandler()不会冒泡。

## 事件冒泡和默认事件



# Jquery动画

## 显示、隐藏

Jquery中显示方法show();隐藏方法hide().

除了可以直接用毫秒数来控制速度以外，jquery还提供了三种预设速度参数字符串：slow、normal、fast，分别对应600毫秒、400毫秒、200毫秒。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Document</title>  </head>  <style type="text/css">  td{  width: 130px;  height: 150px;  }  div{  width: 100px;  height: 100px;  background: #f00;  border: #00f solid 5px;  border-radius: 10px;  margin: 10px;  }  span{  background:orange;  padding: 5px;  width: 25px;  margin:2px;  display: none;  float:left;  }  </style>  <body>  <span>你</span>  <span>好</span>  <span>吗</span>  <span>？</span>  <table border="1">  <tr>  <td><div>abcdefg</div></td>  <td><div>abcdefg</div></td>  <td><div>abcdefg</div></td>  </tr>  <tr id="but" style="text-align: center;">  <td><button class="show">显示</button><button class="hide">隐藏</button></td>  <td><button class="show">显示</button><button class="hide">隐藏</button></td>  <td><button class="show">显示</button><button class="hide">隐藏</button></td>  </tr>  </table>  </body>  <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44=" crossorigin="anonymous"></script>  <script type="text/javascript">  $(function(){  $('#but .show:eq(0)').click(function(){$('div:eq(0)').show();});  $('#but .hide:eq(0)').click(function(){$('div:eq(0)').hide();});  $('#but .show:eq(1)').click(function(){$('div:eq(1)').show(1000);});  $('#but .hide:eq(1)').click(function(){$('div:eq(1)').hide(1000);});  $('#but .show:eq(2)').click(function(){$('div:eq(2)').show('fast');});  $('#but .hide:eq(2)').click(function(){$('div:eq(2)').hide('fast');});  //递归调用实现队列动画  $('span').first().show('fast',function test(){$(this).next().show('fast',test);});  });  </script>  </html> |

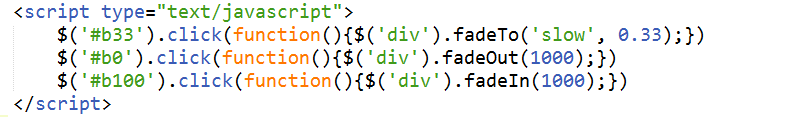
## 滑动、卷动

Jquery提供了一组改变元素高度的方法：slideUp()、slideDown()和slideToggle()。顾名思义，向上收缩（卷动）和向下展开（滑动）。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Document</title>  </head>  <style type="text/css">  td{  width: 130px;  height: 150px;  }  div{  width: 100px;  height: 100px;  background: #f00;  border: #00f solid 5px;  border-radius: 10px;  margin: 10px;  }  </style>  <body>  <table border="1">  <tr>  <td><div>abcdefg</div></td>  <td><div>abcdefg</div></td>  <td><div>abcdefg</div></td>  </tr>  <tr id="but" style="text-align: center;">  <td><button class="show">显示</button><button class="hide">隐藏</button></td>  <td><button class="show">显示</button><button class="hide">隐藏</button></td>  <td><button class="hide">切换</button></td>  </tr>  </table>  </body>  <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44=" crossorigin="anonymous"></script>  <script type="text/javascript">  $(function(){  $('#but .hide:eq(0)').click(function(){$('div:eq(0)').slideUp();});  $('#but .show:eq(0)').click(function(){$('div:eq(0)').slideDown();});  $('#but .hide:eq(1)').click(function(){$('div:eq(1)').slideUp(1000);});  $('#but .show:eq(1)').click(function(){$('div:eq(1)').slideDown(1000);});  $('#but .hide:eq(2)').click(function(){$('div:eq(2)').slideToggle('fast');});  });  </script>  </html> |

