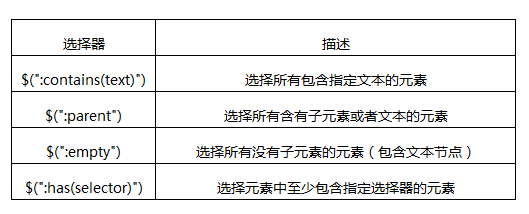
**jQuery选择器之内容筛选选择器**

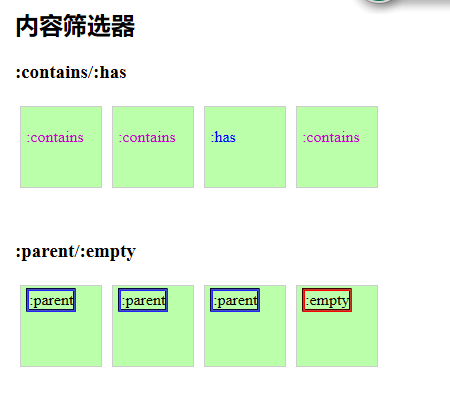
基本筛选选择器针对的都是元素DOM节点，如果我们要通过内容来过滤，jQuery也提供了一组内容筛选选择器，当然其规则也会体现在它所包含的子元素或者文本内容上

内容过滤器描述如下表：

[](http://img.mukewang.com/57cd20bf0001a97f05290214.jpg)

**注意事项：**

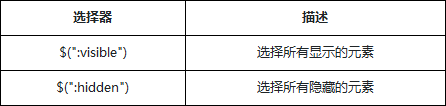
1. :contains与:has都有查找的意思，但是contains查找包含“指定文本”的元素，has查找包含“指定元素”的元素
2. 如果:contains匹配的文本包含在元素的子元素中，同样认为是符合条件的。
3. :parent与:empty是相反的，两者所涉及的子元素，包括文本节点



**jQuery选择器之可见性筛选选择器**

元素有显示状态与隐藏状态，jQuery根据元素的状态扩展了可见性筛选选择器:visible与:hidden

描述如下：

[](http://img.mukewang.com/5590f6de0001e2b204460106.jpg)

这2个选择器都是 jQuery 延伸出来的，看起来比较简单，但是元素可见性依赖于适用的样式

:hidden选择器，不仅仅包含样式是display="none"的元素，还包括隐藏表单、visibility等等

我们有几种方式可以隐藏一个元素：

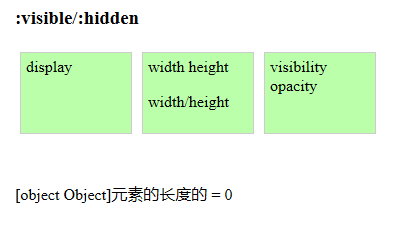
1. CSS display的值是none。
2. type="hidden"的表单元素。
3. 宽度和高度都显式设置为0。
4. 一个祖先元素是隐藏的，该元素是不会在页面上显示
5. CSS visibility的值是hidden
6. CSS opacity的指是0

如果元素中占据文档中一定的空间,元素被认为是可见的。

可见元素的宽度或高度，是大于零。

元素的visibility: hidden 或 opacity: 0被认为是可见的，因为他们仍然占用空间布局。

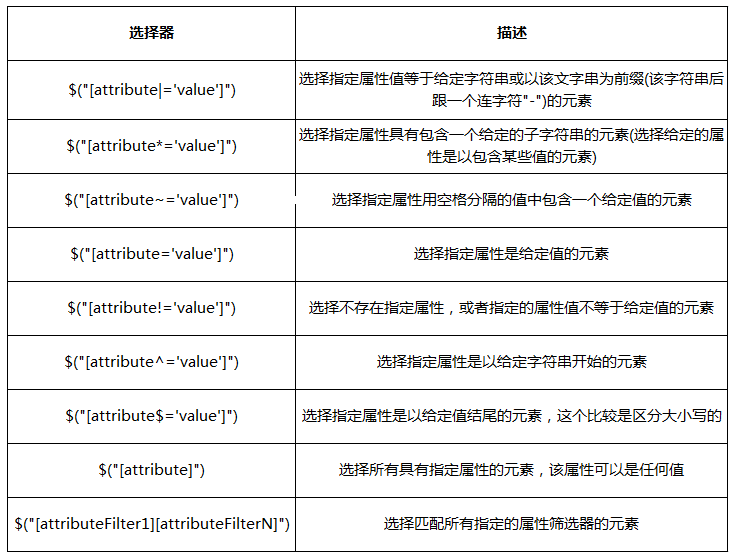
**不在文档中的元素是被认为是不可见的，如果当他们被插入到文档中，jQuery没有办法知道他们是否是可见的，因为元素可见性依赖于适用的样式**



