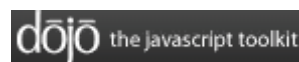


## jQuery 学习笔记

### 1 JavaScript 库

- 1.1 Ajax 技术的产生刺激了 JavaScript 的发展, 使 JavaScript 的重要性提升到了前所未有的程度。但原生的 JavaScript 程序函数名冗长难记, 且存在浏览器兼容性问题。
- 1.2 为了简化 JavaScript 的开发, 一系列 JavaScript 库应运而生。JavaScript 库封装了很多预定义的对象和实用函数。能帮助使用者建立有高难度交互的 Web2.0 特性的富客户端页面, 并且兼容各大浏览器。当前流行的 JavaScript 库有:



### 2 jQuery 简介

- 2.1 jQuery 是继 Prototype 之后又一个优秀的 JavaScript 库。由美国人 John Resig 于 2006 年 1 月发布。如今, jQuery 已经成为最流行的 JavaScript 框架, 在全世界前 10000 个访问最多的网站中, 有超过 55% 在使用 jQuery。



- 2.2 jQuery 的理念是: **写得少, 做得多**。它的优势是:

- 轻量级
- 强大的选择器
- 出色的 DOM 操作的封装
- 可靠的事件处理机制
- 完善的 Ajax
- 出色的浏览器兼容性
- 链式操作方式
- .....

- 2.3 网友膜拜:

John Resig 从未做过旋转木马插件，因为当他去游乐场的时候，所有的木马都支持 jQuery

John Resig 可以用两行代码干掉你！当然，他用一行就够了，但是爆炸的威力会让宇宙崩塌成一个巨大的黑洞

John Resig 的电脑特别定制了钛键盘，因为稍微脆弱一点的会熔化掉

`$('#John Resig').position();` 返回“无所不在！”

当 John Resig 输入 `$.jump()`，控制台输出“多高？”

当 John Resig 输入 `$.require()`，记住，这不是一个请求，这是一个恐吓！

**John Resig does not \$.queue()** 哦！这家伙从来不排队！

John Resig 定义了一个计划任务，在 2012 年的某天运行 `'rm -rf /'`，为什么玛雅人会知道？？[注：`rm -rf /`是一个 Linux 命令，执行这个命令会删除硬盘上的一切，包括系统文件]

上一次 John Resig 用 `$.die()` 的时候，他们修订了日内瓦公约明确禁止这么做！

John Resig 持续 `'onfocus'`

John Resig 从不查找 DOM，他只要扫一眼就会得到他想要的东西

John Resig 查看一个元素时，他看到的是原始的电子！

John Resig 从不等待 DOM 变为 ready 状态，DOM 都在等他 ready

John Resig 检测 `isNaN()` 时，它最好不是一个数字

John Resig 提交新版本不需要写注释，因为 John Resig 也永远说不完

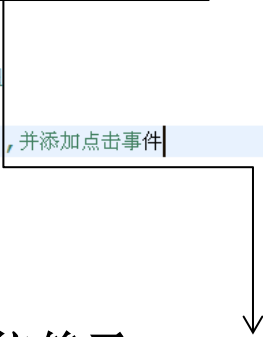
### 3 Hello World

```
<!-- 加载jQuery库 -->
<script type="text/javascript" src="script/jquery-1.7.2.js"></script>
<script type="text/javascript">

    //在DOM结构绘制完毕后执行相应操作，相当于window.onload
    $(document).ready(function() {
        //选取按钮元素，相当于getElementsByTagName(),并添加点击事件
        $("button").click(function() {
            alert("Hello Boy,I am jQuery!");
        });
    });

</script>

<button>Click Me!</button>
```



所有jQuery操作都依赖于jQuery库的支持，千万不要忘记引入！

### 4 认识 jQuery 对象

#### 4.1 认识\$符号

在 jQuery 库中，\$是 jQuery 的简写形式。例如：

`$("#app")`等价于 `jQuery("#app")`

`$.ajax` 等价于 `jQuery.ajax`

这里 jQuery 是一个顶级对象，所有的函数方法都在这个对象之下，\$是它的别名。

#### 4.2 jQuery 对象

4.2.1 jQuery 对象就是通过 `jQuery($())` 包装 DOM 对象后产生的对象，只有通过

jQuery 对象才能调用 jQuery 中的方法。

4.2.2 jQuery 对象是 jQuery 独有的. 如果一个对象是 jQuery 对象, 那么它就可以使用 jQuery 里的方法, 例如: `$( "#persontab" ).html();`

4.2.3 jQuery 对象无法使用 DOM 对象的任何方法

4.2.4 DOM 对象也不能使用 jQuery 里的任何方法

4.2.5 约定: 如果获取的是 jQuery 对象, 那么要在变量前面加上 `$`.

