**jQuery**

**jQuery简介**

1. jQuery并不是一门语言，而是基于JS、DOM、HTML、CSS所实现的一堆丰富功能的代码的封装
2. jQuery框架是当前应用非常广泛的一个js类库，里面封装了大量的方法提供给我们快速的完成常用功能的开发，并且体积小，使用方便
3. 使用jQuery框架能够很解决不同浏览器对DOM实现的不同所造成的差异
4. 只有引用了jquery.js的文件于HTML代码中才能够使用jQuery提供给我们的一些丰富的功能
5. jQuery所封装的函数包含的主要功能

-> JS对象、DOM对象的处理

-> 操作属性、CSS

-> 选择器、筛选

-> 文档处理

-> 事件处理机制（是基于DOM事件的封装和实现，jQuery本身并无事件）

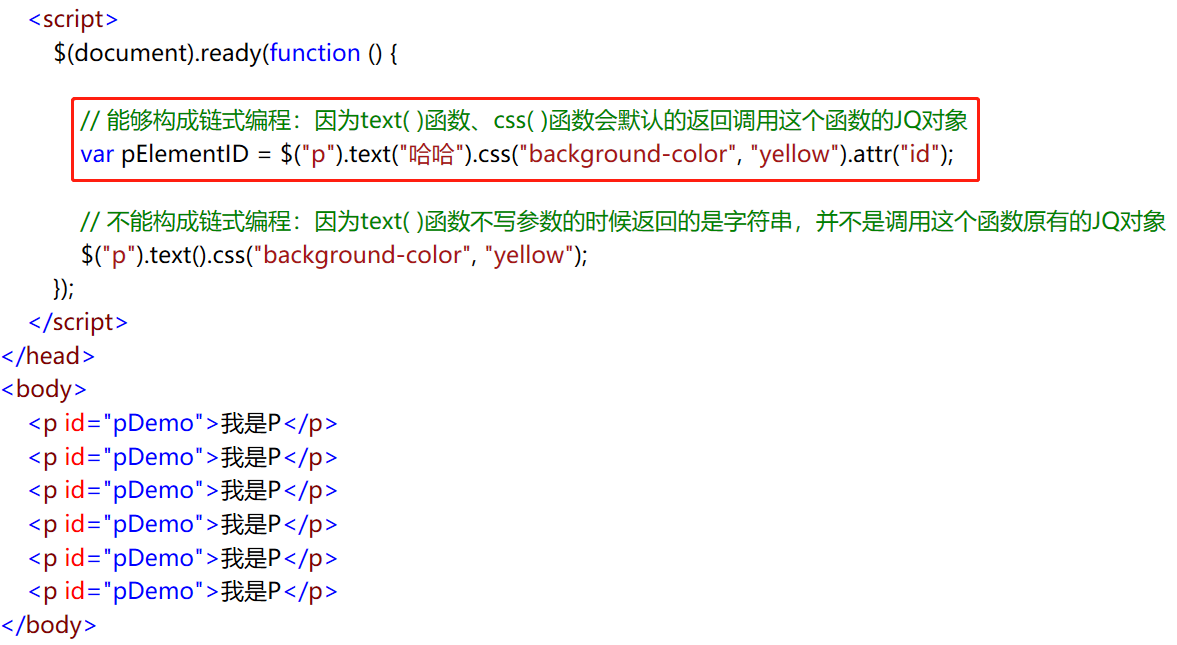
-> 效果

-> ajax实现异步操作

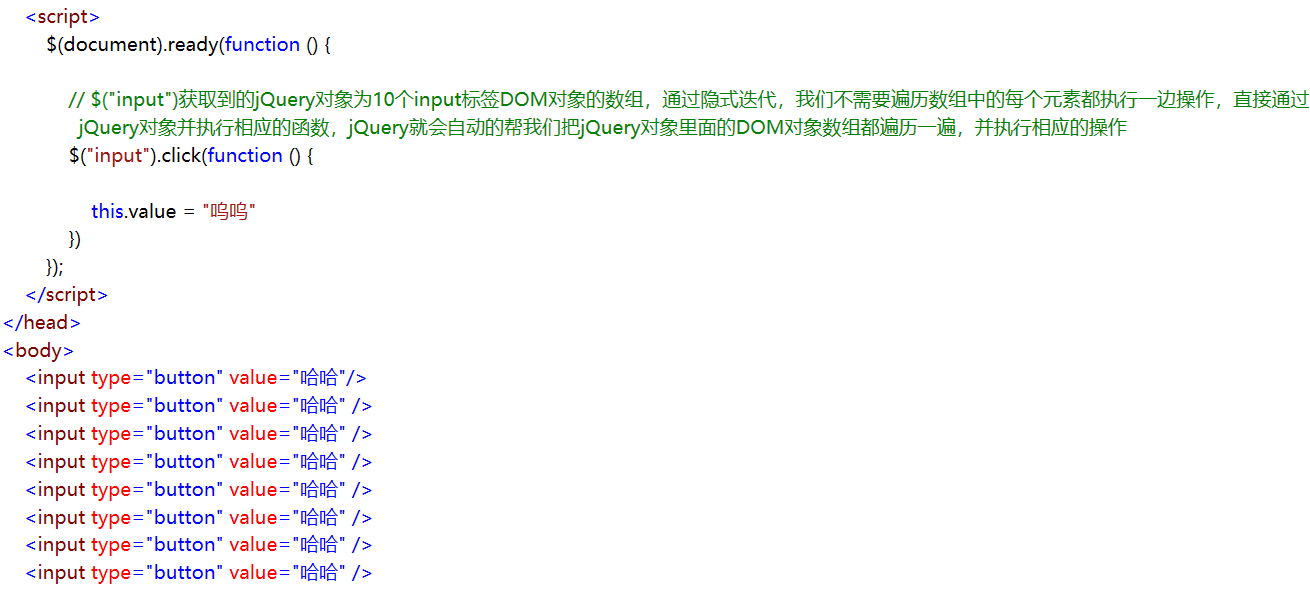
-> 常用的一些工具类

**jQuery框架提供了一些丰富的功能**

1. 链式编程： jQuery中的函数如无特定的返回值，如：text()函数，都会默认的返回调用当前函数的JQ对象，也就是说我们可以利用这一机制，使用返回的JQ对象继续调用其他的函数并作相应的处理，这种方式就叫链式编程
2. 一些函数是有特定类型返回值的，如：字符串，当返回这些值得时候肯定不能使用链式编程了，因为返回的不是JQ对 象所以肯定访问不到jQuery中封装的函数
3. 在一条链中，如果一些函数会返回了其他的JQ对象，这时候就叫做：**链的破坏**，我们能使用**end()函数**来恢复最近一 次被破坏的链.，也就是返回原本承载着这条链的JQ对象



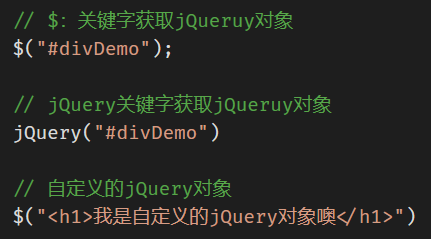
1. 隐式迭代： 不需要遍历jQuery对象中的每一个DOM对象元素执行相应的操作，直接通过jQuery对象并调用相应的函数，jQuery内部会自动的帮我们把jQuery对象里面的每一个DOM对象元素都遍历一遍，并执行相应处理函数



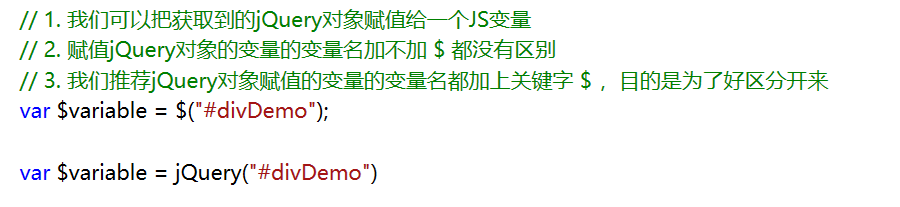
1. 自定义丰富的插件（包括第三方jQuery插件的引用）

**jQuery对象**

1. 通过jQuery选择器，获取符合条件的DOM对象并将之包装为一个数组，这个DOM对象的数组就是jQuery对象
2. 只有通过jQuery对象才能使用jQuery所封装的一系列丰富的函数
3. jQuery对象不能使用DOM对象中的属性和函数，除非通过jQuery对象的下标去访问jQuery对象里的DOM对象，反之，DOM对象不能够使用jQuery对象中的属性和函数，除非通过jQuery所封装的转换DOM对象转换jQuery对象的函数把DOM对象转换成jQuery对象
4. jQuery对象需要通过关键字（$ 或 jQuery）再加上一些相应的选择器才能获取，除此之外，我们还能通过该关键字来声明一个自定义的jQuery对象



1. 我们可以把获取到的jQuery对象赋值给一个JS变量，变量名不加 $ 关键字也能表示该变量为jQuery对象的变量，但是我们推荐jQuery对象所赋值的变量的变量名前面都加上 $，目的是为了让我们区分开来jQuery对象和其他的对象



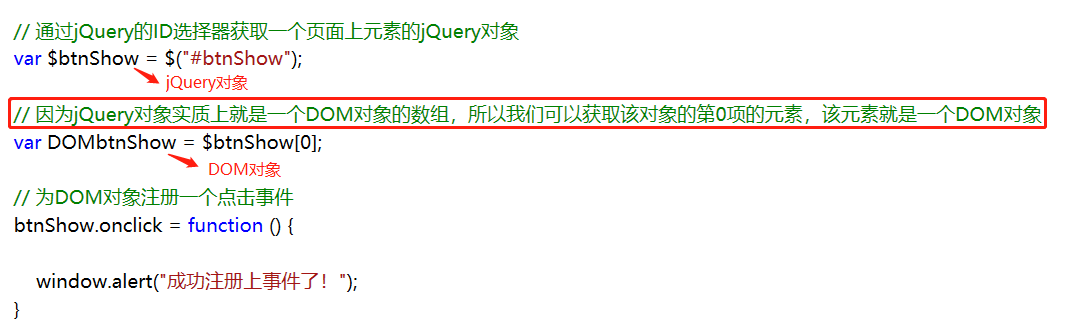
**jQuery的具体使用**

1. **选择器：匹配页面上的HTML元素，并返回该元素的jQuery对象**

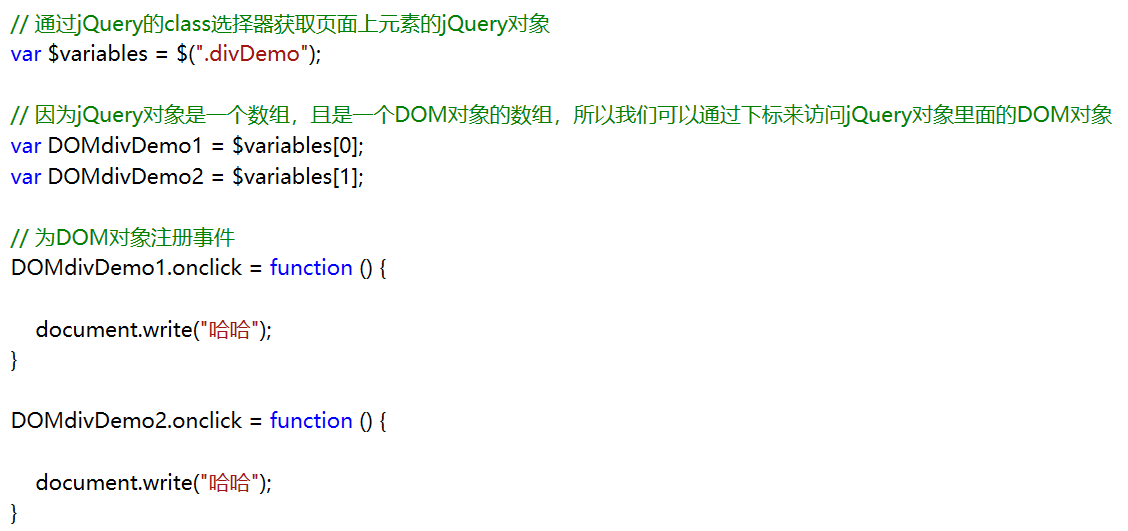
**-> 通过jQuery选择器获取到的对象就是jQuery对象，本质上就是DOM对象的一个数组**

**-> jQuery选择器不管是否能获取到与选择器的条件相匹配的DOM对象，都会返回一个数组，但是这个数组里没有元素**

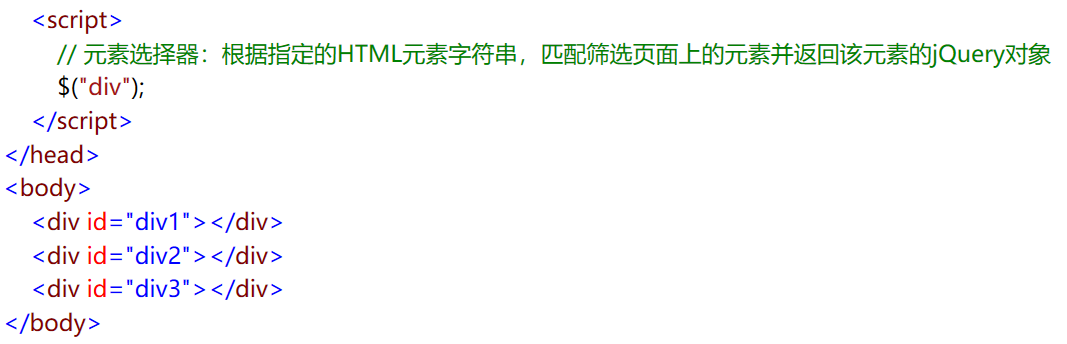
1. 基本选择器
2. $( "#ID" )：ID选择器，根据指定的ID匹配一个HTML元素，并返回该元素的jQuery对象



