jQuery入门

一、jQuery概述

1、JavaScript库

仓库：可以把很多东西放到这个仓库里面。找东西只需要到仓库里面查找到就可以了。

> JavaScript库：即 library，是一个封装好的特定的集合（方法和函数）。从封装一大堆函数的角度理解库，就是在这个库中，封装了很多预先定义好的函数在里面，比如动画animate、hide、show，比如获取元素等。
>
> 简单理解： 就是一个JS 文件，里面对我们原生js代码进行了封装，存放到里面。这样我们可以快速高效的使用这些封装好的功能了。
>
> 比如 jQuery，就是为了快速方便的操作DOM，里面基本都是函数（方法）。

常见的JavaScript库

> - jQuery
> - Prototype
> - YUI
> - Dojo
> - Ext JS
> - 移动端的zepto

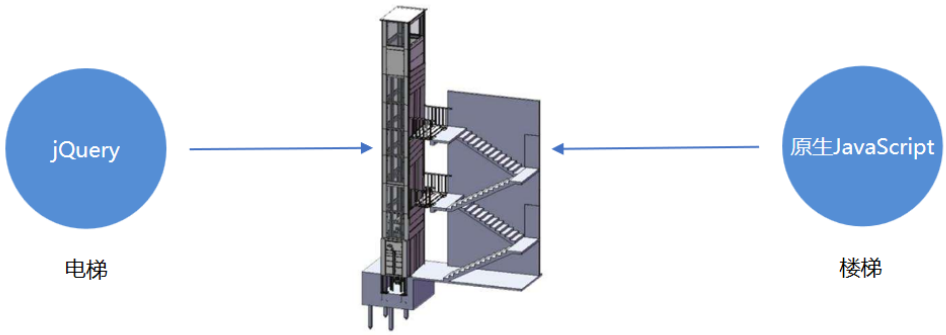
这些库都是对原生 JavaScript 的封装，内**部都是用 JavaScript 实现的**，我们主要学习的是 jQuery。

2、jQuery的概念

jQuery 是一个快速、简洁的 JavaScript 库，其设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。

j 就是 JavaScript； Query 查询； 意思就是查询js，**把js中的DOM操作做了封装，我们可以快速的查询使用里面的功能。**

> jQuery 封装了 JavaScript 常用的功能代码，优化了 DOM 操作、事件处理、动画设计和 Ajax 交互。
>
> 学习jQuery本质： 就是学习调用这些函数（方法）。
>
> jQuery 出现的目的是加快前端人员的开发速度，我们可以非常方便的调用和使用它，从而提高开发效率。



3、jQuery的优点

1. 轻量级。核心文件才几十kb，不会影响页面加载速度
2. 跨浏览器兼容。基本兼容了现在主流的浏览器
3. 链式编程、隐式迭代
4. 对事件、样式、动画支持，大大简化了DOM操作
5. 支持插件扩展开发。有着丰富的第三方的插件，例如：树形菜单、日期控件、轮播图等
6. 免费、开源

二、jQuery的基本使用

官网地址： https://jquery.com/

各个版本的下载：https://code.jquery.com/

1、jQuery使用步骤

> 1. 引入 jQuery 文件
> 2. 使用即可

2、jQuery的入口函数

```
$(function () {
    ...  // 此处是页面 DOM 加载完成的入口
 }) ;
```

```
$(document).ready(function(){
   ...  // 此处是页面DOM加载完成的入口
});
```

> 1. 等着 DOM 结构渲染完毕即可执行内部代码，不必等到所有外部资源加载完成，jQuery 帮我们完成了封装。
> 2. 相当于原生 js 中的 DOMContentLoaded。
> 3. 不同于原生 js 中的 load 事件是等页面文档、外部的 js 文件、css文件、图片加载完毕才执行内部代码。
> 4. 更推荐使用第一种方式。

```
$(function () {
    $('div').hide();
})
```

