%HTML5 基础进阶课程
% HTML 基础知识
% jQuery 基础知识

教学目标

- 让你掌握jQuery中选择器、DOM操作、css设置以及简单的事件响应
- 让你学会查询学习jQuery手册

认识jQuery

jQuery 封装了 JavaScript 常用的功能代码, 提供一种简洁, 快捷的 JavaScript 设计模式, 优化了 HTML 文档操作, 事件处理, 动画设计和 Ajax 交互. 可以说 jQuery 改变了用户编写 JavaScript 代码的方式. jQuery 设计的宗旨是 "Write Less, Do More" , 写更少的代码, 做更多的事情.

jQuery 项目主要包括 jQuery Core (核心库), jQuery UI (界面库), Sizzle (CSS选择器) 和 Qunit (测试套件)四部分, 现在又增加了一个新的部分就是 jQuery Mobile (手机端).

John Resig 是团队的主要核心人物之一, jQuery 是 John Resig 于2006年1月开发的一个开源项目.

- jQuery官方网站 (http://jquery.com)
- jQuery项目组官方网站 (https://jquery.org/)
- John Resig个人网站: (http://ejohn.org/)

jQuery能做什么

jQuery 库为 Web 脚本编程提供了通用的抽象层,使得它几乎适用于任何脚本编程的情形。由于它容易扩展而且不断有新插件面世增强它的功能,所以一本书根本无法涵盖它所有可能的用途和功能。抛开这些不谈,仅就其核心特性而言, jQuery 能够满足下列需求。

1. 取得文档中的元素

如果不使用 JavaScript 库,遍历 DOM (Document Object Model ,文档对象模型)树,以及查找HTML文档结构中某个特殊的部分,必须编写很多行代码。 jQuery 为准确地获取需要检查或操纵的文档元素,提供了可靠而富有效率的选择符机制。

```
$('div.content').find('p');
```

2. 修改页面的外观

CSS 虽然为影响文档呈现的方式提供了一种强大的手段,但当所有浏览器不完全支持相同的标准时,单纯使用 CSS 就会显得力不从心。 jQuery 可以弥补这一不足, 它提供了跨浏览器的标准解决方案。而且,即使在页面已经呈现之后, jQuery 仍然能够改 变文档中某个部分的类或者个别的样式属性。

通过引入 jQuery , 程序开发人员可以很便捷地控制页面的 CSS 文件. 浏览器对页面文件的兼容性, 一直以来都是页面开发者最为头痛的事, 而使用 jQuery 操作页面的样式却可以很好地兼容各种浏览器.

```
$('ul > li:first').addClass('active');
```

3. 对页面事件的处理

jQuery 能够影响的范围并不局限于简单的外观变化,使用少量的代码, jQuery 就能改变文档的内容。可以改变文本、插入或翻转图像、列表重新排序,甚至对 HTML 文档的整个结构都能重写和扩充——所有这些只需一个简单易用的API。

```
$('#container').append('<a href="more.html">more</a>');
```

引入 jQuery 库后, 可以使页面的表现层与功能开发分离, 开发者更多地专注于程序的逻辑与功效; 页面设计者侧重于页面的优化与用户体验. 然后, 通过事件绑定机制, 可能很轻松地实现二者的结合.

即使是最强大和最精心设计的行为,如果我们无法控制它何时发生,那它也毫无用处。 jQuery 提供了截获形形色色的页面事件(比如用户单击某个链接)的适当方式,而不需要使用事件处理程序拆散 HTML 代码。此外,它的事件处理 API 也消除了经常困扰Web开发人员浏览器的不一致性。

```
$('button.show-details').click(function() {
  $('div.details').show();
});
```

4. 大量插件在页面中的运用

在引入 jQuery 库后, 还可以使用大量的插件来完善页面的功能和效果, 如表单插件, UI插件, 这些插件的使用极大的丰富了页面的展示效果, 使原来 JavaScript 代码遥不可及的功能通过插件的引入而软件轻松的实现.

5. 与Ajax技术的完美结合

Ajax 的异步读取服务器数据的方法, 极大地方便了程序的开发, 加深了用户的页面体验度: 而引入 jQuery 库后, 不仅完善了三原有的功能, 而且减少了代码的书写, 通过其内部对象或函数, 加上几行代码就可以实现复杂的功能.

