### 基础特性

1. 可见性

:visible

* display:none
* 元素占据文档中一定的空间，被认为是可见的。（审查元素HTML页面里显示的都为可见）
* 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。
* 所有 option 元素是被认为是隐藏的， 不管他们是否selected（选中）状态.
* 如果一个元素只要有任何布局盒，即使宽度和/或高度为0，那么它将被视为:visible。

:hidden

与visible相反

1. 子元素

注意：:前的元素的父元素下的所有子元素中选择

先找它的父元素，再找每个父元素下的某个子元素

|  |
| --- |
| $("ul li:first-child") //ul父元素下所有的子元素li中的第一个  结果：<li>John</li>，<li>Glen</li>  $("ul li:first) //:first只能选择一个，即把ul下的所有li放一起，找第一个 |

* :first-child

虽然: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)。

* first-of-type

匹配冒号前面元素的同类元素的第一个

:first :first-child :first-of-child的区别

|  |
| --- |
| <body>  <button>:first</button>  <button>:first-child</button>  <button>:first-of-type</button><br><br>  <p>p1</p>  <div style="border:1px solid;">  <p>p2</p>  <p>p3</p>  </div><br>  <div style="border:1px solid;">  <span>span1</span>  <p>p4</p>  <p>p5</p>  <span>span2</span>  </div><br>  <div style="border:1px solid">  <p>p6</p>  <p>p7</p>  </div>    <p>p8</p>  </body> |

$(“p:first”)

先找p标签的集合，然后再这个集合里找第一个p标签

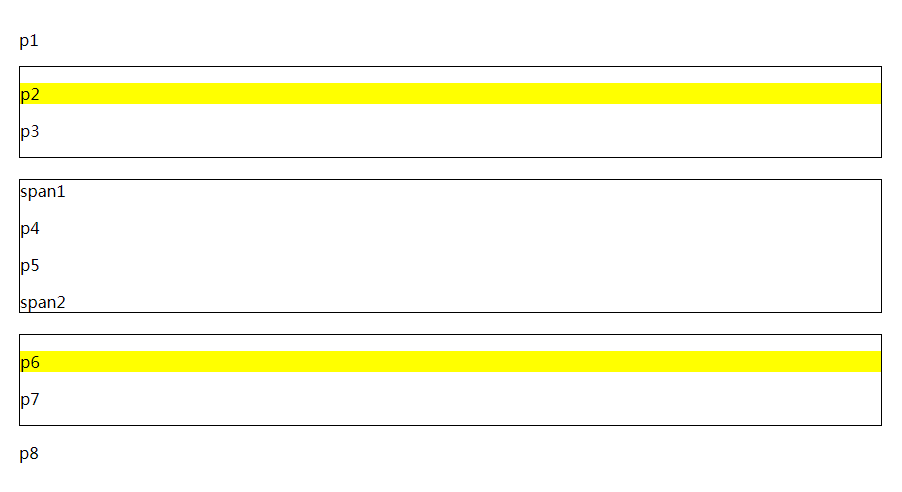
结果：

$(“p:first-child”)

先找所有p标签的所有父标签，再在每个父标签下找各自的第一个子元素，并且这个第一个子元素必须是p标签，不是p标签也匹配不到。

所以，p1不是body父元素的第一个子元素，匹配不到，p4不是第二个div里面的第一个子元素，匹配不到；

结果：



$(“p:first-of-type”)

先找所有p标签的所有父标签，然后在每个父标签下找p标签中的第一个，他并不一定是子元素中的第一个，而是p标签中的第一个。

结果：

* :nth-child(n)

选择的他们所有父元素的第n个子元素。

比如：a:nth-child(1),先找a标签的父元素，若父元素的第一个子元素是a标签的话，匹配，若不是，则匹配不到。

n可以是index索引，从1开始；也可以是even或odd，或方程式4n；

* :nth-of-type(n)

与first-of-type同理，例如： a:nth-of-type(1)先找a标签的父元素，在父元素里找所有a标签集合中的第1个

1. 表单

* :input

选择所有 input, textarea, select 和 button 元素.