1. $( ".Class" )：class选择器，根据指定的class匹配HTML元素，并返回匹配成功元素的jQuery对象



1. $( "Element" )：元素选择器，根据指定的HTML元素字符串，返回匹配成功的jQuery对象

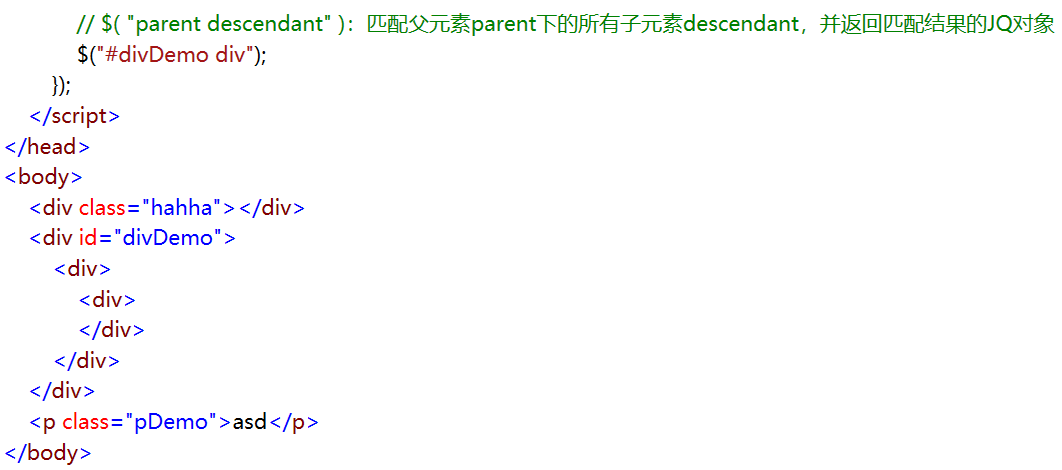


1. 通过 ，分隔多个基本选择器，获取指定的多个选择器所匹配筛选出的jQuery对象

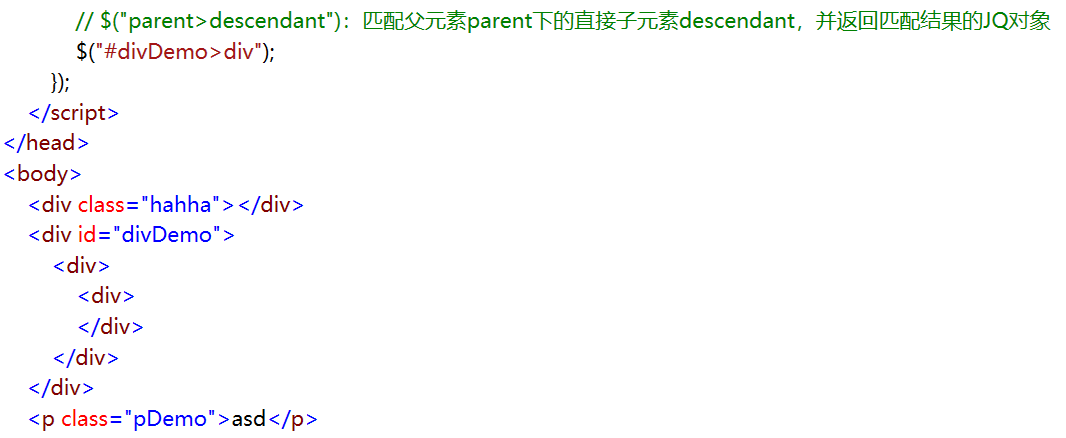
-> 该方式只能用于基本选择器



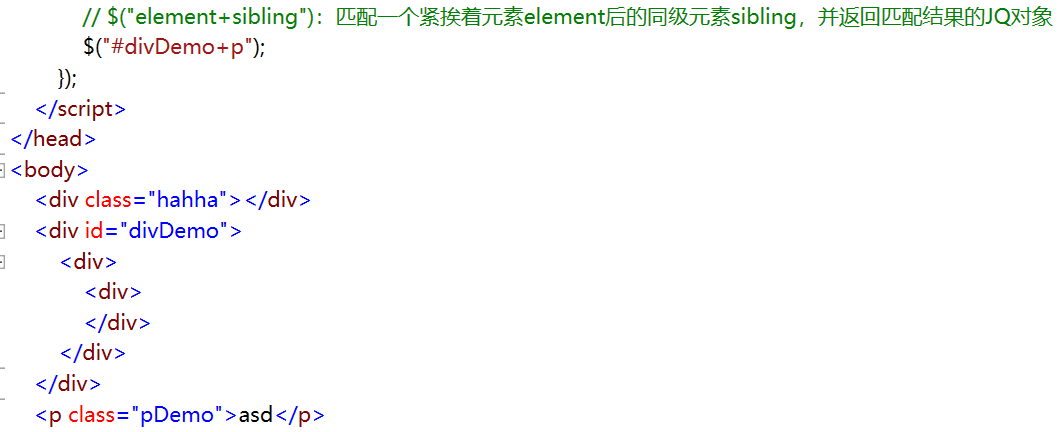
1. 层级选择器
2. $( "parent descendant" )：匹配父元素parent下符合选择器表达式descendant的所有子元素，并返回匹配结 果的JQ对象



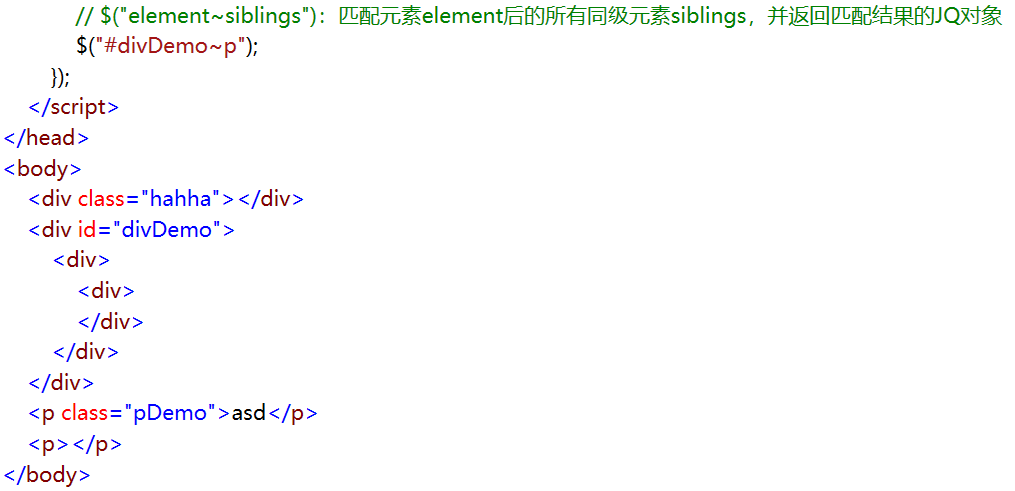
1. $( "parent>descendant" )：匹配父元素parent下符合选择器表达式descendant的直接子元素，并返回匹配结 果的JQ对象



1. $( "element+sibling" )：匹配一个紧挨着元素element后面的符合选择器表达式sibling的同级元素，并返回匹 配结果的JQ对象

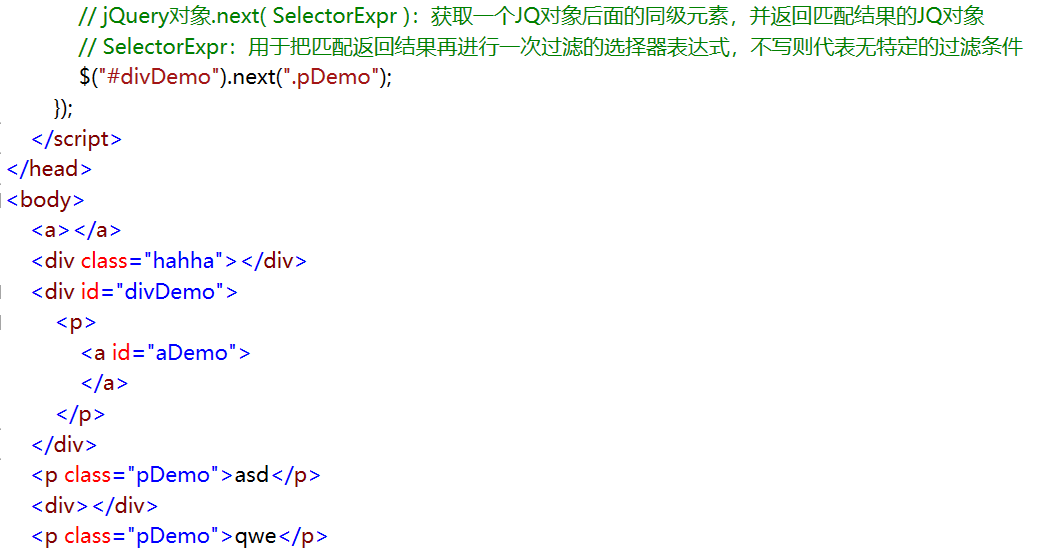


1. $( "element~siblings" )：匹配元素element后面的符合选择器表达式sibling的所有同级元素，并返回匹配结 果的JQ对象



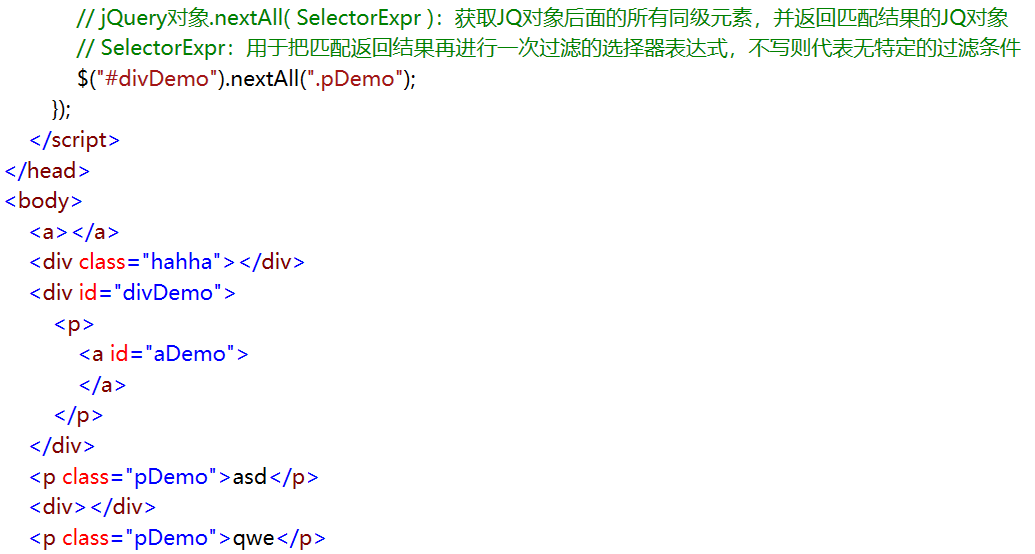
1. 层级筛选器
2. jQuery对象.next( SelectorExpr )：获取一个JQ对象后面的同级元素，并返回匹配结果的JQ对象

-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件



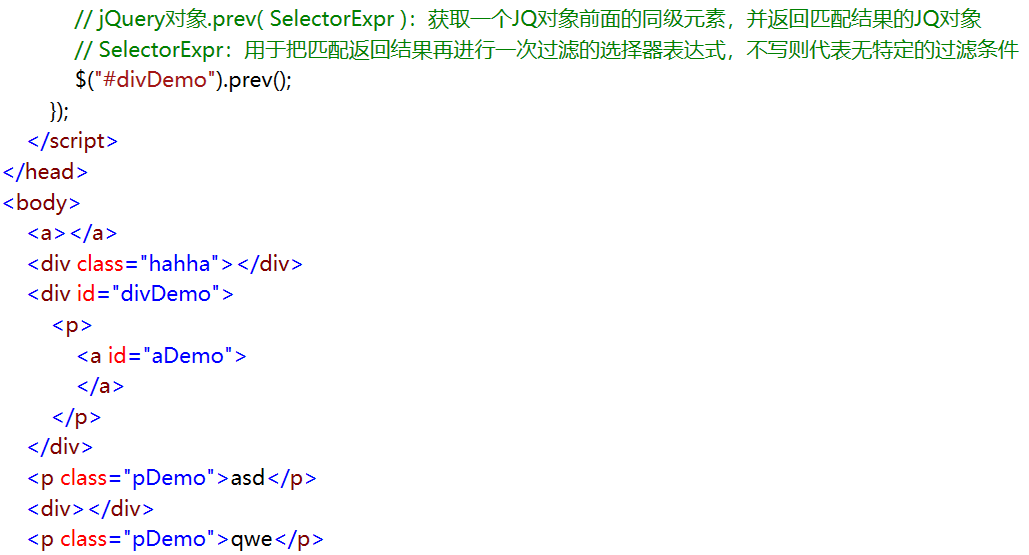
1. jQuery对象.nextAll( SelectorExpr )：获取JQ对象后面的所有同级元素，并返回匹配结果的JQ对象

-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件



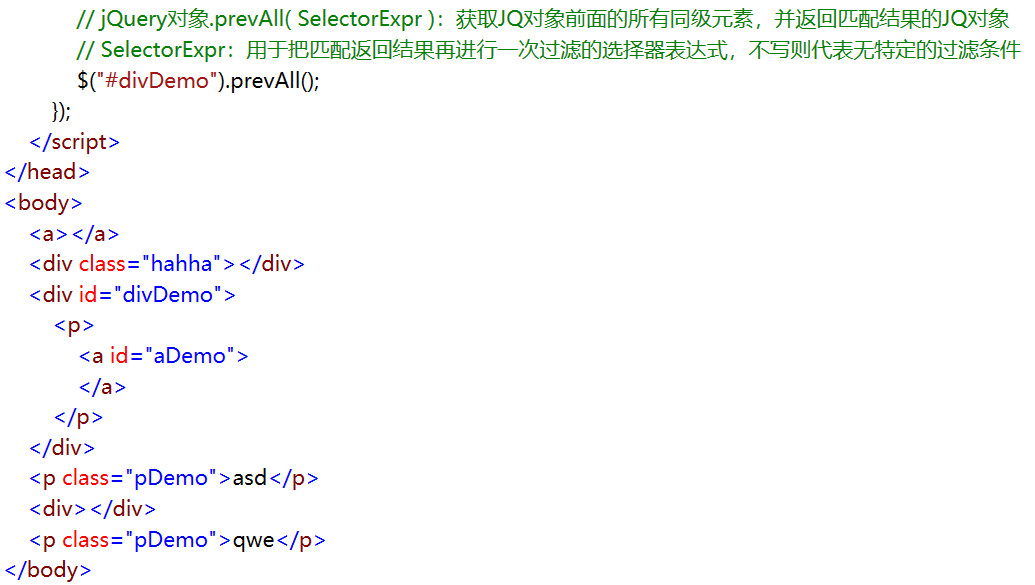
1. jQuery对象.prev( SelectorExpr )：获取一个JQ对象前面的同级元素，并返回匹配结果的JQ对象