**jQuery选择器之属性筛选选择器**

属性选择器让你可以基于属性来定位一个元素。可以只指定该元素的某个属性，这样所有使用该属性而不管它的值，这个元素都将被定位，也可以更加明确并定位在这些属性上使用特定值的元素，这就是属性选择器展示它们的威力的地方。

描述如下：

[](http://img.mukewang.com/57d654200001c46507360560.jpg)

浏览器支持：

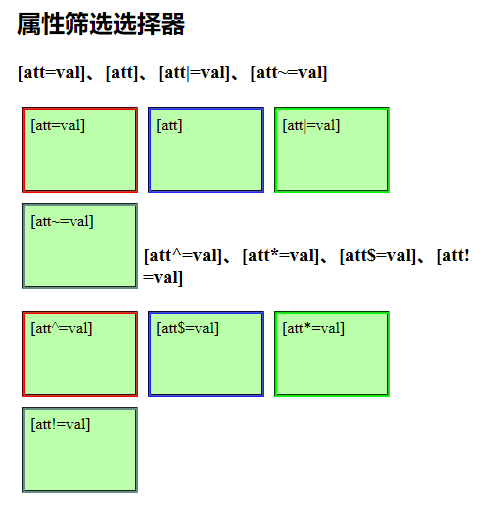
* [att=val]、[att]、[att|=val]、[att~=val]  属于CSS 2.1规范
* [ns|attr]、[att^=val]、[att\*=val]、[att$=val] 属于CSS3规范
* [name!="value"]  属于jQuery 扩展的选择器

CSS选择器无论CSS2.1版本还是CSS3版本，IE7和IE8都支持，webkit、Gecko核心及Opera也都支持，只有IE6以下浏览器才不支持

在这么多属性选择器中[attr="value"]和[attr\*="value"]是最实用的

[attr="value"]能帮我们定位不同类型的元素，特别是表单form元素的操作，比如说input[type="text"],input[type="checkbox"]等

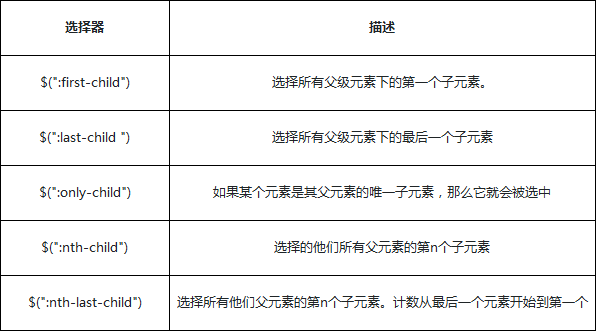
[attr\*="value"]能在网站中帮助我们匹配不同类型的文件



**jQuery选择器之子元素筛选选择器**

子元素筛选选择器不常使用，其筛选规则比起其它的选择器稍微要复杂点

子元素筛选选择器描述表：

[](https://img3.mukewang.com/5bac45120001301105960331.jpg)