无需刷新页面从服务器获取信息。这种编程模式就是众所周知的 Ajax (Asynchronous JavaScript and XML,异步JavaScript和XML),它是一系列在客户端和服务器之间传输数据的强大技术。 jQuery 通过消除这一过程中的浏览器特定的复杂性,使开发人员得以专注于服务器端的功能设计,从而得以创建出反应灵敏、功能丰富的网站。

```
$('div.details').load('more.html #content');
```

6. 为页面添加动态效果

为了实现某种交互式行为,设计者也必须向用户提供视觉上的反馈。 jQuery 中内置的一批淡入、擦除之类的效果,以及制作新效果的工具包,为此提供了便利。

```
$('div.details').slideDown();
```

7. 简化常见的JavaScript任务

除了这些完全针对文档的特性之外, jQuery 也改进了对基本 的 JavaScript 数据结构的操作(例如迭代和数组操作等)。

```
$.each(obj, function(key, value) {
  total += value;
});
```

引入jQuery文件库

从现在开始, 我们将所有的实例都包含以下三部分: HTML 文档, CSS 模式文件, JavaScript 文档和我们需要引入 jQuery 文件.

```
git:(master) ✗ tree
.
├── 1-1.html
├── 1-2.html
├── 1-3.html
├── 1-4.html
├── css
│   └── mycss.css
└── js
    ├── jquery.js
    └── myjs.js
```

下载完 jQuery 框架文件后, 并不需要任何安装, 仅需要使用 <script> 文件导入标记.如下:

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

注意: 我们以后 jQuery 就直接使用 jquery.js 文件

怎样使用jQuery?

我们先还是显示一下弹窗小窗口.

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <script src="./js/jquery.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
        alert("你好, 欢迎来到jQuery世界!");
    })
    </script>

    <title>jQuery Hello world!</title>
</head>
<body>
</body>
</html>
```

使用CDN

什么是CDN? CDN是一种内容分发网络, 当用户请求其数据时, CDN能智能的分配离用户最近的服务器提供服务. CDN优点1, 用户体验好, 加载jQuery速度快. 第二个优点是, 如果使用CDN节省了宝贵的加载jQuery库所需要的带宽.

CDN列表:

- 官方CDN (https://code.jquery.com/)

```
<script   src="https://code.jquery.com/jquery-3.1.1.min.js"
integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
crossorigin="anonymous"></script>
```

- 国内免费CDN (http://www.bootcdn.cn/jquery/)

```
<script src="//cdn.bootcss.com/jquery/3.1.1/jquery.min.js"></script>
```

- 七牛空间CDN (http://www.staticfile.org/)
- 新浪CDN (http://lib.sinaapp.com/)

选择器

根据所获取页面中元素的不同, 可能将jQuery选择器分为: 基本选择器, 层次选择器, 过滤选择器, 表单选择器四大类. 其中, 在过滤选择器中又可分为: 简单过滤选择器, 内容选择器, 可见性过滤选择器, 属性过滤性选择器, 子元素过滤选择器, 表单对象属性过滤选择器.

基本选择器

基本选择器是jQuery中使用最频繁的选择器, 它由元素Id, Class, 元素名, 多个选择符组成, 通过基本选择器可以实现大多数页面元素的查找, 其详细说明如表:

选择器	功能	返回值
#id	根据给定ID匹配一个元素	单个元素
element	根据给定的元素名匹配所有元素	元素集合
.class	根据给定的类匹配元素	元素集合
*	匹配所有元素	元素集合
selector1, selectorN	每一个选择器匹配到的元素合并后一起返回	元素集合

层次选择器

层次选择器通过DOM元素间的层次关系获取元素, 其主要的层次关系包括后代, 父子, 相邻, 兄弟关系, 通过其中某类关系可以方便快捷地定位元素, 其详细说明如下:

[illegible]	CSS	iQuery	[illegible]	

选择器	模式	模式	描述	
后代选择器	ul li a	\$(‘ul li a’)	获取追溯到的多个 DOM 对象	根据祖先元素
子选择器	div > p	\$(‘div p’)	只获取子类节点的多个 DOM 对象	根据父元素匹
next 选择器（相连）	div + p	\$(‘div + p’)	只获取某节点后一个同级DOM对象	匹配所有紧接
nextAll 选择器 (之后所有)	div ~ p	\$(‘div ~ p’)	获取某节点后面所有同级DOM对象	匹配prev元素

因为它属于层次选择器，在高级选择器中，jQuery为这样的选择器都提供了一个相对应的方法。

后代选择

- jQuery为后代选择器提供了一个等价的find( ) 方法：

```
$('#box p').css('color', 'red'); //后代选择器
$('#box').find('p').css('color','red');//和后代选择器等价
```

子选择

- jQuery为子选择器提供了一个等价的children()方法：

```
$('#box > p').css('color','red');//子选择器
$('#box').children('p').css('color','red');//和子选择器等价
```

相邻选择

- jQuery为next选择器提供了一个等价的next()方法：