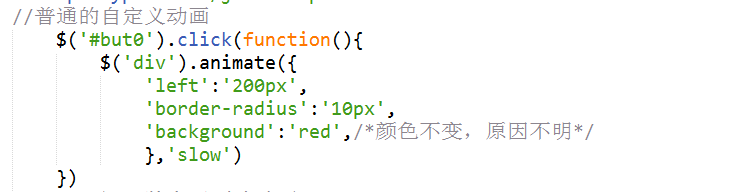
## 淡入、淡出

Jquery提供了一组专门用于透明度变化的方法：fadeIn()和fadeOut()、分别表示淡入、淡出，当然还有一个自动切换的方法fadeToggle();这三个方法都只能设置从0到100之间的变化，如果我们想设置指定值就必需用于另外一个fadeTo()方法。

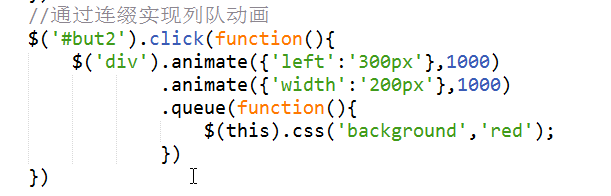


## 自定义动画

我们在动画之后如果连缀普通方法的时候，普通方法会在调用第一个动画的时候一起运行，而这有时不是我们想要的，jquery提供了一个类似于回调函数的方法queue()，他会在动画执行之后执行我们的普通方法



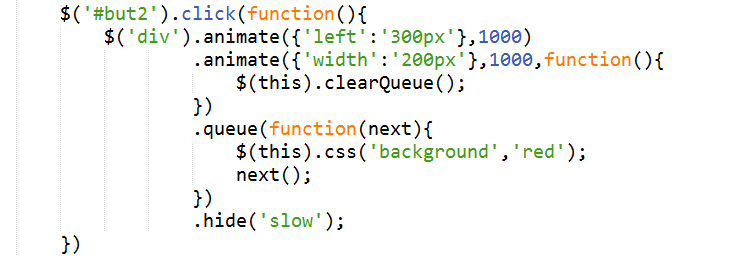




在调用了queue方法后，如果我们再加上一个隐藏动画，发现他居然无法实现了，这个问题是由于queue特性导致的，我们可以给queue()的回调函数传递一个参数，这个参数是一个next函数，在结尾处调用这个函数可以使得函数继续执行。



Jquery还提供了一个清理列队的功能方法，clearQueue().把它放入一个列队的回调函数或queue()方法里，就可以把剩下未执行的列队给移除了，此方法一般用在满足了某些条件我们的动画不需要死循环的执行下去

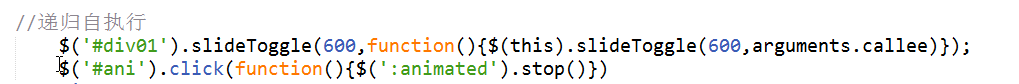


## 动画相关

很多时候需要停止正在运行中的动画，jQuery为此提供了一个stop()方法，它有两个可选参数，stop(clearQueue,gotoEnd):clearQueue传递一个布尔值，代表是否清空未执行完的动画队列，如果为true，动画会即时停止。默认为false，如果存在列队动画，只会停止掉第一个列队动画，然后继续执行后面的列队动画。togoEnd代表是否直接将正在执行的动画跳转到末状态。

.delay(1000)可以将动画延迟





## 动画全局属性

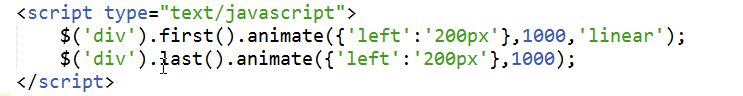
//动画频率，默认值为13，越小会越流畅

$.fx.interval=5;

//关闭动画，默认是false

$.fx.off = true;

//在animate()方法中，还有一个参数，easing运动方式，自带有两个参数：swing(缓动)、linear(匀速),默认为swing.