-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件



1. jQuery对象.prevAll( SelectorExpr )：获取JQ对象前面的所有同级元素，并返回匹配结果的JQ对象

-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件



1. jQuery对象.siblings( SelectorExpr )：获取JQ对象的所有同级元素，并返回匹配结果的JQ对象

-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件



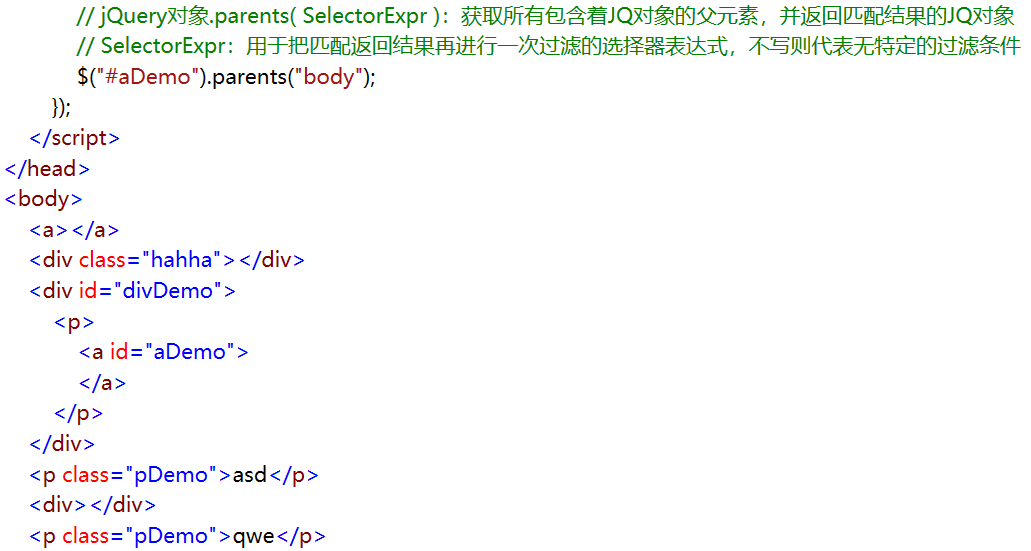
1. jQuery对象.parent( SelectorExpr )：获取JQ对象的直接父元素，并返回匹配结果的JQ对象

-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件



1. jQuery对象.parents( SelectorExpr )：获取所有包含着JQ对象的父元素，并返回匹配结果的JQ对象

-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件

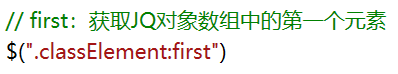


1. jQuery对象.children( SelectorExpr )：获取JQ对象的所有直接子元素，并返回匹配结果的JQ对象

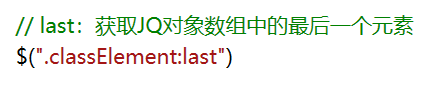
-> SelectorExpr：用于把匹配返回结果再进行一次过滤的选择器表达式，不写则代表无特定的过滤条件



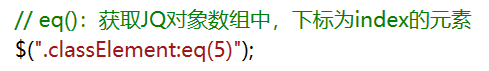
1. 过滤选择器（表达式）
2. $( "选择器表达式 : first" )：获取JQ对象数组中的第一个元素



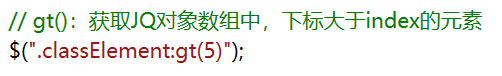
1. $( "选择器表达式 : last" )：获取JQ对象数组中的最后一个元素



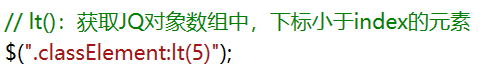
1. $( "选择器表达式 : eq(index)" )：获取JQ对象数组中，下标为index的元素



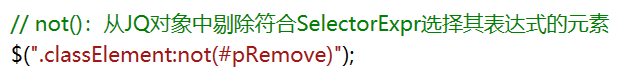
1. $( "选择器表达式 : gt(index)" )：获取JQ对象数组中，下标大于index的元素



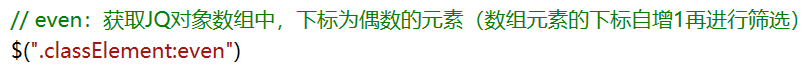
1. $( "选择器表达式 : lt(index)" )：获取JQ对象数组中，下标小于index的元素



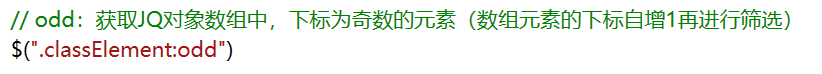
1. $( "选择器表达式 : not( SelectorExpr )" )：从JQ对象中剔除符合SelectorExpr选择器表达式的元素，并返回筛 选后的JQ对象



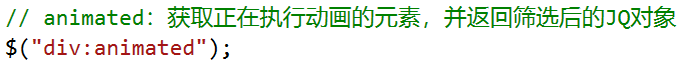
1. $( "选择器表达式 : even" )：获取JQ对象数组中，下标为偶数的元素（数组元素的下标自增1再进行筛选）



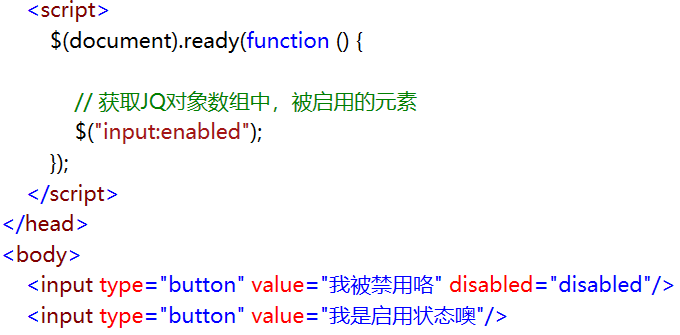
1. $( "选择器表达式 : odd" )：获取JQ对象数组中，下标为奇数的元素（数组元素的下标自增1再进行筛选）



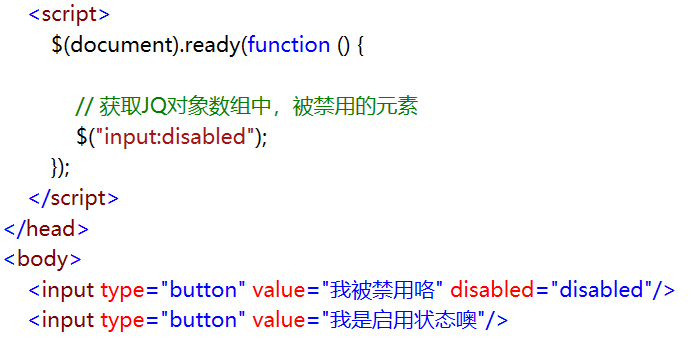
1. $("选择器表达式 : animated")：获取正在执行动画的元素，并返回筛选后的JQ对象



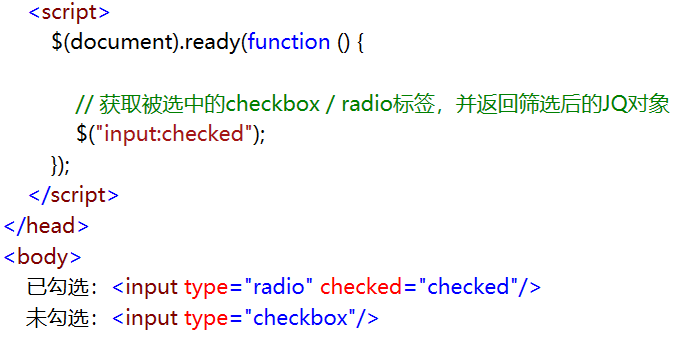
1. 过滤选择器（表达式、针对表单）
2. $( "选择器表达式 : enabled" )：获取JQ对象数组中，被启用的元素



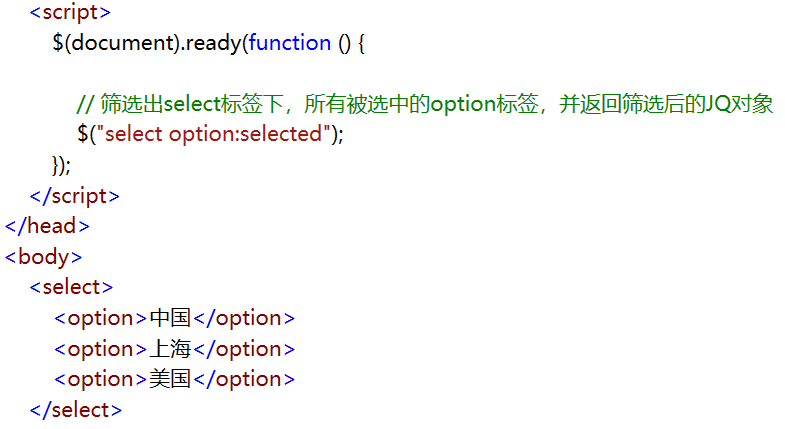
1. $( "选择器表达式 : disabled" )：获取JQ对象数组中，被禁用的元素



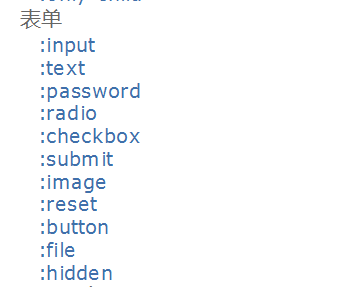
1. $( "选择器表达式 : checked" )：获取被选中的checkbox / radio标签，并返回筛选后的JQ对象



1. $( "select option : selected" )：筛选出select标签下，所有被选中的option标签，并返回筛选后的JQ对象



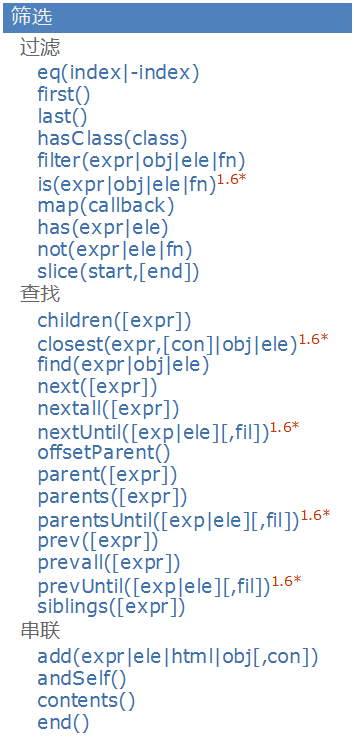
1. 一些表单选择器，具体用法可以查看API文档说明



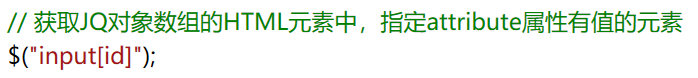
1. 过滤选择器（函数）
2. jQuery对象.find( "Selector" )：使用选择器表达式Selector找出JQ对象的子元素，并返回该子元素的JQ对象



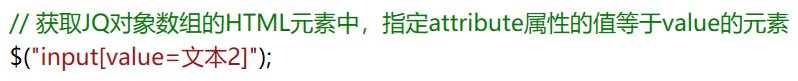
1. 具体参照API文档，作用与表达式形式的过滤选择器一样，只是使用场景不同



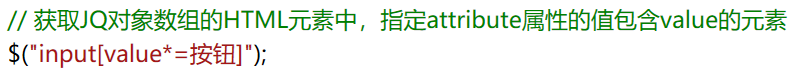
1. 属性选择器
2. $( "选择器表达式[attribute]" )：获取JQ对象数组的HTML元素中，指定attribute属性有值的元素



1. $( "选择器表达式[attribute=value]" )：获取JQ对象数组的HTML元素中，指定attribute属性的值等于value 的元素



1. $( "选择器表达式[attribute\*=value]" )：获取JQ对象数组的HTML元素中，指定attribute属性的值包含value 的元素



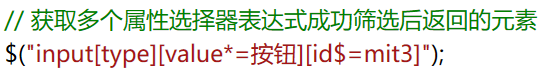
1. $( "选择器表达式[attribute^=value]" )：获取JQ对象数组的HTML元素中，指定attribute属性的值的开头等 于value的元素



1. $( "选择器表达式[attribute$=value]" )：获取JQ对象数组的HTML元素中，指定attribute属性的值的末尾等 于value的元素



1. $( "选择器表达式[attributeSelector1][attributeSelector2]……[attributeSelectorN]" )：获取多个属性选择器表 达式成功筛选后返回的元素



1. 组合选择器的使用

-> $( "组合选择器表达式" )：根据组合选择器表达式，返回匹配成功后的jQuery对象

-> 组合选择器表达式的筛选过滤顺序为：**从左至右**

****

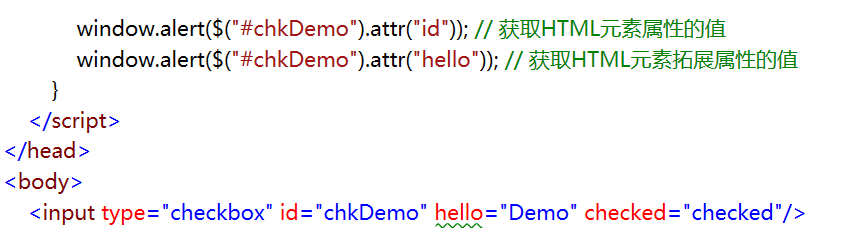
1. **操作属性、CSS：用于操作或移除元素属性的值和该元素的CSS样式**

**-> jQuery所操作的属性只是元素本身的属性，并不是该元素DOM对象所带有的属性**

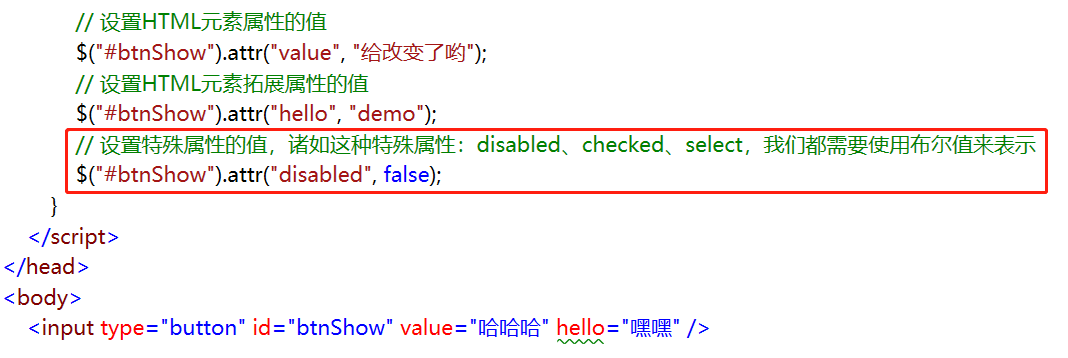
1. attr / removeAttr：获取设置或移除HTML元素的属性和扩展属性