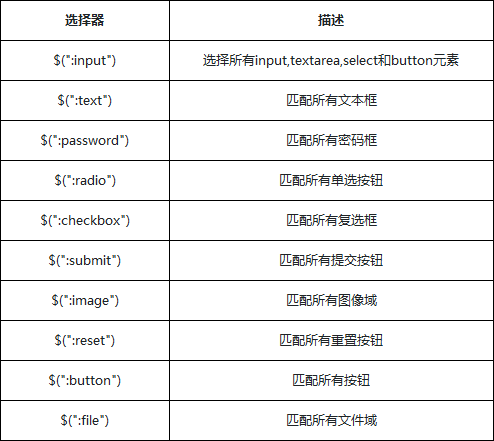
注意事项：

1. :first只匹配一个单独的元素，但是:first-child选择器可以匹配多个：即为每个父级元素匹配第一个子元素。这相当于:nth-child(1)
2. :last 只匹配一个单独的元素， :last-child 选择器可以匹配多个元素：即，为每个父级元素匹配最后一个子元素
3. 如果子元素只有一个的话，:first-child与:last-child是同一个
4. :only-child匹配某个元素是父元素中唯一的子元素，就是说当前子元素是父元素中唯一的元素，则匹配
5. jQuery实现:nth-child(n)是严格来自CSS规范，所以n值是“索引”，也就是说，从1开始计数，:nth-child(index)从1开始的，而eq(index)是从0开始的
6. nth-child(n) 与 :nth-last-child(n) 的区别前者是从前往后计算，后者从后往前计算

## jQuery选择器之表单元素选择器

无论是提交还是传递数据，表单元素在动态交互页面的作用是非常重要的。jQuery中专门加入了表单选择器，从而能够极其方便地获取到某个类型的表单元素

表单选择器的具体方法描述：

[](http://img.mukewang.com/5592040d0001f8eb04940441.jpg)

注意事项：

除了input筛选选择器，几乎每个表单类别筛选器都对应一个input元素的type值。大部分表单类别筛选器可以使用属性筛选器替换。比如 $(':password') == $('[type=password]')

**jQuery选择器之表单对象属性筛选选择器**

除了表单元素选择器外，表单对象属性筛选选择器也是专门针对表单元素的选择器，可以附加在其他选择器的后面，主要功能是对所选择的表单元素进行筛选

表单筛选选择器的描述：

[](http://img.mukewang.com/55920c2f0001198b04940201.jpg)

注意事项：

1. 选择器适用于复选框和单选框，对于下拉框元素, 使用 :selected 选择器
2. 在某些浏览器中，选择器:checked可能会错误选取到<option>元素，所以保险起见换用选择器input:checked，确保只会选取<input>元素



## jQuery选择器之特殊选择器this

相信很多刚接触jQuery的人，很多都会对$(this)和this的区别模糊不清，那么这两者有什么区别呢？

this是JavaScript中的关键字，指的是当前的上下文对象，简单的说就是方法/属性的所有者

下面例子中，imooc是一个对象，拥有name属性与getName方法,在getName中this指向了所属的对象imooc

var imooc = {

name:"慕课网",

getName:function(){

//this,就是imooc对象

return this.name;

}

}

imooc.getName(); //慕课网

当然在JavaScript中this是动态的，也就是说这个上下文对象都是可以被动态改变的(可以通过call,apply等方法)，具体的大家可以查阅相关资料

同样的在DOM中this就是指向了这个html元素对象，因为this就是DOM元素本身的一个引用

假如给页面一个P元素绑定一个事件:

p.addEventListener('click',function(){

//this === p

//以下两者的修改都是等价的

this.style.color = "red";

p.style.color = "red";

},false);

通过addEventListener绑定的事件回调中，this指向的是当前的dom对象，所以再次修改这样对象的样式，只需要通过this获取到引用即可

this.style.color = "red"

但是这样的操作其实还是很不方便的，这里面就要涉及一大堆的样式兼容，如果通过jQuery处理就会简单多了，我们只需要把this加工成jQuery对象

换成jQuery的做法：

$('p').click(function(){

//把p元素转化成jQuery的对象

var $this= $(this)

$this.css('color','red')

})

通过把$()方法传入当前的元素对象的引用this，把这个this加工成jQuery对象，我们就可以用jQuery提供的快捷方法直接处理样式了

**总体：**

**this，表示当前的上下文对象是一个html对象，可以调用html对象所拥有的属性和方法。**

**$(this),代表的上下文对象是一个jquery的上下文对象，可以调用jQuery的方法和属性值。**

**jQuery的属性与样式之.attr()与.removeAttr()**

每个元素都有一个或者多个特性，这些特性的用途就是给出相应元素或者其内容的附加信息。如：在img元素中，src就是元素的特性，用来标记图片的地址。

操作特性的DOM方法主要有3个，getAttribute方法、setAttribute方法和removeAttribute方法，就算如此在实际操作中还是会存在很多问题，这里先不说。而在jQuery中用一个attr()与removeAttr()就可以全部搞定了，包括兼容问题

jQuery中用attr()方法来获取和设置元素属性,attr是attribute（属性）的缩写，在jQuery DOM操作中会经常用到attr()