# Canvas专题

## Canvas绘制直线

获取绘图环境后就可以画图了：

指定线宽：lineWidth=值

指定轮廓颜色：strokeStyle=值

指定填充颜色：fillStyle=值

MoveTo()定义绘制路径的起始点

LineTo()添加一个新点（在我们的直线案例中是定义直线的终点，但是后面继续话，它将变成一个中间点）;

Stroke()绘制已定义的路径

Fill()绘制一个带填充的图形

BeginPath()用来创建新的路径

ClosePath()从当前点回到起始点来封闭路径

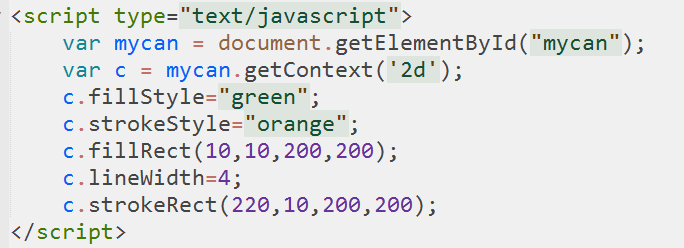


## 矩形的绘制：

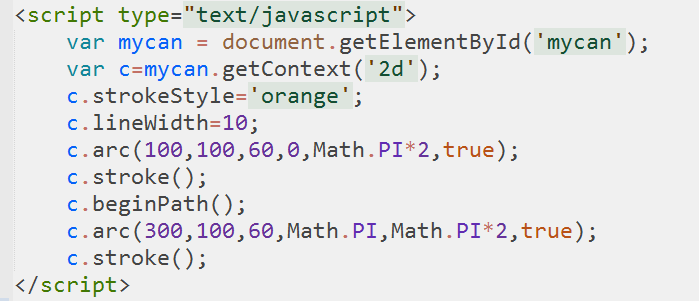
Rect(x,y,w,h)创建一个矩形

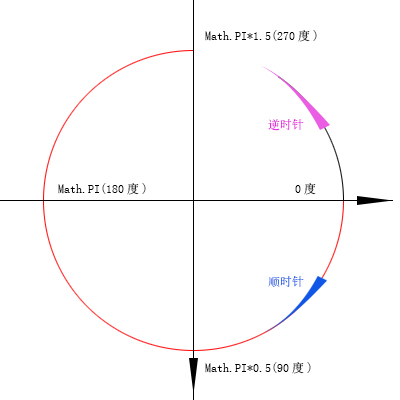
StrokeRect(x,y,w,hx,y,w,h)绘制矩形（没有填充）

FillRect(x,y,w,h)绘制填充矩形



## 圆及弧的绘制





圆的基本参数：

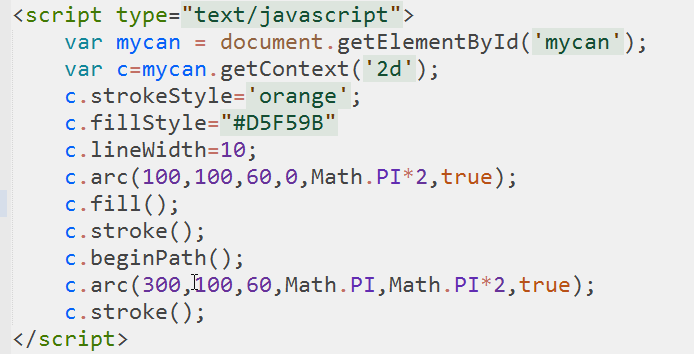
Arc(x,y,r,起始弧度，终止弧度，b)

圆心的坐标：x,y

圆的半径:r

起始弧度和终止弧度：起始角度，终止角度

绘制方向b：true（逆时针）/false（顺时针）



## 文字的绘制

strijeText(“文字内容“,x,y,maxWith)绘制空心文字

fillText(“文字内容”,x,y,maxWith)绘制填充文字

字体样式：font(大小 字体……)