3、jQuery的顶级对象$

1. **$ 是 jQuery 的别称**

在代码中可以使用 jQuery 代替 $

但一般为了方便，通常都直接使用 $ 。

2. $ 是jQuery 的顶级对象， 相当于原生JavaScript中的 window。

把元素利用$包装成jQuery对象，就可以调用jQuery 的方法。

4、jQuery对象和DOM对象

1. 用原生 JS 获取来的对象就是 DOM 对象
2. jQuery 方法获取的元素就是 jQuery 对象。
3. jQuery 对象本质是： 利用$对DOM 对象包装后产生的对象（伪数组形式存储）。

```
<script>
    // 1.DOM对象：用原生 JS 获取来的对象就是 DOM 对象
    var myDiv = document.querySelector('div');
    var mySpan = document.querySelector('span');
    console.log(myDiv);
    // 2.jQuery对象：jQuery 方法获取的元素就是 jQuery 对象。
    $('div');  //$('div')是一个jQuery对象
    console.dir($('div'));
    $('span');
    myDiv.style.display = 'none';
    $('div').hide();
    // myDiv.hide(); myDiv是一个dom对象不能使用 jquery里面的hide方法
</script>
```

注意：

只有 jQuery 对象才能使用 jQuery 方法，DOM 对象则使用原生的 JavaScirpt 方法。

DOM 对象与 jQuery 对象之间是可以相互转换的。

因为原生js 比 jQuery 更大，原生的一些属性和方法 jQuery没有给我们封装。要想使用这些属性和方法需要把jQuery对象转换为DOM对象才能使用。

1. DOM 对象转换为 jQuery 对象： $(DOM对象)

```
$('div')
```

2. jQuery 对象转换为 DOM 对象（两种方式）

$('div') [index]　　index 是索引号

$('div') .get(index)　index 是索引号

```
<script>
    // 1. DOM 对象转换为 jQuery 对象：  $(DOM对象)
    // (1)直接获取视频，得到的就是jQuery对象
    $('video');
    // (2)已经使用原生js获取过来的DOM对象
    var myvideo = document.querySelector('video');
    $(myvideo);
    // 2. jQuery 对象转换为 DOM 对象（两种方式）
    $('video')[0].play();
    $('video').get(0).play();
</script>
```

jQuery常用API

一、jQuery选择器

1、jQuery基础选择器

原生 JS 获取元素方式很多，很杂，而且兼容性情况不一致，因此 jQuery 给我们做了封装，使获取元素统一标准。

$(“选择器”) // 里面选择器直接写 CSS 选择器即可，但是要加引号

名称	用法	描述
ID选择器	$("#id")	获取指定ID的元素
全选选择器	$('*')	匹配所有元素
类选择器	$(".class")	获取同一类class的元素
标签选择器	$("div")	获取同一类标签的所有元素
并集选择器	$("div,p,li")	选取多个元素
交集选择器	$("li.current")	交集元素

2、jQuery层级选择器

名称	用法	描述
子代选择器	`$("ul>li");`	使用>号，获取亲儿子层级的元素；注意，并不会获取孙子层级的元素
后代选择器	`$("ul li");`	使用空格，代表后代选择器，获取ul下的所有li元素，包括孙子等

jQuery 设置样式

```
$('div').css('属性', '值')
```

3、隐式迭代***

遍历内部 DOM 元素（伪数组形式存储）的过程就叫做隐式迭代。

简单理解：给匹配到的所有元素进行循环遍历，执行相应的方法，而不用我们再进行循环，简化我们的操作，方便我们调用。

4、jQuery筛选选择器

语法	用法	描述
:first	$('li:first')	获取第一个li元素
:last	$('li:last')	获取最后一个li元素
:eq(index)	$("li:eq(2)")	获取到的li元素中，选择索引号为2的元素，索引号index从0开始。
:odd	$("li:odd")	获取到的li元素中，选择索引号为奇数的元素
:even	$("li:even")	获取到的li元素中，选择索引号为偶数的元素

```
<script>
    $(function () {
        // 1.第一个元素
        $('ul li:first').css('color', 'red');
        // 2.第三个元素
        $('li:eq(2)').css('color', 'blue');
        // 3.索引号为奇数的元素
        $('ol li:odd').css('color', 'skyblue');
        $('ol li:even').css('color', 'pink');
    })
</script>
```

