# Jquery简介

·一个优秀的js函数库

·封装简化DOM操作(CRUD) / Ajax

·为什么用Jquery？

- 强大选择器: 方便快速查找DOM元素

- 隐式遍历(迭代): 一次操作多个元素

- 读写合一: 读数据/写数据用的是一个函数

- 链式调用: 可以通过.不断调用jQuery对象的方法

- 事件处理

- DOM操作(CUD)

- 样式操作

- 动画

- 浏览器兼容

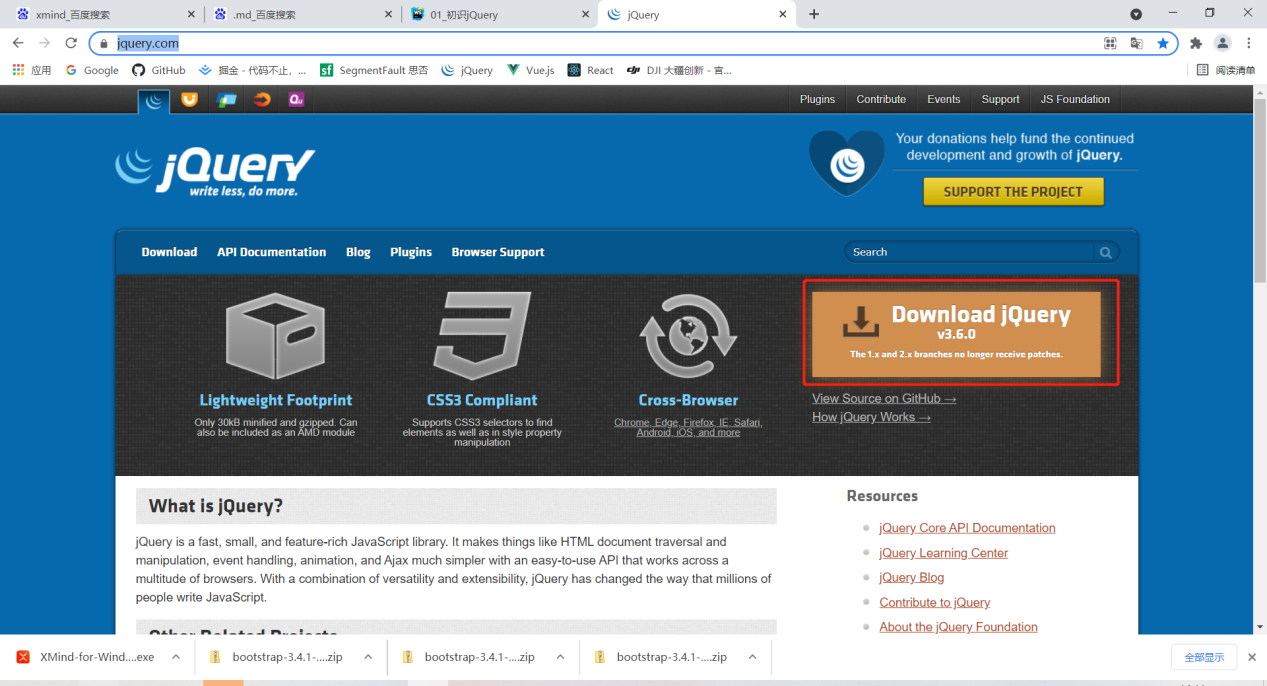
# jQuery的基本使用

## 本地引入jQuery库

·官网

- https://jquery.com/

·下载文件



·jQuery的不同版本

- 1.x

-- 兼容老版本ie

-- 文件更大

- 2.x

-- 部分ie8及以下不支持

-- 文件小，执行效率高

- 3.x

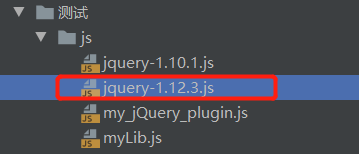
-- 提供了一些新的API

-- 提供不包含ajax/动画API的版本

·测试时用不压缩的

·上线时用压缩版

·将js文件引入到项目



<script type="text/javascript" src="js/jquery-1.12.3.js"></script>

