Jquery

01 jQuery概述

jQuery是什么：是一个优秀的JavaScript框架。

jQuery的作用：

- 获取页面中的元素更方便
- 轻松修改页面的内容和展现样式
- Ajax变得更容易
- 动画效果
- 页面的事件处理更轻松

jQuery的优势：

- 轻量级 （ Lightweight ）
- 强大的选择器
- 出色的DOM操作封装
- 可靠的事件处理机制
- 出色的浏览器兼容性

jQuery使用环境创建

1、下载地址：http://jquery.com提供了最新的jQuery框架下载。

2、下载后在页面中引用，就可以使用JQuery的功能了。

3、 引入的格式：**<script src="jquery.min.js" ></script>**

jQuery核心函数

- $(expression)：expression是一个包含CSS选择器的字符串，其结果是匹配的一组元素
- $(html)：根据提供的原始HTML标记字符串，动态创建由 jQuery 对象包装的 DOM 元素
- $(elements)：将一个或多个DOM元素转化为jQuery对象
- $(function(){...})：$(document).ready(function(){})的简写。仅仅只需要加载所有的DOM结构，在浏览器把所有的HTML放入DOM tree之前就执行js效果。包括在加载外部图片和资源之前。
- window.onload = function() {}; 在整个页面的document全部加载完成以后执行。这种方式不仅要求页面的DOM tree全部加载完成，而且要求所有的外部图片和资源全部加载完成

案例分析

```
<script type=“text/javascript” src=“jquery.js”></script>
<script type=“text/javascript”>
    $(document).ready(function(){
        alert(“你好，jQuery！”);
    });
</script>
window.onload = function() {
$("table tr:nth-child(even)").addClass("even"); //这个是jquery代码
};
```

这段代码会在整个页面的document全部加载完成以后执行。不幸的这种方式不仅要求页面的DOM tree全部加载完成，而且要求所有的外部图片和资源全部加载完成。更不幸的是，如果外部资源，例如图片需要很长时间来加载，那么这个js效果就会让用户感觉失效了。

02 jQuery选择器

什么是jQuery选择器?

jQuery仿照css和xpath的规则，制定了一些用于获取页面HTML元素的方法

选择器是 jQuery 的根基, 在 jQuery 中, 对事件处理, 遍历 DOM 和 Ajax 操作都依赖于选择器

用jQuery的各种选择器获取到的结果是**jQuery**对象**，而不是DOM对象，jQuery

对象可以使用jQuery里的各种方法：

jQuery对象：var a = $("#title");

DOM对象：var b = document.getElementById("title");

jQuery对象和DOM对象之间可以互相转换：

```
var a = $("#title"); //jQuery对象
var b = a[0];    //dom对象
var c = a.get(0);  //dom对象
var d = $(b);    //jQuery对象
```

基本选择器

基本选择器是 jQuery 中最常用的选择器, 也是最简单的选择器, 它通过元素 id, class 和标签名来查找 DOM 元素(在网页中 id 只能使用一次, class 允许重复使用).

1、#id 用法: *$("#**myDiv**");* 返回值 单个元素的组成的集合

说明: 这个就是直接选择html中的*id="**myDiv**"*

2、Element 用法: $("div") 返回值 集合元素

说明: element的英文翻译过来是"元素",所以element其实就是html已经定义的标签元素,例如div, input, a等等.

3、class 用法: $(".myClass") 返回值 集合元素

说明: 这个标签是直接选择html代码中class="myClass"的元素或元素组(因为在同一html页面中class是可以存在多个同样值的).

4、* 用法: $("*") 返回值 集合元素

说明: 匹配所有元素,多用于结合上下文来搜索

5、selector1, selector2, selectorN

用法: $("div,span,p.myClass") 返回值 集合元素

说明: 将每一个选择器匹配到的元素合并后一起返回.你可以指定任意多个选择器, 并将匹配到的元素合并到一个结果内.其中p.myClass是表示匹配元素 p class="myClass"