5、jQuery筛选方法

语法	用法	说明
`parent()`	`$("li").parent();`	查找父级
`children(selector)`	`$("ul").children("li")`	相当于`$("ul>li")`，最近一级（亲儿子）
`find(selector)`	`$("ul").find("li");`	相当于$("ul li"),后代选择器
`siblings(selector)`	`$(".first").siblings("li");`	查找兄弟节点，不包括自己本身
`nextAll([expr])`	`$(".first").nextAll()`	查找当前元素之后所有的同辈元素
`prevtAll([expr])`	`$(".last").prevAll()`	查找当前元素之前所有的同辈元素
`hasClass(class)`	`$('div').hasClass("protected")`	检查当前的元素是否含有某个特定的类，如果有，则返回true
`eq(index)`	`$("li").eq(2);`	相当于`$("li:eq(2)")`,`index`从0开始

重点记住： parent() children() find() siblings() eq()

```
<script>
    //  注意：都是方法，带括号
    $(function () {
        // 1.父 parent()
        // console.log($('.son').parent());
        $('.son').parent();

        // 2.子 children():子代选择器  find():后代选择器
        $('.nav').children('p').css('color', 'blue');
        $('.nav').find('p').css('color', 'red');

        // 3.兄弟元素 siblings：除了自己以外的所有亲兄弟
        // console.log($('.item').siblings());
        $('.item').siblings('li').css('color', 'pink');

        // 当前元素之后的所有兄弟
        $('.item').nextAll('li').css('color', 'blue');
        // 当前元素之前的所有兄弟
        $('.item').prevAll('li').css('color', 'green');

        // 4.第n个元素
        var index = 2;
        $('ul li:eq(2)').css('color', 'pink');
        // $('ul li:eq('+index+')').css('color','pink');
        $('ul li').eq(4).css('color', 'orange');
        // $('ul li').eq(index).css('color', 'orange');

        // 5.判断是否有某个类名  Boolean
        console.log($('.has div:first').hasClass('current'));
        console.log($('.has div:last').hasClass('current'));
    })
</script>
```

6、排他思想

> 想要多选一的效果，排他思想：当前元素设置样式，其余的兄弟元素清除样式。

```
$(this).css( “color” ,” red” );
$(this).siblings(). css( “color” ,” ” );
```

7、链式编程

链式编程是为了节省代码量，看起来更优雅。

```
$(this).css('color', 'red').sibling().css('color', '');
```

使用链式编程一定注意是哪个对象执行样式.

二、jQuery样式操作

1、操作CSS方法

jQuery 可以使用 css 方法来修改简单元素样式； 也可以操作类，修改多个样式。

1. 参数只写属性名，则是返回属性值

```
$(this).css("color");
```

2. 参数是属性名，属性值，逗号分隔，是设置一组样式，属性必须加引号，值如果是数字可以不用跟单位和引号

```
$(this).css("color", "red");
```

3. 参数可以是对象形式，方便设置多组样式。属性名和属性值用冒号隔开， 属性可以不用加引号，

```
$(this).css({ "color":"white","font-size":"20px"});
```

```
// 3.参数可以是对象形式，方便设置多组样式。
// 属性名和属性值用冒号隔开 属性名可以不用加引号，
$('div').css({
    background: 'green',
    // 'font-size': 40
    // 复合属性采用驼峰命名法，属性值若不为数字则需要加引号
    fontSize: 40
```

2、设置类样式方法

作用等同于以前的 classList，可以操作类样式， 注意操作类里面的参数不要加点。

1. 添加类

```
$( “div” ).addClass("current");
```

2. 移除类

```
$( “div” ).removeClass(''current'');
```

3. 切换类

```
$( “div” ).toggleClass(''current'');
```

3、类操作与className区别

> 原生 JS 中 className 会覆盖元素原先里面的类名。
>
> jQuery 里面类操作只是对指定类进行操作，不影响原先的类名。

三、jQuery效果

jQuery 给我们封装了很多动画效果，最为常见的如下：

1、显示隐藏效果

1. 显示语法规范

```
show([speed,[easing],[fn]])
```

显示参数

> - 参数都可以省略， 无动画直接显示。
> - **speed：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)。**
> - easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
> - fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