```
$('#box+p').css('color','red');//下一个同级节点
$('#box').next('p').css('color','red');//和next选择器等价
```

兄弟选择

- jQuery为nextAll选择器提供了一个等价的方法nextAll()：

```
$('#box ~ p').css('color','red');//后面所有同级节点
$('#box').nextAll('p').css('color', 'red'); //和 nextAll 选择器等价
```

* 需要注意的是：

层次选择器对节点的层次都是有要求的，比如子选择器，有子节点才可以被选择到，孙子节点和重孙子节点都无法选择到。next和nextAll选择器，必须是同一个层次的后一个和后N个，不在同一个层次就无法选取到了。

在 find()、children()、next()和nextAll()和这四个方法中，如果不传递参数，就相当于传递了“”，选择所有符合条件的元素，*建议尽量保持参数的传递
jQuery独有补充方法。CSS未含有

```
$('#box').prev('p').css('color','red');//同级上一个元素
$('#box').prevAll('p').css('color','red');//同级上面所有的元素
$('#box').prevUntil('p').css('color','red');//同级上非指定元素选定，遇到则停止
$('#box').nextUntil('p').css('color','red');//同级下非指定元素选定，遇到则停止
$('#box').siblings('p').css('color','red');//siblings()方法正好集成了prevAll()和nextAll()两个功能的效果，及上下相邻的所有元素进行选定：
```

扩展：

```
$('p', '#box');//jQuery会自动把这条语句转成$('#box').find('p')，这会导致一定的性能损失。
$('p', $('#parent'));//jQuery内部会也将这条语句转成$('#box').find('p')
```

这里，推荐使用jQuery提供的方法。使用“+”或“~”从字面上没有next和nextAll更加语义化，更加清晰，jQuery的方法更加丰富，提供了相对的prev和prevAll。

并且有时需要能够灵活的拆分和组合选择器。所以，如果jQuery提供了独立的方法来代替某些选择器的功能，推荐优先使用独立的方法。

jQuery 支持的基础CSS选择器

选择器	描述
*	匹配所有元素
E	匹配标签名为 E 的所有元素
E F	匹配标签名为 F 的所有元素, 这些元素是 E 的子节点.(可以是孙子)

选择器	描述
E > F	匹配标签名为 F 的所有元素, 这些元素是 E 的直接子节点.(必须是儿子)
E + F	匹配标签名为 F 的所有元素, 这些元素是 E 的后面的第一个兄弟节点
E ~ F	匹配标签名为 F 的所有元素, 这些元素是 E 的后面的兄弟节点之一
E . C	匹配标签名为 E 的所有元素, 这些元素拥有 CSS 类名为 C , 如果省略 E 则相当于 *.C
E # I	匹配标签名为 E 的所有元素, 这些元素的 ID 特性为 I , 如果省略 E 则相当于 *#I
E[A]	匹配标签名为 E 的所有元素, 这些元素拥有特性 A
E[A=V]	匹配标签名为 E 的所有元素, 这些元素的 A 特性值为 V
E[A^=V]	匹配标签名为 E 的所有元素, 这些元素的 A 特性值以 V 开头
E[A$=V]	匹配标签名为 E 的所有元素, 这些元素的 A 特性值以 V 结束
E[A!=V]	匹配标签名为 E 的所有元素, 这些元素的 A 特性值不等于 V , 或者根本就不存在 A 特性.
E[A*=V]	匹配标签名为 E 的所有元素, 这些元素的 A 特性值包含 V

正则表达式匹配

从开头进行匹配

我们可以通过下面的选择器来匹配那些 href 特性值以 http:// 开头的链接：

```
a[href^='http://']
```

这会匹配 href 特性值以 http:// 开头的所有链接元素。 插入符（^）用来指定匹配字符串 开头的字符 。这也是大部分正则表达式处理器使用的字符， 表示匹配候选字符串开头的字符， 记住这个用法应该很容易。

从结尾进行匹配

我们可以通过下面的选择器来匹配那以.png文件类型：