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $("#button").click(function(){
    alert($("#hello").text());
    $("#hello").css("background-color","red");
    alert($(".java").text());
    $(".java").css("background-color","green");
    alert($("p").text());
    $("p").css("background-color","blue");
    alert($("body *").text());
    $("body *").css("background-color","gray");
    alert($("#hello,.java").text());
    $("#hello,.java,input").css("background-color","yellow");
  });
});
</script>
</head>
<body>
<p id="hello">hello</p>
<p class="java">java</p>
<input type="button" id="button" value="基本选择器" />
</body>
</html>
```

层次选择器

如果想通过 DOM 元素之间的层次关系来获取特定元素, 例如后代元素, 子元素, 相邻元素, 兄弟元素等, 则需要使用层次选择器。

1 、ancestor descendant

用法: *$("form input")*; 返回值 集合元素

说明: 在给定的祖先元素下匹配所有后代元素.这个要和下面讲的"parent > child"区分开.

2、parent > child

用法: *$("form > input")*; 返回值 集合元素

说明: 在给定的父元素下匹配所有子元素.注意:要区分好后代元素与子元素

3、prev + next

用法: *$("label + input")*; 返回值 集合元素

说明: 匹配所有紧接在 prev 元素后的 next 元素

4、prev ~ siblings

用法: *$("form ~ input")*; 返回值 集合元素

说明: 匹配 prev 元素之后的所有 siblings 元素.注意:是匹配之后的元素,不包含该元素在内,并且siblings匹配的是和prev同辈的元素,其后辈元素不被匹配。

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("#button").click(function(){
        //alert($("form #name").val());(匹配所有后代元素)
        //alert($("form > #id").val());
        //alert($("form > #name").val()); //name取不到(只匹配子元素)
        //$("div + p").css("color","red"); //第一个p变色
        $("div ~ p").css("color","red");    //两个p都变色

    });

});
</script>
</head>
<body>
<form>
    <span>id</span><input id="id" type="text" />
    <div>
        <span>name</span><input id="name" type="text" /><br/>
        <span>age</span><input id="age" type="text" />
    </div>
    <p>div平级</p>
    <p>变色</p>
    <br/>
    <input type="button" id="button" value="层次选择器" />
</form>

</body>
</html>
```

过滤选择器

过滤选择器主要是通过特定的过滤规则来筛选出所需的 DOM 元素, 该选择器都以 ":" 开头。

按照不同的过滤规则, 过滤选择器可以分为基本过滤, 内容过滤, 可见性过滤, 属性过滤, 子元素过滤和表单对象属性过滤选择器。

基础过滤选择器

1、:first 用法: $("tr:first")； 返回值 单个元素的组成的集合

说明: 匹配找到的第一个元素

2、:last 用法: $("tr:last") 返回值 集合元素

说明: 匹配找到的最后一个元素.与 :first 相对应.

3、:not(selector) 用法: $(”input:not(:checked)”)返回值 集合元素

说明: 去除所有与给定选择器匹配的元素.有点类似于”非”,意思是没有被选中的input(当input的type=”checkbox”).

4、:even 用法: $(”tr:even”) 返回值 集合元素

说明: 匹配所有索引值为偶数的元素，从 0 开始计数.js的数组都是从0开始计数的.例如要选择table中的行,因为是从0开始计数,所以table中的第一个tr就为偶数0.

5、: odd 用法: $(”tr:odd”) 返回值 集合元素

说明: 匹配所有索引值为奇数的元素,和:even对应,从 0 开始计数.

6、:eq(index) 用法: $(”tr:eq(0)”) 返回值 集合元素

说明: 匹配一个给定索引值的元素.eq(0)就是获取第一个tr元素.括号里面的是索引值,不是元素排列数.

7、:gt(index) 用法: $(”tr:gt(0)”) 返回值 集合元素

说明: 匹配所有大于给定索引值的元素.

8、:lt(index) 用法: $(”tr:lt(2)”) 返回值 集合元素

说明: 匹配所有小于给定索引值的元素.

9、:header(固定写法) 用法: $(”:header”).css(”background”, “#EEE”) 返回值 集合元素

说明: 匹配如 h1, h2, h3之类的标题元素.这个是专门用来获取h1,h2这样的标题元素.

10、:animated(固定写法) 返回值 集合元素

说明: 匹配所有正在执行动画效果的元素

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    //alert($("li:first").text());  //1
    //alert($("li:last").text());       //4
    //alert($("li:not(.a)").text()); //li标签除了class=a的其他都被选中
    //alert($("li:even").text()); //索引值为偶数：0,2
    //alert($("li:odd").text()); //索引值为偶数：1,3
    //alert($("li:eq(1)").text());  //匹配索引值为1的元素值
    //alert($("li:gt(1)").text());  //匹配索引值大于1的元素值
    //alert($("li:lt(1)").text());  //匹配索引值小于1的元素值
    alert($(":header").text());          //匹配所有标题
    function aa() {
        $("#ani").animate({width:300},"slow");
        $("#ani").animate({width:100},"slow",aa);
    }
    aa();
    $("#run").click(function(){
      $(":animated").css("background-color","blue");
```