`var $variable = jQuery 对象`

`var variable = DOM 对象`

4.3 jQuery 对象转换 DOM 对象

jQuery 对象不能使用 DOM 中的方法, 但有时想要的方法在 jQuery 中没有封装, 必须使用 DOM 对象解决。这时可以通过两种方法由 jQuery 获取 DOM 对象:

① **jQuery 对象是一个数组对象**, 可以通过 `[index]` 数组下标的方法得到对应的 DOM 对象

```
//jQuery对象
var $btnEle = $("button");
//转换为DOM对象
var btnElement = $btnEle[0];
//使用DOM方法操作
btnElement.onclick = function() {
    alert("dd");
}
```

② 使用 jQuery 中的 `get(index)` 方法得到相应的 DOM 对象

```
//jQuery对象
var $btnEle = $("button");
//转换为DOM对象
var btnElement = $btnEle.get(0);
//使用DOM方法操作
btnElement.onclick = function() {
    alert("ee");
}
```

4.4 DOM 对象转换 jQuery 对象

对于一个 DOM 对象, 只需要用 `$()` 把 DOM 对象包装起来, 就可以获得一个 jQuery 对象, 因为本来 jQuery 对象就是通过 jQuery 包装 DOM 对象后产生的对象。DOM 对象转换成 jQuery 对象后就可以使用 jQuery 中的方法了

```
//DOM对象
var btnElement = document.getElementsByTagName("button")[0];
//转换为jQuery
var $btnEle = $(btnElement);
//调用jQuery方法
$btnEle.click(function() {
    alert("You Clicked a jQuery Object");
});
```

5 jQuery 代码怎么写

5.1 导入 jQuery 库

<!-- 导入 jQuery 库, 使用 `scrip` 标签的 `src` 属性 -->

`<script type="text/javascript" src="script/jquery-1.7.2.js"></script>`

`<script type="text/javascript">`

`$(function(){`

```
alert("--");
```

```
});
```

```
</script>
```

## 5.2 加载文档

`$(document).ready(function(){});` 作用相当于 JS 当中的 `window.onload = function(){};`

`$(document).ready(function(){});` 可以简写为 `$(function(){});`

## 6 jQuery 常用方法

### 6.1 val()方法

#### 6.1.1 返回 value 值: jQuery 对象.val()

对多选框而言: `val()`方法返回的是被选中的第一个值, 而不是全部被选中的值组成的数组

#### 6.1.2 设置 value 值: 传入数组适应性更强

##### ①jQuery 对象.val("value")

文本框

密码框

单选下拉列表

##### ②jQuery 对象.val(["value01","value02"])

文本框

密码框

单选框

多选框

单选下拉列表

多选下拉列表

### 6.2 each()方法

用来遍历一组元素

```
each(function(){
```

在函数体内使用 `this` 关键字引用当前正在被遍历的元素

`this` 永远是 DOM 对象

`$(this)`转为 jQuery 对象

```
});
```

### 6.3 text()方法

返回文本值: jQuery 对象.text()

设置文本值: jQuery 对象.text("文本值")

### 6.4 attr()方法

返回属性值: jQuery 对象.attr("属性名")

设置属性值: jQuery 对象.attr("属性名","属性值")

### 6.5 \$.trim()方法: 去除字符串前后空格

### 6.6 jQuery 特性

#### 6.6.1 隐式迭代

自动遍历所有被选中的 `p` 元素, 把单击响应函数绑定到每一个 `p` 元素上

```
$("p").click(function(){
```

```
//jQuery 对象.text()方法可以返回元素节点内的文本值  
var text = $(this).text();  
alert(text);  
});
```