1. 表单对象属性

:enabled

:disabled

:checked

:selected

1. 内容

:contains 匹配包含指定文本的元素（文本可以在后代中）

:has(selector) 后代匹配selector的父元素

:empty 匹配没有子元素的元素（子元素包括文本节点）

:parent匹配含有子元素的元素（子元素包括文本节点）

1. 属性

* .attr(attrName) 获取匹配的元素集合中的第一个元素的属性的值。

.attr(attr,value) 为每一个元素设置属性值

1）.attr( attributeName, value )

.attr(‘id’,’act’)

2）.attr( attributes ) //传对象

.attr({alt:”xxx”,title:”xxx”})

3）.attr( attributeName, function )

|  |
| --- |
| $('#greatphoto').attr('title', function(i, oldVal) { //i代表索引，oldVal代表原来的属性值  return val + ' - photo by Kelly Clark'  }); |

.removeAttr(attrName) 为匹配到的每一个元素移除一个属性

* .prop(propName) 获取匹配元素集中第一个元素的property值
* .prop(propName,value)
* .prop({})
* .prop( propertyName, function(index, oldPropertyValue) )

注意：Properties 属性一般影响 DOM 元素的动态状态并不会改变序列化的 HTML attribute 属性。所以input的disabled和checked属性，应该使用prop()而不是使用attr();

但是在3.xx的jquery，使用attr()也能达到效果，只是attr()改变的是html的checked属性。

html的属性值，相当于初始值。

HTML标签上设置的属性就是Attribute, 而直接在JavaScript中用点运算符操作的DOM对象属性就是Property; Attribute的数据类型永远都是字符串，而Property就可以非常丰富。

1. HTML代码/文本/值

html() 获取（第一个）/设置（所有匹配）的html

text() 获取（第一个）/设置（所有匹配）的文本

val() 获取input select textarea的值

1. css()

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

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

所以，css()的优先级大于addClass()

1. 内部插入

选择器.append(追加的内容[,more]) 在匹配到的每个元素的内部追加

注意：追加的内容可以是新创建的，也可以是页面上已存在的；若是选择页面已存在的，是移动而不是复制；

1. 外部插入

选择器.after(插入的内容) 在每个匹配到的元素后面插入，作为其兄弟元素

插入的内容.insertAfter(选择器) 在每个匹配到的元素后面插入，作为其兄弟元素

1. 删除节点

empty() 移除匹配元素的所有子节点，他本身并没有被移除；它移除了元素身上的所有事件，不可恢复；不能传参数

remove([selector]) 除了移除所有子节点外，还将自身移除，并移除所有绑定的事件。

detach([selector]) 删除节点（包括自身），但保留事件

1. 克隆

clone()

clone(true) 连事件一起克隆，默认false

1. 替换

oldContent.replaceWith(newContent) 使用新内容替换匹配内容

注意：用来替换的内容若是页面上原有的元素，它是被移动了，并非复制。

会删除与节点相关联的所有数据和事件

返回被替换掉的元素的集合

newContent.replaceAll(oldContent) 目标和源相反

1. 串联

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

等同于js里的childNodes

children() 只获取子元素，不含文本节点及注释节点，只是子代，不含后代

1. 包裹

wrap(wrappingElement) 用wrappingElement包裹匹配到的每一个元素（一个个单独包裹）

wrapAll( wrappingElement ) 用wrappingElement包裹匹配到的元素的集合，把匹配的集合凑一块，用一个东西包裹。匹配元素间的任何元素都被排除在外。

wrapInner(wrappingElement) 在匹配的元素的内容外部包一层结构，即在每个匹配元素的innerHTML的外面包一层。

1. 查找

children（）只获取子元素，不含文本节点及注释节点，只是子代，不含后代

find() 在后代中查找，不包括自己