```
    });
});
</script>
</head>
<body>
<h1>一级</h1>
<h2>二级</h2>
<div>
    <li class="a">1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
</div>
<button id="run">Run</button>
<div id="ani" class="clear" style="width:100px;height:100px;background-
color:red"></div><br/>
</body>
</html>
```

内容过滤选择器

内容过滤选择器的过滤规则主要体现在它所包含的子元素和文本内容上

1、:contains(text) 用法: $(”div:contains(‘John’)”) 返回值 集合元素

说明: 匹配包含给定文本的元素.这个选择器比较有用，当我们要选择的不是dom标签元素时,它就派上了用场了,它的作用是查找被标签”围”起来的文本内容是否符合指定的内容的.

2、:has(selector) 用法: $(”div:has(p)”).addClass(”test”) 返回值 集合元素

说明: 匹配含有选择器所匹配的元素的元素.这个解释需要好好琢磨,但是一旦看了使用的例子就完全清楚了:给所有包含p元素的div标签加上class=”test”.

3、:empty 用法: $(”td:empty”) 返回值 集合元素

说明: 匹配所有不包含子元素或者文本的空元素

4、:parent 用法: $(”td:parent”) 返回值 集合元素

说明: 匹配含有子元素或者文本的元素.注意:这里是”:parent”,和:empty用法相反.

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    //alert($("p:contains('tom')").text()); //元素内是否含有某个值，含有则输出元素内所
有内容
    //$("div:has(span)").css("background-color","red"); //给所有包含span元素的div背
景变红
    $("td:empty").css("background-color","blue"); //查找空元素，设置背景颜色
    $("td:parent").css("background-color","red"); //查找空元素，设置背景颜色
});
</script>
```

```
</head>
<body>
<div>
    <p>joy</p>
    <p>tom</p>
    <p>sun</p>
</div>
<div>
    <span>hello</span>
</div>
<table width="25%">
    <tr><td>01</td><td></td></tr>
    <tr><td></td><td>jim</td></tr>
</table>
</body>
</html>
```

可见度过滤选择器

可见度过滤选择器是根据元素的可见和不可见状态来选择相应的元素

1、:hidden 用法: $(”tr:hidden”) 返回值 集合元素

说明: 匹配所有的不可见元素，input 元素的 type 属性为 “hidden” 的话也会被匹配到.意思是css中display:none和input type=”hidden”的都会被匹配到.同样,要在脑海中彻底分清楚冒号”:”, 点号”.”和逗号”,”的区别.

2、:visible 用法: $(”tr:visible”) 返回值 集合元素

说明: 匹配所有的可见元素.

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    alert($("tr:hidden").text()); //输出不可见元素，即隐藏元素
    alert($("tr:visible").text()); //处处可见元素
});
</script>
</head>
<body>
<table border="1" width="25%">
    <tr style="display:none;"><td>01</td><td>sun</td></tr>
    <tr><td>02</td><td>jim</td></tr>
</table>
</body>
</html>
```

属性过滤选择器

属性过滤选择器的过滤规则是通过元素的属性来获取相应的元素

1、[attribute] 用法: $(”div[id]“) ; 返回值 集合元素

说明: 匹配包含给定属性的元素. 例子中是选取了所有带”id”属性的div标签.

2、[attribute=value] 用法: $(”input[name='newsletter']“); 返回值 集合元素

说明: 匹配给定的属性是某个特定值的元素.例子中选取了所有 name 属性是 newsletter 的 input 元素.

3、[attribute!=value] 用法: $(”input[name!='newsletter']“); 返回值 集合元素

说明: 匹配所有不含有指定的属性，或者属性不等于特定值的元素.此选择器等价于:not([attr=value]),要匹配含有特定属性但不等于特定值的元素,请使用[attr]:not([attr=value]).之前看到的 :not 派上了用场.

4、[attribute^=value] 用法: $(”input[name^='news']“) 返回值 集合元素

说明: 匹配给定的属性是以某些值开始的元素。

5、[attribute$=value] 用法: $(”input[name$='letter']“) 返回值 集合元素

说明: 匹配给定的属性是以某些值结尾的元素.

6、[attribute*=value] 用法: $(”input[name*='man']“) 返回值 集合元素

说明: 匹配给定的属性是以包含某些值的元素.

7、[attributeFilter1][attributeFilterN] 用法: $(”input[id][name$='man']“) 返回值 集合元素

说明: 复合属性选择器,需要同时满足多个条件时使用.又是一个组合,这种情况

我们实际使用的时候很常用.这个例子中选择的是所有含有 id 属性,并且它的 name 属性是以 man 结尾的元素。

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
   //alert($("div[id]").text()); //获取div中含有id属性的元素
   //alert($("div[class='a1']").text()); //获取div中含有class='a1'的元素
   //alert($("div[class!='a1']").text()); //获取div中不含有class='a1'的div元素
   //alert($("div[class!='a1']").text()); //获取div中不含有class='a1'的div元素
   //alert($("div:not([class='a1'])").text());//等价于
alert($("div[class!='a1']").text());
   //alert($("div[style^='width:100px']").text()); //获取div中style属性以某个值开始
的div元素
   //alert($("div[style$='color:red;']").text()); //获取div中style属性以某个值结尾的
div元素
   //alert($("div[style*='color']").text()); //获取div中style属性包含某个值的div元素
   alert($("div[id][style*='color']").text()); //复合属性,含有id，且style中含有
color
});
</script>
</head>
<body>
```

```
<div class="a1">
    <p>joy</p>
</div>
<div id="a2">
    <span>hello</span>
</div>
<div id="a3" style="width:100px;color:red;">
    <span>jquery</span>
</div>
</body>
</html>
```