```
a[href$='.png']
```

这会匹配 `href` 特性值以 `.pdf` 结尾的所有链接元素。 插入符（$）用来指定匹配字符串 结尾的字符 。

jQuery通过位置选择元素

选择器	描述
:first	匹配上下文中的第一个元素 li a:first 返回列表项后代节点中的第一个链接
:last	匹配上下文中的最后一个元素 li a:last 返回列表项后代节点为的最后一个链接
:first-child	匹配上下文中的第一个子节点 li:first-child 返回每个列表中的第一个列表项
:last-child	匹配上下文中的最后一个子节点 li:last-child 返回每个列表的最后一个列表项
:only-child	返回所有没有兄弟节点的元素
:nth-child(n)	匹配上下文中的第n个子节点. li:nth-child(2) 返回每个列表中的第二个列表项
:nth-child(even|odd)	匹配上下文中的偶数或奇数子节点. li:nth-child(even) 返回每个列表中的偶数列表项.
:nth-child(Xn+Y)	匹配上下文中的由提供的公式计算出的子节点.
:even	匹配上下文中的偶数元素, li:even返回所有偶数的列表项
:odd	匹配上下文中的奇数元素. li:odd返回所有奇数的列表项
:eq(n)	匹配第n个元素
:gt(n)	匹配第n个元素之后的元素(不包含第n个元素)
:lt(n)	匹配第n个元素之前的元素(不包含第n个元素)

下面就是准备练习使用的table样式.

```
<table id=lang>
            <thead>
                <tr>
                    <th>姓名</th>
                    <th>性别</th>
                    <th>年龄</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>张一</td>
                    <td>男</td>
                    <td>28</td>
                </tr>
                <tr>
                    <td>钱二</td>
                    <td>女</td>
                    <td>24</td>
                </tr>
                <tr>
                    <td>李三</td>
                    <td>男</td>
                    <td>23</td>
                </tr>
                <tr>
                    <td>赵四</td>
                    <td>男</td>
                    <td>23</td>
                </tr>
                </tbody>
        </table>
```

如果我们想要选择第一行标题行时, 可以使用如下:

```
$("thead > tr").addClass("myblue");
```

如果选择第一列时, 可以使用如下:

```
$("tr td:first-child").addClass("myred");
```

如果选择最后一列时, 可以使用如下:

```
$("tbody td:last-child").addClass("myblue");
```

如果选择中间这列时, 怎么办? 给大家几分钟思考.

```
$("tr td:nth-child(2)").addClass("mygreen");
```

自定义的过滤器

选择器	描述
:animated	选择处于动画状态的元素
:button	选择按钮元素(包括input[type=submit], input[type=reset], input[type=button]和button)
:checkbox	选择复选框元素(input[type=checkbox])
:contains(food)	选择包含文本 food 的元素
:disabled	选择处于禁用状态的元素
:enabled	选择处于启用状态的元素
:file	选择文件输入元素(input[type=file])
:has(selector)	选择至少包含一个匹配指定选择器的元素的元素
:header	选择标题元素, 比如 <h1> ~ <h6>
:hidden	选择隐藏元素
:image	选择图片输入元素(input[type=image])
:input	选择表单元素(input, select, textarea, button)
:not(selector)	选择不匹配指定选择器的元素
:parent	选择有子节点(包含文本)的元素, 空元素除外
:password	选择口令元素(input[type=password])
:radio	选择单选框元素(input[type=radio])
:reset	选择重置按钮元素(input[type=reset])或者(button[type=reset])
:selected	选择列表中处于选中状态的 元素
:submit	选择提交按钮 (input[type=submit])或者(button[type=submit])
:text	选择文本输入框元素 (input[type=text])
:visible	选择可见的元素

看一下面的列子, 让我们去练习一下.上面的所给出的选择器.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>使用jQuery的CSS选择器</title>
    <script language="javascript" type="text/javascript"
            src="js/jquery.js"></script>
    <style type="text/css">
           body{font-size:30px;text-align:center;}
           div{
               margin-left: 200px;
               margin-top: 100px;
           }
           table, th, td{
               border: 1px solid blue;
           }
           .myblue { color: blue;
               background-color: #ccc;
           }
           .myred {
               color: red;
               background-color: #ccc;
           }
           .mygreen{
               color: green;
               background-color: #ccc;
           }
    </style>
    <script type="text/javascript">
            $(function(){
                //$("div :button").addClass("mygreen");
                $("input[type=text]").addClass("myred");
                $(":disabled").addClass("myblue");
                $(":checkbox").addClass("mygreen");
                $(":header").addClass("mygreen");
                $(":password").addClass("mygreen");
            })
    </script>
</head>
<body>
    <div>
        <button name="sure">确定</button>
        <button name="cancel" disabled="true" >取消</button>
    </div>

    <div>
        <input type="text" name="input" value="Type here">
    </div>
    <div>
        喜欢H5: <input type="checkbox" value="1" >
        <h1>北京菜鸟在线</h1>
        <h2>H5全栈工程师</h2>
    </div>
    <div class="div">
```

```
        password: <input type="password">
    </div>
</body>
</html>
```

操作DOM

通过我们前面的学习, 感觉到可以能方便的选择元素, 而利用这个我们也很容易操作HTML文档本身的结构, 也是就是操作DOM, 这是jQuery 最强大的一个功能. 我们将会学习到改变页面的结构的种种方法.

