一、基础

1、搭建环境—在<head>中导入包

<script type="text/javascript" src="http://libs.baidu.com/jquery/1.9.1/jquery.js"></script>

2、jQuery对象和DOM对象的区别

DOM对象：var p = document.getElementById('imooc');

jQuery对象：var $p = $('#imooc');

* 通过jQuery方法包装后的对象，是一个类数组对象。它与DOM对象完全不同，唯一相似的是它们都能操作DOM。jQuery是一个类数组对象，而DOM对象就是一个单独的DOM元素。

3、jQuery对象转化成DOM对象

* HTML代码

<div>元素一</div>

<div>元素二</div>

<div>元素三</div>

* JavaScript代码

第一种方法：jQuery对象是一个类数组对象，根据下标获得DOM对象

var $div = $('div') //jQuery对象

var div = $div[0] //转化成DOM对象

div.style.color = 'red' //操作dom对象的属性

第二种方法：jQuery对象自带的get(index)方法

var $div = $('div') //jQuery对象

var div = $div.get(0) //通过get方法，转化成DOM对象

div.style.color = 'red' //操作dom对象的属性

4、DOM对象转化成jQuery对象

* HTML代码

<div>元素一</div>

<div>元素二</div>

<div>元素三</div>

* JavaScript代码
* $(dom)方法将普通的dom对象加工成jQuery对象

var div = document.getElementsByTagName('div'); //dom对象

var $div = $(div); //jQuery对象

var $first = $div.first(); //找到第一个div元素

$first.css('color', 'red'); //给第一个元素设置颜色

5、选择器

1. id选择器：$( "#id" )；

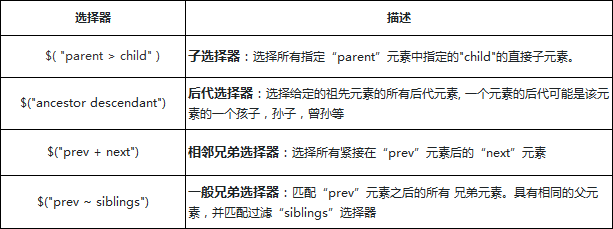
id是唯一的，每个id值在一个页面中只能使用一次，而且选择中的只有一个节点。如果多个元素分配了相同的id，将只匹配该id选择集合的第一个DOM元素。但这种行为不应该发生;有超过一个元素

的页面使用相同的id是无效的。

1. 类选择器：$( ".class" )：

类选择器可以选择多个类，jQuery对象是一个数组对象，选择中的有多个节点。

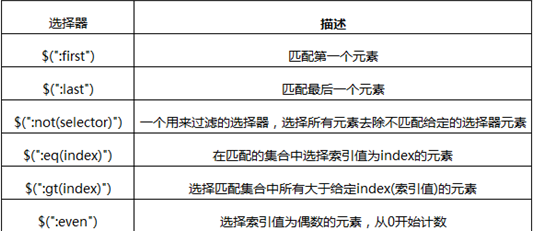
1. 元素选择器：$( "element" )，选择的是标签名tagname
2. 全选择器（\*选择器）：$( "\*" )，选取的是html界面中所有的元素节点。
3. 层级选择器：

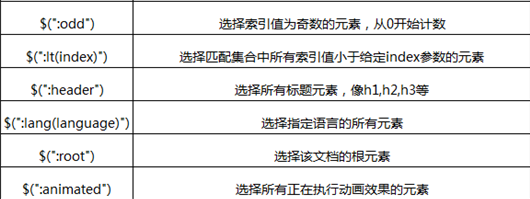


* 层级选择器都有一个参考节点
* 子选择器包含的是直接子元素。
* 后代选择器包含子选择器的选择的内容
* 一般兄弟选择器包含相邻兄弟选择的内容
* 相邻兄弟选择器和一般兄弟选择器所选择到的元素，必须在同一个父元素下

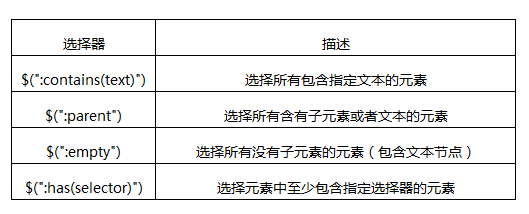
6、特殊选择器

1. 基本筛选：选择器用冒号“：”开头





1. 内容筛选



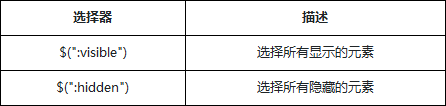
* 注意事项

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

1. 可见性筛选

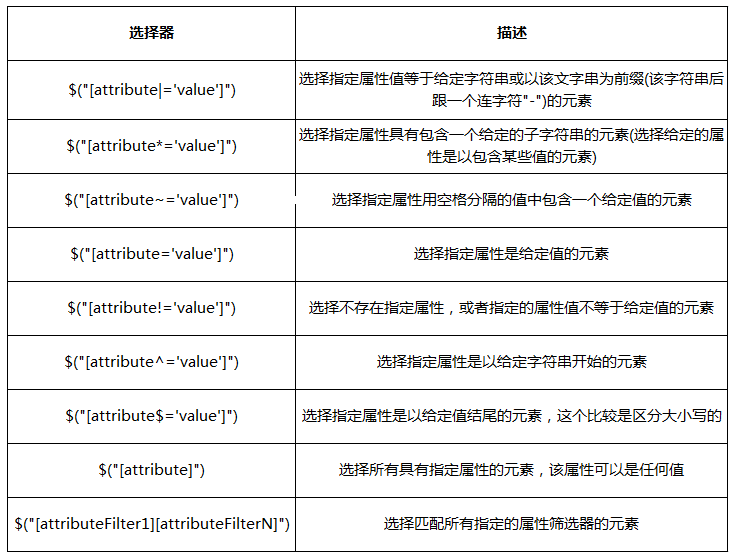


* :hidden选择器，不仅仅包含样式是display="none"的元素，还包括隐藏表单、visibility等等
* 不在文档中的元素是被认为是不可见的，如果当他们被插入到文档中，jQuery没有办法知道他们是否是可见的，因为元素可见性依赖于适用的样式