子元素过滤选择器

1、:nth-child(index/even/odd) 用法: $(”ul li:nth-child(2)”) 返回值 集合元素

说明: 匹配其父元素下的第N个子或奇偶元素.这个选择器和之前说的基础过滤(Basic Filters)中的eq() 有些类似,不同的地方就是前者是从0开始,后者是从1开始.

2、:first-child 用法: $(”ul li:first-child”) 返回值 集合元素

说明: 匹配第一个子元素.':first' 只匹配一个元素,而此选择符将为每个父元素匹配一个子元素.这里需要特别点的记忆下区别.

3、:last-child 用法: $(”ul li:last-child”) 返回值 集合元素

说明: 匹配最后一个子元素.':last'只匹配一个元素,而此选择符将为每个父元素匹配一个子元素.

4、: only-child 用法: $(”ul li:only-child”) 返回值 集合元素

说明: 如果某个元素是父元素中唯一的子元素,那将会被匹配.如果父元素中含有其他元素,那将不会被匹配.意思就是:只有一个子元素的才会被匹配!

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    //alert($("#a2 li:nth-child(3n+1)").text());
            // index:从1开始的索引值，even偶数，odd奇数，3n+1循环
    //alert($("ul li:first-child").text());
            //为每个ul元素下的li都匹配一个子元素
    //alert($("ul li:last-child").text());
            //为每个ul元素下的li都匹配一个子元素
    alert($("ul li:only-child").text());
            //匹配ul下含有只含有一个li的元素
});
</script>
</head>
<body>
<ul>
    <li>第一个</li>
    <li>第二个</li>
```

```
    <li>第三个</li>
</ul>
<ul id="a2">
    <li>index1</li>
    <li>index2</li>
    <li>index3</li>
</ul>
<ul id="a3">
    <li>唯一</li>
</ul>
</body>
</html>
```

表单对象属性过滤选择器

此选择器主要对所选择的表单元素进行过滤

1、:enabled 用法: $(”input:enabled”) 返回值 集合元素

说明: 匹配所有可用元素.意思是查找所有input中不带有disabled=”disabled”的input.不为disabled,当然就为enabled啦.

2、:disabled 用法: $(”input:disabled”) 返回值 集合元素

说明: 匹配所有不可用元素.与上面的那个是相对应的.

3、:checked 用法: $(”input:checked”) 返回值 集合元素

说明: 匹配所有选中的被选中元素(复选框、单选框等，不包括select中的option).

4、:selected 用法: $(”select option:selected”) 返回值 集合元素

说明: 匹配所有选中的option元素.

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("input:enabled").css("background-color","green"); //将所有可用元素背景变为绿色
    $("input:disabled").css("background-color","blue");////将所有不可用元素背景变为蓝色

    $("input:checked").each(function(i){
        alert($(this).val());
    });   //获取所有被选中的值，循环输出
    alert($("select option:selected").val());
});
</script>
</head>
<body>
<input type="text" disabled="disabled" value="disabled" />
<input type="text" value="enabled" />
<input type="text" value="enabled2" />
```

```
<br/>
  <input type="checkbox" name="week" checked="checked" value="周一">周一</input>
  <input type="checkbox" name="week" value="周二">周二</input>
  <input type="checkbox" name="week" checked="checked" value="周三">周三</input>
 <br/>
<select>
  <option value="1">唱歌</option>
  <option value="2" selected="selected">跳舞</option>
  <option value="3">学习</option>
</select>
</body>
</html>
```

表单选择器

1、:input 用法: $(":input") ; 返回值 集合元素

说明:匹配所有 input, textarea, select 和 button 元素

2、:text 用法: $(":text") ; 返回值 集合元素

说明: 匹配所有的单行文本框.