## 外部引入

打开网站：https://www.bootcdn.cn/



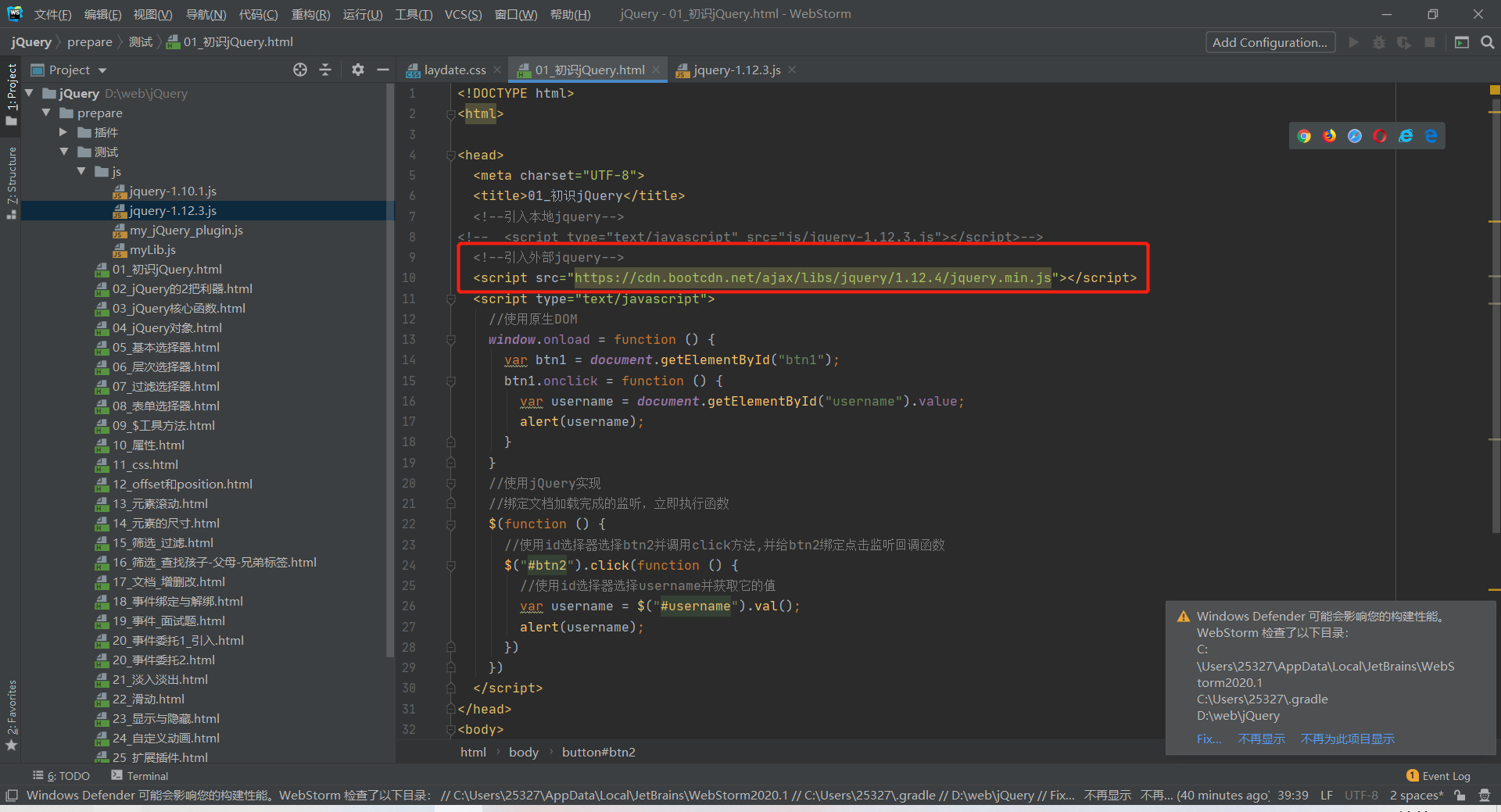
找到想要引入的版本



复制



引入到项目



<!--引入外部jquery-->  
<script src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

·测试时使用本地引入

## 使用jQuery

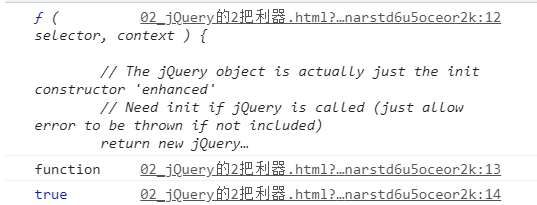
### jQuery的两把利器

·jQuery核心函数

- $

- jQuery

***console***.log($);  
***console***.log(typeof $);//function  
***console***.log(***jQuery*** === $);//true



- jQuery向外暴露的就是jQuery/$函数, 可以直接使用

·jQuery核心对象

- 执行jQuery核心函数返回的对象

- 执行jQuery/$函数返回的就是jQuery对象

- jQuery对象内部包含的是DOM元素对象的伪数组（可能只有一个元素）

- jQuery对象有很多有用的属性和方法

//使用id选择器获取btn2对象  
var $btn2 = $("#btn2");