创建一个新元素

我们先来创建一个新的元素. 有以下几种方法.

使用$()来创建一个元素

我们可以使用$()来创建一个的元素, 并将它追加到相应的元素中去.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('<p>').appendTo('.new');
      $('span').text(" is full stack develop engineer.")
    });
    </script>
</head>
<body>
  <div class="name">Richard.Wang</div>
  <p class="new"></p>
</body>
</html>
```

src/demo-3.html

使用DOM API创建新元素

我们可以使用DOM API直接生成HTMLElement对象. 其实jQuery在幕后悄悄地调用DOM API.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('.name').append(document.createElement("span"));
      $('span').text(" is full stack develop engineer.")
    });
    </script>
</head>
<body>
  <div class="name">Richard.Wang</div>
  <p class="new"></p>
</body>
</html>
```

src/demo-1.html

我们使用document.createElement(“span”)创建一个span元素, 然后并将这个元素插入到class等于name的div元素中.

用clone创建新的元素

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $(".name").clone().appendTo('.new');
    });
    </script>
</head>
<body>
  <div class="name">Richard.Wang</div>
  <p class="new"></p>
</body>
</html>
```

src/demo-2.html

添加子元素或后代元素

创建完元素后, 就需要将它们添加到文档中, 我们先从把一个元素插入到另一个元素内. 生成该元素的子元素或后代元素开始.

append()

.append()函数将特定内容插入到每个匹配元素里面的最后面，作为它的最后一个子元素（last child）.插入所有选中的元素中.

1. 可以将HTML元素直接插入到div元素中.

```
$('div').append('<p>教育科技有限公司.</p>');
```

1. 可以选择一个元素然后插入在另一个元素里面.

```
$('div').append($('p'));
```

1. 可以回调函数动态插入元素

```
$('div').append(function(index, html){
  return ($('p'));
});
```

根据上面的例子, 现在看完整例子.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $("#new").append('<span>教育科技有限公司</span>');
      $("#new").append($('.new'));
      $("#new").append(function(index, html){
        console.log(index);
        console.log(html);
        return ($('.name'));
      })
    });
    </script>
</head>
<body>
  <div class="name">Richard.Wang</div>
  <p class="new">HTML5全栈开发</p>
  <div id="new">北京菜鸟在线</div>
</body>
</html>
```

src/append.html

- index: 是元素的索引值.
- html: 是元素的内容.

练习: 验证当选中多个元素时, 是否会分别都会插入相应的元素.

prepend()

prepend()方法会插入新元素到jQuery所有内含元素之内, 使之成为相应元素的第一个子元素, 也就是说插到最前面. 而append是追加, 将元插到最后面.

1. 可以将HTML元素直接插入到div元素中.

```
$('div').prepend('<p>教育科技有限公司.</p>');
```

1. 可以选择一个元素然后插入在另一个元素里面.

```
$('div').prepend($('p'));
```

1. 可以回调函数动态插入元素

```
$('div').prepend(function(index, html){
  return ($('p'));
});
```

好, 我们现在试着完成一个整的例子.

- 根据append的例子来实现一下prepend的例子.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $("#new").prepend('<span>教育科技有限公司</span>');
      $(".name").prepend($('#new'));
      $(".name").prepend(function(index, html){
        console.log(index);
        console.log(html);
        return ($('.new'));
      });
    });
    </script>
</head>
<body>
  <div class="name">Richard.Wang</div>
  <p class="new">HTML5全栈开发</p>
  <div id="new">北京菜鸟在线</div>
</body>
</html>
```

src/prepend.html

appendTo() 和 prependTO()

- appendTo()

.append()和.appendTo()两种方法功能相同，主要的不同是语法——内容和目标的位置不同。对于.append(), 选择表达式在函数的前面，参数是将要插入的内容。对于.appendTo()刚好相反，内容在方法前面，无论是一个选择器表达式 或创建作为标记上的标记，它都将被插入到目标容器的末尾。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('<span>教育科技有限公司</span>').appendTo('#new');
      $("#new").appendTo($('.new'));
    });
    </script>
</head>
<body>
  <div class="name">Richard.Wang</div>
  <p class="new">HTML5全栈开发</p>
  <div id="new">北京菜鸟在线</div>
</body>
</html>
```

src/appendto.html

- prependTo()

