jQuery

1 jQuery简介

jQuery 是一个 JavaScript 代码库，它的核心思想是“write less, do more” （写得更少，做得更多）。

2 jQuery核心函数

`$`是 jQuery 的核心函数，使用`$()`调用该函数。`$`函数具有很多功能：

1. 传入参数为函数时，在页面加载完成后执行该函数，相当于`window.onload = function(){};`。
2. 传入参数为 HTML 字符串时，根据这个字符串创建元素节点对象。
3. 传入参数为选择器字符串时，根据这个字符串查找元素节点对象。
4. 传入参数为 DOM 对象时，将 DOM 对象包装为 jQuery 对象返回。

例如，用 jQuery 为按钮绑定单击响应函数的代码如下：

```
<head>
    <!-- 引入 jQuery -->
    <script type="text/javascript" src="jquery-1.7.2.js"></script>
    <script type="text/javascript">
        // 页面加载完成后执行该函数
        $(function(){
            // 用选择器查找按钮元素对象
            // jQuery 对象的名字习惯上以 $ 打头
            var $btn = $("#btn");
            // 绑定单击响应函数
            $btn.click(function(){
                alert("hello world");
            })
        });
    </script>
</head>
<body>
    <button id="btn">SayHello</button>
</body>
```

3 jQuery对象

3.1 什么是jQuery对象

1. 通过 jQuery 提供的 API 创建的对象，是 jQuery 对象。
2. 通过 jQuery 包装的 DOM 对象，是 jQuery 对象。
3. 通过 jQuery 提供的 API 查询到的对象，是 jQuery 对象。

jQuery 对象的本质：jQuery 对象 = DOM 对象的数组 + jQuery 提供的一系列功能函数

3.2 jQuery对象与DOM对象的区别

1. 在获取方式上：通过`getElementById()`、`getElementByTagName()`、`getElementByName()`、`createElement()`得到的对象是 DOM 对象，通过 jQuery 提供的 API 得到的对象是 jQuery 对象。
2. 在输出效果上：使用`alert()`输出对象信息时，DOM 对象输出为`[object HTMLxxxElement]`，jQuery对象输出为`[object Object]`。
3. 在使用方法上：jQuery 对象不能使用 DOM 对象的属性和方法，DOM 对象也不能使用 jQuery 对象的属性和方法。

3.3 jQuery对象与DOM对象的相互转换

DOM 对象转换为 jQuery 对象：`$(DomObject)`。

jQuery 对象转换为 DOM 对象：`jQueryObject[index]`。

4 jQuery选择器

4.1 基本选择器

1. id 选择器：`#id`，根据`id`属性值查找元素。
2. 类选择器：`.class`，根据`class`属性值查找元素。
3. 标签选择器：`element`，根据标签名查找元素。
4. 全选选择器：`*`，查找所有元素。
5. 组合选择器：`selector1,selector2,selectorN`，将每个选择器匹配到的元素合并后一起返回。

通过选择器查找到的元素按照 HTML 文档中从上到下的顺序排列。

4.2 层级选择器

1. 后代选择器：`ancestor descendant`，在给定的祖先元素下匹配所有的后代元素。
2. 子代选择器：`parent>child`，在给定的父元素下匹配所有的子元素。
3. 相邻兄弟选择器：`prev+next`，匹配所有紧接在 prev 元素后的 next 元素。
4. 后继兄弟选择器：`prev~siblings`，匹配 prev 元素之后的所有 siblings 元素。

4.3 过滤选择器

4.3.1 基本过滤器

1. `:first`：获取匹配到的第一个元素。
2. `:last`：获取匹配到的最后一个元素。
3. `:not(selector)`：去除所有与给定选择器匹配的元素。
4. `:even`：匹配所有索引值为偶数的元素，从 0 开始计数。
5. `:odd`：匹配所有索引值为奇数的元素，从 0 开始计数。
6. `:eq(index)`：匹配一个给定索引值的元素。
7. `:gt(index)`：匹配所有大于给定索引值的元素。
8. `:lt(index)`：匹配所有小于给定索引值的元素。
9. `:header`：匹配 `h1` ~ `h6` 的标题元素。
10. `:animated`：正在执行动画的所有元素。