-> 了解：该函数虽能够获取到HTML元素本身自带的属性和我们所写的拓展的属性的值，但是所获取到的值只是我们在代码中所写的属性的值的文本，比如说我们设置了HTML元素的disabled属性的值为"哈哈"，那么通过attr函数获取到该属性的值也是"哈哈"，并不能提取到该元素更深一层内部所制定该属性disabled的值（在HTML元素的内部，该属性的值其实为true/false），所以诸如特殊的属性：**disabled、checked、select**，使用**prop函数**更为专业

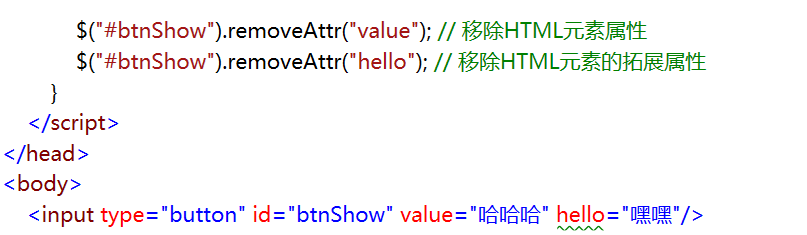
1. jQuery对象.attr( "属性名" )：获取指定HTML元素的属性或拓展属性的值



1. jQuery对象.attr( "属性名" , "属性值" )：设置指定HTML元素的属性或拓展属性的值



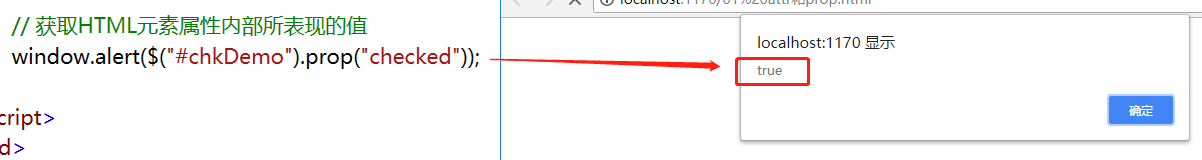
1. jQuery对象.removeAttr( "属性名" )：移除指定HTML元素的属性或拓展属性**（移除整个属性，并不是属性的值）**



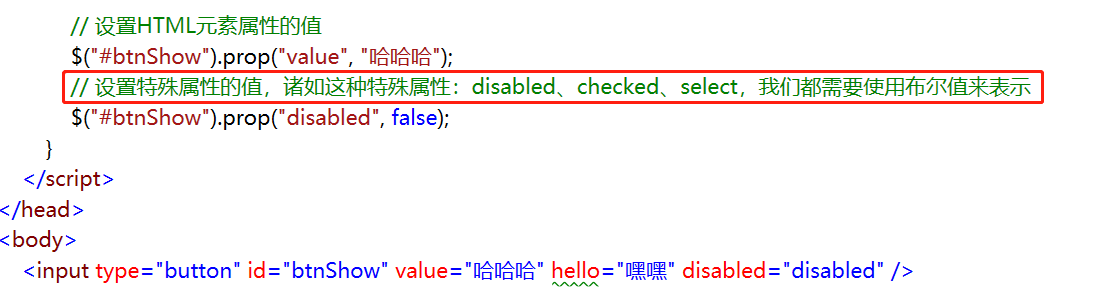
1. prop/ removeProp：获取设置或移除HTML元素的属性

-> 了解：该函数虽然只能够获取到HTML元素属性的值，并不能获取到我们所写的HTML元素拓展属性的值，但是，如上attr函数所带来的弊端，prop函数所获取到的HTML元素属性的值，都是HTML元素内部真实的值，如：我们所指定的HTML元素的disabled的属性值为 "disabled"，那么该属性在HTML元素内部所表现的值其实为true/false，prop函数就能够达到这一目的，所以诸如特殊的属性：**disabled、checked、…… 、selected** 使用**prop函数**更为专业

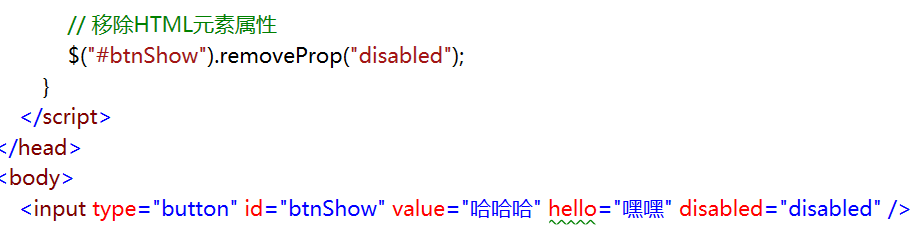
1. jQuery对象.prop( "属性名" )：获取指定HTML元素的属性的内部属性的值



1. jQuery对象.prop( "属性名" , "属性值" )：设置指定HTML元素的属性



1. jQuery对象.removeAttr( "属性名" )：移除指定HTML元素的属性**（移除整个属性，并不是属性的值）**

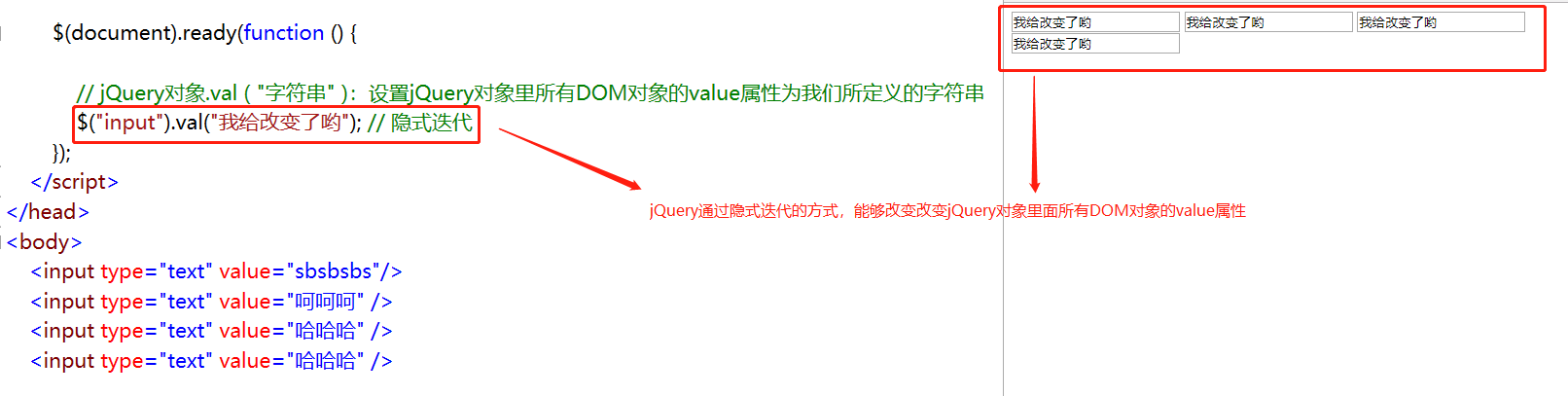


1. val()：获取或设置HTML标签的value属性
2. jQuery对象.val( )：获取JQ对象里DOM对象的HTML标签value属性的值

-> 如果jQuery对象里存在多个DOM对象元素，则**只获取下标为0的DOM对象HTML标签的value属性**



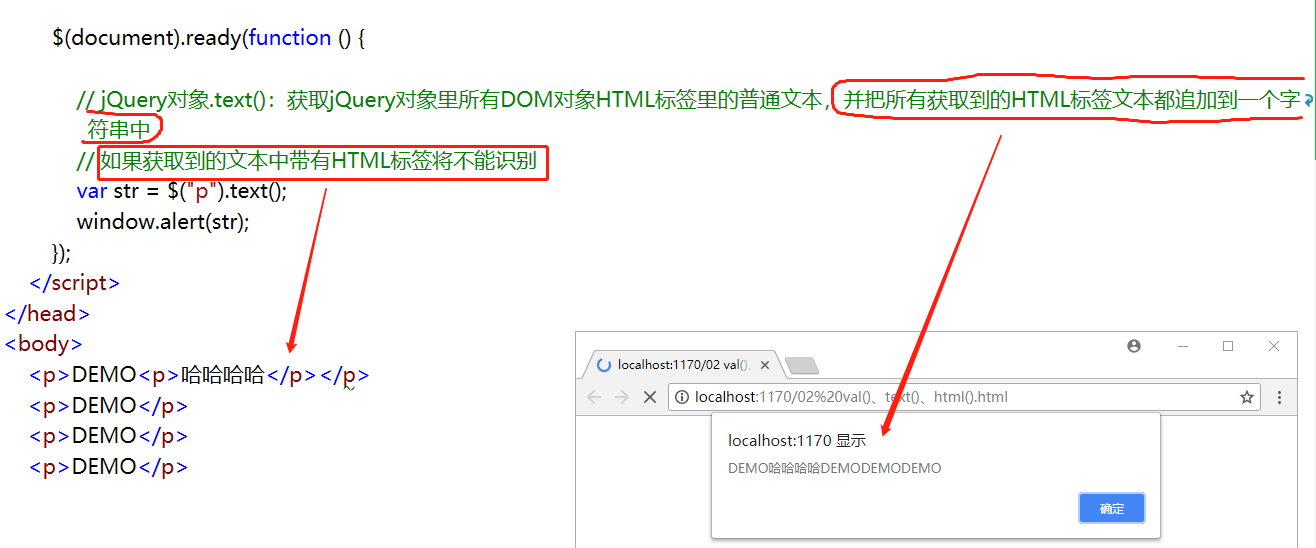
1. jQuery对象.val ( "字符串" )：设置JQ对象里所有DOM对象的HTML标签value属性为我们所定的字符串



1. text()：获取或设置HTML标签的文本内容
2. jQuery对象.text( )：获取JQ对象里DOM对象的HTML标签文本内容

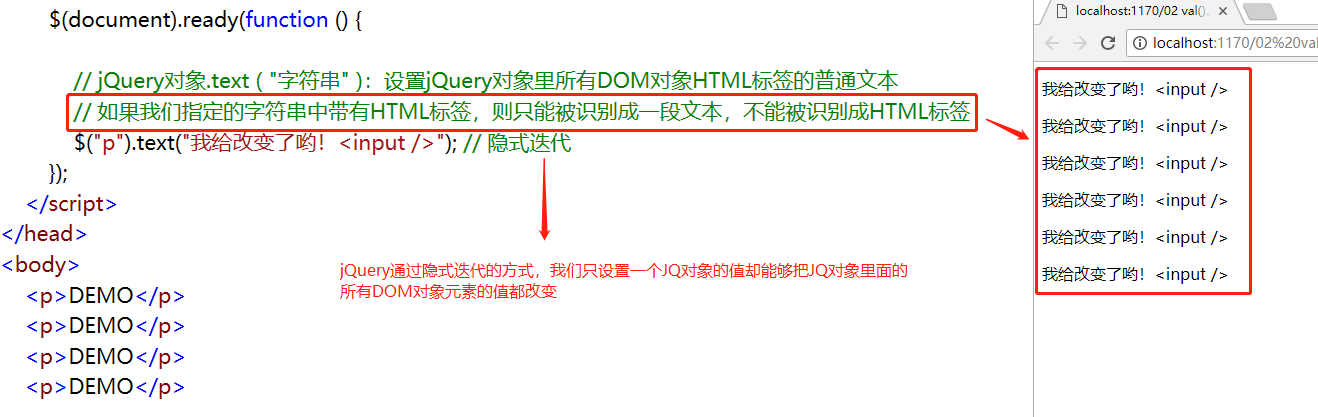
-> 如果JQ对象里存在多个DOM对象元素，则会**获取所有DOM对象的HTML标签文本内容，并追加至一个字符 串中**

-> 如果所定义的字符串里存在HTML标签，则不能够被识别



1. jQuery对象.text ( "字符串" )：设置JQ对象里所有DOM对象的HTML标签文本内容为我们所定义的字符串

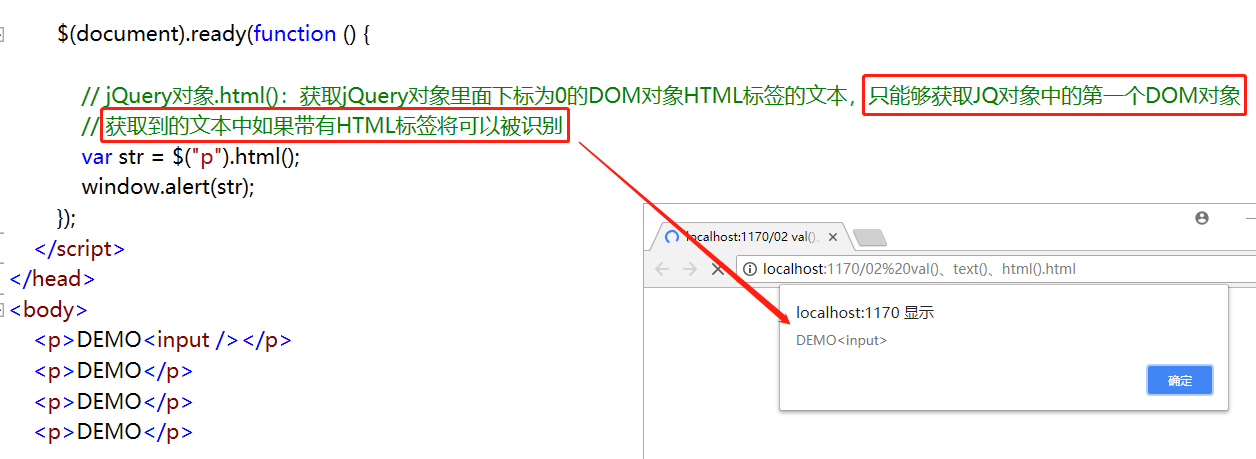
-> 如果所定义的字符串里存在HTML标签，则不能够被识别



1. html()：获取或设置HTML标签的文本内容，能够识别文本中带有的HTML标签
2. jQuery对象.html( )：获取JQ对象里DOM对象的HTML标签文本内容