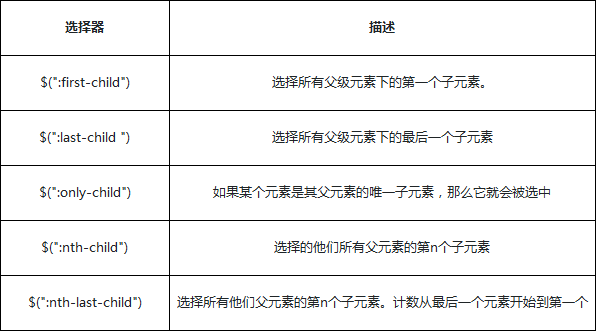
隐藏一个元素的方法：

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
7. 属性筛选

* 属性选择器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，



1. 子元素筛选



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

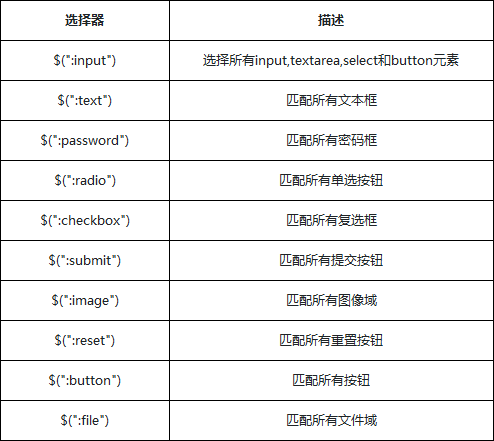
③ 如果子元素只有一个的话，:first-child与:last-child是同一个

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

1. 表单元素选择



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]')

1. 表单对象属性筛选

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



1. 特殊选择器this

① 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。

② $(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。

7、jQuery的属性与样式 相关**方法**

1. 关于元素节点的属性 -- attribute就是dom节点自带的属性

* .attr()有4个表达式

.attr(传入属性名)：获取属性的值

.attr(属性名, 属性值)：设置属性的值

.attr(属性名,函数值)：设置属性的函数值

.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）

1. 读取、修改元素的html结构或者元素的文本内容

* .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
6. 用于处理表单元素的值 -- .val()方法

* .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：

通过.val()处理select元素， 当没有选择项被选中，它返回null

.val()方法多用来设置表单的字段的值

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。
4. 增加样式 -- .addClass()

* .addClass( className )方法

.addClass( className ) : 为每个匹配元素所要增加的一个或多个样式名

.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

1. 删除样式.removeClass()

* .removeClass( )方法

.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

.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

1. 切换样式.toggleClass()

* .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：

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

toggleClass会保留原有的Class名后新增，通过空格隔开

1. 样式操作.css()

* .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

1. 获取：

.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

.css( propertyNames )：传递一组数组，返回一个对象结果

1. 设置：

.css(propertyName, value )：设置CSS

.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

.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
4. 元素的数据存储

* jQuery提供的存储接口

1. jQuery.data( element, key, value ) //静态接口,存数据
2. jQuery.data( element, key ) //静态接口,取数据
3. .data( key, value ) //实例接口,存数据
4. .data( key ) //实例接口,存数据

* jQuery的删除接口

1. jQuery.removeData( element [, name ] )
2. .removeData( [name ] )

二、DOM属性操作

1、JS中创建节点以及相关方法

* 创建流程：

创建节点(常见的：元素、属性和文本)

添加节点的一些属性

加入到文档中

* 相关方法：

创建元素：document.createElement

设置属性：setAttribute

添加文本：innerHTML

加入文档：appendChild

2、jQuery创建节点

* 通过$()函数处理，$("html结构")

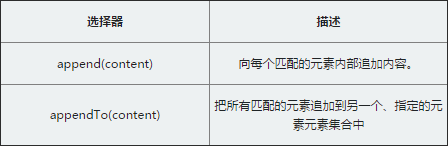
$("<div></div>") //创建元素节点

$("<div>我是文本节点</div>") //创建文本节点

$("<div id='test' class='aaron'>我是文本节点</div>") //创建属性节点

<DOM内部和外部插入操作>

3、DOM内部插入 – 追加内容（其他html标签）



append()前面是被插入的对象，后面是要在对象内插入的元素内容

appendTo()前面是要插入的元素内容，而后面是被插入的对象

4、DOM外部插入



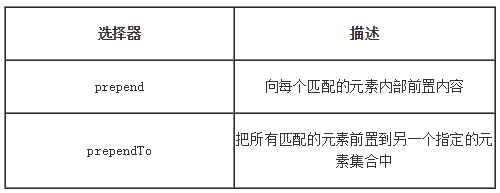
1. before与after都是用来对相对选中元素外部增加相邻的兄弟节点
2. 2个方法都是都可以接收HTML字符串，DOM 元素，元素数组，或者jQuery对象，用来插入到集合中每个匹配元素的前面或者后面
3. 2个方法都支持多个参数传递after(div1,div2,....) 可以参考右边案例代码

* 注意事项：

after向元素的后边添加html代码，如果元素后面有元素了，那将后面的元素后移，然后将html代码插入

before向元素的前边添加html代码，如果元素前面有元素了，那将前面的元素前移，然后将html代码插

5、DOM内部插入 – 前置内容



* 注意事项：

1. .prepend()方法将指定元素插入到匹配元素里面作为它的第一个子元素 (如果要作为最后一个子元素插入用.append()).
2. .prepend()和.prependTo()实现同样的功能，主要的不同是语法，插入的内容和目标的位置不同
3. 对于.prepend() 而言，选择器表达式写在方法的前面，作为待插入内容的容器，将要被插入的内容作为方法的参数
4. 而.prependTo() 正好相反，将要被插入的内容写在方法的前面，可以是选择器表达式或动态创建的标记，待插入内容的容器作为参数。

* 内部插入四种方法比较：

1. append()向每个匹配的元素内部追加内容
2. prepend()向每个匹配的元素内部前置内容
3. appendTo()把所有匹配的元素追加到另一个指定元素的集合中
4. prependTo()把所有匹配的元素前置到另一个指定的元素集合中

6、DOM外部插入 – 前置内容



1. .before()和.insertBefore()实现同样的功能。主要的区别是语法——内容和目标的位置。 对于before()选择表达式在函数前面，内容作为参数，而.insertBefore()刚好相反，内容在方法前面，它将被放在参数里元素的前面
2. .after()和.insertAfter() 实现同样的功能。主要的不同是语法——特别是（插入）内容和目标的位置。 对于after()选择表达式在函数的前面，参数是将要插入的内容。对于 .insertAfter(), 刚好相反，内容在方法前面，它将被放在参数里元素的后面
3. before、after与insertBefore。insertAfter的除了目标与位置的不同外，后面的不支持多参数处理