3、:password 用法: $(":password") ; 返回值 集合元素

说明: 匹配所有密码框.

4、:radio 用法: $(":radio") ; 返回值 集合元素

说明: 匹配所有单选按钮.

5、:checkbox 用法: $(":checkbox") ; 返回值 集合元素

说明: 匹配所有复选框

6、:submit 用法: $(":submit") ; 返回值 集合元素

说明: 匹配所有提交按钮.

> 案例展示

```
<!DOCTYPE html >
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript" src="jquery-1.11.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $(":input").each(function(i){
        alert($(this).val());
    });   //输出所有input元素
    /*$(":text").each(function(i){
        alert($(this).val());
    });   //输出所有type="text"的元素
    $(":password").each(function(i){
        alert($(this).val());
    });  //输出所有type="password"的元素
    $(":radio").each(function(i){
        alert($(this).val());
```

```
    }); //输出所有type="radio"的元素
    $(":checkbox").each(function(i){
        alert($(this).val());
    }); //输出所有type="checkbox"的元素
    $(":submit").each(function(i){
        alert($(this).val());
    });*/
});
</script>
</head>
<body>
   <input type="text" value="name" /><br/>
   <input type="password" value="password" /><br/>
   <input type="radio" value="男">男</input><br/>
   <input type="checkbox" value="爱好">爱好</input><br/>
   <input type="submit" value=" 提交" />
</body>
</html>
```

03 jQuery中DOM操作

jQuery对象与DOM对象

jQuery 对象就是通过jQuery包装DOM对象后产生的对象

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: $(“#tab”).html();

jQuery 对象无法使用 DOM 对象的任何方法, 同样 DOM 对象也不能使用 jQuery 里的任何方法

建议约定：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

jQuery 对象不能使用 DOM 中的方法, 但如果 jQuery 没有封装想要的方法, 不得不使用 DOM方法的时候, 有如下两种处理方法:

(1) jQuery 对象是一个数组对象, 可以通过 [index] 的方法得到对应的 DOM对象.

$("#msg")[0]

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

$("#msg").get(0)

jQuery中DOM操作

- **DOM**

(Document Object Model—文档对象模型)：一种与浏览器, 平台, 语言无关的接口, 使用该接口可以轻松地访问页面中所有的标准组件。

- **DOM操作的分类**

DOM Core: DOM Core 并不专属于 JavaScript, 任何一种支持 DOM 的程序设计语言都可以使用它. 它的用途并非仅限于处理网页, 也可以用来处理任何一种是用标记语言编写出来的文档, 例如: XML

HTML DOM: 使用 JavaScript 和 DOM 为 HTML 文件编写脚本时， 有许多专属于 HTML-DOM 的属性

CSS-DOM:针对于 CSS 操作, 在 JavaScript 中, CSS-DOM 主要用于获取和设置 style 对象的各种属性

DOM操作节点属性和方法

属性节点

attr(): 获取html属性和设置属性

当为该方法传递一个参数时, 即为某元素的获取指定属性：$("p").attr("style");

当为该方法传递两个参数时, 即为某元素设置指定属性的值：$("p").attr("style","color:red;");

jQuery 中有很多方法都是一个函数实现获取和设置。

如: attr(), html(), text(), val(), height(), width(), css() 等.

text() - 设置或返回所选元素的文本内容

html() - 设置或返回所选元素的内容（包括 HTML 标记）

val() - 设置或返回表单字段的值

removeAttr(): 删除指定元素的指定属性。

addClass():为每个匹配的元素添加指定的类名，可添加多个内容(以空格隔开)。

removeClass()：从所有匹配的元素中删除全部或者指定的类。

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(document).ready(function(){
                $("#insertAttr").click(function(){
                    $("#attr1 p").attr("style","color:red;");
                });

                $("#getAttr").click(function(){
                    alert($("#attr2 p").attr("id"));
                });

                $("#removeAttr").click(function(){
                    $("#attr3 p").removeAttr("style");
                });
            });
```

```
        </script>
    </head>
    <body>
        <div id="attr1"><p style="color:yellow;background-color: gray;">给p标签插入属性style=“color:red;”</p></div>
        <input type="button" id="insertAttr" value="插入属性" />

        <div id="attr2"><p id="getA">获取p标签的id属性getA</p></div>
        <input type="button" id="getAttr" value="获取属性" />