**attr()有4个表达式**

1. attr(传入属性名)：获取属性的值
2. attr(属性名, 属性值)：设置属性的值
3. attr(属性名,函数值)：设置属性的函数值
4. attr(attributes)：给指定元素设置多个属性值，即：{属性名一: “属性值一” , 属性名二: “属性值二” , … … }

**removeAttr()删除方法**

.removeAttr( attributeName ) : 为匹配的元素集合中的每个元素中移除一个属性（attribute）

**优点：**

attr、removeAttr都是jQuery为了属性操作封装的，直接在一个 jQuery 对象上调用该方法，很容易对属性进行操作，也不需要去特意的理解浏览器的属性名不同的问题

**注意的问题：**

dom中有个概念的区分：Attribute和Property翻译出来都是“属性”，《js高级程序设计》书中翻译为“特性”和“属性”。简单理解，Attribute就是dom节点自带的属性

例如：html中常用的id、class、title、align等：

<div id="immooc" title="慕课网"></div>

而Property是这个DOM元素作为对象，其附加的内容，例如,tagName, nodeName, nodeType,, defaultChecked, 和 defaultSelected 使用.prop()方法进行取值或赋值等

**获取Attribute就需要用attr，获取Property就需要用prop**

**jQuery的属性与样式之html()及.text()**

读取、修改元素的html结构或者元素的文本内容是常见的DOM操作，jQuery针对这样的处理提供了2个便捷的方法.html()与.text()

**.html()方法**

获取集合中第一个匹配元素的HTML内容 或 设置每一个匹配元素的html内容，具体有3种用法：

1. .html() 不传入值，就是获取集合中第一个匹配元素的HTML内容
2. .html( htmlString )  设置每一个匹配元素的html内容
3. .html( function(index, oldhtml) ) 用来返回设置HTML内容的一个函数

**注意事项：**

.html()方法内部使用的是DOM的innerHTML属性来处理的，所以在设置与获取上需要注意的一个最重要的问题，**这个操作是针对整个HTML内容（不仅仅只是文本内容）**

**.text()方法**

得到匹配元素集合中每个元素的文本内容结合，包括他们的后代，或设置匹配元素集合中每个元素的文本内容为指定的文本内容。，具体有3种用法：

1. .text() 得到匹配元素集合中每个元素的合并文本，包括他们的后代
2. .text( textString ) 用于设置匹配元素内容的文本
3. .text( function(index, text) ) 用来返回设置文本内容的一个函数

**注意事项：**

.text()结果返回一个字符串，包含所有匹配元素的合并文本

**.html与.text的异同:**

1. .html与.text的方法操作是一样，只是在具体针对处理对象不同
2. .html处理的是元素内容，.text处理的是文本内容
3. .html只能使用在HTML文档中，.text 在XML 和 HTML 文档中都能使用
4. 如果处理的对象只有一个子文本节点，那么html处理的结果与text是一样的
5. 火狐不支持innerText属性，用了类似的textContent属性，.text()方法综合了2个属性的支持，所以可以兼容所有浏览器

**jQuery的属性与样式之.val()**

jQuery中有一个.val()方法主要是用于处理表单元素的值，比如 input, select 和 textarea。

**.val()方法**

1. .val()无参数，获取匹配的元素集合中第一个元素的当前值
2. .val( value )，设置匹配的元素集合中每个元素的值
3. .val( function ) ，一个用来返回设置值的函数

**注意事项：**

1. 通过.val()处理select元素， 当没有选择项被选中，它返回null
2. .val()方法多用来设置表单的字段的值
3. 如果select元素有multiple（多选）属性，并且至少一个选择项被选中， .val()方法返回一个数组，这个数组包含每个选中选择项的值