<DOM节点的删除>

7、DOM节点删除 --- empty()方法 – 删除此节点的子元素

* 这个方法不仅移除子元素（和其他后代元素），同样移除元素里的文本。因为，根据说明，元素里任何文本字符串都被看做是该元素的子节点。
* 具体代码：

//通过empty处理

$('.hello').empty()

//结果：<p>慕课网</p>被移除

<div class="hello"></div>

8、删除 – remove()方法

* remove会将元素自身移除，同时也会移除元素内部的一切，包括绑定的事件及与该元素相关的jQuery数据。

无参数：删除选中的所有节点。如：$("p").remove();

有参数：参数为选择器筛选器。如：$("p").remove(":contains('3')");

* empty()和remove()的区别：

**empty方法**

严格地讲，empty()方法并不是删除节点，而是清空节点，它能清空元素中的所有后代节点

empty不能删除自己本身这个节点

**remove方法**

该节点与该节点所包含的所有后代节点将同时被删除，还会删除绑定的事件。

提供传递一个筛选的表达式，删除指定合集中的元素

9、保留数据的删除操作 – detach()

* 这个方法不会把匹配的元素从jQuery对象中删除，因而可以在将来再使用这些匹配的元素。与remove()不同的是，所有绑定的事件、附加的数据等都会保留下来。

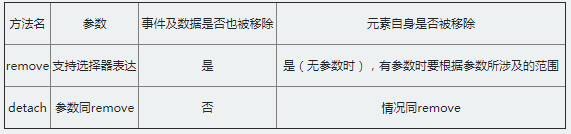
$("div").detach()这一句会移除对象，仅仅是显示效果没有了。但是内存中还是存在的。当你append之后，又重新回到了文档流中。就又显示出来了。

* 具体代码：

p = $("p").detach(); -- 删除

$("body").append(p); -- 添加

* detach()和remove()的区别



remove：移除节点

无参数，移除自身整个节点以及该节点的内部的所有节点，包括节点上事件与数据

有参数，移除筛选出的节点以及该节点的内部的所有节点，包括节点上事件与数据

detach：移除节点

移除的处理与remove一致

与remove()不同的是，所有绑定的事件、附加的数据等都会保留下来

例如：$("p").detach()这一句会移除对象，仅仅是显示效果没有了。但是内存中还是存在的。当你append之后，又重新回到了文档流中。就又显示出来了。

<DOM的复制和替换>

10、克隆方法 -- .clone()

* .clone()方法深度 复制所有匹配的元素集合，包括所有匹配元素、匹配元素的下级元素、文字节点。
* （传入参数）如果节点有事件或者数据之类的其他处理，我们需要通过clone(ture)传递一个布尔值ture用来指定，这样不仅仅只是克隆单纯的节点结构，还要把附带的事件与数据给一并克隆了。
* HTML部分

<div></div>

JavaScript部分

$("div").on('click', function() {//执行操作})

//clone处理一

$("div").clone() //只克隆了结构，事件丢失

//clone处理二

$("div").clone(true) //结构、事件与数据都克隆

* 注意事项：

1. clone()方法时，在将它插入到文档之前，我们可以修改克隆后的元素或者元素内容，如右边代码我 $(this).clone().css('color','red') 增加了一个颜色
2. 通过传递true，将所有绑定在原始元素上的事件处理函数复制到克隆元素上
3. clone()方法是jQuery扩展的，只能处理通过jQuery绑定的事件与数据
4. 元素数据（data）内对象和数组不会被复制，将继续被克隆元素和原始元素共享。深复制的所有数据，需要手动复制每一个

11、替换方法replaceWith()

* .replaceWith( newContent )：用提供的内容替换集合中所有匹配的元素并且返回被删除元素的集合

具体代码：

<div>

<p>第一段</p>

<p>第二段</p>

<p>第三段</p>

</div>

$("p:eq(1)").replaceWith('<a style="color:red">替换第二段的内容</a>')

<div>

<p>第一段</p>

<a style="color:red">替换第二段的内容</a>'

<p>第三段</p>

</div>

* .replaceAll( target ) ：用集合的匹配元素替换每个目标元素

.replaceAll()和.replaceWith()功能类似，但是目标和源相反，用上述的HTML结构，我们用replaceAll处理

具体代码：

$('<a style="color:red">替换第二段的内容</a>').replaceAll('p:eq(1)')

* 注意点：

1. .replaceAll()和.replaceWith()功能类似，主要是目标和源的位置区别
2. .replaceWith()与.replaceAll() 方法会删除与节点相关联的所有数据和事件处理程序
3. .replaceWith()方法，和大部分其他jQuery方法一样，返回jQuery对象，所以可以和其他方法链接使用。
4. .replaceWith()方法返回的jQuery对象引用的是替换前的节点，而不是通过replaceWith/replaceAll方法替换后的节点。

12、包裹方法wrap() --如果要将元素用其他元素包裹起来，也就是给它增加一个父元素

* .wrap( wrappingElement )：在集合中匹配的每个元素周围包裹一个HTML结构

具体代码：

（原始） <p>p元素</p>

（包裹处理） $('p').wrap('<div></div>')

（处理完结果） <div>

<p>p元素</p>

</div>

* .wrap( function ) ：一个**回调函数，**返回用于包裹匹配元素的 HTML 内容或 jQuery 对象，使用后的效果与直接传递参数是一样，只不过可以把代码写在函数体内部，写法不同而已。

具体代码：

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

return '<div></div>'; //与第一种类似，只是写法不一样

})

* 注意要点：

.wrap()函数可以接受任何字符串或对象，可以传递给$()工厂函数来指定一个DOM结构。这种结构可以嵌套了好几层深，但应该只包含一个核心的元素。每个匹配的元素都会被这种结构包裹。该方法返回原始的元素集，以便之后使用链式方法。

13、除包裹方法 – 删除父元素unwrap() – 不接受任何参数

* jQuery提供了一个unwrap()方法 ，作用与wrap方法是相反的。将匹配元素集合的父级元素删除，保留自身（和兄弟元素，如果存在）在原来的位置。
* （原始数据）<div>

<p>p元素</p>

</div>

（unwrap处理） $('p').unwrap();

（处理后） <p>p元素</p>

14、集中包裹方法 – wrapAll --将集合中的元素用其他元素包裹起来，也就是给他们增加一

个父元素

* .wrapAll( wrappingElement )：给集合中匹配的元素增加一个外面包裹HTML结构

(HTML结构) <p>p元素</p>

<p>p元素</p>

(wrapAll处理) $('p').wrapAll('<div></div>')