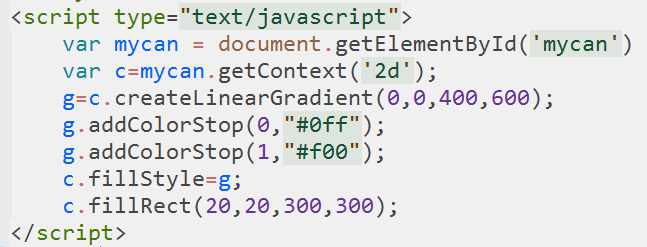
## 渐变色

createLinearGradient()创建线性渐变

createLinearGradient(x1,y1,x2,y2)颜色渐变的起始点和终点坐标

addColorStop(位置，颜色值)位置0-1

createRadialGradient(x1,y1,r1,x2,y2,r2)创建放射状渐变



## 线条样式

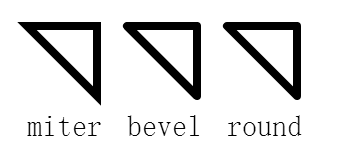
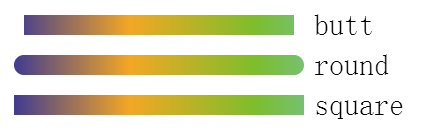
LineCap设置或返回线条的结束端点样式

Butt默认。向线条的每个末端添加平直的边缘

Round向线条的每个末端添加圆形线帽

Square向线条的每个末端添加正方形线帽

Round和square会使线条略微变长



Linejoin设置或返回两条线相交时，所创建的拐角类型

**Miter/bevel/round**

<script type="text/javascript">

var canvas = document.getElementById("mycan");

var c = canvas.getContext('2d');

c.lineJoin="miter";

c.lineWidth=8;

c.moveTo(30,30);

c.lineTo(100,30);

c.lineTo(100,100);

c.closePath();

c.stroke();

c.beginPath();

c.lineJoin="bevel";

c.moveTo(130,30);

c.lineTo(200,30);

c.lineTo(200,100);

c.closePath();

c.stroke();

c.lineJoin="round";

c.moveTo(230,30);

c.lineTo(300,30);

c.lineTo(300,100);

c.closePath();

c.stroke();

c.font="30px 宋体";

c.fillText("miter",30,140);

c.fillText("bevel",130,140);

c.fillText("round",230,140);

</script>

**Butt/round/square**

<script type="text/javascript">

var can = document.getElementById('can');

var c = can.getContext('2d');

g=c.createLinearGradient(0,0,500,0);

g.addColorStop(0,"#333399");

g.addColorStop(0.25,"#F4A824");

g.addColorStop(0.5,"#7DBE2C");

g.addColorStop(0.75,"#6AC6D9");

g.addColorStop(1,"#483993");

c.strokeStyle=g;

c.fillStype=g;

c.lineWidth=20;

c.lineCap="butt"

c.moveTo(20,20);

c.lineTo(290,20);

c.stroke();

c.beginPath();

c.lineCap="round"

c.moveTo(20,60);

c.lineTo(290,60);

c.stroke();

c.beginPath();

c.lineCap="square"

c.moveTo(20,100);

c.lineTo(290,100);

c.stroke();

c.font="30px 宋体";

c.fillText("butt",310,30);

c.fillText("round",310,70);

c.fillText("square",310,110);

</script>

Miter默认。创建尖角

Bevel创建斜角

Round创建圆角

## 贝赛尔曲线

Arc()创建弧/曲线

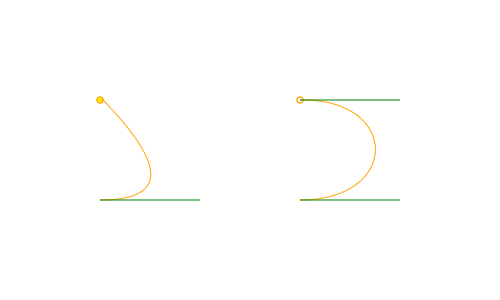
ActTo(x1,y1,x2,y2,r)创建两切线之间的弧/曲线

QuadraticCurveTo(cx,cy,x,y)创建二次贝赛尔曲线

二次贝赛尔曲线需要两个点，第一个点是用于二次贝赛尔计算中的控制点，第二个点是曲线的结束点。曲线的开始点是当前路径中的最后一个点，如果路径不存在，那么使用beginPath()和moveTo()方法来定义开始点。