**.html(),.text()和.val()的差异总结：**

1. .html(),.text(),.val()三种方法都是用来读取选定元素的内容；只不过.html()是用来读取元素的html内容（包括html标签），.text()用来读取元素的纯文本内容，包括其后代元素，.val()是用来读取表单元素的"value"值。其中.html()和.text()方法不能使用在表单元素上,而.val()只能使用在表单元素上；另外.html()方法使用在多个元素上时，只读取第一个元素；.val()方法和.html()相同，如果其应用在多个元素上时，只能读取第一个表单元素的"value"值，但是.text()和他们不一样，如果.text()应用在多个元素上时，将会读取所有选中元素的文本内容。
2. .html(htmlString),.text(textString)和.val(value)三种方法都是用来替换选中元素的内容，如果三个方法同时运用在多个元素上时，那么将会替换所有选中元素的内容。
3. .html(),.text(),.val()都可以使用回调函数的返回值来动态的改变多个元素的内容。

**jQuery的属性与样式之增加样式.addClass()**

通过动态改变类名（class），可以让其修改元素呈现出不同的效果。在HTML结构中里，多个class以空格分隔，当一个节点（或称为一个标签）含有多个class时，DOM元素响应的className属性获取的不是class名称的数组，而是一个含有空格的字符串，这就使得多class操作变得很麻烦。同样的jQuery开发者也考虑到这种情况，增加了一个.addClass()方法，用于动态增加class类名

**.addClass( className )方法**

1. .addClass( className ) : 为每个匹配元素所要增加的一个或多个样式名
2. .addClass( function(index, currentClass) ) : 这个函数返回一个或更多用空格隔开的要增加的样式名

**注意事项：**

.addClass()方法不会替换一个样式类名。它只是简单的添加一个样式类名到元素上

简单的描述下：在p元素增加一个newClass的样式

<p class="orgClass">

$("p").addClass("newClass")

那么p元素的class实际上是 class="orgClass newClass"样式只会在原本的类上继续增加，通过空格分隔

**jQuery的属性与样式之删除样式.removeClass()**

jQuery通过.addClass()方法可以很便捷的增加样式。如果需要样式之间的切换，同样jQuery提供了一个很方便的.removeClass()，它的作用是从匹配的元素中删除全部或者指定的class

**.removeClass( )方法**

1. .removeClass( [className ] )：每个匹配元素移除的一个或多个用空格隔开的样式名
2. .removeClass( function(index, class) ) ： 一个函数，返回一个或多个将要被移除的样式名

**注意事项**

如果一个样式类名作为一个参数,只有这样式类会被从匹配的元素集合中删除 。 如果没有样式名作为参数，那么所有的样式类将被移除

**jQuery的属性与样式之切换样式.toggleClass()**

在做某些效果的时候，可能会针对同一节点的某一个样式不断的切换，也就是addClass与removeClass的互斥切换，比如隔行换色效果

jQuery提供一个toggleClass方法用于简化这种互斥的逻辑，通过toggleClass方法动态添加删除Class，一次执行相当于addClass，再次执行相当于removeClass

**.toggleClass( )方法：**在匹配的元素集合中的每个元素上添加或删除一个或多个样式类,取决于这个样式类是否存在或值切换属性。即：如果存在（不存在）就删除（添加）一个类

1. .toggleClass( className )：在匹配的元素集合中的每个元素上用来切换的一个或多个（用空格隔开）样式类名
2. .toggleClass( className, switch )：一个布尔值，用于判断样式是否应该被添加或移除
3. .toggleClass( [switch ] )：一个用来判断样式类添加还是移除的 布尔值
4. .toggleClass( function(index, class, switch) [, switch ] )：用来返回在匹配的元素集合中的每个元素上用来切换的样式类名的一个函数。接收元素的索引位置和元素旧的样式类作为参数

**注意事项：**

1. toggleClass是一个互斥的逻辑，也就是通过判断对应的元素上是否存在指定的Class名，如果有就删除，如果没有就增加
2. toggleClass会保留原有的Class名后新增，通过空格隔开

**jQuery的属性与样式之样式操作.css()**

通过JavaScript获取dom元素上的style属性，我们可以动态的给元素赋予样式属性。在jQuery中我们要动态的修改style属性我们只要使用css()方法就可以实现了

**.css() 方法：获取元素样式属性的计算值或者设置元素的CSS属性**

**获取：**

1. .css( propertyName ) ：获取匹配元素集合中的第一个元素的样式属性的计算值
2. .css( propertyNames )：传递一组数组，返回一个对象结果