2. 隐藏语法规范

```
hide([speed,[easing],[fn]])
```

隐藏参数

> - 参数都可以省略， 无动画直接显示。
> - speed：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)。
> - easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。

- fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

3. 切换语法规范

```
toggle([speed,[easing],[fn]])
```

切换参数

- 参数都可以省略， 无动画直接显示。
- speed：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)。
- easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
- fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。
- 建议：平时一般不带参数，直接显示隐藏即可。

2、滑动效果

1. 下滑效果语法规范

```
slideDown([speed,[easing],[fn]])
```

下滑效果参数

- 参数都可以省略。
- speed: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)。
- easing:(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
- fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

2. 上滑效果语法规范

```
slideUp([speed,[easing],[fn]])
```

上滑效果参数

- 参数都可以省略。
- speed：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)。
- easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
- fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

3. 滑动切换效果语法规范

```
slideToggle([speed,[easing],[fn]])
```

滑动切换效果参数

- 参数都可以省略。
- speed：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)。
- easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
- fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

3、事件切换

```
hover([over,]out)
```

1. over：鼠标移到元素上要触发的函数（相当于mouseenter）
2. out：鼠标移出元素要触发的函数（相当于mouseleave）
3. 如果只写一个函数，则鼠标经过和离开都会触发它

4、动画队列及其停止排队方法

1. 动画或效果队列

动画或者效果一旦触发就会执行，如果多次触发，就造成多个动画或者效果排队执行。

2. 停止排队

```
stop()
```

1. stop() 方法用于停止动画或效果。
2. **注意： stop() 写到动画或者效果的前面， 相当于停止结束上一次的动画。**

5、淡入淡出效果

1. 淡入效果语法规范

```
fadeIn([speed,[easing],[fn]])
```

淡入效果参数

> - 参数都可以省略。
> - speed：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)。
> - easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
> - fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

2. 淡出效果语法规范

```
fadeOut([speed,[easing],[fn]])
```

淡出效果参数

> - 参数都可以省略。
> - speed：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)。
> - easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
> - fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

3. 淡入淡出切换效果语法规范

```
fadeToggle([speed,[easing],[fn]])
```

淡入淡出切换效果参数

> - 参数都可以省略。
> - speed：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)。

- easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
- fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

4. 渐进方式调整到指定的不透明度

```
fadeTo([[speed],opacity,[easing],[fn]])
```

效果参数

- opacity 透明度必须写，取值 0~1 之间。
- speed：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)。必须写
- easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
- fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

6、自定义动画 animate

1、语法

```
animate(params,[speed],[easing],[fn])
```

2、参数

1. params：想要更改的样式属性，以对象形式传递，必须写。 属性名可以不用带引号， 如果是复合属性则需要采取驼峰命名法 borderLeft。其余参数都可以省略。
2. speed：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. easing：(Optional) 用来指定切换效果，默认是“swing”，可用参数“linear”。
4. fn: 回调函数，在动画完成时执行的函数，每个元素执行一次。

```
<script>
    $(function () {
        $("button").click(function () {
            $("div").animate({
                left: 500,
                top: 300,
                opacity: .4,
                width: 500
            }, 1000);
        })
    })
</script>
```

四、jQuery属性操作

1、设置或获取元素固有属性值 prop()

所谓元素固有属性就是元素本身自带的属性，比如 元素里面的 href ， 比如 元素里面的 type。

1. 获取属性语法

```
prop("属性")
```

2. 设置属性语法

```
prop("属性", "属性值")
```

2、设置或获取元素自定义属性值 attr()