(处理后结构) <div>

<p>p元素</p>

<p>p元素</p>

</div>

* .wrapAll( function ) ：一个回调函数，返回用于包裹匹配元素的 HTML 内容或 jQuery 对象。（HTML结构和上面一样）

(处理) $('p').wrapAll(function() {

return '<div><div/>';

})

(处理结果) <div>

<p>p元素</p>

</div>

<div>

<p>p元素</p>

</div> //为每个p元素包裹一层div

15、包裹子元素 – wrapInner()方法

* 将合集中的元素内部所有的子元素用其他元素包裹起来，并当作指定元素的子元素。

.wrapInner( wrappingElement )：给集合中匹配的元素的内部，增加包裹的HTML结构。

（HTML结构） <div>p元素</div>

<div>p元素</div>

（处理） $('div').wrapInner('<p></p>')

（处理结构） <div>

<p>p元素</p>

</div>

<div>

<p>p元素</p>

</div>

* .wrapInner( function ) ：允许我们用一个callback函数做参数，每次遇到匹配元素时，该函数被执行，返回一个DOM元素，jQuery对象，或者HTML片段，用来包住匹配元素的内容

（处理） $('div').wrapInner(function() {

return '<p></p>';

})

* 注意点：

当通过一个选择器字符串传递给.wrapInner() 函数，其参数应该是格式正确的 HTML，并且 HTML 标签应该是被正确关闭的。

<jQuery遍历方法>

16、childeren()遍历方法 --快速查找合集里面的**第一级子元素**

* .children(selector) 方法是返回匹配元素集合中每个元素的所有子元素（仅儿子辈，这里可以理解为就是父亲-儿子的关系） -- 可以加上筛选器

jQuery是一个合集对象，所以通过children是匹配合集中每一给元素的第一级子元素。

* 具体代码：

$('.div').children(':last').css('border', '3px solid blue')

17、find()方法，查找后代元素（包括父子关系） -- children是父子关系查找，find是后代关系（包含父子关系）

（HTML结构） <div class="div">

<ul class="son">

<li class="grandson">1</li> </ul></div>

$(‘.div’).find(‘li’)，查找后代元素中，li标签元素。

注意要点：

1. find是遍历当前元素集合中每个元素的后代。只要符合，不管是儿子辈，孙子辈都可以。
2. 与其他的树遍历方法不同，**选择器表达式对于 .find() 是必需的参数**。如果我们需要实现对所有后代元素的取回，可以传递通配选择器 '\*'。
3. find只在后代中遍历，不包括自己。
4. 选择器 context 是由 .find() 方法实现的；因此，$('.item-ii').find('li') 等价于 $('li', '.item-ii')(找到类名为item-ii的标签下的li标签)。

18、遍历parent()方法 --快速查找合集里面的每一个元素的父元素 – 只会向上查找一层

（HTML结构） <div class="div">

<ul class="son">

<li class="grandson">1</li>

</ul>

</div>

查找div：div, $(ul).parent() //可以传入一个筛选器

* parent()方法允许我们能够在DOM树中搜索到这些元素的父级元素，从有序的向上匹配元素，并根据匹配的元素创建一个新的 jQuery 对象。

19、遍历parents()方法 – 查找一个元素的所有祖先节点，同样可以接受一个筛选器。

* 注意事项：

1. parents()和.parent()方法是相似的，但后者只是进行了一个单级的DOM树查找
2. $( "html" ).parent()方法返回一个包含document的集合，而$( "html" ).parents()返回一个空集合。
3. parent()只是查找一个父级元素，parents()查找所有的祖先元素。

20、closest() – 查找符合条件的祖先元素，只会选中0个或者1个元素。

* closest()方法接受一个匹配元素的选择器字符串
* 注意事项：
  1. 起始位置不同：.closest开始于当前元素 .parents开始于父元素
  2. 遍历的目标不同：.closest要找到指定的目标，.parents遍历到文档根元素，closest向上查找，直到找到一个匹配的就停止查找，parents一直查找到根元素，并将匹配的元素加入集合
  3. 结果不同：.closest返回的是包含零个或一个元素的jquery对象，parents返回的是包含零个或一个或多个元素的jquery对象

21、遍历next()方法 --快速查找指定元素集合中每一个元素紧邻的后面同辈元素的元素集合

* next()可以加一个筛选器。-- $('.item-1').next().css('border','1px solid red');

22、遍历prev()方法 --快速查找指定元素集合中每一个元素紧邻的前面同辈元素的元素集合

* prev()可以加一个筛选器。--$('.item-2').prev().css('border','2px solid red')

23、siblings() – 找兄弟节点，快速查找指定元素集合中每一个元素的同辈元素

* 可以加一个筛选器

24、add()方法 – 添加到合集中，在html中的位置是不改变的

* jQuery是一个合集对象，通过$()方法找到指定的元素合集后可以进行一系列的操作。$()之后就意味着这个合集对象已经是确定的，如果后期需要再往这个合集中添加一新的元素

（HTML结构）

<ul>

<li>list item 1</li>

<li>list item 3</li>

</ul>

<p>新的p元素</p>

（处理方法）

$('li').add('p') -- $('li')包含两个<li>，.add('p')执行方法后包含两个<li>和一个<p>,html中结构不改变。

* 还有一种改变html结构的方法

$('li').add('<p>新的p元素</p>').appendTo(目标位置)

--- 先加入到合集中，在添加到html中的具体位置

25、each() -- 遍历方法

* 通过$()方法找到指定的元素合集后可以进行一系列的操作，就是对合集中的每一个元素进行操作。
* .each() 方法就是一个for循环的迭代器，它会迭代jQuery对象合集中的每一个DOM元素。每次回调函数执行时，会传递当前循环次数作为参数(从0开始计数
* 注意要点：

1. each是一个for循环的包装迭代器。
2. each通过回调的方式处理，并且会有2个固定的实参，索引与元素。
3. each回调方法中的this指向当前迭代的dom元素。

* 具体语法：

$("li").each(function(index, element) {

index 索引 0,1

element是对应的li节点 li,li

this 指向的是li

})

* 修改偶数行的css样式：

$("li").each(function(index, element) {

if (index % 2) {

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

}

})

三、jQuery事件

<鼠标事件>

1、 单击事件click() – 点击只有在松手后触发

使用：

* 1. $ele.click() ：$("ele").click(function(){}
  2. $ele.click( handler(eventObject) )
  3. $ele.click( [eventData ], handler(eventObject) ) ：可以接受一个数据参数，这样的处理是 为了解决不同作用域下数据传递的问题