(c1x,c1y.c2x,c2y,x,y)创建三次方贝赛尔曲线

三次贝赛尔曲线需要三个点，前两个点是用于三次贝赛尔计算中的控制点，第三个点是曲线的结束点，曲线的开始是当前路径中最后一个点，如果路径不存在，可以使用beginPath()或moveTo()方法定义开始点。



g.moveTo(100,100);

g.quadraticCurveTo(200,200,100,200);

g.moveTo(300,100);

g.bezierCurveTo(400,100,400,200,300,200);

## 裁剪

Clip()从原始画布剪切任意形状的区域

g.rect(0,0,500,150);

g.clip();

isPointInPath()如果指定的点位于当前路径 中，则返回true,否则返回false

# 附件：

#### 【计算器】原代码：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>cal</title>

</head>

<style>

table{

margin:15px auto;

font-size: 22px;

border: 5px outset orange;

}

#tab-03,#tab-02,#tab-01{

border: 3px outset rgba(10,10,10,0.3)

}

input{

outline: none;

box-shadow: 3px 5px 2px rgba(100,100,100,0.8) inset

}

input[type="button"]{

width: 60px;

height: 40px;

border-radius: 5px;

background: #fff;

}

#result{

text-align: right;

height: 50px;

width: 100%;

background: #fff;

font-size: 22px;

}

td{s

height: 30px;

background:#ccc;