用户自己给元素添加的属性，我们称为自定义属性。 比如给 div 添加 index =“1”。

1. 获取属性语法

```
attr("属性")    // 类似原生 getAttribute()
```

2. 设置属性语法

```
attr("属性", "属性值")  // 类似原生 setAttribute()
```

该方法也可以获取 H5 自定义属性

3、数据缓存 data()

data() 方法可以在指定的元素上存取数据，并不会修改 DOM 元素结构。一旦页面刷新，之前存放的数据都将被移除。

1. 附加数据语法

```
data("name","value")  // 向被选元素附加数据
```

2. 获取数据语法

```
date("name")          //  向被选元素获取数据
```

同时，还可以读取 HTML5 自定义属性 data-index ， 得到的是数字型

```
<script>
    $(function () {
        // 1、设置或获取元素固有属性值 prop()
        // (1)获取属性值：prop('属性')
        alert($("a").prop("href"));
        // (2)设置属性值：prop('属性','属性值')
        $("a").prop("href", "http://www.baidu.cn");
        $("input").change(function () {
            alert($(this).prop("checked"));
        })
        // 2、设置或获取元素自定义属性值 attr()
        // (1)attr(''属性'')      // 类似原生 getAttribute()
        console.log($("div").attr("index"));
        // attr(''属性'', ''属性值'')   // 类似原生 setAttribute()
        $("div").attr("index", 0);
        console.log($("div").attr("index"));
        // 3.H5自定义属性：data-
        console.log($("div").attr("data-index"));
        // 4.数据缓存 data()方法可以在指定的元素上存取数据，并不会修改 DOM 元素结构
        // data(''name'',''value'')   // 向被选元素附加数据
        // date(''name'')             //   向被选元素获取数据
        // 可以读取 HTML5 自定义属性  data-index，得到的是数字型
        $("span").data("uname", "rose");
        console.log($("span").data("uname"));
        // 获取data-index自定义属性  不用加data-前缀
        console.log($("div").data("index"));


    })
</script>
body>
```

五、jQuery内容文本值

主要针对元素的内容还有表单的值操作。

1. 普通元素内容 html()（相当于原生inner HTML)

```
html()          // 获取元素的内容
```

```
html("内容")  // 设置元素的内容
```

2. 普通元素文本内容 text() (相当与原生 innerText)

```
text()              // 获取元素的文本内容
```

```
text("文本内容")  // 设置元素的文本内容
```

3. 表单的值 val()（相当于原生value)

```
val()          // 获取表单的值
```

```
val("内容")  // 设置表单的值
```

```
<script>
    $(function () {
        // 1.获取设置元素内容 html()
        console.log($("div").html());
        // $("div").html("123");

        // 2.获取设置元素文本内容 text()
        console.log($("div").text());
        $("div").text("123");

        // 3.获取设置表单值 val()
        console.log('input的值为: ',$("input").val());
        // $("input").val("123");
```

```
<script>
    // parents('选择器') 可以返回指定祖先元素
    console.log($(".four").parent().parent().parent());
    console.log($(".four").parents());
    console.log($(".four").parents(".one"));
</script>
```

六、jQuery元素操作

主要是遍历、创建、添加、删除元素操作。

1、遍历元素

jQuery 隐式迭代是对同一类元素做了同样的操作。 如果想要给同一类元素做不同操作，就需要用到遍历。

语法1

```
$("div").each(function (index, domEle) { xxx; })
```

1. each() 方法遍历匹配的每一个元素。主要用DOM处理。 each 每一个
2. 里面的回调函数有2个参数： index 是每个元素的索引号; **demEle 是每个DOM元素对象，不是jquery对象**
3. **所以要想使用jquery方法，需要给这个dom元素转换为jquery对象 $(domEle)**

```
var arr = ['red', 'green', 'blue'];
var sum = 0;
$("div").each(function (index, domEle) {
    // 回调函数第一个参数一定是索引号  可以自己指定索引号号名称
    console.log(index);
    // console.log(i);
    // 回调函数第二个参数一定是 dom元素对象 也是自己命名
    console.log(domEle);
    // $(domEle).css("color", "blue");
    $(domEle).css("color", arr[index]);
    // 字符转换为数字型
    // sum += parseInt($(domEle).text());
    sum += Number($(domEle).text());
})
console.log(sum);
```