.prepend()和.prependTo()实现同样的功能，主要的不同是语法，插入的内容和目标的位置不同。 对于 .prepend() 而言，选择器表达式写在方法的前面，作为待插入内容的容器，将要被插入的内容作为方法的参数。而 .prependTo() 正好相反，将要被插入的内容写在方法的前面，可以是选择器表达式或动态创建的标记，待插入内容的容器作为参数。

- 根据上面的例子, 自己实现这个方法的使用.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('<span>教育科技有限公司</span>').prependTo('#new');
      $("#new").prependTo($('.new'));
    });
    </script>
</head>
<body>
  <div class="name">Richard.Wang</div>
  <p class="new">HTML5全栈开发</p>
  <div id="new">北京菜鸟在线</div>
</body>
</html>
```

src/prependto.html

封装

jQuery还提供了一些把新元素作为现有元素的父元素或者祖先元素插入到页面的方法. 把这些方法称为封装方法.

wrap()

.wrap()函数可以接受任何字符串或对象，可以传递给$()工厂函数来指定一个DOM结构。这种结构可以嵌套了好几层深，但应该只包含一个核心的元素。每个匹配的元素都会被这种结构包裹。该方法返回原始的元素集，以便之后使用链式方法。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      var newElem = $("<div></div>").css('border', '1px solid red');
      $('#new').wrap(newElem);
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="name">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/wrap.html

wrapAll()

.wrapAll()函数可以接受任何字符串或对象，可以传递给$()工厂函数来指定一个DOM结构。这种结构可以嵌套多层，但是最内层只能有一个元素。所有匹配元素将会被当作是一个整体，在这个整体的外部用指定的 HTML 结构进行包裹。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      var newElem = $("<div></div>").css('border', '1px solid red');
      $('.new').wrapAll(newElem);
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/wrapall.html

- 思考: wrap()和wrapAll()的区别是什么?
- 当我们使用wrap和wrapall时, 调换顺序就会有新的样式的改变.

wrapInner()

.wrapInner()函数可以接受任何字符串或对象，可以传递给$()工厂函数来指定一个DOM结构。这种结构可以嵌套多层，但是最内层只能有一个元素。每个匹配元素的内容都会被这种结构包裹。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    .new { border: 2px solid blue;};
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      var newElem = $('<div class="inner"></div>').css('border', '1px solid
red');
      $('.new').wrapInner(newElem);
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/wrapinner.html

- 练习: 如果在class=new后增加新的文本显示.
- 注意: warp()和wrapInner()都可以实现回调函数.
- 练习: 实现这两个函数的回调函数的使用.

```
$('div').warp(function(index){
  return ();
});
```

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      var newElem1 = $('<div class="inner"></div>').css('border', '1px solid
red');
      var newElem2 = $('<div class="inner"></div>').css('border', '1px solid
blue');
      $('.new').wrap(function(index){
        console.log(index);
        if(index / 2){
          return newElem1;
        }else {
          return newElem2;
        }
      });
      $('.new').wrapInner(function(index){
        if(index / 2){
          return newElem1;
        }else {
          return newElem2;
        }
      })
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/wrapinner-fun.html

插入兄弟元素

jQuery也提供了插入兄弟元素到页面其他元素旁的方法.

- after()

.after() 和.insertAfter()实现同样的功能。主要的不同是语法——内容和目标的位置不同。对于.after(),要插入的内容来自方法的参数：$(target).after(contentToBeInserted)。 对于.insertAfter()， 刚好相反，内容在方法前面并且插入到目标的前面， 而目标是传递给.insertAfter()方法的参数： $(contentToBeInserted).insertAfter(target)。

例子:

请看下面的HTML:

```
<div class="container">
  <h2>HTML5</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

我们可以创建内容然后同时插在好几个元素后面:

```
$('.inner').after('<p>Test</p>');
```

每个内部的

元素得到新的内容:

```
<div class="container">
  <h2>HTML5</h2>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
  <p>Test</p>
</div>
```

- 练习, 根据上面的例子, 请写出完整示例代码.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('.new').after('<span>开源世界!!!</span>');
      $('#new').after(function(index, html){
        return ('<h3>'+ index +'</h3>');
      })
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/after.html

- insertAfte()

The .after()和.insertAfter() 实现同样的功能。主要的不同是语法——特别是（插入）内容和目标的位置。 对于 .after(), 选择表达式在函数的前面，参数是将要插入的内容。对于 .insertAfter(), 刚好相反，内容在方法前面，它将被放在参数里元素的后面。

请看下面的HTML：

```
<div class="container">
  <h2>HTML5</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

我们可以创建内容然后同时插在好几个元素后面：