#### 6.6.2 函数连缀

很多 jQuery 方法，返回值都是调用对象本身，所以可以以“连缀”的方式，继续进行调用 例如：`$("#text").focus(...).blur(...)`

注意：在使用方法连缀时，一定要留意每个参与连缀的方法的返回值是不是原来调用的那个对象，如果不是了，就不能继续连缀，除非你要操作的是这个新的对象

### 7 基本选择器

7.1 id 选择器：`#id` `$("#one")`

7.2 class 选择器：`.class` `$(".mini")`

7.3 选择元素名：`$(元素名)` `$("#div")`

7.4 所有元素：`*` `$("*")`

7.5 “,”表示并列关系 `$("#span,#two")` 此时所有 span 元素和 id 为 two 的都会被选中

7.6 选择特定 class 属性的元素 元素名.class `$("#div.one")`

### 8 层次选择器

8.1 选择后代元素：空格 父元素 后代元素

8.2 选择直接子元素：`>` 父元素>子元素

8.3 紧邻的下一个兄弟元素：`+` 前元素+后元素 必须是紧邻的

8.4 紧邻的后面所有的兄弟元素：`~` 前元素~后元素 必须是紧邻的

8.5 所有兄弟元素 `siblings()`这个函数 本身元素.`siblings()`

8.6 选取不相邻的下一个兄弟元素 `nextAll(元素名:first)`

8.7 选取前面的兄弟元素 `prev()` `prevAll()`

### 9 基本过滤选择器

9.1 `:first` 选取第一个元素

9.2 `:last` 选取最后一个元素

9.3 `:not(选择器)`选择指定选择器之外的元素

9.4 `:even` 选择索引值为偶数的元素，注意：索引值从 0 开始

9.5 `:odd` 选择索引值为奇数的元素

9.6 `:gt(索引值)` 选择索引值大于指定索引值的元素

9.7 `:eq(索引值)` 选择索引值等于指定索引值的元素

9.8 `:lt(索引值)` 选择索引值小于指定索引值的元素

9.9 `:header` 选择所有的标题元素

9.10 `:animated` 选择所有正在执行动画的元素

### 10 内容过滤选择器

10.1 `:contains(text)` 选择含有指定文本的选择器

10.2 `:empty` 选择空的元素

- 10.3 :has(选择器) 选择包含匹配指定选择器的元素
- 10.4 :parent 选择有子元素或文本元素的元素（当爹的）
- 11 可见性过滤选择器
  - 11.1 :visible 选择可见的元素
  - 11.2 :hidden 选择所有不可见的元素（注意：选中不等于在页面上显示，要想在页面上显示，需借助 show()）
  - 11.3 表单隐藏域无法使用 show()方法在页面上显示，但是选中后，可以获取操作其值
- 12 属性过滤选择器
  - 12.1 [attribut]含有指定属性的元素
  - 12.2 [attribut=value]含有指定属性，且属性值为指定值的元素
  - 12.3 [attribut!=value]属性值不等于 value 的元素，注意：没有指定属性的元素也会被选中，如果想准确指定，需使用组合选择器，例如：[title][title!=test]
  - 12.4 [attribut^=value]属性值以 value 开始的元素
  - 12.5 [attribut\$=value]属性值以 value 结尾的元素
  - 12.6 [attribut\*=value]属性值包含 value 的元素
  - 12.7 [selector1][selector2]注意：[selector2]在[selector1]的选择后的结果的范围内进行选择
- 13 子元素过滤选择器
  - 13.1 父元素:nth-child(索引值) 每一个父元素下指定索引值的子元素，索引值从 1 开始
  - 13.2 父元素:first-child(索引值) 每一个父元素下第一个子元素
  - 13.3 父元素:last-child(索引值) 每一个父元素下最后一个子元素
  - 13.4 父元素:only-child(索引值) 如果一个父元素下存在着一个单独的子元素，那么就选择它
- 14 表单元素过滤选择器
  - 14.1 表单元素:enabled 选择可用的表单元素
  - 14.2 表单元素:disabled 选择不可用的表单元素
  - 14.3 单选框（或复选框）:checked 匹配单选框或多选框被选中的
  - 14.4 对于多选框来说，所有被选中的元素返回值是一个数组，不能直接使用 val()返回所有的值，需要使用 each()方法进行遍历
  - 14.5 "select :selected" 选择被选中的下拉列表项

```
$("#btn5").click(function(){
    $("select :selected").each(function(){
        alert($(this).val());
    });
});
```
- 15 DOM 操作
  - 15.1 查找节点