语法2

```
$.each(object, function (index, element) { xxx; })
```

1. $.each()方法可用于遍历任何对象。主要用于数据处理，比如数组，对象
2. 里面的函数有2个参数： index 是每个元素的索引号; element 遍历内容

```
$.each($("div"), function (i, ele) {
    console.log(i);
    console.log(ele);
})

$.each(arr, function (i, ele) {
    console.log(i);
    console.log(ele);
})

$.each({
    name: "rose",
    age: 21
}, function (i, ele) {
    console.log(i);
    console.log(ele);
})
```

2、创建元素

语法

```
$("<li></li>");
```

动态的创建了一个 `<li>`

3、添加元素

1. 内部添加

```
element.append("内容")
```

把内容放入匹配元素内部最后面，类似原生 appendChild。

```
element.prepend("内容")
```

把内容放入匹配元素内部最前面。

2. 外部添加

```
element.after("内容")      // 把内容放入目标元素后面
```

```
element.before("内容")    // 把内容放入目标元素前面
```

> 内部添加元素，生成之后，它们是父子关系。

> 外部添加元素，生成之后，他们是兄弟关系。

4、删除元素

```
element.remove()  // 删除匹配的元素（本身）
```

```
element.empty()  // 删除匹配的元素集合中所有的子节点
```

```
element.html("")  // 清空匹配的元素内容
```

> remove 删除元素本身。
>
> empt() 和 html("") 作用等价，都可以删除元素里面的内容，只不过 html 还可以设置内容。

```
$(function () {
    // 1.创建元素
    // $("<li></li>");  创建了一个li
    var li = $("<li>我是后来创建的li</li>");

    // 2.添加元素
    // 内部添加元素，生成之后，它们是父子关系。
    // 外部添加元素，生成之后，他们是兄弟关系。
    // (1)内部添加
    // element.append("内容")
    // 把内容放入匹配元素内部最后面，类似原生 appendChild
    $("ul").append(li);
    // element.prepend("内容")
    // 把内容放入匹配元素内部最前面。
    $("ul").prepend(li);

    // (2)外部添加
    // element.after("内容")        //  把内容放入目标元素后面
    // element.before("内容")    //  把内容放入目标元素前面
    var div = $("<div>我是后来的div</div>");
    $(".test").after(div);
    $(".test").before(div);

    // 3.删除元素
    // remove 删除元素本身。
    // empt() 和  html('''') 作用等价，都可以删除元素里面的内容，只不
    // (1)element.remove()   //  删除匹配的元素（本身）
    $("ul").remove();
    // (2)element.empty()    //  删除匹配的元素集合中所有的子节点
    $("ul").empty();
    // (3)element.html("")   //  清空匹配的元素内容
    $("ul").html("");
```

```
        $( "ul" ).html( "" );


    })
</script>
```

七、jQuery尺寸、位置操作

1、jQuery尺寸

语法	用法
width() / height()	取得匹配元素宽度和高度值 只算 width / height
innerWidth() / innerHieght()	取得匹配元素宽度和高度值 包含 padding
outerWidth() / outerHeight()	取得匹配元素宽度和高度值 包含 padding 、border
outerWidth(true) / outerHeight(true)	取得匹配元素宽度和高度值 包含 padding 、borde、margin

1. 以上参数为空，则是获取相应值，返回的是数字型。
2. 如果参数为数字，则是修改相应值。
3. 参数可以不必写单位。

```
    <script>
        $(function () {
            // 1. width() / height() 获取设置元素 width和height大小
            console.log($("div").width());
            // 设置元素宽度和高度
            // $("div").width(300);

            // 2. innerWidth() / innerHeight()  获取设置元素 width和height + padding 大小
            console.log($("div").innerWidth());

            // 3. outerWidth()  / outerHeight()  获取设置元素 width和height + padding + border 大小
            console.log($("div").outerWidth());

            // 4. outerWidth(true) / outerHeight(true) 获取设置 width和height + padding + border + margin
            console.log($("div").outerWidth(true));

        })
    </script>
</body>
```

2、jQuery位置

位置主要有三个： offset()、position()、scrollTop()/scrollLeft()