4.3.2 内容过滤器

1. `:contains(text)`：匹配包含给定文本的元素。
2. `:empty`：匹配所有不包含子元素或文本的空元素。
3. `:parent`：匹配含有子元素或文本的元素。
4. `:has(selector)`：匹配含有选择器所匹配的元素的元素。

4.3.3 属性过滤器

1. `[attribute]`：匹配含有给定属性的元素。
2. `[attribute=value]`：匹配给定的属性是某个特定值的元素。
3. `[attribute!=value]`：匹配不含有指定的属性，或者属性不等于特定值的元素。
4. `[attribute^=value]`：匹配给定的属性是以某些值开始的元素。
5. `[attribute$=value]`：匹配给定的属性是以某些值结尾的元素。
6. `[attribute*=value]`：匹配给定的属性包含某些值的元素。
7. `[selector1][selector2][selectorN]`：复合属性选择器，选择同时满足所有条件的元素。

4.3.4 表单过滤器

1. `:input`：匹配所有 `input`、`textarea`、`select` 和 `button` 元素。
2. `:text`：匹配所有的 `type` 属性值为 `text` 的 `input` 元素，即所有单行文本框。
3. `:password`：匹配所有的 `type` 属性值为 `password` 的 `input` 元素，即所有密码框。
4. `:radio`：匹配所有的 `type` 属性值为 `radio` 的 `input` 元素，即所有单选按钮。
5. `:checkbox`：匹配所有的 `type` 属性值为 `checkbox` 的 `input` 元素，即所有复选框。
6. `:submit`：匹配所有的 `type` 属性值为 `submit` 的 `input` 元素，即所有提交按钮。
7. `:reset`：匹配所有的 `type` 属性值为 `reset` 的 `input` 元素，即所有重置按钮。
8. `:button`：匹配所有按钮。
9. `:file`：匹配所有文件域。

4.3.5 表单对象属性过滤器

1. `:enabled`：匹配所有可用元素。
2. `:disabled`：匹配所有不可用元素。
3. `:checked`：匹配所有选中的元素（复选框、单选框等），不包括`select`中的`option`。
4. `:selected`：匹配所有选中的`option`元素。

4.3.6 可见性过滤器

1. `:hidden`：匹配所有不可见元素，或者`type`为`hidden`的元素。
2. `:visible`：匹配所有的可见元素。

5 jQuery元素筛选函数

1. `eq(index)`：获取索引为 n 的元素。
2. `first()`：获取第一个元素。
3. `last()`：获取最后一个元素。
4. `filter(expr|obj|ele|fn)`：筛选出与指定表达式匹配的元素集合。其参数可以为：
   （1）`expr`：选择器字符串。
   （2）`obj`：jQuery 对象。
   （3）`ele`：DOM 元素对象。
   （4）`fn`：回调函数。它接收一个参数`index`，表示元素在 jQuery 集合中的索引。回调函数中的`this`表示当前的 DOM 元素。
5. `is(expr|obj|ele|fn)`：判断元素集合中是否包含与指定表达式匹配的元素。如果有则返回`true`；如果没有元素符合，或者表达式无效，都返回`false`。
6. `has(expr|ele)`：保留包含特定后代的元素，去掉那些不含有指定后代的元素。
7. `not(expr|ele|fn)`：删除与指定表达式匹配的元素。
8. `children([expr])`：取得一个包含匹配的元素集合中每一个元素的所有子元素的元素集合。可以通过可选的表达式来过滤所匹配的子元素。
9. `find(expr|obj|ele)`：搜索所有与指定表达式匹配的元素。这个函数可以用于查找正在处理的元素的后代元素。
10. `next([expr])`：取得一个包含匹配的元素集合中每一个元素的下一个相邻兄弟元素的元素集合。
11. `nextAll([expr])`：查找当前元素之后所有的兄弟元素。可以用表达式过滤。
12. `nextUntil([exp|ele][,fil])`：查找当前元素之后所有的兄弟元素，直到遇到匹配的那个元素为止。如果没有选择器匹配到，或者没有提供参数，那么跟在后面的所有兄弟元素都会被选中，这种情况下与没有参数的`nextAll()`效果一样。
13. `parent([expr])`：取得一个包含着所有匹配元素的父元素的元素集合。可以用可选的表达式筛选。