- [1]元素节点: jQuery 选择器
- [2]元素属性: attr()或其他方法
- [3]文本节点: text()读写

#### 15.2 创建节点

\$(html 代码) \$("-

注意: HTML 代码格式规范

新创建的节点, 不会自动添加到文档中

#### 15.3 插入节点

- [1]根据父子关系插入

父对象.append() 干爹.append(郭美美)

子对象.appendTo(父对象) 郭美美.appendTo(干爹)

append()

\$("#city").append(\$("<li id='gz'> 广 州 </li>")); // 相 当 于 dom 对  
象.appendChild()

appendTo()

\$("#<li id='sz'>深圳</li>").appendTo("#city");

prepend()

\$("#game").prepend("<li>丢沙包</li>");

prependTo()

\$("#<li>跳房子</li>").prependTo("#game");

- [2]根据兄弟关系插入

after() 大哥.after(小弟) 大哥后面跟着一个小弟

\$("#bj").after("<li>长春</li>");

insertAfter() 小弟.insertAfter(大哥) 大哥我要跟着你

\$("#<li>重庆</li>").insertAfter("#bj");

before() 小弟.before(大哥) 大哥请你罩着我

\$("#rl").before("<li>跳皮筋</li>");

insertBefore() 大哥.insertBefore 大哥说: 我来罩着你

\$("#<li>弹球</li>").insertBefore("#rl");

#### 15.4 删除节点

在 JS 中, 是父元素.removeChild(要删除的节点) 父亲把儿子扫地出门

在 jQuery 中, 要删除的节点.remove() 儿子离家出走

\$("#jpf").remove();

父元素.empty()删除全部子节点和后代节点

\$("#city").empty();

#### 15.5 复制节点

clone()不复制行为

clone(true)连同绑定的事件一同复制

\$("#bj").click(function(){

alert(\$(this).text());

}).clone(true).appendTo("#game");

#### 15.6 替换节点

replaceWith(): 将所有匹配的元素都替换为指定的 HTML 或 DOM 元素

```
$("#bj").replaceWith("<li>大连</li>");
```

replaceAll(): 颠倒了的 replaceWith() 方法

```
$("#<li>青岛</li>").replaceAll("#city li"); city 下所有的 li 都被换成了<li>青岛</li>
```

#### 15.7 包裹节点

```
<UL id=city>
```

```
    <FONT color=blue><LI id=bj>北京</LI></FONT>
```

```
    <FONT color=blue><LI>上海</LI></FONT>
```

```
    <FONT color=blue><LI>东京</LI></FONT>
```

```
    <FONT color=blue><LI>首尔</LI></FONT>
```

```
</UL>
```

```
$("#city li").wrap("<font color='blue'></font>");
```

```
<UL id=city>
```

```
<FONT color=red>
```

```
<LI id=bj>北京</LI>
```

```
<LI>上海</LI>
```

```
<LI>东京</LI>
```

```
<LI>首尔</LI>
```

```
</FONT>
```

```
</UL>
```

```
$("#city li").wrapAll("<font color='red'></font>");
```

```
    <LI id=bj><FONT color=blue>北京</FONT></LI>
```

```
$("#city li").wrapInner("<font color='blue'></font>");
```

#### 15.8 属性操作

attr(属性名,属性值) 设置属性

```
$("#username").attr("disabled","disabled");
```

removeAttr(属性名) 移除属性

```
$("#username").removeAttr("disabled");
```

#### 15.9 html() 方法

```
alert($("#city").html());
```

不是替换元素节点本身的 HTML 代码, 而是替换其内部的 HTML 代码

```
$("#bj").html("<font color='blue'>南 京 </font>"); <li>北 京 </li> --> <li><font color='blue'>南京</font></li>
```

