JS中$的作用

<https://www.cnblogs.com/jokerjason/p/7404649.html>

$在JS中本身只是一个符号而异，在JS里什么也不是。

但在JS应用库JQUERY的作者将之做为一个自定义函数名了，这个函数是获取指定网页元素的函数，使用非常之频繁，所以好多新手不知道，还以为$是JS的什么特殊语法。

后来，可能有些程序员JQUERY用得多了，**发现$这个函数很好用，很方便**，所以，在不用JQUERY的情况，一般自己也会自定义一个$函数。

即：获取html中的元素。

**function $(Nid){**

**return document.getElementById(Nid);**

**}**

就这么简单(在JSQUERY里可能稍微功能再强大一点，但主要还是我写的这个功能。)

以后在网页就不用每次使用document.getElementById("ID名")来获取元素，只用$('ID名')即可，非常简使了。

三种具体用法：

1、$()可以是$(expresion)，即css选择器、Xpath或html元素，也就是通过上述表达式来匹配目标元素。

比如：$("a")构造的这个对象，是用CSS选择器构建了一个jQuery对象——它选择了所有的<a/>这个标签。如：

$("a").click(function(){...})

就是在点击页面上的任何一个链接时的触发事件。确切地说，就是jQuery用<a/>这个标签构建了一个对象$("a")，函数 click()是这个jQuery对象的一个（事件）方法。

比如有这样一段HTML代码：

<p>one</p>

<div>

<p>two</p>

</div>

<p>three</p>

<a href="#" id="test" onClick="jq()" >jQuery</a>

而操作这段HTML的是如下一条语句：

alert($("div>p").html());

$()中的是一个查询表达式，也就是用“div>p”这样一个查询表达式构建了一个jQuery对象，然后的“html()”意思是显示其html内容，也就是上面HTML代码段的[two]。再如：

$("<div><p>Hello</p></div>").appendTo("body");

$()中的是一个字符串，用这样一段字串构建了jQuery对象，然后向<body/>中添加这一字串。

2、$()可以是$(element)，即一个特定的DOM元素。如常用的DOM对象有document、location、form等。如这样一行代码：

$(document).find("div>p").html());

$()中的document是一个DOM元素，即在全文寻找带<p>的<div>元素，并显示<p>中的内容。

3、$()可以是$(function)，即一个函数，它是$(document).ready()的一个速记方式。如常见的形式是这样的：

$(document).ready(function(){

alert("Hello world!");

});

可变形作：

$(function(){

alert("Hello world!");

});

对于选择HTML文档中的elements，jQuery有两种方法：

1）如$("div>ul a")，它的意思是div标签中的ul标签中的a标签

不过，$('div>ul')和$('div ul')是有区别的，

$('div>ul')是<div>的直接后代里找<ul>；

而$('div ul')是在<div>的所有后代里找<ul>。

2）用jQuery对象的几个方法（如方法find()、each()等）

$("#orderedlist).find("li") 就像 $("#orderedlist li"). each()一样迭代了所有的li，而表达式中的“#”表示HTML中的ID，如上例中的“#orderedlist”就表示“ID为orderedlist所在的标签”。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、标签选择器$('p')、类选择器$('.myClass')、id选择器$('#myId')相对简单，不多说。不过有一点——$('div>ul')和$('div ul')是有区别的，

$('div>ul')是<div>的直接后代里找<ul>；而$('div ul')是在<div>的所有后代里找<ul>。

所以，$('#sId>li')所选择的是id为"sId"的所有<li>孩子节点，即使这个<li>的后代还有<li>也不是它所找的范围（所找到的DOM对象，只是它本级的DOM对象。）。而$('#sId li:not(.horizontal)')，就是指类名"sId"里面的所有li的子孙中没有horizontal类的所有元素。——这里的not()是一个negation pseudo class.

这里返回的是一个jQurey对象，一个数组对象，这个jQuery对象的长度可用.length()得到。

2、XPath选择器

如：选择所有带有title 属性的链接，我们会这样写：$('a[@title]')

[]里带@，说明[]里的是元素的属性；是个属性选择器

[]里没@，说明[]里的是元素的子孙。

$('ul li')和$('ul[li]')虽然返回的都是一个jQuery数组，但两者的含义正好相反。前者是要找<ul>下所有<li>子孙，而后者却是在找所有子孙为<li>的<ul>数组。

在XPath中，要找一个“以...开头”的属性，用^=，如找一个name属性是以mail开头的input元素，就用

$('input[@name^="mail"]')

要找一个“以...结尾”的属性，要用$=

要找一个“不头不尾”的属性，用\*=

3、不属于上述的CSS和XPath的选择器，就是自定义的选择器了，用“:”表示，这里要用的就是:first，:last，:parent ，:hidden，:visible，:odd，:even，:not('xxx')， ":eq(0)"（始于0)，:nth(n），:gt(0)，:lt(0)，:contains("xxx")

如：$('tr:not([th]):even')意为<tr>元素的子孙中不含<th>的所有子孙的偶数项

4、还有几个，简单不解释了

$('th').parent()——

$('td:contains("Henry")').prev()——内容包含有"Henry"的<td>的上一个节点

$('td:contains("Henry")').next()——内容包含有"Henry"的<td>的下一个节点

$('td:contains("Henry")').siblings()——内容包含有"Henry"的<td>的所有兄弟节点

还有一个，就是end()，这个方法肯定是用在某个DOM节点执行了某一动作之后，还想在与其相关的节点上执行类似动作，这里就要用到end()。用过end()方法之后，所返回的是执行动作的那个节点的父节点上。举个例子

$(...).parent().find(...).addClass().end()

这里执行动作的节点是find(...)，是一个数组对象，它所做的动作是“addClass()”，之后，用了个end()，这时所返回的东东就是指向了parent()所指向的节点，也就是执行“addClass()”动作的那个数组对象的父节点。

5、要直接访问DOM元素，可用get(0)的方法，如

$('#myelement').get(0)，也可缩写成$('#myelement')[0]