1. offset() 设置或获取元素偏移

1. offset() 方法设置或返回被选元素相对于文档的偏移坐标，跟父级没有关系。

2. 该方法有2个属性 left、top 。offset().top 用于获取距离文档顶部的距离，offset().left 用于获取距离文档左侧的距离。
3. 可以设置元素的偏移：offset({ top: 10, left: 30 });

2. position() 获取元素偏移

1. position() 方法用于返回被选元素相对于带有定位的父级偏移坐标，如果父级都没有定位，则以文档为准。
2. 该方法有2个属性 left、top。position().top 用于获取距离定位父级顶部的距离，position().left 用于获取距离定位父级左侧的距离。
3. 该方法只能获取。

3. scrollTop()/scrollLeft() 设置或获取元素被卷去的头部和左侧

1. scrollTop() 方法设置或返回被选元素被卷去的头部。
2. 不跟参数是获取，参数为不带单位的数字则是设置被卷去的头部。

jQuery事件

一、jQuery事件注册

1、单个事件注册

语法：

```
element.事件(function(){})
```

```
$("div").click(function(){  事件处理程序 })
```

其他事件和原生基本一致。

```
$(function () {
    // 1.单个事件注册
    $("div").click(function () {
        $(this).css("background", "purple");
    });

    $("div").mouseenter(function () {
        $(this).css("background", "skyblue");
    });
```

比如mouseover、mouseout、blur、focus、change、keydown、keyup、resize、scroll 等

二、jQuery事件处理

1、事件处理 on() 绑定事件

on() 方法在匹配元素上绑定一个或多个事件的事件处理函数

语法：

```
element.on(events,[selector],fn)
```

1. events:一个或多个用空格分隔的事件类型，如"click"或"keydown" 。
2. selector: 元素的子元素选择器 。
3. fn:回调函数 即绑定在元素身上的侦听函数。

on()优势

1、可以绑定多个事件，多个处理事件处理程序。

```
$("div").on({
 mouseover: function(){},
 mouseout: function(){},
 click: function(){}
});
```

```
$("div").on("mouseenter mouseleave", function () {
    $(this).toggleClass("current");
})
```

如果事件处理程序相同

```
$("div").on("mouseover mouseout", function() {
  $(this).toggleClass("current");
 });
```

2、可以事件委派操作 。**事件委派的定义就是，把原来加给子元素身上的事件绑定在父元素身上，就是把事件委派给父元素。**

```
$('ul').on('click', 'li', function() {
    alert('hello world!');
});
```

在此之前有bind(), live() delegate()等方法来处理事件绑定或者事件委派，最新版本的请用on替代他们。

动态创建的元素用on()绑定事件

3、**动态创建的元素，click() 没有办法绑定事件， on() 可以给动态生成的元素绑定事件**！！！

```
$("div").on("click","p", function(){
    alert("俺可以给动态生成的元素绑定事件")
});
```

```
$("div").append($("<p>我是动态创建的p</p>"));
```

```
<script>
    $(function () {
        // (2)on可以事件事件委托（委派）
        // $("ul li").click();
        $("ul").on("click", "li", function () {
            alert("hello world!");
        });
        // 事件是绑定在ul上面的 但触发的对象是li

        // (3)on可以给动态创建的元素绑定事件
        // $("ol li").click(function () {
        //     alert("hello world!");
        // })
        $("ol").on("click", "li", function () {
            alert("hello world!");
        })
        var li = $("<li>王一博</li>");
        $("ol").append(li);
    })
</script>
```

2、事件处理 off() 解绑事件

off() 方法可以移除通过 on() 方法添加的事件处理程序。

```
$("p").off() // 解绑p元素所有事件处理程序
$("p").off( "click")  // 解绑p元素上面的点击事件 后面的 foo 是侦听函数名
$("ul").off("click", "li");   // 解绑事件委托
```

如果有的事件只想触发一次， 可以使用 one() 来绑定事件。

```
<script>
    $(function () {
        $("div").on({
            click: function () {
                alert("我点击了");
            },
            mouseover: function () {
                alert("我经过了");
            }
        });
        // 委托事件
        $("ul").on("click", "li", function () {
            console.log("hi@you");
        })
        // 1.事件解绑 off
        // 解除div上的所有事件
        $("div").off();

        // 解除div上的click事件
        $("div").off("click");

        // 解除委托事件
        $("ul").off("click", "li");
```