        <div id="attr3"><p style="color:goldenrod;">删除p标签的颜色属性</p></div>
        <input type="button" id="removeAttr" value="删除属性" />
    </body>
</html>
```

创建节点

使用 jQuery 的工厂函数 $(): $(html); 会根据传入的 html 标记字符串创建一个 DOM 对象, 并把这个 DOM 对象包装成一个 jQuery 对象返回.

注意:

动态创建的新元素节点不会被自动添加到文档中, 而是需要使用其他方法将其插入到文档中;

当创建单个元素时, 需注意闭合标签和使用标准的 XHTML 格式. 例如创建一个 元素, 可以使用 $(“ ”) 或 $(“ ”), 但不能使用 $(“ ”) 或 $(“<P>”)

创建文本节点就是在创建元素节点时直接把文本内容写出来; 创建属性节点也是在创建元素节点时一起创建

插入节点

动态创建了 HTML 元素之后, 还需要将新创建的节点插入到文档中, 即成为文档中某个节点的子节点。

方法	描述	示例
append()	向每个匹配的元素的内部的结尾处	a.append(b):将b追加到a中结尾处
appendTo()	将每个匹配的元素追加到指定元素中的内部的结尾处	a.appendTo(b):将a追加到b中结尾处
prepend()	将每个匹配的元素的内部的开始处插入内容	a.prepend(b):将b追加到a中开始处
prependTo()	将每个匹配的元素插入到指定的元素内部的开始处	a.prependTo(b):将a追加到b中开始处
after()	向每个匹配的元素之后插入内容	a.after(b):将b追加到a后面
insertAfter()	在被选元素之后插入 HTML 标记或已有的元素	a.insertAfter(b):将a追加到b后面
before()	向每个匹配的元素之前插入内容	a.before(b):将b追加到a前面
insertBefore()	方法在被选元素之前插入 HTML 标记或已有的元素	a.insertBefore(b):将a追加到b前面

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(document).ready(function(){
                //$("div").append("<div>java</div>");
                      //在div内结尾处追加<div>java</div>（内部）
               //$("<div>java</div>").appendTo("div");
                      //将<div>java</div>追加到div内结尾处（内部）
               //$("div").prepend("<div>world</div>");
                     //在div内最前面加入<div>world</div>（内部）
               //$("<div><p>world</p></div>").prependTo("div");
                     //将<div>world</div>加入到div内部最前面（内部）
               //$("div").after("<div>java</div>");
                     //在div后追加<div>java</div>(外部)
              //$("<div>java</div>").insertAfter("div");
                     //将<div>java</div>追加到div后面(外部)
              //$("div").before("<div>world</div>");
                     //在div前面追加<div>world</div>(外部)
              $("<div>world</div>").insertBefore("div");
                    //将<div>world</div>追加到div前面(外部)
            });
        </script>
    </head>
    <body>
        <div style="background-color: yellow;">hello</div>
    </body>
```

```
</html>
```

删除节点

remove(): 从 DOM 中删除所有匹配的元素, 传入的参数用于根据 jQuery 表达式来筛选元素. 当某个节点用 remove() 方法删除后, 该节点所包含的所有后代节点将被同时删除. 这个方法的返回值是一个指向已被删除的节点的引用。

empty(): 清空节点 – 清空元素中的所有后代节点(不包含属性节点)。

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(document).ready(function(){
                //$("p").remove(); //直接删除所有含有p标签的节点
                $("p").empty(); //直接删除p标签内的信息
            });
        </script>
    </head>
    <body>
        <p id="" style="background-color: gold; width: 100px; height:
30px;">demo1</p>
        <p style="background-color: greenyellow; width: 100px; height:
30px;">demo2</p>
    </body>
</html>
```

复制节点

clone(): 克隆匹配的 DOM 元素, 返回值为克隆后的副本. 但此时复制的新节点不具有任何行为.不克隆绑定事件。

clone(true): 复制元素的同时也复制元素中的的事件 ，克隆绑定事件。

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(document).ready(function(){
                $("#source p").click(function(){
                    //$(this).clone().appendTo("#target");
```

```
                        //将选中的p标签加入#target中，不包含绑定事件
                    $(this).clone(true).appendTo("#target");
                        //将选中的p标签加入#target中，会有和p一样的效果，点击会插入
target后面
                });
            });
        </script>
    </head>
    <body>
        <div id="source">
            <p>goods1</p>
            <p>goods2</p>
            <p>goods3</p>
    </div>
    <br />
        <div id="target" style="border:1px dashed red; width: 200px;height:
200px;">
        </div>
    </body>
</html>
```

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;

即：使用后面的元素替换前面的元素：$(old).replaceWith(new)

replaceAll(): 颠倒了的 replaceWith() 方法.