2、 双击事件 dblclick() – 由两个click()叠加而成 – 使用方法与click一样

* dblclick又是由2个click叠加而来的，所以dblclick事件只有在满足以下条件的情况下才能被触发
  1. 鼠标指针在元素里面时点击。
  2. 鼠标指针在元素里面时释放。
  3. 鼠标指针在元素里面时再次点击，点击间隔时间，是系统而定。
  4. 鼠标指针在元素里面时再次释放。
  5. 在同一元素上同时绑定 click 和 dblclick 事件是不可取的

3、 鼠标按下事件mousedown() 鼠标松手触发事件mouseup() – 使用方法与click一样

* mousedown事件触发需要以下几点：

1. mousedown强调是按下触发
2. 如果在一个元素按住了鼠标不放，并且拖动鼠标离开这个元素，并释放鼠标键，这仍然是算作mousedown事件
3. 任何鼠标按钮被按下时都能触发mousedown事件
4. 用event 对象的which区别按键，敲击鼠标左键which的值是1，敲击鼠标中键which的值是2，敲击鼠标右键which的值是3

* mouseup事件触发需要以下几点：

1. mouseup强调是松手触发，与mousedown是相反的
2. mouseup与mousedown组合起来就是click事件
3. 如果用户在一个元素上按下鼠标按键，并且拖动鼠标离开这个元素，然后释放鼠标键，这仍然是算作mouseup事件
4. 任何鼠标按钮松手时都能触发mouseup事件
5. 用event 对象的which区别按键，敲击鼠标左键which的值是1，敲击鼠标中键which的值是2，敲击鼠标右键which的值是3

* click与mousedown的区别：

click事件其实是由mousedown于mouseup 2个动作构成，所以点击的动作只有在松手

后才触发

4、监听用户移动的的操作 mousemove() --使用方法与click一样

* mousemove事件触发需要以下几点：

1. mousemove事件是当鼠标指针移动时触发的，即使是一个像素。
2. 如果处理器做任何重大的处理，或者如果该事件存在多个处理函数，这可能造成浏览器的严重的性能问题。

5、 鼠标移入事件mouseover 鼠标移出事件mouseout -- 使用方法与click一样

鼠标移入事件mouseenter 鼠标移出事件mouseleave -- 使用方法与click一样

* mouseenter事件和mouseover的区别关键点就是：冒泡的方式处理问题

<div class="aaron2">

<p>鼠标离开此区域触发mouseleave事件</p>

</div>

mouseover事件：当div块和p块都绑定mouseover事件时候（鼠标移入事件），当鼠标移入div块时触发div块的mouseover，当鼠标移入p块时触发p块和div块的mouseover，因为子元素触发后会向上传导，当父元素也绑定时也算触发，就像冒泡一样。

mouseenter事件：移入div块只会触发div块事件，移入p块只会触发p块事件，不会有传导。mouseout是冒泡，mouseleave不是冒泡。

6、 鼠标移入移出事件：hover

* $(selector).hover(handlerIn, handlerOut)

handlerIn(eventObject)：当鼠标指针进入元素时触发执行的事件函数

handlerOut(eventObject)：当鼠标指针离开元素时触发执行的事件函数

<表单事件>

7、 （input输入框）点击聚焦事件focusin，失焦事件focusout --使用方法与click一样

点击聚焦事件focus，失焦事件blur --使用方法与click一样

* focusin和focusout是冒泡事件

focus和blur是非冒泡事件

8、change事件

* <input>元素，<textarea>和<select>元素的值都是可以发生改变的，开发者可以通过change事件去监听这些改变的动作

**input元素**

监听value值的变化，当有改变时，失去焦点后触发change事件。对于单选按钮

和复选框，当用户用鼠标做出选择时，该事件立即触发。

**select元素**

对于下拉选择框，当用户用鼠标作出选择时，该事件立即触发

**textarea元素**

多行文本输入框，当有改变时，失去焦点后触发change事件

9、select事件 – 选择时触发事件 --使用方法与click一样

* select事件只能用于<input>元素与<textarea>元素
* 这个函数会调用执行绑定到select事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。

10、submit提交事件 -- 使用方法与click一样 – 绑定在form元素上面

* 具体能触发submit事件的行为：

<input type="submit">

<input type="image">

<button type="submit">

当某些表单元素获取焦点时，敲击Enter（回车键）

* form元素是有默认提交表单的行为，如果通过submit处理的话，需要禁止浏览器的这个默认行为。传统的方式是调用事件对象 e.preventDefault() 来处理， jQuery中可以直接在函数中最后结尾return false即可

jQuery处理如下：

$("#target").submit(function(data) {

return false; //阻止默认行为，提交表单

});

<键盘事件>

11、keydown()和keyup()

* 键盘字母按下事件 – keydown()

当用户在一个元素上第一次按下键盘上字母键的时候，就会触发它。

* 键盘字母弹起事件 – keyup()

当用户在一个元素上第一次松手键盘上的键的时候，就会触发它。使用方法与keydown是一致的只是触发的条件是方法的

* 注意：

1. keydown是在键盘按下就会触发
2. keyup是在键盘松手就会触发
3. 理论上它可以绑定到任何元素，但keydown/keyup事件只是发送到具有焦点的元素上，不同的浏览器中，可获得焦点的元素略有不同，但是表单元素总是能获取焦点，所以对于此事件类型表单元素是最合适的。

12、keypress()事件 – 和keyup keydown事件有点像

* keypress事件与keydown和keyup的主要区别

1. 只能捕获单个字符，不能捕获组合键
2. 无法响应系统功能键（如delete，backspace）
3. 不区分小键盘和主键盘的数字字符

* KeyPress主要用来接收字母、数字等ANSI字符，而 KeyDown 和 KeyUP 事件过程可以处理任何不被 KeyPress 识别的击键。诸如：功能键（F1-F12）、编辑键、定位键以及任何这些键和键盘换档键的组合等。

<事件的绑定和解绑>

13、on的多事件绑定

1. 多事件绑定同一个函数： $("#elem").on("mouseover mouseout",function(){ });
2. 多事件绑定多个函数： $("#elem").on({

mouseover:function(){}, //多个事件用逗号隔开

mouseout:function(){}

});

1. 将数据传导处理程序：

function greet( event ) {

alert( "Hello " + event.data.name ); //Hello 慕课网

}

$( "button" ).on( "click", {

name: "慕课网"

}, greet );

* 可以通过第二参数（对象），当一个事件被触发时，要传递给事件处理函数的

14、on的高级用法

* 委托机制：.on( events ,[ selector ] ,[ data ], handler(eventObject) )