}

</style>

<body>

<table cellspacing="0px">

<tr>

<td colspan="2"><input type="text" id="result" value=0></td>

<td style="text-align: center;">

<input type="button" value="清除" onclick="cle()">

<input type="button" value="退格" onclick="tuige()">

</td>

</tr>

<tr>

<td>

<table id="tab-01">

<tr>

<td><input type="button" value="sin" onclick="math('sin')"></td>

<td><input type="button" value="cos" onclick="math('cos')"></td>

<td><input type="button" value="tan" onclick="math('tan')"></td>

</tr>

<tr>

<td><input type="button" value="asin" onclick="math('asin')"></td>

<td><input type="button" value="acos" onclick="math('acos')"></td>

<td><input type="button" value="atan" onclick="math('atan')"></td>

</tr>

<tr>

<td><input type="button" value="PI" onclick="math('PI')"></td>

<td><input type="button" value="1/x" onclick="math('1/x')"></td>

<td><input type="button" value="exp" onclick="math('exp')"></td>

</tr>

<tr>

<td><input type="button" value="inx" onclick="math('inx')"></td>

<td><input type="button" value="lgx" onclick="math('lgx')"></td>

<td><input type="button" value="n!" onclick="math('n!')"></td>

</tr>

</table>

</td>

<td>

<table id="tab-02">

<tr>

<td><input type="button" value="7" onclick="num(7)"></td>

<td><input type="button" value="8" onclick="num(8)"></td>

<td><input type="button" value="9" onclick="num(9)"></td>

</tr>

<tr>

<td><input type="button" value="4" onclick="num(4)"></td>

<td><input type="button" value="5" onclick="num(5)"></td>

<td><input type="button" value="6" onclick="num(6)"></td>

</tr>

<tr>

<td><input type="button" value="1" onclick="num(1)"></td>

<td><input type="button" value="2" onclick="num(2)"></td>

<td><input type="button" value="3" onclick="num(3)"></td>

</tr>

<tr>

<td><input type="button" value="0" onclick="num(0)"></td>

<td><input type="button" value="." onclick="decimal()"></td>

<td><input type="button" value="=" onclick="compute('=')"></td>

</tr>

</table>

</td>

<td>

<table id="tab-03">

<tr>

<td><input type="button" value="+" onclick="compute('+')"></td>

<td><input type="button" value="取整" onclick="math('i')"></td>

</tr>

<tr>

<td><input type="button" value="-" onclick="compute('-')"></td>

<td><input type="button" value="取余" onclick="compute('%')"></td>

</tr>

<tr>

<td><input type="button" value="\*" onclick="compute('\*')"></td>

<td><input type="button" value="x^y" onclick="compute('x^y')"></td>

</tr>

<tr>

<td><input type="button" value="/" onclick="compute('/')"></td>

<td><input type="button" value="+/-" onclick="plusMinus()"></td>

</tr>

</table>

</td>

</tr>

</table>

</body>

<script type="text/javascript">

var Boo =false;

var result =0;

var ope;

function $(xid){return document.getElementById(xid);}

function num(Num){

var txt = $('result');

if(Boo){

txt.value =Num;

Boo=false;

}else{

if(txt.value=='0'){txt.value=Num;}else{txt.value+=Num;}}

}

function decimal(){

var txt = $('result');

if(Boo){

txt.value='0';Boo=false;

}else{

if(txt.value.indexOf('.')==-1){

txt.value+='.';

}

}

}

function compute(op){

var txt = $('result');

var onum = txt.value;

if(txt.value==''){txt.value=0};

Boo=true;

switch(ope){

case '+':

result += parseFloat(onum);break;

case '-':

result -= parseFloat(onum);break;

case '\*':

result \*= parseFloat(onum);break;

case '/':

result /= parseFloat(onum);break;

case '=':

result =parseFloat(onum);break;

case '%':

result %= onum;break;

case 'x^y':

result =Math.pow(result,onum);break;

default:

result = parseFloat(onum);break;

}

txt.value=result;

ope=op;

}

function plusMinus(){

var ounm = $('result').value;

if(onum=''){

alert('不可以为空取反');

}else{

$('result').value\*=-1;

}

}

function tuige(){

var txt = $('result');

txt.value=txt.value.substring(0,txt.value.length-1);

}

function jecheng(a){

if(a==1){return 1;}else{return jecheng(a-1)\*a}

}

function math(op){

var onum=$('result').value;

if(onum==''){alert('数据不可以为空')};

Boo = true;

with(Math){

switch(op){

case 'sin':result=sin(onum);break;

case 'cos':result=cos(onum);break;

case 'tan':result=tan(onum);break;

case 'asin':result=asin(onum);break;

case 'acos':result=acos(onum);break;

case 'atan':result=atan(onum);break;

case 'PI':result=PI;break;

case '1/x':result=1/onum;break;

case 'exp':result=exp(onum);break;

case 'inx':result=Inx(onum);break;

case 'lgx':result=log(onum);break;

case 'i':result=floor(onum);break;

case 'n!':result=jecheng(onum);break;

}

}

$('result').value=result;

}

function cle(){

Boo =false;

result =0;

ope=null;

$('result').value='0';

}

</script>

</html>

#### 【绑定数组弹出颜色】原代码：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<style type="text/css">

div{

width: 80px;

height: 80px;

font-size: 20px;

text-align: center;

border:4px solid #222;

border-radius: 5px;

margin: 3px;

float: left;

}

</style>

<body>

<div id="div0" style="background: red">red</div>

<div id="div1" style="background: green">green</div>

<div id="div2" style="background: blue">blue</div>

<div id="div3" style="background: orange">orange</div>

</body>

<!-- <script type="text/javascript">

var colorarr=["red","green","blue","orange"];

var divs = document.getElementsByTagName("div");

/\*var divs = document.querySelectorAll('div');\*/

for(var i=0;i<divs.length;i++){

(function(l){divs[l].onclick=function(){alert(colorarr[l])}})(i)

}

alert(divs.length);

</script> -->

<script>

var colorarr = ["red","green","blue","orange"];

var divs = document.querySelectorAll("div");

for(var i=0;i<divs.length;i++){

(function(){

var c=i;

divs[i].onclick=function(){alert(colorarr[c])}

})()