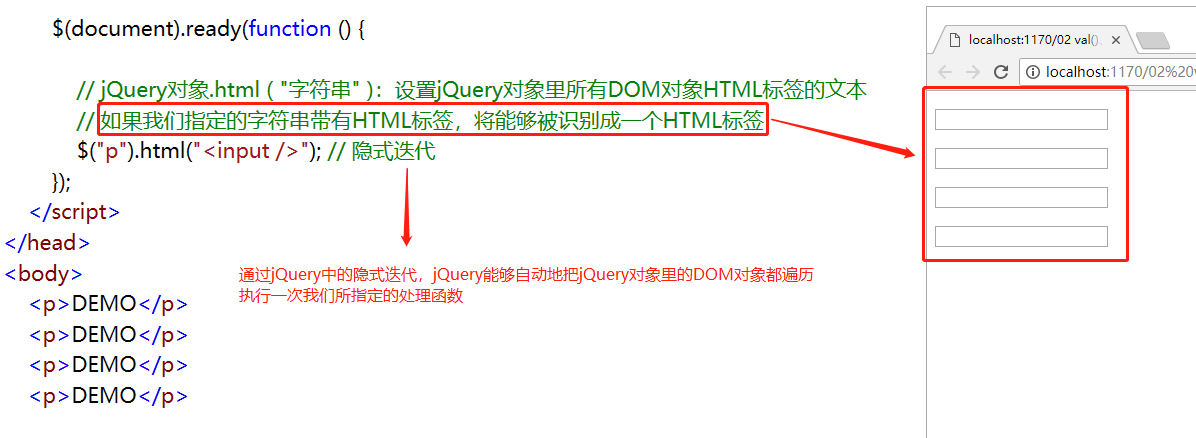
-> 如果JQ对象里存在多个DOM对象元素，则**只会获取下标为0的DOM对象的HTML标签的文本内容**

-> 能够识别所定义的字符串中存在着的HTML标签



1. jQuery对象.html( "字符串" )：设置JQ对象里所有DOM对象的HTML标签文本内容为我们所定义的字符串

-> 能够识别所定义的字符串中存在着的HTML标签



1. **事件处理机制**

**-> jQuery事件处理机制只是对DOM事件的进一层的封装和实现，本身并无事件**

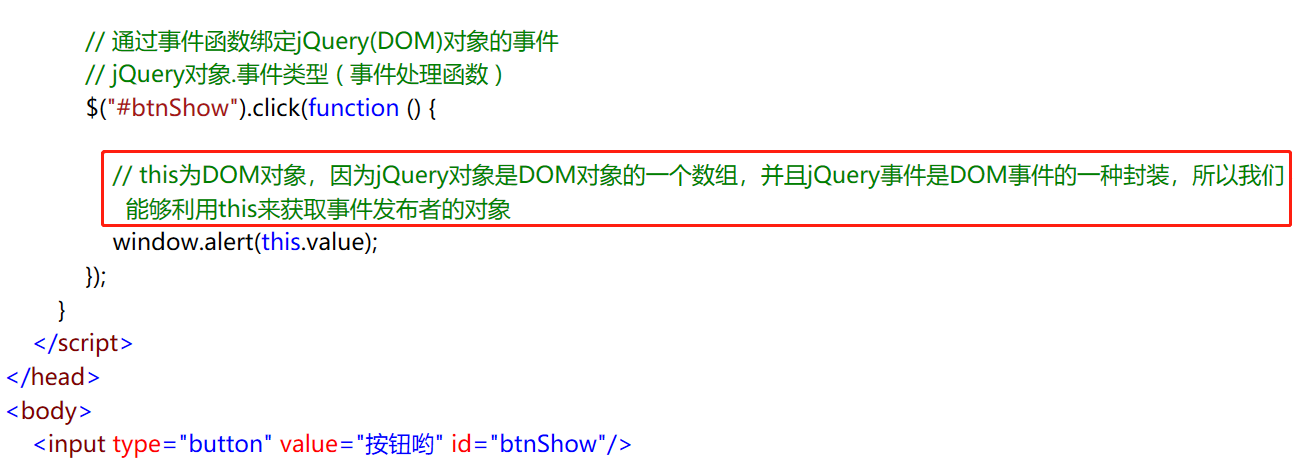
**-> jQuery的事件类型名和DOM事件类型名一样，但是要去掉on的文本，具体可参考API文档**

**-> jQuery事件支持多播形式，也就是说一个事件类型能够绑定一个或多个事件处理函数，而DOM事件是无法实现多播的**

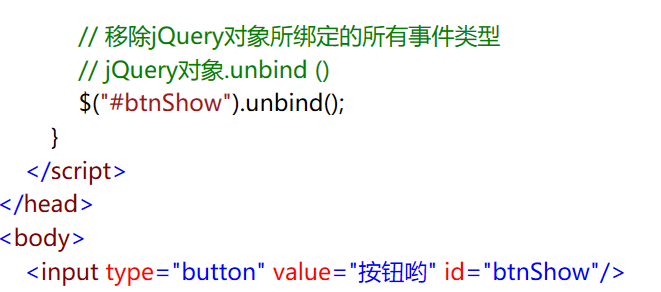
1. 绑定事件的两种方式
2. jQuery对象.bind ( "事件类型名" , 事件处理函数 )：为jQuery对象的指定事件类型绑定事件处理函数



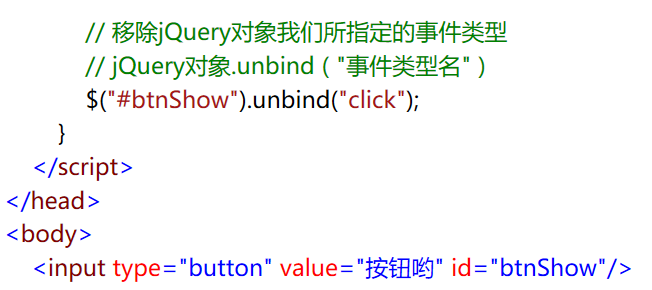
1. jQuery对象.事件类型 ( 事件处理函数 )：为jQuery对象的指定事件类型绑定事件处理函数



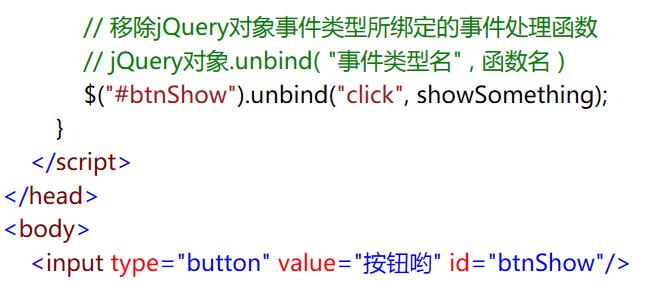
1. 移除事件的三种方式
2. jQuery对象.unbind()：移除jQuery对象所绑定的所有事件类型



1. jQuery对象.unbind ( "事件类型名" )：移除jQuery对象我们所指定的事件类型



1. jQuery对象.unbind( "事件类型名" , 函数名 )：移除jQuery对象事件类型所绑定的事件处理函数



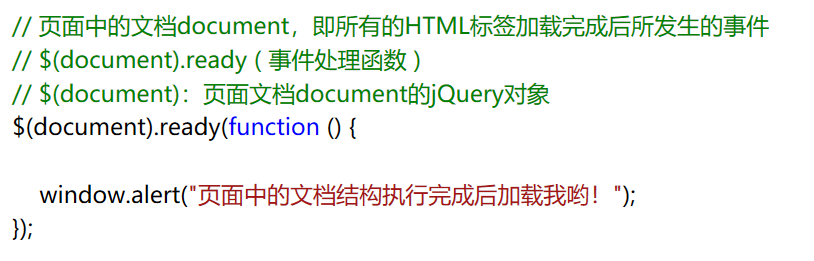
1. jQuery的页面加载事件：ready

-> jQuery页面加载事件ready：当页面中的文档结构加载完成就会执行，即页面中的HTML元素加载完成

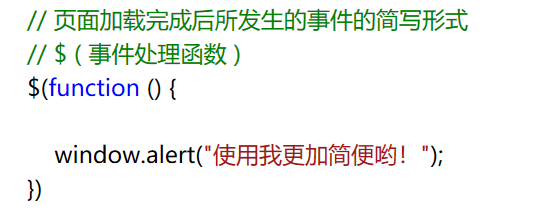
-> DOM页面加载事件onload：当页面中的文档结构和页面中所占用的所有资源加载完成就会执行

-> 了解：该事件也能够用于其他HTML标签，但是这样使用的话就会显得矛盾，第一：如果我们把该事件写在指定该事件发布者的HTML标签的上面，那么就必须在页面加载完成后的事件里面才能为指定的HTML标签注册该事件，如果我们不写在页面加载完成后的事件里面是获取不到指定HTML标签的jQuery对象的，因为文档流的加载原因，这时候的HTML标签是未加载完成的，第二：如果我们写在指定该事件发布者的HTML标签的下面，是能够获取到指定的HTML标签，但是我们写在指定HTML标签的下面的话，HTML标签不都加载完了吗，直接写一段JS脚本不就代表着当该HTML标签加载完成后执行我们所写的JS脚本，哪用获取jQuery对象的等等工作，所以，如果把该事件用在HTML标签中是相当矛盾的

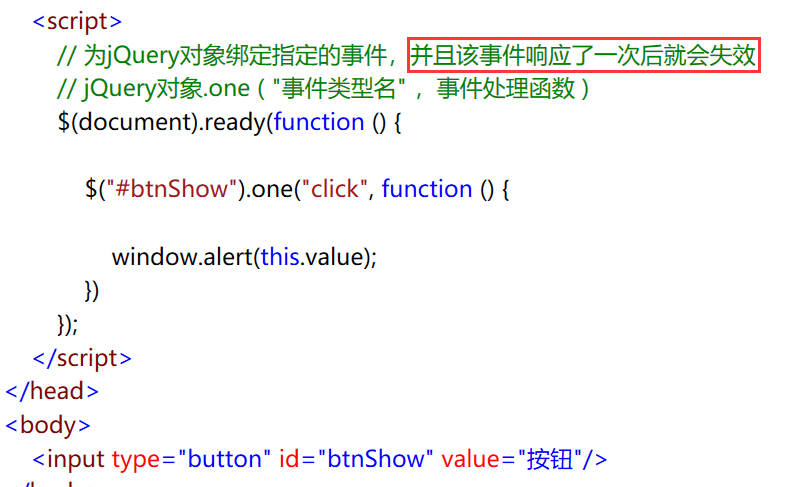
1. jQuery文档对象.ready ( 事件处理函数 )：当页面中的文档结构加载完成执行事件处理函数



1. $ ( 事件处理函数 )：当页面中的文档结构加载完成执行事件处理函数



1. jQuery中的合成事件：合成事件也是jQuery对JS DOM事件的一个封装，我们自己也可以写出来，但是使用它能够减少代码量
2. jQuery对象.one ( "事件类型名" , 事件处理函数)：为jQuery对象的指定事件类型绑定一个事件处理函数，但是 这个事件响应一次后便会失效



1. jQuery对象.hover ( Mouseover事件处理函数 ，Mouseout事件处理函数 )：为指定jQuery对象绑定一个合 成事件，该事件是由Mouseover事件和Mouseout事件所组成，也就是说这一个函数能够完成两个事件的实现



1. jQuery对象.toggle ( 事件处理函数1 , 事件处理函数2，……，事件处理函数N )：为指定jQuery对象绑定一个 点击事件，第一次触发点击事件则执行事件处理函数1，并且每下一次的触发点击事件，则该事件的事件处理函数按我 们所写的顺序执行下一个，直至触发到了第N次点击事件并执行处理函数N，当事件处理函数N的下一次点击事件又 被触发，则回到原点，重新开始执行事件处理函数1，简而言之，这是一个轮流执行我们所规定的事件处理函数的过程



1. jQuery对象.trigger( "事件类型名" )：手动的在其他地方触发JQ对象的指定事件



1. jQueryInput对象.bind( "input propertychange" , 事件处理函数 )：为input标签的JQ对象绑定一个当input标签的value属性发生改变时所发生的事件



1. jQuery中所集成的事件（具体参照API文档）



1. **DOM对象与jQuery对象之间的转换**

**-> jQuery对象本质上就是一个DOM对象的数组，但是，DOM对象是访问不到jQuery对象中的成员的，同理jQuery对象也访问不到DOM对象中的成员，因此我们可以利用一些方案实现它们相互间的转换**

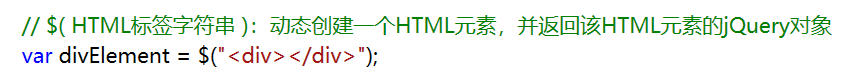
1. $( DOM对象/DOM对象数组 )：把一个DOM对象或者一个DOM对象的数组转换成一个jQuery对象