第一个参数：事件名

第二个参数：匹配的选择器

第三个参数：

第四个参数：绑定的函数

* 具体代码：$("div").on("click","p",fn)

事件绑定在最上层div元素上，当用户触发在a元素上，事件将往上冒泡，一直会冒泡在div元素上。如果提供了第二参数，那么事件在往上冒泡的过程中遇到了选择器匹配的元素，将会触发事件回调函数

15、卸载事件off()方法

* 根据on绑定事件的一些特性，off方法也可以通过相应的传递组合的事件名，名字空间，选择器或处理函数来移除绑定在元素上指定的事件处理函数。当有多个过滤参数时，只有与这些参数完全匹配的事件处理函数才会被移除

1. 绑定2个事件：

$("elem").on("mousedown mouseup",fn)

1. 删除1个事件：

$("elem").off("mousedown")

1. 删除所有事件：

$("elem").off("mousedown mouseup")

$("elem").off()

<事件对象event>

16、事件对象的作用

* 事件对象是用来记录一些事件发生时的相关信息的对象。事件对象只有事件发生时才会产生，并且只能是事件处理函数内部访问，在所有事件处理函数运行结束后，事件对象就被销毁
* 属性：event.target

event.target代表当前触发事件的DOM元素（就是HTML中绑定的），可以通过当前元素对象的一系列属性来判断是不是我们想要的元素。可变的。

this一般表示当前绑定的对象，一般是不可变的。

17、事件对象的属性和方法

1. event.type：获取事件的类型
2. event.pageX 和 event.pageY：获取鼠标当前相对于页面的坐标
3. event.preventDefault() 方法：阻止默认行为
4. event.stopPropagation() 方法：阻止事件冒泡
5. event.which：获取在鼠标单击时，单击的是鼠标的哪个键

。event.which也将正常化的按钮按下(mousedown 和 mouseupevents)，左键报告1，中间键报告2，右键报告3

1. event.currentTarget : 在事件冒泡过程中的当前DOM元素

冒泡前的当前触发事件的DOM对象, 等同于this.

* this和event.target的区别：

js中事件是会冒泡的，所以this是不可以变化的（表示祖先元素是可变的），但event.target（表示当前正在触发的那个事件）会变化，它永远是直接接受事件的目标DOM元素；

* .this和event.target都是dom对象

如果要使用jquey中的方法可以将他们转换为jquery对象。比如this和$(this)的使用、event.target和$(event.target)的使用；、

<自定义事件>

18、trigger事件

* 绑定click事件：

$('#elem').on('click', function() {

alert("触发系统事件")

});

触发事件的方法有两种：

一种是点击触发

一种是利用trigger事件触发：$('#elem').trigger('click');当执行到这一句的时候会自动触发事件。

* Trigger事件可以传入参数，具体还要了解。

19、triggerHandler事件

* trigger事件还有一个特性：会在DOM树上冒泡，所以如果要阻止冒泡就需要在事件处理程序中返回false或调用事件对象中的.stopPropagation() 方法可以使事件停止冒泡
* triggerHandler与trigger的用法是一样的，重点看不同之处：

1. triggerHandler不会触发浏览器的默认行为，.triggerHandler( "submit" )将不会调用表单上的.submit()
2. .trigger() 会影响所有与 jQuery 对象相匹配的元素，而 .triggerHandler() 仅影响第一个匹配到的元素
3. 使用 .triggerHandler() 触发的事件，并不会在 DOM 树中向上冒泡。 如果它们不是由目标元素直接触发的，那么它就不会进行任何处理
4. 与普通的方法返回 jQuery 对象(这样就能够使用链式用法)相反，.triggerHandler() 返回最后一个处理的事件的返回值。如果没有触发任何事件，会返回 undefined

四、jQuery动画效果

<动画基础显示和隐藏>

1. 隐藏方法hide() 显示方法show() -- show方法与hide()方法是相反的

* 原理：让页面上的元素不可见，一般可以通过设置css的display为none属性。Hide是jquery方法控制css的页面属性。
* 无参数：$elem.hide() ， 元素的隐藏。

**提供参数**：

.hide( options ) 当提供hide方法一个参数时，.hide()就会成为一个动画方法。.hide()方法将会匹配元素的宽度，高度，以及不透明度，同时进行动画操作。

**快捷参数**：.hide("fast / slow")

这是一个动画设置的快捷方式，'fast' 和 'slow' 分别代表200和600毫秒的延时，就是元素会执行200/600毫秒的动画后再隐藏

* 注意：

jQuery在做hide操作的时候，是会保存本身的元素的原始属性值，再之后通过对应的方法还原的时候还是初始值。比如一个元素的display属性值为inline，那么隐藏再显示时，这个元素将再次显示inline。一旦透明度 达到0，display样式属性将被设置为none，这个元素将不再在页面中影响布局

* show方法注意事项：

show与hide方法是修改的display属性，通过是visibility属性布局需要通过css方法单独设置

如果使用!important在你的样式中，比如display: none !important，如果你希望.show()方法正常工作，必须使用.css('display', 'block !important')重写样式

如果让show与hide成为一个动画，那么默认执行动画会改变元素的高度，高度，透明度

1. 显示和隐藏切换方法 toggle() – 显示的变为隐藏，隐藏的变为显示

* 无参数操作：toggle()

如果元素是最初显示，它会被隐藏。如果隐藏的，它会显示出来

display属性将被储存并且需要的时候可以恢复。如果一个元素的display值为inline，然后是隐藏和显示，这个元素将再次显示inline。

* 提供参数：.toggle( [duration ] [, complete ] )
* 直接定位：toggle(display)

<上卷下拉效果>

3、.slideDown()：用滑动动画显示一个匹配元素，下拉动画

* .slideDown( [duration ] [, complete ] )

duration ：指动画的延迟效果，毫秒。

complete ：执行完成后的函数。

具体代码： $("ele").slideDown(1000, function() {

//等待动画执行1秒后,执行别的动作....