}

//这种写法i将以局部变量c的方式保存；

</script>

</html>

#### 【弹出框】原代码

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<style type="text/css">

#box{

width: 300px;

height: 200px;

background: #eee;

border: solid black 1px;

border-radius: 2px;

display: none;

}

#box h3{

color: white;

margin-top: 1px;

background: orange;

}

#box span{

background: rgba(100,100,100,0.4);

border-radius: 2px;

float: right;

cursor: pointer;

}

#box p{

padding: 10px;

text-indent: 20px;

word-wrap: break-word;

word-break:keep-all;

}

</style>

</head>

<body>

<button onclick="show()">出来吧！</button>

<div id="box">

<h3>标题<span id="clo">关闭</span></h3>

<p>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*</p>

</div>

</body>

<script type="text/javascript" src="../JQuery/jquery-3.1.1.js"></script>

<script type="text/javascript">

function show(){$('#box').css('display','block')}

$('#clo').click(function close(){$('#box').css('display','none')})

</script>

</html>

#### 【选项卡】原代码

方案1：【CSS方案】

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<style type="text/css">

#xuanxiangka \*{

margin:0px;

}

#xuanxiangka{

position: absolute;

height: 200px;

width: 300px;

border: solid orange 1px;

background: #eee;

}

#list{

width: 100%;

height: 20px;

overflow: hidden;

}

#list a{

width: 100px;

height: 20px;

text-decoration: none;

padding:5px;

font-size: 14px;

background: orange;

margin: 0px;

text-align: center;

}

#list a:hover{

opacity: 0.5;

}

#content{

overflow: hidden;

height: 180px;

width: 300px;

}

#content div{

padding: 10px;

background: #ddd;

height: 180px;

width: 300px;

border-top: dashed orange 1px;

}

</style>

<body>

<div id="xuanxiangka">

<div id="list">

<a href="#tab01">选项卡1</a>

<a href="#tab02">选项卡2</a>

<a href="#tab03">选项卡3</a>

</div>

<div id="content">

<div id="tab01">div01:\*\*\*\*\*\*\*\*\*\*\*</div>

<div id="tab02">div02:\*\*\*\*\*\*\*\*\*\*\*</div>

<div id="tab03">div03:\*\*\*\*\*\*\*\*\*\*\*</div>

</div>

</div>

</body>

</html>

方案1：【javaScript方案】

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<style type="text/css">

#xuanxiangka{

width: 300px;

height: 200px;

background: #eee;

border: black solid 1px;

}

#list{

width: 300px;

height: 20px;

overflow: hidden;

}

#list a{

text-decoration: none;

padding: 5px;

width: 100px;

height: 20px;

background: orange;

}

#list a:hover{

opacity: 0.4;

}

#content{

width: 300px;

height: 180px;

border-top: dashed orange 1px;

}

#content div{

width: 300px;

height: 180px;

padding: 10px;

}

</style>

<body>

<div id="xuanxiangka">

<div id="list">

<a href="javascript:void(0);">选项卡01</a>

<a href="javascript:void(0);">选项卡02</a>

<a href="javascript:void(0);">选项卡03</a>

</div>

<div id="content">

<div>内容1：\*\*\*\*\*\*\*\*\*\*\*</div>

<div>内容2：\*\*\*\*\*\*\*\*\*\*\*</div>

<div>内容3：\*\*\*\*\*\*\*\*\*\*\*</div>

</div>

</div>

</body>

<script type="text/javascript">

var ass = document.getElementById('list').getElementsByTagName('a');

var cs = document.getElementById('content').getElementsByTagName('div');

for(var i=0;i<ass.length;i++){

if(i!=0){

cs[i].style.display='none';

}

ass[i].index=i;

ass[i].onclick=function(){

for(var inde=0;inde<cs.length;inde++){

cs[inde].style.display='none';

}

cs[this.index].style.display='block';

}

}

</script>

</html>