***console***.log($() instanceof ***Object***);//true

<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="UTF-8">  
 <title>jQuery的二把利器</title>  
</head>  
<body>  
<button>测试</button>  
<script type="text/javascript" src="js/jquery-1.12.3.js"></script>  
<script type="text/javascript">  
 //jquery函数，直接可用  
 ***console***.log($);  
 ***console***.log(typeof $);//function  
 ***console***.log(***jQuery*** === $);//true  
 //执行jquery函数得到对象  
 ***console***.log($() instanceof ***Object***);//true  
</script>  
</body>  
</html>

# jQuery函数的使用

·作为一般函数调用：$(param)

- param为function: 相当于window.onload = function(文档加载完成的监听)

//绑定文档加载完成的监听  
$(function () {  
   
})

-- 能隐藏功能实现

- param是选择器字符串: 查找所有匹配的DOM元素, 返回包含所有DOM元素的**jQuery对象**，此时回调函数的this是btn元素的对象

var $btn = $("#btn");

//为btn绑定点击响应事件  
$btn.click(function () {  
 //这里的this是btn元素的对象  
 alert(this.innerHTML);  
})

- param是DOM元素: 将DOM元素对象包装为jQuery对象返回，包装后可使用jQuery对象的方法

//将btn对象包装为jQuery对象返回，并使用jQuery对象的html方法，html方法既可以读取值，又可以写入新的值，读写合一  
alert($(this).html());//获取btn的值  
$(this).html("修改");//修改btn的值

- param是标签字符串: 创建标签DOM元素对象并包装为jQuery对象返回

//创建一个新的输入框  
var input = $("<input type=\"text\" name=\"msg3\"/><br/>");

·appendTo("选择器")

- 该方法可以在某个选择器后边添加新的元素

//将创建好的元素插入到id选择器d1中  
input.appendTo("#d1");

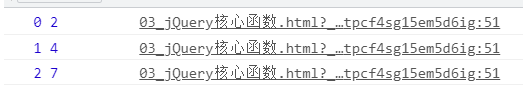
·作为对象使用: $.xxx()

- each(数组，回调函数)

- 隐式遍历数组或函数

- 回调函数传入的是数组下标和本身的值

var ***arr*** = [2,4,7];  
$.each(***arr***,function (index,item) {  
 ***console***.log(index,item);  
})



- $.trim(字符串)

- 去除两端的空格

//原生js方式去除两端的空格  
// console.log(str.trim());  
//jq方式去除两端的空格  
***console***.log($.trim(***str***));

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>03\_jQuery核心函数</title>  
</head>  
  
<body>  
  
<div id="d1">  
 <button id="btn">测试</button>  
 <br/>  
  
 <input type="text" name="msg1"/><br/>  
 <input type="text" name="msg2"/><br/>  
</div>  
  
  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
  
  
 /\*需求1. 点击按钮: 显示按钮的文本, 显示一个新的输入框\*/  
 //1). 参数为函数 : 当DOM加载完成后，执行其中的函数 回调函数  
 //绑定文档加载完成的监听  
 $(function () {  
 //根据id选择器获取btn按钮  
 var $btn = $("#btn");  
 //为btn绑定点击响应事件  
 $btn.click(function () {  
 //这里的this是btn元素的对象  
 // alert(this.innerHTML);  
 //将btn对象包装为jQuery对象返回，并使用jQuery对象的html方法，html方法既可以读取值，又可以写入新的值，读写合一  
 alert($(this).html());//获取btn的值  
 // $(this).html("修改");//修改btn的值  
 //创建一个新的输入框  
 var input = $("<input type=\"text\" name=\"msg3\"/><br/>");  
 //将创建好的元素插入到id选择器d1中  
 input.appendTo("#d1");  
 })  
 })  
  
 /\*需求2. 遍历输出数组中所有元素值\*/  
 var ***arr*** = [2,4,7];  
 $.each(***arr***,function (index,item) {  
 ***console***.log(index,item);  
 })  
  
 /\*需求3. 去掉" my atguigu "两端的空格\*/  
 var ***str*** = " my atguigu ";  
 //原生js方式去除两端的空格  
 // console.log(str.trim());  
 //jq方式去除两端的空格  
 ***console***.log($.trim(***str***));  
  
</script>  
</body>  
  
</html>

# jQuery对象的使用

·包含所有匹配的n个DOM元素的伪数组对象

·length/size()