14. `prev([expr])`：取得一个包含匹配的元素集合中每一个元素的上一个相邻兄弟元素的元素集合。可以用可选的表达式进行筛选。
15. `prevAll([expr])`：查找当前元素之前所有的兄弟元素。
16. `prevUntil([expr|ele][,fil])`：查找当前元素之前所有的兄弟元素，直到遇到匹配的那个元素为止。如果没有选择器匹配到，或者没有提供参数，那么前面的所有兄弟元素都会被选中，这种情况下与没有参数的`prevAll()`效果一样。
17. `siblings([expr])`：取得一个包含匹配的元素集合中每一个元素的所有同辈元素的元素集合。可以用可选的表达式进行筛选。
18. `add(expr|ele|html|obj[,con])`：把与表达式匹配的元素添加到 jQuery 对象中。这个函数可以用于连接分别与两个表达式匹配的元素结果集。
    参数：
    （1）`expr`：选择器字符串。
    （2）`ele`：DOM 元素对象。
    （3）`html`：HTML 字符串。
    （4）`obj`：jQuery 对象。
    （5）`con`：可选。待查找的 DOM 元素集、文档或 jQuery 对象。

6 jQuery的属性操作

6.1 HTML属性

1. `html()`：获取或设置标签中的内容。与 DOM 属性`innerHTML`作用相同。
2. `text()`：获取或设置标签中的文本。与 DOM 属性`innerText`作用相同。
3. `val()`：获取或设置表单项的`value`属性值。与 DOM 属性`value`作用相同。

以上三个函数，不传参数时用于获取，传参数时用于设置。

`val()`函数可以用于设置表单项的选中状态，例如：

```
<head>
    <script src="jquery-1.7.2.js"></script>
    <script>
        $(function(){
            // 单选按钮
            $(":radio").val(["radio2"]);
            // 多选按钮
            $(":checkbox").val(["checkbox2", "checkbox1", "checkbox3"]);
            // 多选下拉列表
            $("#multiple").val(["mul2", "mul3"]);
            // 单选下拉列表
            $("#single").val(["sin2"]);
            // 同时操作多种表单项
            $(":radio,:checkbox,#multiple,#single").val(
                ["radio1", "checkbox1", "checkbox2", "mul1", "mul3", "sin1"]);
        });
    </script>
</head>
<body>
    单选:
    <input name="radio" type="radio" value="radio1">radio1
    <input name="radio" type="radio" value="radio2">radio2
    <br>
    多选:
    <input name="checkbox" type="checkbox" value="checkbox1">checkbox1
    <input name="checkbox" type="checkbox" value="checkbox2">checkbox2
    <input name="checkbox" type="checkbox" value="checkbox3">checkbox3
    <br>
    多选下拉列表:
    <select id="multiple" multiple="multiple" size="4">
        <option value="mul1">mul1</option>
        <option value="mul2">mul2</option>
        <option value="mul3">mul3</option>
    </select>
    <br>
    单选下拉列表:
    <select id="single">
        <option value="sin1">sin1</option>
        <option value="sin2">sin2</option>
        <option value="sin3">sin3</option>
    </select>
</body>
```

6.2 任意属性

1. `attr(name|pro|key, val|fn)`：获取在匹配的元素集合中的第一个元素的属性值，或者设置所有被选元素的一个或多个属性值。
2. `prop(name|pro|key, val|fn)`：获取在匹配的元素集合中的第一个元素的属性值，或者设置所有被选元素的一个或多个属性值。

参数：

1. `name`：属性名称。用于获取指定属性的值。
2. `pro`：“键/值对”对象。用于设置多个属性的值。
   注意：键的引号可以省略。但如果属性名为`class`，引号不能省略，因为`class`是 JavaScript 关键字。
3. `key, val`：属性名称、属性值。用于设置单个属性的值。
4. `key, fn`：属性名称、回调函数。回调函数可以有 2 个参数，第一个参数为当前元素的索引值，第二个参数为原先的属性值。回调函数的返回值为要设置的属性值。如果回调函数没有返回值或返回值为`undefined`，则不修改属性值，利用这一特性可以在满足某些条件时才选择性地设置属性值。

示例：

```
// 获取所有 img 标签的 src 属性值
$("img").attr("src");

// 为所有 img 标签设置 src 和 alt 属性值
$("img").attr({
    src: "test.jpg",
    alt: "Test Image"
});

// 为所有 img 标签设置 src 属性
$("img").attr("src", "test.jpg");

// 根据 title 属性的旧值计算新值并更新
$("img").attr("title", function(i, val){
    return val + " - photo by Kelly Clark";
});
```