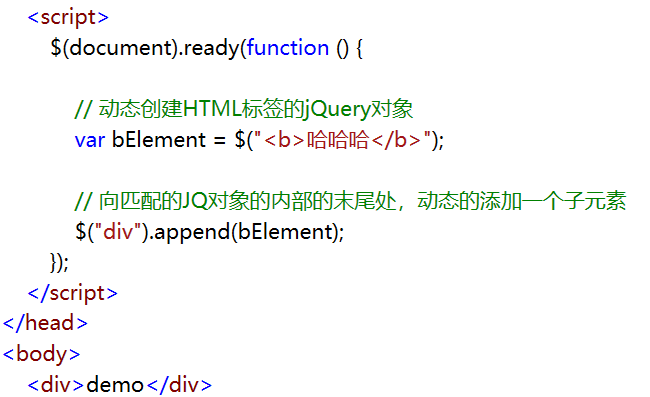
1. jQuery对象[ 下标 ] / jQuery对象.get ( 下标 )：jQuery对象本身就是一个DOM对象的数组，因此我们能通过下标来访问JQ对象中的DOM对象数组中的元素



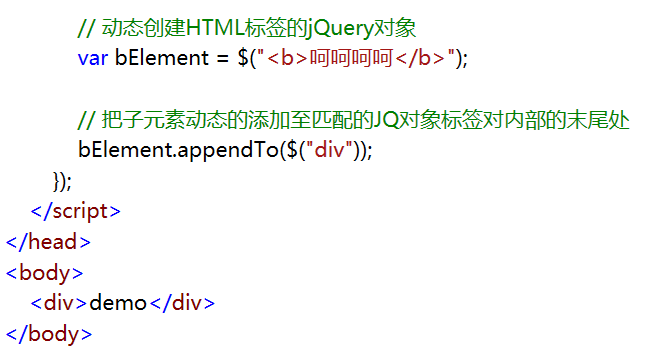
1. **jQuery动态操作元素**
2. $ ( "HTML标签字符串" )：创建指定HTML标签字符串的HTML标签，并返回该HTML标签的jQuery对象



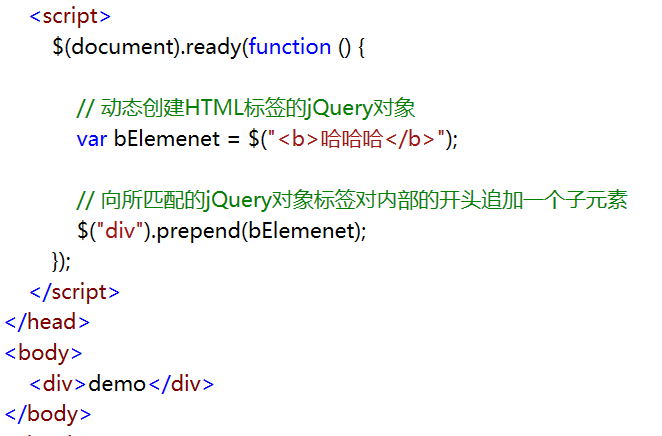
1. Append：末尾追加子元素
2. JQ对象A.append( JQ对象B )：在JQ对象A标签对内部的末尾处，添加一个子元素JQ对象B



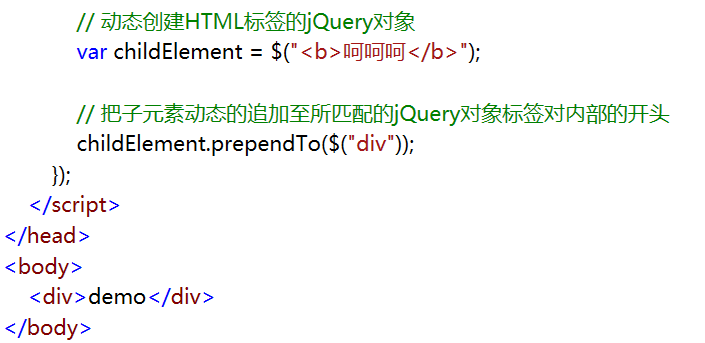
1. JQ对象A.appendTo( JQ对象B )：把子元素JQ对象A添加至JQ对象B标签对内部的末尾处



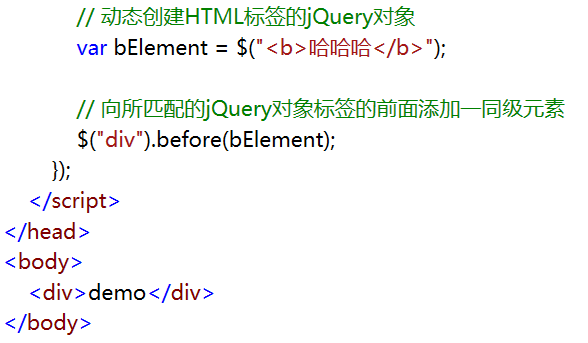
1. prepend：开头追加子元素
2. JQ对象A.prepend( JQ对象B )：在JQ对象A标签对内部的开头处，添加一个子元素JQ对象B



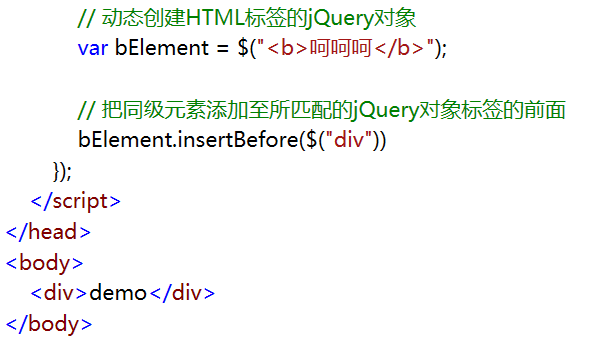
1. JQ对象A.prependto( JQ对象B )：把子元素JQ对象A添加至JQ对象B标签对内部的开头处



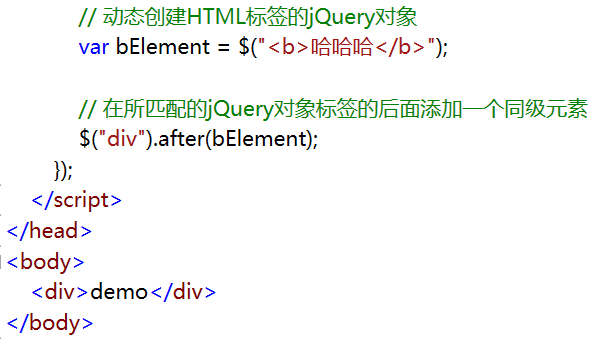
1. Before：开头添加同级元素
2. JQ对象A.before( JQ对象B )：在JQ对象A标签的前面，添加一个同级元素JQ对象B



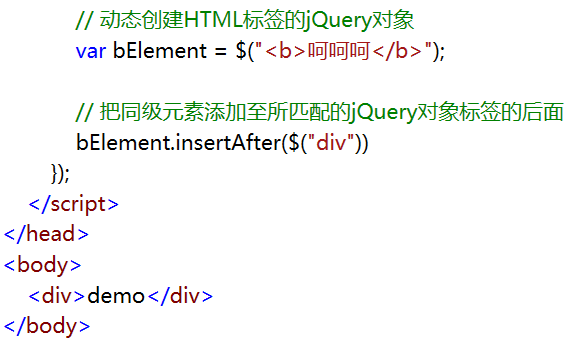
1. JQ对象A.insertBefore( JQ对象B )：把同级元素JQ对象A添加至JQ对象B标签的前面



1. After：末尾添加同级元素
2. JQ对象A.after( JQ对象B )：向JQ对象A标签的后面，添加一个同级元素JQ对象B



1. JQ对象A.insertAfter( JQ对象B )：把同级元素JQ对象A添加至JQ对象B标签的后面



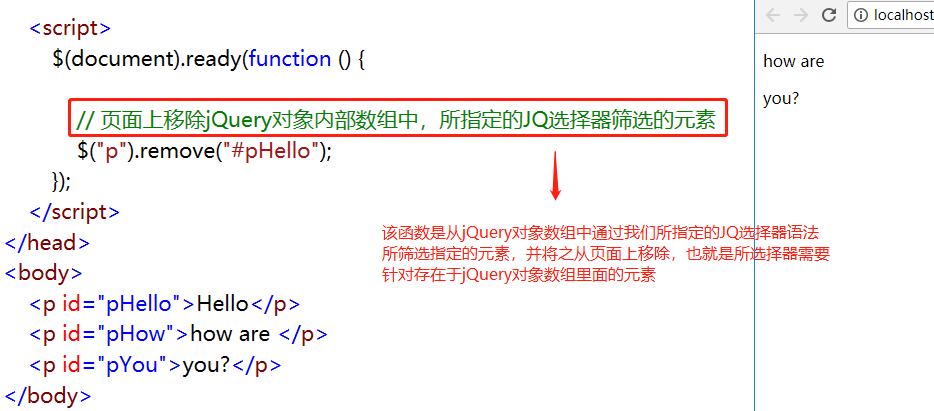
1. jQuery对象.empty()：清空jQuery对象HTML标签内部的所有子元素



1. remove：移除
2. jQuery对象.remove()：从页面上移除jQuery对象里的所有DOM元素所指向的HTML元素



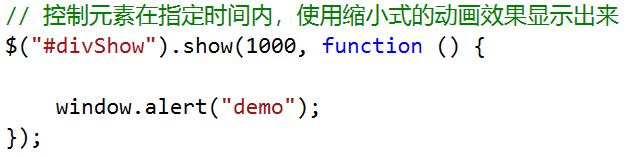
1. jQuery对象.remove ( "JQ选择器语法" )：通过我们所指定JQ选择器的语法，从jQuery对象内部数组中筛选出该元素，并在页面中将至移除（**该JQ选择器语法需仅针对存在于jQuery对象数组中的元素**）



1. **jQuery呈现动画效果的函数**
2. jQuery对象.show( time , function() { } )：控制元素在指定时间内，使用缩小式的动画效果显示出来

-> time：指定元素显示出来所使用的时间，单位为毫秒

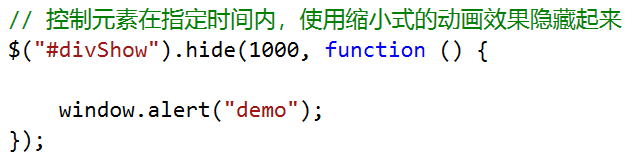
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.hide( time , function() { } )：控制元素在指定时间内，使用缩小式的动画效果隐藏起来

-> time：指定元素隐藏起来所使用的时间，单位为毫秒

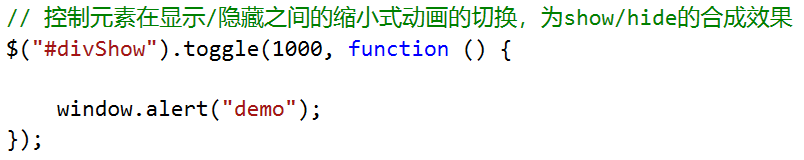
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.toggle( time , function() { } )：控制元素在 显示/隐藏 之间的缩小式动画的切换，为 show/hide 的合成效果

-> time：指定元素 隐藏起来/显示出来 所使用的时间，单位为毫秒

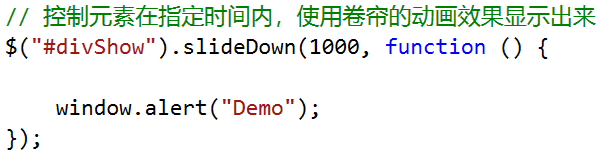
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.slideDown( time , function() { } )：控制元素在指定时间内，使用卷帘的动画效果显示出来

-> time：指定元素显示出来所使用的时间，单位为毫秒

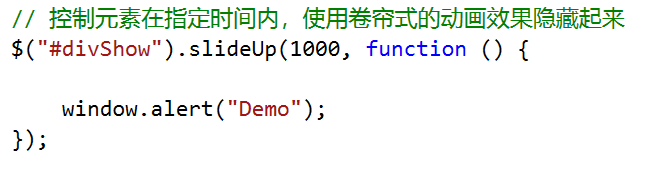
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.slideUp( time , function() { } )：控制元素在指定时间内，使用卷帘式的动画效果隐藏起来

-> time：指定元素隐藏起来所使用的时间，单位为毫秒

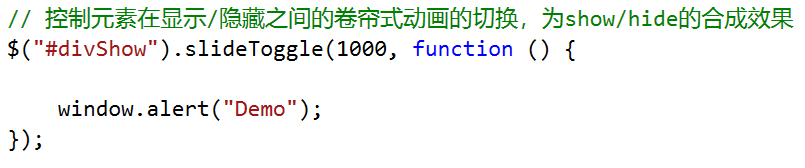
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.slideToggle( time , function() { } )：控制元素在 显示/隐藏 之间的卷帘式动画的切换，为 slideDown/slideUp 的合成效果

-> time：指定元素 隐藏起来/显示出来 所使用的时间，单位为毫秒

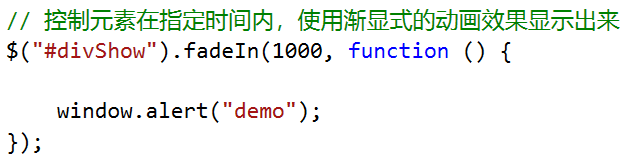
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.fadeIn( time , function() { } )：控制元素在指定时间内，使用渐显式的动画效果显示出来

-> time：指定元素显示出来所使用的时间，单位为毫秒

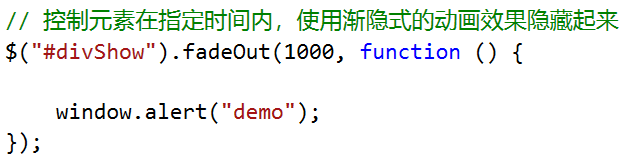
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.fadeOut( time , function() { } )：控制元素在指定时间内，使用渐隐式的动画效果隐藏起来

-> time：指定元素隐藏起来所使用的时间，单位为毫秒

-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数

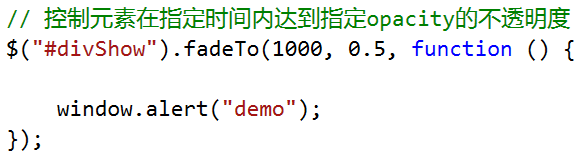


1. jQuery对象.fadeTo( time , opacity , function() { } )：控制元素在指定时间内达到指定opacity的不透明度

-> time：指定元素达到指定不透明度所使用的时间，单位为毫秒

-> opacity：不透明度，范围 0~1

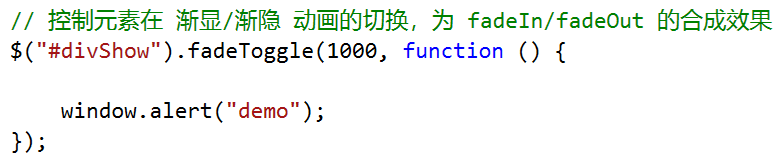
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