```
$('<p>Test</p>').insertAfter('.inner');
```

结果如下：

```
<div class="container">
  <h2>HTML5</h2>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
  <p>Test</p>
</div>
```

- 练习想一下, 我们学过的那几对函数和这个函数的用法一样, 写一下例子.

完整参考事例:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('<span>开源世界!!!</span>').insertAfter('.new');
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/insertafter.html

- before()

.before() 和.insertBefore()实现同样的功能。主要的不同是语法——内容和目标的位置不同。 对于.before(),要插入的内容来自方法的参数：$(target).before(contentToBeInserted)。 对于.insertBefore()， 刚好相反，内容在方法前面并且插入到目标的前面， 而目标是传递给.insertBefore()方法的参数：$(contentToBeInserted).insertBefore(target)。

例子:

请看下面的HTML:

```
<div class="container">
  <h2>HTML5</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

我们可以创建内容然后同时插在好几个元素后面:

```
$('.inner').before('<p>Test</p>');
```

每个内部的

元素得到新的内容:

```
<div class="container">
  <h2>HTML5</h2>
  <p>Test</p>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
</div>
```

- 练习, 根据上面的例子, 请写出完整示例代码.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('.new').before('<span>开源世界!!!</span>');
      $('#new').before(function(index, html){
        return ('<h3>'+ index +'</h3>');
      })
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/before.html

- insertbefore()
  The .before()和.insertBefore()实现同样的功能。主要的区别是语法——内容和目标的位置。 对于 .before()，选择表达式在函数前面，内容作为参数，而.insertBefore()刚好相反，内容在方法前面，它将被放在参数里元素的前面。

请看下面的HTML：

```
<div class="container">
  <h2>HTML5</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

我们可以创建内容然后同时插在好几个元素后面：

```
$('<p>Test</p>').insertBefore('.inner');
```

结果如下：

```
<div class="container">
  <h2>HTML5</h2>
  <p>Test</p>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
</div>
```

- 练习: 根据前面的学习, 现在自己去写一个例子来验证这个函数的使用.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('<span>开源世界!!!</span>').insertBefore('.new');
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/insertbefore.html

替换元素

jQuery也提供了一组, 替换元素的函数, 我们接下来看一下他们的使用.

replaceWith()

.replaceWith()可以从DOM中移除内容，然后在这个地方插入新的内容。

请看下面的例子：

```
<div class="container">
  <div class="inner first">Hello</div>
  <div class="inner second">And</div>
  <div class="inner third">Goodbye</div>
</div>
```

我们可以用指定的HTML替换第二个 inner

：

```
$('div.second').replaceWith('<h2>New heading</h2>');
```

结果如下：

```
<div class="container">
  <div class="inner first">Hello</div>
  <h2>New heading</h2>
  <div class="inner third">Goodbye</div>
</div>
```

完整示例代码:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
     mydiv {border: 1px solid red}
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      //$('#new').replaceWith('<span>开源世界!!!</span>');
      $('#new').replaceWith(function(index){
        console.log(index);
        return $('#name').html($(this).contents());
      });

    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
    <p id="name"></p>
  </body>
  </div>

</html>
```

src/replacewith.html

- 练习: 如果实现当单击按钮后, 将替换内容.

replaceAll()

.replaceAll()和.replaceWith()功能类似，但是目标和源相反。

请看下面的HTML:

```
<div class="container">
  <div class="inner first">Hello</div>
  <div class="inner second">And</div>
  <div class="inner third">Goodbye</div>
</div>
```

我们可以创建一个元素然后用它替换其它元素：

```
$('<h2>New heading</h2>').replaceAll('.inner');
```

结果如下：

```
<div class="container">
  <h2>New heading</h2>
  <h2>New heading</h2>
  <h2>New heading</h2>
</div>
```

或者，我们也可以选择一个元素用来替换其它的:

```
$('.first').replaceAll('.third');
```

DOM结构的结果：

```
<div class="container">
  <div class="inner second">And</div>
  <div class="inner first">Hello</div>
</div>
```

从这个例子可以看出，用来替换的元素从老地方移到新位置，而不是复制。

- 注意事项: .replaceAll() 方法会删除与节点相关联的所有数据和事件处理程序 。
- 练习: 根据上面的例子, 写出一个完整事例代码.
- 练习: 增加一个按钮, 当单击按钮进, 将进行替换操作.