即：使用前面的元素替换后面的元素：$(new).replaceWith(old)

注意: 若在替换之前, 已经在元素上绑定了事件, 替换后原先绑定的事件会与原先的元素一起消失

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            function replaceWith(){
                $("div #with").replaceWith("<p>你好，世界</p>");
                    //将div中id=“with”的标签及内容全部替换成<p>你好，世界</p>
            }

            function replaceAll(){
                $("<p>今晚吃鸡</p>").replaceAll("div #all");
                    //把<p>今晚吃鸡</p>标签及内容全部提换div #all，包括#all标签，和
replaceWith相反
            }
        </script>
    </head>
    <body>
        <div>
```

```
            <p id="with" style="color: red;">hello world!!!</p>
            <input type="button" onclick="return replaceWith();"
value="replaceWith"></input>
            <p id="all" style="color: red;">大吉大利</p>
            <input type="button" onclick="return replaceAll();"
value="replaceAll"></input>
        </div>
    </body>
</html>
```

遍历节点

函数	描述
.children()	获得匹配元素集合中每个元素的所有子元素。该方法只考虑子元素而不考虑任何后代元素
.each()	对 jQuery 对象进行迭代，为每个匹配元素执行函数。
.filter()	将匹配元素集合缩减为匹配选择器或匹配函数返回值的新元素。
.next()	获得匹配元素集合中每个元素紧邻的同辈元素。
.prev()	获得匹配元素集合中每个元素紧邻的前一个同辈元素，由选择器筛选（可选）。
.siblings()	获得匹配元素集合中所有元素的同辈元素，由选择器筛选（可选）。
.slice()	将匹配元素集合缩减为指定范围的子集。

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(function(){
              $("button").click(function(){
               fun1();
              });
            });
            function fun1(){
            /*  $("#parent1").children().each(function(){
                    alert("child:"+$(this).text());  //获取子类中所有元素
                });
                $("#parent2").next().each(function(){ //获取后面的紧邻元素
                    alert("next:"+$(this).text());
                });
                $("#parent2").prev().each(function(){ //获取前面的紧邻元素
                    alert("prev:"+$(this).text());
                });*/
```

```
                $("#parent2").siblings().each(function(){ //获取同辈所有元素
                    alert("siblings:"+$(this).text());
                });
            }
</script>
</head>
<body>
<div id="parent1">
    <div id="sun1">
        内容1-1
        <p style="margin-left: 50px;">内容1-1-1</p>
    </div>
    <div id="sun2">
        内容1-2
        <p style="margin-left: 50px;">内容1-2-1</p>
    </div>

</div>
<div id="parent2">内容2</div>
<div id="parent3">内容3</div>
<button id="but1">按钮</button>
</body>
</html>
```

04 jQuery事件处理

页面载入事件

$(document).ready()在页面加载完成后执行，和window.load效果类似，区别在于：

window.load需要在页面文档和资源都加载完成后，才执行

$(document).ready()中的事件，在页面文档加载完成后即执行

$(document).ready()可以多次写入新函数，这样在页面加载完成后，所有函数都会执行

简写方式：$(function(){ ... });

例：

```
$(function(){
    alert(“jquery函数");
});
```

Jquery处理事件方法

事件处理

bind：为元素绑定事件处理函数

- 一个事件绑定一个处理函数
    - 多个事件绑定同一个处理函数
        - 多个事件绑定不同的处理函数

- 阻止事件提交：preventDefault()；

one：绑定一次性的事件处理函数

unbind：删除绑定的事件

trigger： 触发被选元素的指定事件

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(function(){
                $("#button1").bind("click",function(){
                    alert($("#a1").text());
                });

                $("#button2").bind("click mouseout",function(){
                    alert($("#a2").text());
                });

                $("#button3").bind({
                    "mouseover":function(){
                        $("#a3").css("color","green");},
                    "click":function(){
                        alert($("#a3").text());},
                    "mouseout":function(){
                        $("#a3").css("color","red");}
                });

                $("form").submit(function(event){
                    //event.preventDefault();
                });
                $("p").one("change",function(){
                        //为所有p标签绑定一个一次性的事件处理函数，即只能弹出一次
                    alert($(this).text());
                });

                $("p").unbind( "click" ); //给所有p标签取消click事件
                $("input").trigger("focus"); //触发被选元素的指定事件
            });
        </script>
    </head>
    <body>
        <p id="a1">一个事件绑定一个处理函数</p>
        <input type="button" value="事件绑定" id="button1" />


        <p id="a2">多个事件绑定同一处理函数</p>
        <input type="button" value="事件绑定" id="button2" />