#### 15.10 父对象.children() 获取全部子节点, 不包括后代节点

```
$lis = $("#city").children();
```

#### 15.11 eq(索引值) 可以返回指定索引对应的对象

```
alert($lis.eq(0).text());
```

```
alert($lis.eq(1).text());
```

```
alert($lis.eq(2).text());
```

```
alert($lis.eq(3).text());
```



## 16 CSS 操作

1.hasClass(class 名) 返回布尔值, 真: 存在, 假: 不存在

```
var $sub = $("div.SubCategoryBox");  
alert($sub.hasClass("SubCategoryBox"));
```

2.removeClass(class 名) 移除 class

```
$sub.removeClass("SubCategoryBox");  
alert($sub.hasClass("SubCategoryBox"));
```

3.addClass(class 名) 添加 class

```
$sub.addClass("SubCategoryBox");
```

4.toggleClass(class 名) 切换 class

```
$("#div.showmore a").click(function(){  
    $sub.toggleClass("SubCategoryBox");  
    return false;  
});
```

5.css(样式属性) 返回属性值 css(样式属性,属性值) 设置样式为指定的属性值

```
$sub.css("background","#bbffaa");
```

6.opacity 属性透明度

```
alert($sub.css("opacity"));  
$sub.css("opacity",0.5);
```

7.height()高度width()宽度

```
alert("height="+$sub.height());  
alert("width="+$sub.width());  
$sub.height("200px");  
$sub.width("800px");
```

8.offset()获取元素在当前视窗中的相对位移。返回值属性: top 距视窗上边框距离, left 距视窗左边框距离

```
alert("top="+$sub.offset().top);  
alert("left="+$sub.offset().left);  
$sub.offset().top = "120px";
```

## 17 从 jQuery 的角度看事件

1.window.onload 和\$(document).ready()的区别

[1]执行的时机不同

window.onload 要等整个窗口(包括图片)都加载完才触发执行  
\$(document).ready()在 DOM 结构绘制完成后就可执行

[2]编写的个数

window.onload 编写多个时, 只有最后的那个起作用  
\$(document).ready()可以编写多个, 都生效

[3]简化写法

window.onload 无  
\$(document).ready()可简写为 \$()

Tip: jQuery 中真正与 window.onload 完全等价的是\$(window).load()方法

## 2.事件的绑定

```
bind(type,[data],fn)
```

type:事件类型 [data]:与事件有关的数据 fn:响应函数

## 3.事件的合成

hover()方法 hover(function(){鼠标移入时的操作},function(){鼠标移出时的操作})

mouseover+mouseout

toggle(fn1,fn2,...,fnn) 用于把多个鼠标单击事件合并在一起，所传入的响应函数会依次执行，并循环

切换元素的显示状态，例：

```
$("#header").click(function(){
    $("#div.content").toggle();
})
```

## 4.事件的冒泡

产生的原因是：监听区域的重合。取消的方式是 return false;

return false 还可以取消元素的默认行为

表单里提交按钮的提交行为

超链接的跳转行为

## 5.事件对象的属性

事件对象：封装了与事件有关的一系列信息，在事件触发时创建

接收事件对象：给响应函数增加一个形式参数 function(event){event.pageX...}

event.pageX 事件发生时，鼠标距离页面左边框的距离

```
var x = event.pageX;
```

event.pageY 事件发生时，鼠标距离页面上边框的距离

```
var y = event.pageY;
```

## 6.事件的移除

```
$("#li").click(function(){
    alert($(this).text());
});
```

//使用 unbind(事件类型)函数 移除事件

```
$("#bj").unbind("click");
```

//使用 one(事件类型,响应函数) 事件只执行一次，然后会被移除

```
$("#bj").one("click",function(){
    alert($(this).text());
});
```

## 18 动画

## 1.show() hide()

可以传入一个毫秒值控制显示/隐藏的速度，也可以是三种预定速度之一的字符串 ("slow", "normal", or "fast")

## 2.fadeIn()淡入 fadeOut()淡出

三种预定速度之一的字符串 ("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

3.slideDown(), slideUp()

三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如: 1000)

4.slideToggle()

通过高度变化来切换所有匹配元素的可见性

5.fadeTo(速度值, 不透明度) 不透明度 0 为完全透明, 1 为完全不透明