`attr()`函数可以操作自定义属性，例如：

```
$("div").attr("abc", "abcValue"); // 为 div 设置 abc 属性
var abc = $("div").attr("abc"); // 获取 div 的 abc 属性值
```

如果属性未设置值，使用`attr()`函数获取该属性值时将返回`undefined`。例如，当单选框未选中时，使用`attr()`函数获取`checked`属性值时将返回`undefined`；当单选框选中时，使用`attr()`函数获取`checked`属性值时将返回`"checked"`。

对于布尔属性，如`checked`、`selected`、`disabled`、`readonly`、`required`等，使用`prop()`函数将返回`true`或`false`，而`attr()`函数则不会。因此操作布尔属性时适合用`prop()`函数，操作其他属性时使用`attr()`函数。

练习：全选、全不选、反选。

```
<head>
    <script src="jquery-1.7.2.js"></script>
    <script>
        $(function(){
            // 全选按钮
            $("#checkedAllBtn").click(function(){
                // 将所有复选框设置为选中
                $(":checkbox").prop("checked", true);
            });

            // 全不选按钮
            $("#checkedNoBtn").click(function(){
                // 将所有复选框设置为未选中
                $(":checkbox").prop("checked", false);
            });

            // 反选按钮
            $("#checkedRecBtn").click(function(){
                // 获取 name 属性值为 items 的复选框，用 each() 函数进行遍历
                $(":checkbox[name='items']").each(function(){
                    // this 是当前遍历到的 DOM 对象
                    this.checked = !this.checked;
                });
                // 获取复选框的个数
                var allCount = $(":checkbox[name='items']").length;
                // 获取选中的复选框的个数
                var checkedCount = $(":checkbox[name='items']:checked").length;
                // 如果所有复选框都被选中，则全选复选框应选中，否则不选中
                $("#checkedAllBox").prop("checked", allCount == checkedCount);
            });

            // 提交按钮
            $("#sendBtn").click(function(){
                // 获取选中的复选框
                $(":checkbox[name='items']:checked").each(function(){
                    alert(this.value);
                });
            });

            // 全选复选框
            $("#checkedAllBox").click(function(){
                // 点击全选复选框时，将其他复选框设置为与自己状态相同
                $(":checkbox[name='items']").prop("checked", this.checked);
            });

            // 为每个复选框绑定单击响应函数
            $(":checkbox[name='items']").click(function(){
                // 获取复选框的个数
                var allCount = $(":checkbox[name='items']").length;
                // 获取选中的复选框的个数
                var checkedCount = $(":checkbox[name='items']:checked").length;
                // 如果所有复选框都被选中，则全选复选框应选中，否则不选中
```

```
                $("#checkedAllBox").prop("checked", allCount == checkedCount);
            });
        });
    </script>
</head>
<body>
    你爱好的运动是? <input type="checkbox" id="checkedAllBox">全选
    <br>
    <input type="checkbox" name="items" value="足球">足球
    <input type="checkbox" name="items" value="篮球">篮球
    <input type="checkbox" name="items" value="羽毛球">羽毛球
    <input type="checkbox" name="items" value="乒乓球">乒乓球
    <br>
    <input type="button" id="checkedAllBtn" value="全选">
    <input type="button" id="checkedNoBtn" value="全不选">
    <input type="button" id="checkedRecBtn" value="反选">
    <input type="button" id="sendBtn" value="提交">
</body>
```

7 DOM的增删改

内部插入:

1. `a.appendTo(b)`: 将 a 插入到 b 的末尾，使 a 成为 b 的最后一个子元素。
2. `a.prepentTo(b)`: 将 a 插入到 b 的开始，使 a 成为 b 的第一个子元素。

外部插入:

1. `a.insertAfter(b)`: 将 a 插入到 b 的后面。
2. `a.insertBefore(b)`: 将 a 插入到 b 的前面。

替换:

1. `a.replaceWith(b)`: 用 b 替换 a。
2. `a.relpaceAll(b)`: 用 a 替换 b。

删除:

1. `a.remove([expr])`: 删除所有匹配的元素。
2. `a.empty()`: 清空元素中的内容。

8 CSS样式操作

1. `addClass()`: 向被选元素添加一个或多个 `class` 属性值。

2. `removeClass()`：从被选元素删除一个或多个`class`属性值。当参数为空时，删除所有`class`属性值。
3. `toggleClass()`：切换`class`属性值，有就删除，没有就添加。
4. `offset()`：获取或设置第一个匹配元素相对于整个文档的位置。

9 jQuery动画操作

1. `show()`：将隐藏的元素显示。
2. `hide()`：将可见的元素隐藏。
3. `toggle()`：切换，可见就隐藏，不可见就显示。
4. `fadeIn()`：淡入。
5. `fadeOut()`：淡出。
6. `fadeToggle()`：切换，可见就淡出，不可见就淡入。

以上动画函数可以添加可选参数：

1. 动画的执行速度，可以是三种预定速度之一的字符串（`slow`、`normal`或`fast`），也可以是表示动画时长的毫秒数值。
2. 切换效果，默认是`swing`，可用参数`linear`。
3. 回调函数，当动画完成后自动执行。

`fadeTo([[speed], opacity, [easing], [fn]])`：在指定的时间内将透明度以渐进方式调整到指定的值。透明度取值为 0-1，0 为完全透明，1 为完全可见。参数为：

1. `speed`：三种预定速度之一的字符串（`slow`、`normal`或`fast`），或表示动画时长的毫秒数值。
2. `opacity`：目标透明度。
3. `easing`：用来指定切换效果，默认是`swing`，可用参数`linear`。
4. `fn`：回调函数，动画完成后自动执行。

10 jQuery事件操作

10.1 文档加载事件

原生 JavaScript 的页面加载完成事件为`window.onload = function(){};`。

jQuery 的页面加载完成事件为`$(document).ready(function(){});`，简写为`$(function(){});`。

原生 JavaScript 与 jQuery 的页面加载完成响应函数的区别为：

1. 页面加载完成后，jQuery 的响应函数先执行，原生 JavaScript 的后执行。
2. 对于 jQuery ，在浏览器的内核解析完页面并创建好 DOM 对象后就执行响应函数；对于原生 JavaScript，除了要等待浏览器的内核解析完页面并创建好 DOM 对象，还要等标签显示的内容加载完成。
3. 注册多个响应函数时，jQuery 会在页面加载完成后把所有函数按顺序依次执行。

10.2 常用的事件

1. `click([[data], fn])`：单击事件。
2. `mouseover([[data], fn])`：鼠标移入事件。
3. `mouseout([[data], fn])`：鼠标移出事件。

可选的参数：

1. `data`：可传入`data`供函数`fn`处理。
2. `fn`：事件响应函数。

当传递一个函数作为参数时，为事件绑定响应函数；当不传递参数时，触发事件的响应函数；当两个参数都传递时，将数据传入响应函数。

10.3 事件处理

1. `bind()`：为元素的一个或多个事件绑定事件处理函数。在 jQuery 3.0 中已弃用此函数，应使用`on()`代替。
2. `one()`：与`bind()`的用法相同，但响应函数只响应一次。
3. `unbind()`：与`bind()`的用法相同，用于解除绑定的事件处理函数。在 jQuery 3.0 中已弃用此函数，应使用`off()`代替。
4. `live()`：给所有匹配的元素绑定一个事件处理函数，即使这个元素是以后再添加进来的也有效。从 jQuery 1.7 开始，不再建议使用`live()`方法。

10.4 事件的冒泡

当子元素的事件被触发时，父元素的相同事件也被触发。

在事件响应函数的函数体内使用`return false;`可以阻止事件的冒泡。

10.5 事件对象

事件对象是封装有事件信息的 JavaScript 对象。当事件被触发时，浏览器自动向事件响应函数传递一个事件对象，可以在响应函数中添加一个形参来接收事件对象。

当用`bind()`函数为多个事件绑定同一个响应函数时，可以使用事件对象的`type`属性确定触发的事件类型，为不同事件提供不同的响应操作。例如：

```
$("div").bind("click mouseover", function(event){
    if (event.type == "click")
    {
        // 单击事件的响应操作
    }
    else if (event.type == "mouseover")
    {
        // 鼠标移入事件的响应操作
    }
});
```