});

* 注意事项：

1. 下拉动画是从无到有，所以一开始元素是需要先隐藏起来的，可以设置display:none
2. 如 果提供回调函数参数，callback会在动画完成的时候调用。将不同的动画串联在一起按顺序排列执行是非常有用的。这个回调函数不设置任何参数，但是 this会设成将要执行动画的那个DOM元素，如果多个元素一起做动画效果，那么要非常注意，回调函数会在每一个元素执行完动画后都执行一次，而不是这组 动画整体才执行一次。

4、slideup( )：上卷动画

* 不带参数：$("elem").slideUp();
* 带参数：.slideUp( [duration ] [, easing ] [, complete ] )

因为动画是异步的，所以要在动画之后执行某些操作就必须要写到回调函数里面，这里要特别注意。

5、slideToggle – 上卷下拉切换方法

无参数：slideToggle();

提供参数：.slideToggle( [duration ] ,[ complete ] )

* 注意：

1. display属性值保存在jQuery的数据缓存中，所以display可以方便以后可以恢复到其初始值
2. 当一个隐藏动画后，高度值达到0的时候，display 样式属性被设置为none，以确保该元素不再影响页面布局

<淡入淡出效果>

6、fadeOut() -- 用于隐藏所有匹配的元素，并带有淡出的过渡动画效果

* .fadeOut( [duration ], [ complete ] )

1. 通过不透明度的变化来实现所有匹配元素的淡出效果，并在动画完成后可选地触发一个回调函数。这个动画只调整元素的不透明度，也就是说所有匹配的元素的高度和宽度不会发生变化。
2. 字符串 'fast' 和 'slow' 分别代表200和600毫秒的延时。如果提供任何其他字符串，或者这个duration参数被省略，那么默认使用400毫秒的延时

7、fadeIn –用于显示所有匹配的元素，并带有淡入的过渡动画效果。

* .fadeIn( [duration ], [ complete ] )

1. duration：指定过渡动画运行多长时间(毫秒数)，默认值为400。该参数也可以为字符串"fast"(=200)或"slow"(=600)。
2. 元素显示完毕后需要执行的函数。函数内的this指向当前DOM元素。
3. 淡入的动画原理：操作元素的不透明度从0%逐渐增加到100%
4. 如果元素本身是可见的，不对其作任何改变。如果元素是隐藏的，则使其可见

8、淡入淡出切换fadeToggle

* fadeToggle切换fadeOut与fadeIn效果，所谓"切换"，即如果元素当前是可见的，则将其隐藏(淡出)；如果元素当前是隐藏的，则使其显示(淡入)。

具体代码：.fadeToggle( [duration ] ,[ complete ] )

9、淡入效果fadeTo – 修改透明度

fadeIn：淡入效果，内容显示，opacity是0到1

fadeOut：淡出效果，内容隐藏，opacity是1到0

* 语法：.fadeTo( duration, opacity ,callback)

必需的 duration参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。fadeTo() 方法中必需的 opacity 参数将淡入淡出效果设置为给定的不透明度（值介于 0 与 1 之间）。可选的 callback 参数是该函数完成后所执行的函数名称。

10、动画切换的比较

* 操作元素的显示和隐藏原理：

1. 改变样式display为none
2. 设置位置高度为0
3. 设置透明度为0

* toggle、sildeToggle以及fadeToggle的区别：

1. toggle：切换显示与隐藏效果
2. sildeToggle：切换上下拉卷滚效果
3. fadeToggle：切换淡入淡出效果

* toggle与slideToggle细节区别：

1. toggle：动态效果为从右至左。横向动作，toggle通过display来判断切换所有匹配元素的可见性
2. slideToggle：动态效果从下至上。竖向动作，slideToggle 通过高度变化来切换所有匹配元素的可见性

* fadeToggle方法

1. fadeToggle() 方法在 fadeIn() 和 fadeOut() 方法之间切换。
2. 元素是淡出显示的，fadeToggle() 会使用淡入效果显示它们。
3. 元素是淡入显示的，fadeToggle() 会使用淡出效果显示它们。
4. 注释：隐藏的元素不会被完全显示（不再影响页面的布局）

11、animate()动画

* 语法：

.animate( properties ,[ duration ], [ easing ], [ complete ] )

参数说明：

1. **properties**：一个或多个css属性的键值对所构成的Object对象。要特别注意所有用于动画的属性必须是数字的，除非另有说明；这些属性如果不是数字的将不能使用基本的jQuery功能。特别注意单位，属性值的单位像素（px）,除非另有说明。单位em 和 %需要指定使用。除了定义数值，每个属性能使用'show', 'hide', 和 'toggle'。这些快捷方式允许定制隐藏和显示动画用来控制元素的显示或隐藏。
2. **duration**：

动画执行的时间，持续时间是以毫秒为单位的；值越大表示动画执行的越慢，不是越快。还可以提供'fast' 和 'slow'字符串，分别表示持续时间为200 和 600毫秒。

1. **easing**动画运动的算法

jQuery库中默认调用 swing。如果需要其他的动画算法，请查找相关的插件

1. **complete**回调

动画完成时执行的函数，这个可以保证当前动画确定完成后发会触发

* .animate( properties, options )

1. **options参数**

duration - 设置动画执行的时间

easing - 规定要使用的 easing 函数，过渡使用哪种缓动函数

step：规定每个动画的每一步完成之后要执行的函数

progress：每一次动画调用的时候会执行这个回调，就是一个进度的概念

complete：动画完成回调

* 如果多个元素执行动画，回调将在每个匹配的元素上执行一次，不是作为整个动画执行一次。

12、停止动画stop()

* 语法：

.stop( [clearQueue ], [ jumpToEnd ] )

.stop( [queue ], [ clearQueue ] ,[ jumpToEnd ] )

* 参数说明：

1. .stop(); 停止当前动画，点击在暂停处继续开始
2. .stop(true); 如果同一元素调用多个动画方法，尚未被执行的动画被放置在元素的效果队列中。这些动画不会开始，直到第一个完成。当调用.stop()的时候，队列中的下一个动画立即开始。如果clearQueue参数提供true值,那么在队列中的动画其余被删除并永远不会运行
3. .stop(true,true); 当前动画将停止，但该元素上的 CSS 属性会被立刻修改成动画的目标值

* 简单的说：参考下面代码、-- 三个动画

$("#aaron").animate({

height: 300

}, 5000)

$("#aaron").animate({

width: 300

}, 5000)

$("#aaron").animate({

opacity: 0.6

}, 2000)

stop()：只会停止第一个动画，第二个第三个继续

stop(true)：停止第一个、第二个和第三个动画

stop(true ture)：停止动画，直接跳到第一个动画的最终状态

五、jQuery核心方法

1、$.each方法 – 用来遍历对象和数组

jQuery会提供$(selector).each()**来遍历jQuery对象** -- $().each()

一个通用的jQuery.each方法，用来处理**对象和数组**的遍历 -- $.each()

* 语法

jQuery.each(array, callback )

jQuery.each( object, callback )