删除元素

detach()

.detach() 方法和.remove()一样, 除了 .detach()保存所有jQuery数据和被移走的元素相关联。当需要移走一个元素，不久又将该元素插入DOM时，这种方法很有用。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
     #name {border: 1px solid red}
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('#name').click(function(){
        $(this).detach();
      });
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="new">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
    <p id="name">请单击这里, 这一行将被删除!</p>
  </body>
  </div>

</html>
```

src/detach.html

empty()

这个方法不仅移除子元素（和其他后代元素），同样移除元素里的文本。因为，根据说明，元素里任何文本字符串都被看做是该元素的子节点。

请看下面的HTML:

```
<div class="container">
  <div class="hello">Hello</div>
  <div class="goodbye">Goodbye</div>
</div>
```

我们可以移除里面的任何元素

```
$('.hello').empty();
```

结果文本 Hello文本被删除：

```
<div class="container">
  <div class="hello"></div>
  <div class="goodbye">Goodbye</div>
</div>
```

如果

里面包含任何数量的嵌套元素，他们也会被移走。

为了避免内存泄漏，jQuery先移除子元素的数据和事件处理函数，然后移除子元素。

如果你想删除元素，不破坏他们的数据或事件处理程序（这些绑定的信息还可以在之后被重新添加回来），请使用.detach()代替 。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('#new').click(function(){
        $(this).empty();
      })
    });
    </script>
</head>
<body>
  <div id="all">
    <div id="new">单击这里，这一行和下面的子元素将会被删除.
      <p class="new">HTML5全栈开发</p>
    </div>
    <div class="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/empty.html

remove()

.remove() 将元素移出DOM。 当我们想将元素自身移除时我们用 .remove()，同时也会移除元素内部的一切，包括绑定的事件及与该元素相关的jQuery数据。要删除的元素不删除数据和事件的情况下，使用.detach()来代替。

请看下面的HTML:

```
<div class="container">
  <div class="hello">Hello</div>
  <div class="goodbye">Goodbye</div>
</div>
```

可以移除任何想要移除的元素:

```
$('.hello').remove();
```

结果如下：

```
<div class="container">
  <div class="goodbye">Goodbye</div>
</div>
```

如果

元素里面有子元素，他们同样会被移除。还有元素上的事件及 jQuery 数据也会被删除。

我们也可以添加一个可选的选择器参数来过滤匹配的元素。例如，前面的代码可以重写为：

```
$('div').remove('.hello');
```

结果将一样：

```
<div class="container">
  <div class="goodbye">Goodbye</div>
</div>
```

- 思考, remove和empty有什么区别.

代码示例:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      $('#new').click(function(){
        $(this).remove();
      })
    });
    </script>
</head>
<body>
  <div id="all">
    <div id="new">单击这里，这一行和下面的子元素将会被删除.
      <p class="new">HTML5全栈开发</p>
    </div>
    <div class="new">北京菜鸟在线</div>
  </body>
  </div>

</html>
```

src/remove.html

unwrap()

.unwrap()删除元素的父级元素。和 .wrap()的功能相反。匹配的元素（以及他们的兄弟元素，如果有的话）取代他们的父母在DOM结构。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>创建一个新的元素</title>
    <style media="screen">
    </style>
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
      var newElem = $("<div></div>").css('border', '1px solid red');
      $('#new').wrap(newElem);
      $('button#delete').click(function(){
        $('#new').unwrap();
      })
      $('button#add').click(function(){
        $('#new').wrap(newElem);
      })
    });
    </script>
</head>
<body>
  <div id="all">
    <div class="name">Richard.Wang</div>
    <p class="new">HTML5全栈开发</p>
    <div id="new">北京菜鸟在线</div>
    <button id = "delete" type="button" name="button">去掉红框</button>
    <button id = "add" type="button" name="button">加上红框</button>
  </body>
  </div>

</html>
```

单击事件

我们现在实现一个非常简单的单击事件, jQuery提供了一种简化事件操作的方法 — 简写事件方法, 简写事件方法的原理与对应的.on()调用相同, 可以减少一定的代码输入量.

代码事例: 单击按钮会显示你已经单击按钮了.

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <script src="../../jquery/jquery-3.0.0.min.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
        $("#btn").click(function(){
            $("#div1").text("你已经单击按钮了.");
        });
    });
    </script>

    <title>jQuery Hello world!</title>
</head>
<body>
    <div id="div2"></div>
    <div id = "div1"></div>
    <button id="btn" type="button" name="button">按钮</button>
</body>
</html>
```

课后作业

1. 描述： jquery的优点？
2. 描述： 描述页面的加载顺序(包括css文件和js文件)？
3. 描述： $(document).ready()的作用？
4. 描述： 简述scirpt标签的async 和defer属性？
5. 描述： jQuery中$符号的作用？
6. 练习： 练习课堂中css-code event-code selector-code dom-code代码