1. jQuery对象.fadeToggle( time , function() { } )：控制元素在 渐显/渐隐 动画的切换，为 fadeIn/fadeOut 的合成效果

-> time：指定元素 显示出来/隐藏起来 所使用的时间，单位为毫秒

-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数



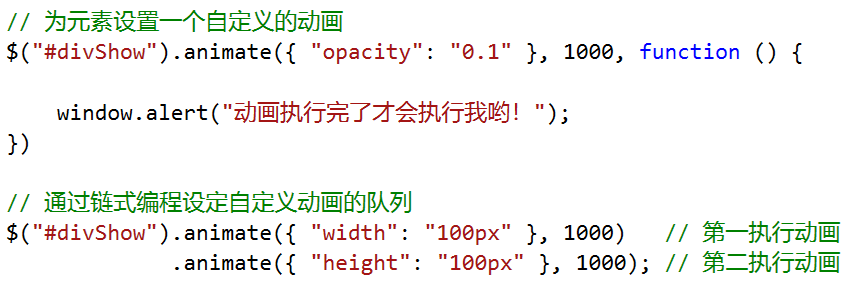
1. jQuery对象.animate( cssTarget , time , function() { } )：为元素设置一个自定义的动画

-> cssTarget：指定元素的样式改变为我们所指定的目标样式cssTarget，**cssTarget只能为数字类型的CSS样式**

-> time：指定元素改成指定样式所需的事件

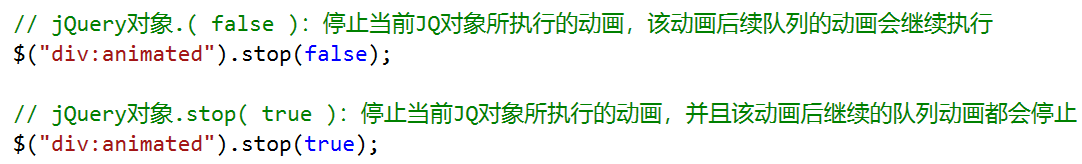
-> function() { }：可选参数，当动画完成后为JQ对象中的每一个元素所执行的回调函数

-> 可以通过链式编程设定该元素的动画执行队列



1. jQuery对象.stop( isStopAll )：停止指定JQ对象当前所执行的动画

-> isStopAll：停止了当前执行的动画后是否停止该动画后续队列的动画（true、false）



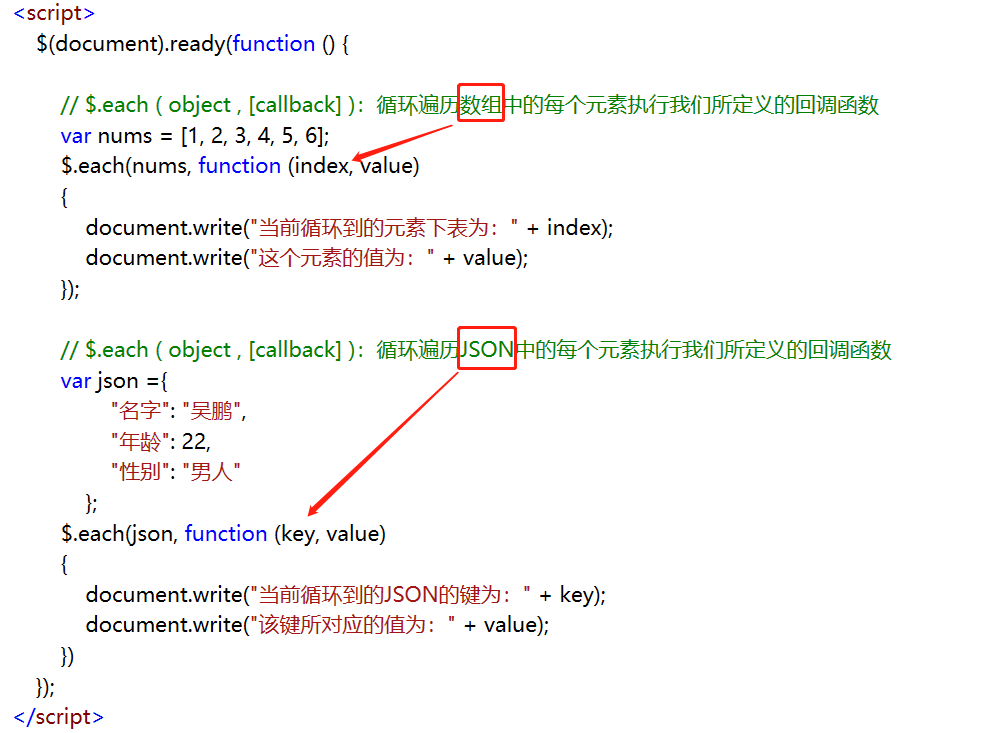
1. **jQuery工具**
2. $.each ( 数组/Json，function( index/key , value ) { } )：遍历我们所定义的数组/Json，为数组/Json执行我们所定的回调函数function( index/key , value ) { }

-> function( index/key , value ) { }：所定义的数组/Json中的每一个元素都需执行一遍操作的回调函数

-> index：数组元素的索引

-> key：Json中的键

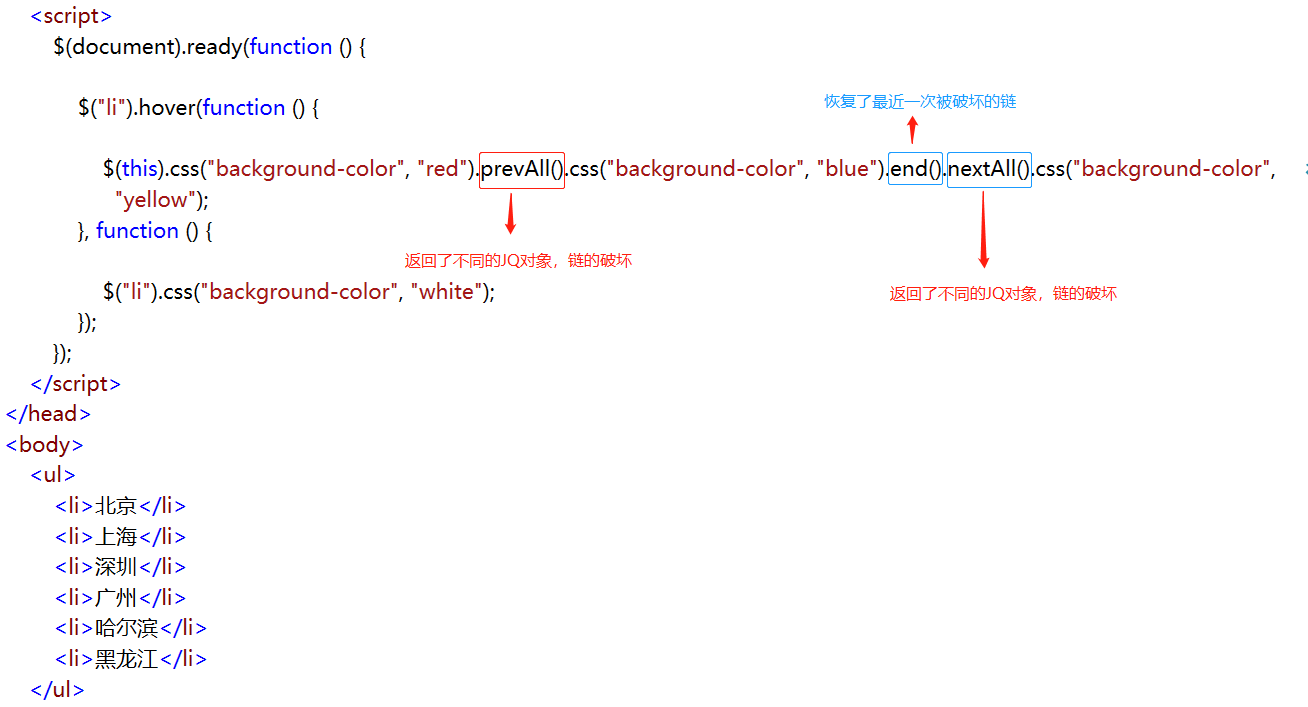
-> value：Json/数组 中的 键/索引 所对应的值



1. jQuery对象.end( )：恢复最近一次被破坏的链（返回原本承载着这条链的JQ对象）

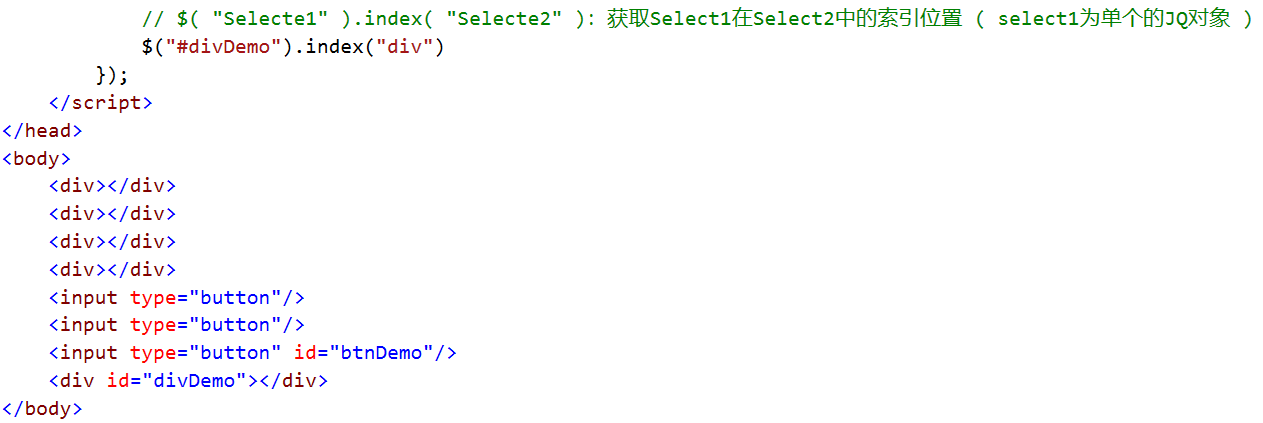
-> 我们之所以能够使用链式编程是因为jQuery中的函数如不存在特定返回值的特性都会返回执行该函数的JQ对象的特性，但是当一条链中指定了其他函数返回了其他的JQ对象，那么这条链就被破坏了，因为返回的是其他的JQ对象，我们就不能获取到原先承载着这条链的JQ对象了，但是我们能在破坏链的函数后面写end( )函数，来恢复被破坏的链

-> **如果最近一次无被破坏的链，则返回一个空的数组**



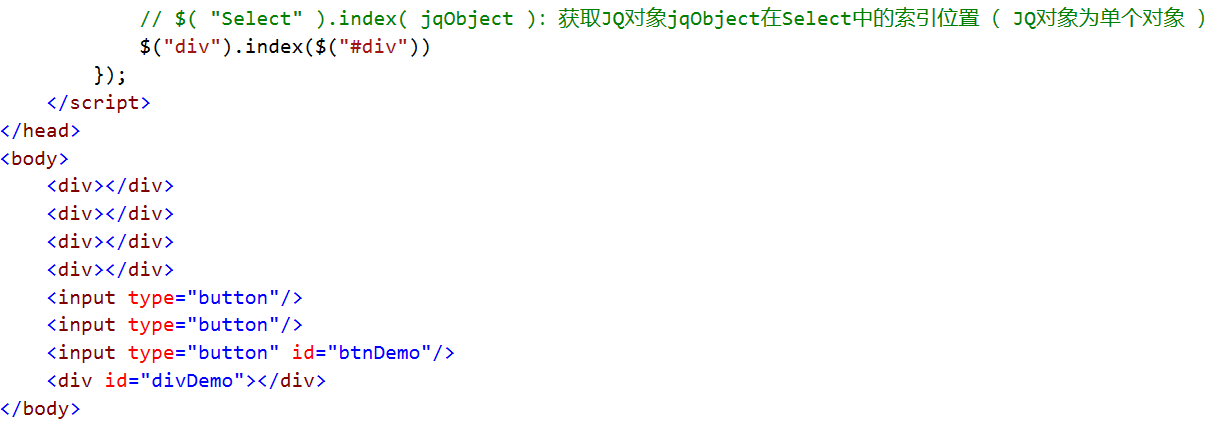
1. jQuery对象.index：获取JQ对象索引的位置
2. $( "Select1" ).index( "Select2" )：获取 Select1 在 Select2 中的索引位置