第一个参数传递的就是一个对象或者数组，第二个是回调函数

* 具体代码：

$.each(["Aaron", "慕课网"], function(index, value) {

//index是索引,也就是数组的索引

//value就是数组中的值了});

1. each就是for循环方法的一个包装，内部就是通过for遍历数组与对象，通过回调函数返回内部迭代的一些参数，第一个参数是当前迭代成员在对象或数组中的索引值(从0开始计数)，第二个参数是当前迭代成员(与this的引用相同
2. jQuery.each()函数还会根据每次调用函数callback的返回值来决定后续动作。如果返回值为false，则停止循环(相当于普通循环中的break)；如果返回其他任何值，均表示继续执行下一个循环。
3. 查找数组中的索引 – inArray

* jQuery.inArray()函数用于在数组中搜索指定的值，并返回其索引值。如果数组中不存在该值，则返回 -1。

**语法**：

jQuery.inArray( value, array ,[ fromIndex ] )

**注意**：

用法非常简单，传递一个检测的目标值，然后传递原始的数组，可以通过fromIndex规定查找的起始值，默认数组是0开始

**实例**：

$.inArray(5,[1,2,3,4,5,6,7])

1. $.trim()方法 -- 用于去除字符串两端的空白字符

* 需要注意： -- 不需要参数

移除字符串开始和结尾处的所有换行符，空格(包括连续的空格)和制表符（tab）

如果这些空白字符在字符串中间时，它们将被保留，不会被移除

1. get()方法 – 从jQuery对象合集中获取某一个元素

* HTML结构：

<a>1</a>

<a>2</a>

<a>3</a>

语法：.get( [index ] ) -- $(‘a’).get(2)

* 注意

get方法是获取的dom对象，也就是通过document.getElementById获取的对象

get方法是从0开始索引

* 负索引值参数

get方法还可以从后往前索引，传递一个负索引值，注意的负值的索引起始值是-1

同样是找到第二元素，可以传递 $(a).get(-2)

1. index()方法 -- 从匹配的元素中搜索给定元素的索引值，从0开始计数。

* 语法：参数接受一个jQuery或者dom对象作为查找的条件

.index()

.index( selector )

.index( element )

1. 如果不传递任何参数给 .index() 方法，则返回值就是jQuery对象中第一个元素相对于它同辈元素的位置，就是同辈元素的排列位置，返回索引
2. 如果在一组元素上调用 .index() ，并且参数是一个DOM元素或jQuery对象， .index() 返回值就是传入的元素相对于原先集合的位置
3. 如果参数是一个选择器， .index() 返回值就是原先元素相对于选择器匹配元素的位置。如果找不到匹配的元素，则 .index() 返回 -1

六、jQuery中的Ajax应用

1、 使用load()方法异步请求数据 --- 使用load()方法通过Ajax请求加载服务器中的数据，并把返回的数据放置到指定的元素中

* 语法格式：

load(url,[data],[callback])

参数url为加载服务器地址，可选项data参数为请求时发送的数据，callback参数为数据请求成功后，执行的回调函数。

2、使用getJSON()方法异步加载JSON格式数据 -- 通过Ajax异步请求的方式

* 语法格式

jQuery.getJSON(url,[data],[callback])

或 $.getJSON(url,[data],[callback])

* 参数说明：

url参数为请求加载json格式文件的服务器地址，可选项data参数为请求时发送的数据，callback参数为数据请求成功后，执行的回调函数。

3、 使用getScript()方法异步请求服务器中的js文件并执行

* 语法格式

jQuery.getScript(url,[callback]) 或 $.getScript(url,[callback])

参数url为服务器请求地址，可选项callback参数为请求成功后执行的回调函数。

4、 使用get()方法时，采用GET方式向服务器请求数据，并通过方法中回调函数的参数返

回请求的数据

* 语法格式：

$.get(url,[callback],”json”)

url为请求地址，callback为回调函数，json为数据格式

5、使用post()方法以POST方式向服务器发送数据

* 语法格式：

$.post(url,[data],[callback])

参数url为服务器请求地址，可选项data为向服务器请求时发送的数据，可选项callback参数为请求成功后执行的回调函数。

6、 使用serialize()方法**序列化**表单元素值 -- 可以将表单中有name属性的元素值进行序列

化，生成标准URL编码文本字符串，直接可用于ajax请求，

* 语法格式： -- 针对表单的值 – 没有参数

$(selector).serialize()

其中selector参数是一个或多个表单中的元素或表单元素本身。

* 具体语法：

$("#litest").html($("form").serialize());

7、 使用$.ajax()方法加载服务器数据

使用ajax()方法是最底层、功能最强大的请求服务器数据的方法，它不仅可以获取服务器返回的数据，还能向服务器发送请求并传递数值.

* 具体语法：jQuery.ajax([settings])或$.ajax([settings])

其中参数settings为发送ajax请求时的配置对象

url表示服务器请求的路径，

data为请求时传递的数据，

dataType为服务器返回的数据类型，

success为请求成功的执行的回调函数，

error为请求失败的执行的回调函数，

type为发送数据请求的方式，默认为get。

* 实例：

$.ajax({

url:"http://www.imooc.com/data/check.php",

data: { num: $("#txtNumber").val() },

type:"post",

success: function (data) {

$("ul").append("<li>你输入的<b> "

+ $("#txtNumber").val() + " </b>是<b> "

+ data + " </b></li>");

}

});

8、 使用ajaxSetup()方法**设置全局Ajax默认选项** -- 使用ajaxSetup()方法可以设置Ajax请求

的一些全局性选项值，设置完成后，后面的Ajax请求将不需要再添加这些选项值。

* 具体语法：

jQuery.ajaxSetup([options]) 或 $.ajaxSetup([options])

可选项options参数为一个对象，通过该对象设置Ajax请求时的全局选项值。

* 示范： -- 设置了一次，之后的ajax的请求都不用设置

$.ajaxSetup({

dataType:"text",

success:function(data){

$("ul").append("<li>你输入的<b> "

+ $("#txtNumber").val() + " </b>是<b> "

+ data + " </b></li>");

}});

9、 ajaxStart()和ajaxStop()方法 – 绑定ajax事件 – 选择器可以随意绑定

* ajaxStart()方法用于在Ajax请求发出前触发函数。

$(selector).ajaxStart(function())

* ajaxStop()方法用于在Ajax请求完成后触发函数。

$(selector).ajaxStop(function())

* 两个方法中括号都是绑定的函数，当发送Ajax请求前执行ajaxStart()方法绑定的函数，请求成功后，执行ajaxStop ()方法绑定的函数。