3、自动触发事件 trigger()

有些事件希望自动触发, 比如轮播图自动播放功能跟点击右侧按钮一致。可以利用定时器自动触发右侧按钮点击事件，不必鼠标点击触发。

```
element.click()  // 第一种简写形式
```

```
element.trigger("type") // 第二种自动触发模式
```

```
$("p").on("click", function () {
  alert("hi~");
});

$("p").trigger("click"); // 此时自动触发点击事件，不需要鼠标点击
```

```
element.triggerHandler(type)  // 第三种自动触发模式
```

triggerHandler模式不会触发元素的默认行为，这是和前面两种的区别。

4、one()事件

```
$(function () {
    // 有的事件只想触发一次， 可以使用 one() 来绑定事件
    $("div").one("click", function () {
        alert("hi@you!");
    })
})
```

三、jQuery事件对象

事件被触发，就会有事件对象的产生。

```
element.on(events,[selector],function(event) {})
```

> 阻止默认行为：event.preventDefault() 或者 return false
>
> 阻止冒泡： event.stopPropagation()

```
<script>
    $(function () {
        $(document).on("click", function () {
            console.log("点击了document! ");
        })
        $("div").on("click", function (event) {
            // 阻止默认行为：event.preventDefault()   或者 return false
            // 阻止冒泡：  event.stopPropagation()
            console.log("点击了div!");
            event.stopPropagation();
        })
    })
</script>
```

jQuery其他方法

一、jQuery对象拷贝

如果想要把某个对象拷贝（合并） 给另外一个对象使用，此时可以使用 **$.extend() 方法**

语法：

```
$.extend([deep], target, object1, [objectN])
```

1. deep：如果设为true 为深拷贝， 默认为false 浅拷贝
2. target：要拷贝的目标对象
3. object1：待拷贝到第一个对象的对象。
4. objectN：待拷贝到第N个对象的对象。
5. 浅拷贝是把被拷贝的对象复杂数据类型中的地址拷贝给目标对象，修改目标对象会影响被拷贝对象。
6. 深拷贝，前面加true， 完全克隆(拷贝的对象,而不是地址)，修改目标对象不会影响被拷贝对象。

二、多库共存

jQuery使用$作为标示符

随着jQuery的流行,其他 js 库也会用这$作为标识符， 这样一起使用会引起冲突。

需要一个解决方案，让jQuery 和其他的js库不存在冲突，可以同时存在，这就叫做多库共存。

jQuery解决方案

1. 把里面的 $ 符号 统一改为 jQuery。 比如 jQuery("div")
2. jQuery 变量规定新的名称：

```
$.noConflict()
var xx = $.noConflict();
```

三、jQuery插件

jQuery 功能比较有限，想要更复杂的特效效果，可以借助于 jQuery 插件完成。

注意：这些插件也是依赖于jQuery来完成的，所以必须要先引入jQuery文件，因此也称为 jQuery 插件。

jQuery 插件常用的网站：

1. jQuery 插件库 http://www.jq22.com/
2. jQuery 之家 http://www.htmleaf.com/

jQuery 插件使用步骤：

1. 引入相关文件。 (jQuery 文件 和 插件文件)
2. 复制相关html、css、js (调用插件)。

jQuery 插件演示：

1. 瀑布流

2. 图片懒加载（图片使用延迟加载在可提高网页下载速度。它也能帮助减轻服务器负载）

   当我们页面滑动到可视区域，再显示图片。

   我们使用jquery 插件库 EasyLazyload。 注意，此时的js引入文件和js调用必须写到 DOM元素（图片）最后面

3. 全屏滚动（fullpage.js） gitHub： https://github.com/alvarotrigo/fullPage.js 中文翻译网站： http://www.dowebok.com/demo/2014/77/

boostrap

bootstrap JS 插件：

bootstrap 框架也是依赖于 jQuery 开发的，因此里面的 js插件使用 ，也必须引入jQuery 文件。