-> Select1选择器所筛选出来的JQ对象必须为单个的JQ对象

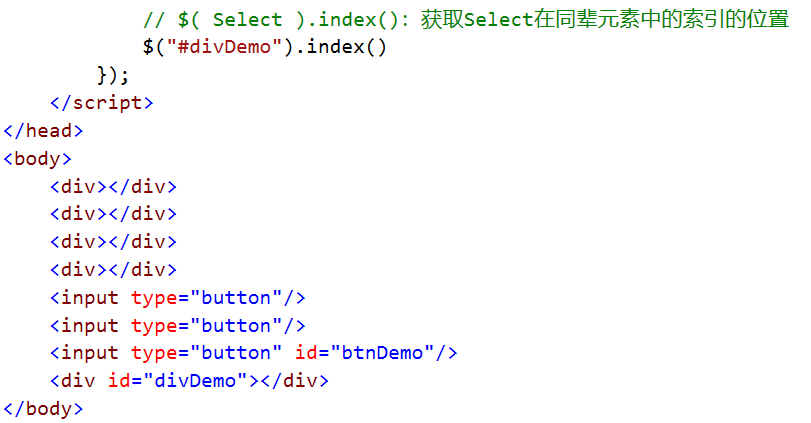


1. $( "Select" ).index( jqObject )：获取JQ对象jqObject在Select中的索引位置

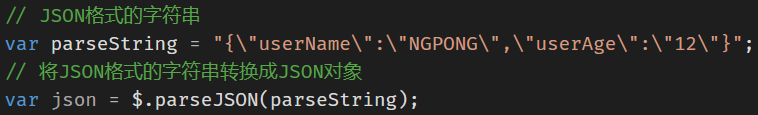
-> jqObject必须为单个的JQ对象



1. $( "Select" ).index( )：获取Select在同辈元素中的索引的位置

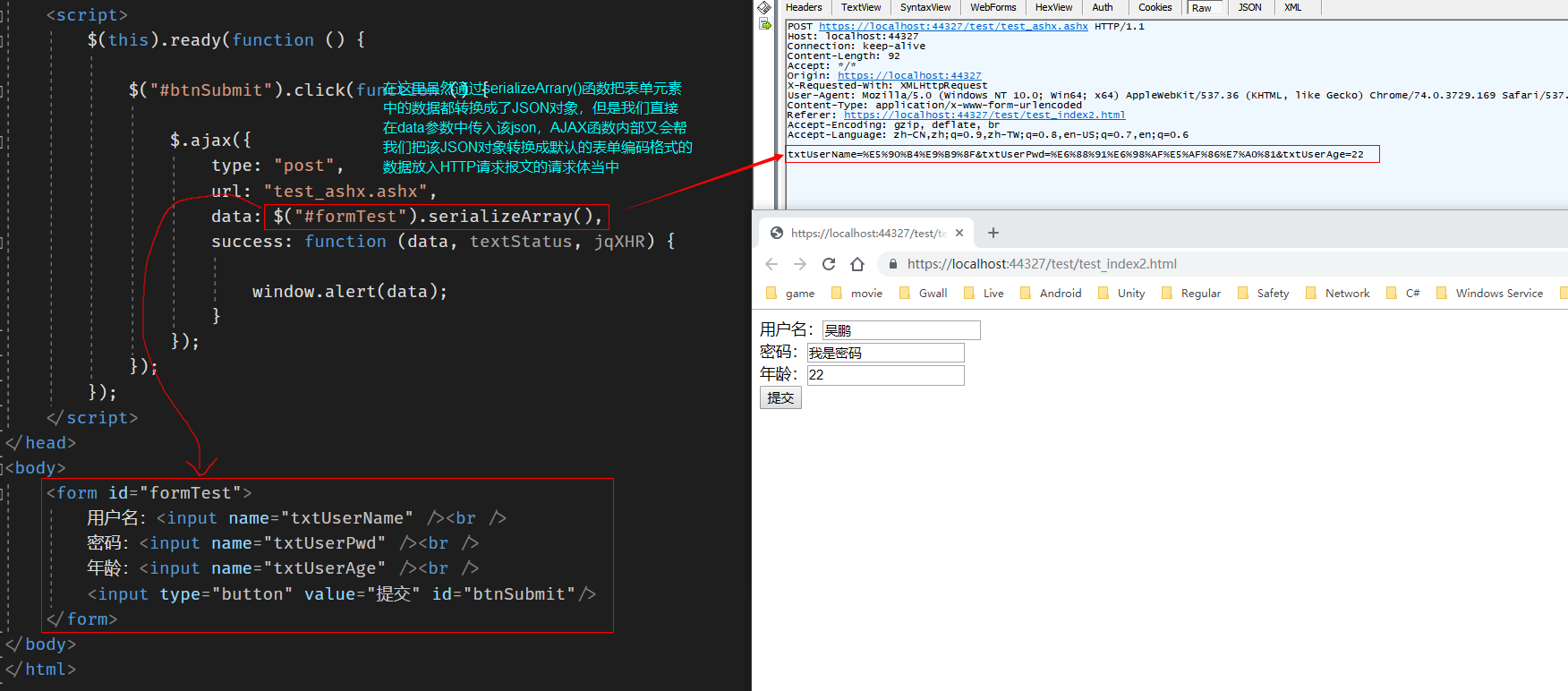


1. $.parseJSON ( parseString )：将JSON格式的parseString字符串转换成一个JSON对象

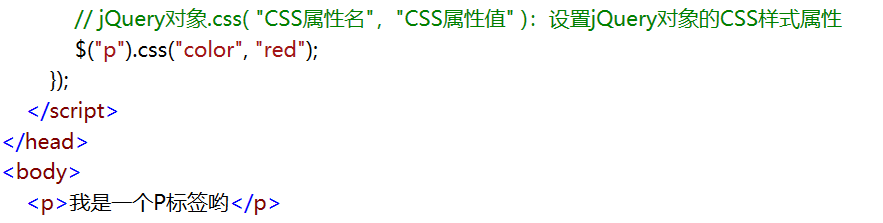


1. 表单/表单元素jQuery对象.serializeArray()：将表单元素/表单的jQuery对象序列化成一个JSON对象

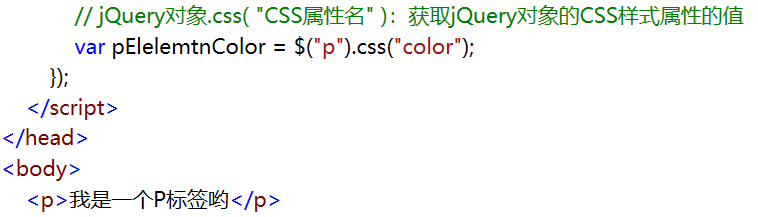
-> 该函数常用于在jQuery的AJAX函数中写入参数的数据，我们可以直接在data参数中直接传入用当前函数序列化后的JSON对象，然后在jQuery中的AJAX函数会自动地帮我们把该JSON对象重新组织成默认的form编码格式的字符串放入HTTP请求体当中



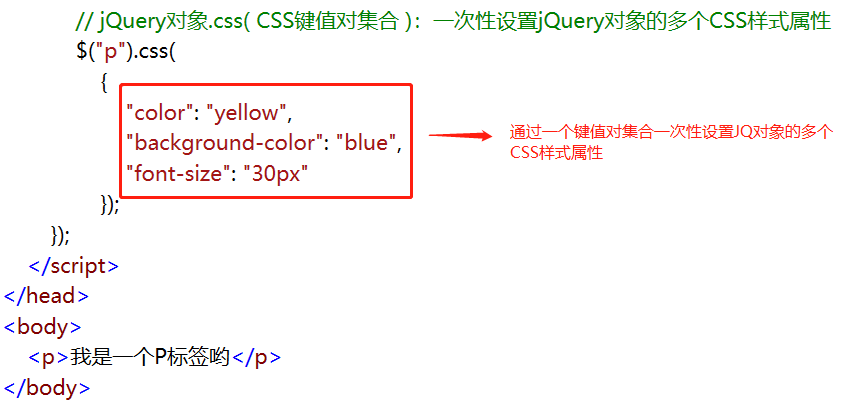
1. **jQuery操作CSS样式**
2. jQuery对象.css( )：设置或获取jQuery对象中的CSS样式属性
3. jQuery对象.css( "CSS属性名"，"CSS属性值" )：设置jQuery对象的CSS样式属性



1. jQuery对象.css( "CSS属性名" )：获取jQuery对象的CSS样式属性的值

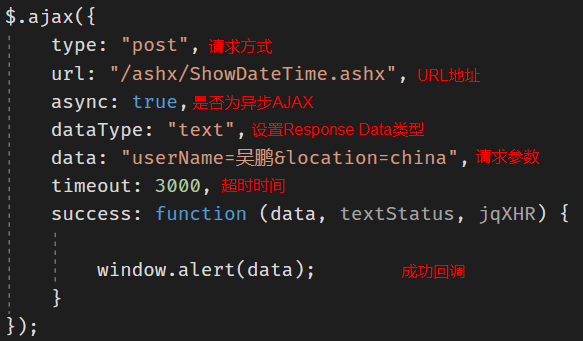


1. jQuery对象.css( CSS键值对集合 )：一次性设置jQuery对象的多个CSS样式属性

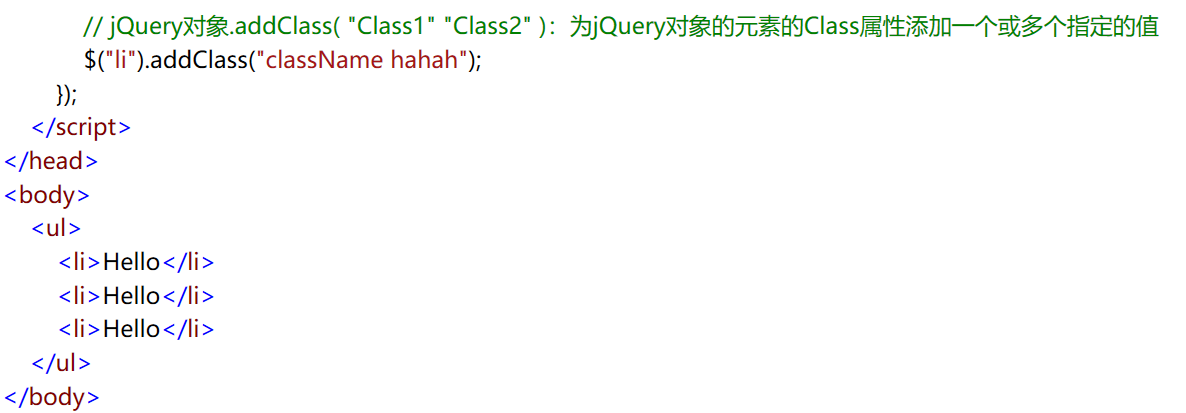


1. **jQuery中操作AJAX**
2. jQuery.ajax( Json )：完成一次AJAX交互工作

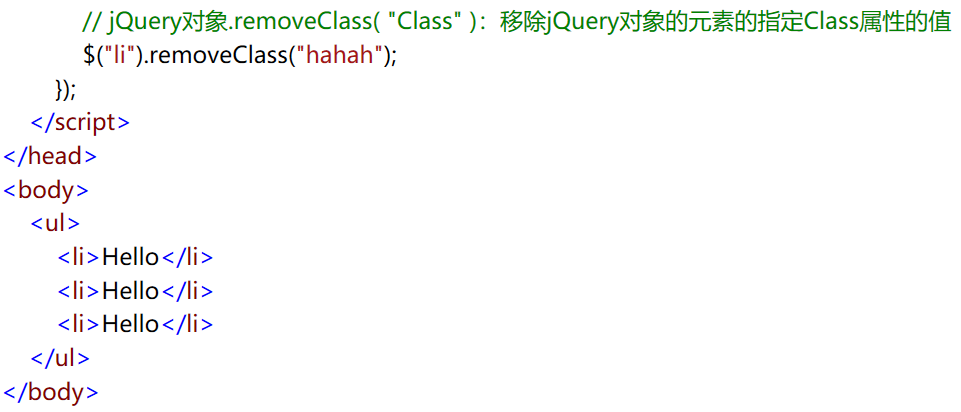
-> JSON为此次AJAX中的一些设置，关于键的含义，请参阅[jQuery的开发文档](../../../开发工具/scripts/jQueryAPI_1.7.1_CN.chm)，以下代码实例只用常见的一些参数设置



1. **jQuery操作元素的Class**
2. jQuery对象.addClass( "Class1"，"Class2" )：为jQuery对象的元素的Class属性添加一个或多个指定的值



1. jQuery对象.removeClass( "Class属性名" )：移除jQuery对象的元素的指定Class属性的值



1. **基于jQuery的第三方插件**

**-> 基于jQuery开发的第三方插件，需要先引入jQuery的JS文件然后再引入第三方插件的JS文件，这样才能保证第三方插件的正常读取和使用**

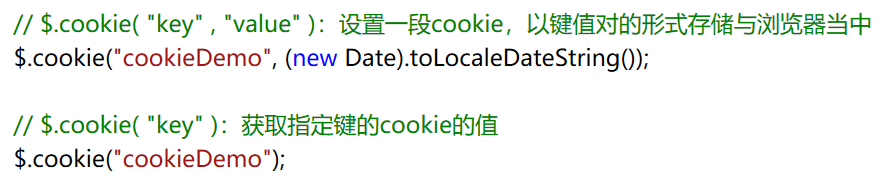
1. 操作cookie

-> 引入文件：微信截图_20180729131715

-> [cookie简介](../Cookie/Cookie.docx)

-> 设置/获取cookie的语法：

1. $.cookie( "key" , "value" )：设置cookie指定key的value
2. $.cookie( "key" )：获取cookie指定key的value



1. jQuery-UI

-> 引入文件：微信截图_20180730212204 微信截图_20180730212236

-> 使用流程：jquery-ui-1.8.20 > demos > 找到需要使用的jQuery-UI，并参照源代码撰写即可

1. jQuery-easyui

-> 使用流程：参考【[Demo](../../../开发工具/scripts/jquery-easyui-1.3.6/demo)】文件夹下给的案例的使用方式和需要引入的文件，个别属性参考【[开发文档](../../../开发工具/scripts/jquery-easyui-1.3.6/doc/jQuery EasyUI中文手册.chm)】

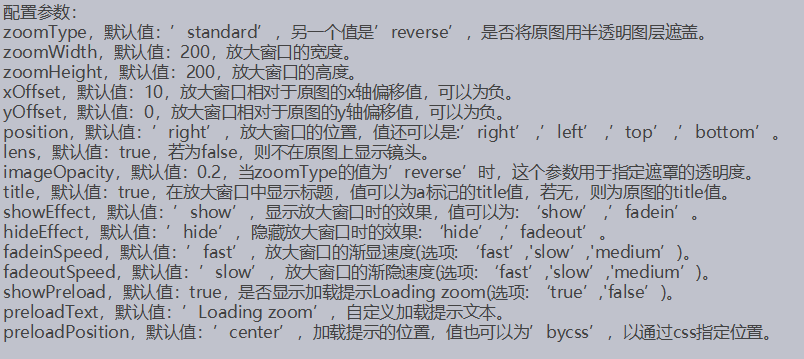
1. jQuery-zoom

-> 引入文件：微信截图_20180731210753 微信截图_20180731210740

-> HTML元素书写格式：<a id = "aElement" href = "大图片" > <img src= "小图片" /> </a>

-> 语法格式：$("#aElement").jqzoom( { JSON用于设置放大镜属性 } )

-> 可用于JSON设置的属性：



-> 代码实现：