        <p id="a3">多个事件绑定多个处理函数</p>
```

```
        <input type="button" value="事件绑定" id="button3" />
        <form action="../index.html">
            <input type="text" /><br />
            <input type="text" /><br />
            <input type="text" /><br />
            <input type="submit" value="阻止事件提交" />
        </form>

    </body>
</html>
```

普通事件

click：绑定或触发单击事件

dbclick：绑定或触发双击事件

focus：绑定或触发focus事件

blur：绑定或触发blur事件

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(function(){
                $("#button1").click(function(){
                    alert($("#a1").text());
                });

                $("#button2").dblclick(function(){
                    alert($("#a2").text());
                });

                $("#button3").focus(function(){
                    alert($("#a3").text());
                });
                $("#button4").blur(function(){
                    alert($("#a4").text());
                });

            });
        </script>
    </head>
    <body>
        <p id="a1">事件click</p>
        <input type="button" value="事件click" id="button1" />
```

```
        <p id="a2">事件dbclick</p>
        <input type="button" value="事件dbclick" id="button2" />

        <p id="a3">事件focus</p>
        <input type="text" value="事件focus" id="button3" />

        <p id="a4">事件blur</p>
        <input type="text" value="事件blur" id="button4" />

    </body>
</html>
```

交互处理

hover：模仿悬停事件

> 案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{
                border: 1px dotted mediumvioletred;
                height: 200px;
                width: 200px;
            }
        </style>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(function(){
              $("div").hover(function(){
                $("div").css("background-color","yellow"); //鼠标放上去in
              },function(){
                $("div").css("background-color","pink"); //鼠标移除后
              });
            });
        </script>
    </head>
    <body>
        <div>
        </div>
    </body>
</html>
```

05 jQuery动画

动画效果

show()：显示元素

hide()：隐藏元素

fadeIn()：淡入

fadeOut()：淡出

fadeToggle()：淡入淡出

fadeTo()：调整元素的透明度到某个特定的值

animate()：用于创建自定义动画效果

stop()：停止指定元素上的的动画效果

delay()：设置一个动画延迟开始的值

案例展示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript" src="../js/jquery-1.11.1.min.js" >
</script>
        <script type="text/javascript">
            $(function(){
                $("#show").click(function(){
                    $("#showing").show(3000);
                        //从隐藏到显示,要求#showing刚开始为隐藏状态
                });
                $("#hide").click(function(){
                    $("#hiding").hide(3000);
                        //从显示到隐藏,要求#hiding刚开始为显示状态
                });

                $("#fadeIn").click(function(){
                    $("#fadeIning").fadeIn(3000);
                        //从隐藏到显示，淡淡的出来,要求#showing刚开始为隐藏状态
                });
                $("#fadeOut").click(function(){
                    $("#fadeOuting").fadeOut(3000);
                        //从显示到隐藏,淡淡的出来，要求#hiding刚开始为显示状态
                });
                $("#fadeOut").click(function(){
                    $("#fadeOuting").fadeOut(3000);
                        //从显示到隐藏,淡淡的出来，要求#hiding刚开始为显示状态
                });

                $("#fadeToggle").click(function(){
                    $("#fadeToggleing").fadeToggle(3000);
                        //从显示到隐藏,淡淡的出来，要求#hiding刚开始为显示状态
                });

                $("#fadeTo").click(function(){
                    $("#fadeToing").fadeTo(3000,0.3); //透明度
```

```
                });

                function animate() {
                    $("#ani").animate({width:100},1000);
                    $("#ani").animate({width:300},1000,animate);
                }
                animate();

                $("#stop").click(function(){
                    $("#ani").stop(); //停止动画
                });

                $("#delay").click(function(){
                    $("#ani").delay(5000); //延迟动画执行
                });
            });
        </script>
    </head>
    <body>
        <p style="display: none;" id="showing">显示了</p>
        <input type="button" id="show" value="show" />

        <p id="hiding">隐藏了</p>
        <input type="button" id="hide" value="hide" />
        <br />
        <p style="display: none;" id="fadeIning">淡入</p>
        <input type="button" id="fadeIn" value="fadeIn" />

        <p id="fadeOuting">淡出</p>
        <input type="button" id="fadeOut" value="fadeOut" />

        <p id="fadeToggleing">淡入淡出</p>
        <input type="button" id="fadeToggle" value="fadeToggle" />

        <p id="fadeToing">透明度</p>
        <input type="button" id="fadeTo" value="fadeTo" />

        <div id="ani" style="background-color: darkgreen;
                width: 100px;height: 100px;">
        </div>
        <input type="button" id="stop" value="stop" />
        <input type="button" id="delay" value="delay" />
    </body>
</html>
```