**设置：**

1. .css(propertyName, value )：设置CSS
2. .css( propertyName, function )：可以传入一个回调函数，返回取到对应的值进行处理
3. .css( properties )：可以传一个对象，同时设置多个样式

**注意事项：**

1. 浏览器属性获取方式不同，在获取某些值的时候都jQuery采用统一的处理，比如颜色采用RBG，尺寸采用px
2. .css()方法支持驼峰写法与大小写混搭的写法，内部做了容错的处理
3. 当一个数只被作为值（value）的时候， jQuery会将其转换为一个字符串，并添在字符串的结尾处添加px，例如 .css("width",50}) 与 .css("width","50px"})一样

**jQuery的属性与样式之.css()与.addClass()设置样式的区别**

对于样式的设置，我们学了addClass与css方法，那么两者之间有什么区别？

**可维护性：**

.addClass()的本质是通过定义个class类的样式规则，给元素添加一个或多个类。css方法是通过JavaScript大量代码进行改变元素的样式

通过.addClass()我们可以批量的给相同的元素设置统一规则，变动起来比较方便，可以统一修改删除。如果通过.css()方法就需要指定每一个元素是一一的修改，日后维护也要一一的修改，比较麻烦

**灵活性：**

通过.css()方式可以很容易动态的去改变一个样式的属性，不需要在去繁琐的定义个class类的规则。一般来说在不确定开始布局规则，通过动态生成的HTML代码结构中，都是通过.css()方法处理的

**样式值：**

.addClass()本质只是针对class的类的增加删除，不能获取到指定样式的属性的值，.css()可以获取到指定的样式值。

**样式的优先级：**

css的样式是有优先级的，当外部样式、内部样式和内联样式同一样式规则同时应用于同一个元素的时候，优先级如下

外部样式 < 内部样式 < 内联样式

1. **.addClass()方法是通过增加class名的方式，那么这个样式是在外部文件或者内部样式中先定义好的，等到需要的时候在附加到元素上**
2. **通过.css()方法处理的是内联样式，直接通过元素的style属性附加到元素上的**

**通过.css方法设置的样式属性优先级要高于.addClass方法**

**总结：**

.addClass与.css方法各有利弊，一般是静态的结构，都确定了布局的规则，可以用addClass的方法，增加统一的类规则

如果是动态的HTML结构，在不确定规则，或者经常变化的情况下，一般多考虑.css()方式

## jQuery的属性与样式之元素的数据存储

html5 dataset是新的HTML5标准，允许你在普通的元素标签里嵌入类似data-\*的属性，来实现一些简单数据的存取。它的数量不受限制，并且也能由JavaScript动态修改，也支持CSS选择器进行样式设置。这使得data属性特别灵活，也非常强大。有了这样的属性我们能够更加有序直观的进行数据预设或存储。那么在不支持HTML5标准的浏览器中，我们如何实现数据存取?  jQuery就提供了一个.data()的方法来处理这个问题

使用jQuery初学者一般不是很关心data方式，这个方法是jquery内部预用的，可以用来做性能优化，比如sizzle选择中可以用来缓存部分结果集等等。当然这个也是非常重要的一个API了，常常用于我们存放临时的一些数据，因为它是直接跟DOM元素对象绑定在一起的

jQuery提供的存储接口

jQuery.data( element, key, value ) //静态接口,存数据

jQuery.data( element, key ) //静态接口,取数据

.data( key, value ) //实例接口,存数据

.data( key ) //实例接口,存数据

2个方法在使用上存取都是通一个接口，传递元素，键值数据。在jQuery的官方文档中，建议用.data()方法来代替。

我们把DOM可以看作一个对象，那么我们往对象上是可以存在基本类型，引用类型的数据的，但是这里会引发一个问题，可能会存在**循环引用的内存泄漏风险**

通过jQuery提供的数据接口，就很好的处理了这个问题了，我们不需要关心它底层是如何实现，只需要按照对应的data方法使用就行了

同样的也提供2个对应的删除接口，使用上与data方法其实是一致的，只不过是一个是增加一个是删除罢了

jQuery.removeData( element [, name ] )

.removeData( [name ] )

参考右边的代码区域，2个代码段分别描述了静态与实例data的使用