- 得到dom元素的个数

var ***$buttons*** = $("button");  
//打印$buttons的长度，两种方法都可以  
***console***.log(***$buttons***.size(),***$buttons***.length);

·[index]

- 得到指定下标对应的dom元素

//输出按钮2和3的内容  
***console***.log(***$buttons***[1].innerHTML,***$buttons***.get(2).innerHTML);

·each(function(index, domEle){})

- 遍历所有dom元素

- index是下标

- domEle是遍历到的dom元素

***$buttons***.each(function (index,domEle) {  
 ***console***.log(index,domEle,this);//这里的this就是domEle  
 });  
***$buttons***.each(function () {  
 ***console***.log(this);//这里的this就是domEle  
});



·index()

- 得到当前dom元素在所有兄弟中的下标

***console***.log($("#btn3").index());//2

·伪数组

- 类型

-- Object对象

- 有length属性

- 有index数组下标属性

- 没有数组Array特别的方法，如不能使用forEach()，push()，pop()等方法

·证明一个对象是不是数组

***console***.log(***$buttons*** instanceof ***Array***);//false

<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="UTF-8">  
 <title>04\_jQuery对象</title>  
</head>  
  
<body>  
<button>测试一</button>  
<button>测试二</button>  
<button id="btn3">测试三</button>  
<button>测试四</button>  
  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 // 需求1. 统计一共有多少个按钮  
 /\*size()/length: 包含的DOM元素个数\*/  
 //通过标签选择器查找button标签  
 var ***$buttons*** = $("button");  
 //打印$buttons的长度，两种方法都可以  
 ***console***.log(***$buttons***.size(),***$buttons***.length);  
 // 需求2. 取出第2个button的文本  
 /\*[index]/get(index): 得到对应位置的DOM元素\*/  
 //输出按钮2和3的内容  
 ***console***.log(***$buttons***[1].innerHTML,***$buttons***.get(2).innerHTML);  
 // 输出所有button标签的文本  
 ***$buttons***.each(function (index,domEle) {  
 ***console***.log(index,domEle,this);//这里的this就是domEle  
 });  
 ***$buttons***.each(function () {  
 ***console***.log(this);//这里的this就是domEle  
 });  
 // 需求4. 输出'测试三'按钮是所有按钮中的第几个  
 /\*index(): 得到在所在兄弟元素中的下标\*/  
 ***console***.log($("#btn3").index());//2  
 ***console***.log(***$buttons*** instanceof ***Array***);//false  
</script>  
</body>  
</html>

# jQuery基本选择器

·选择器是有特定语法规则(css选择器)的字符串

·用来查找特定页面元素

·#id

- id选择器

//获取div1，并修改其背景颜色和字体颜色，此方法可设置多个属性  
// $("#div1").css({background:"red",color:"#fff"});  
//获取div1，并修改其背景颜色，此方法设置一个属性  
$("#div1").css("background","red");

·element

- 元素选择器

//获取所有div元素，并修改它们的背景颜色  
$("div").css("background","blue");

·.class

- 属性选择器

//选择所有class属性为box的元素,并修改它们的背景颜色  
$(".box").css("background","yellow");

·\*

- 任意标签

//选择所有元素  
$("\*").css("background","black");

·selector1,selector2,selectorN

- 取多个选择器的并集(组合选择器)

//选择所有的div和span元素,并修改它们的背景颜色  
$("div,span").css("background","green");

·selector1selector2selectorN

- 取多个选择器的交集(相交选择器)

//选择所有class属性为box的div元素,并修改它们的背景颜色  
$("div.box").css("background","#654756");

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>05\_基本选择器</title>  
</head>  
<body>  
<div id="div1" class="box">div1(class="box")</div>  
<div id="div2" class="box">div2(class="box")</div>  
<div id="div3">div3</div>  
<span class="box">span(class="box")</span>  
<br>  
<ul>  
 <li>AAAAA</li>  
 <li title="hello">BBBBB(title="hello")</li>  
 <li class="box">CCCCC(class="box")</li>  
 <li title="hello">DDDDDD(title="hello")</li>  
</ul>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
// 1. 选择id为div1的元素  
 //获取div1，并修改其背景颜色和字体颜色，此方法可设置多个属性  
 // $("#div1").css({background:"red",color:"#fff"});  
 //获取div1，并修改其背景颜色，此方法设置一个属性  
 $("#div1").css("background","red");  
// 2. 选择所有的div元素  
 //获取所有div元素，并修改它们的背景颜色  
 $("div").css("background","blue");  
 //选择所有class属性为box的元素,并修改它们的背景颜色  
 $(".box").css("background","yellow");  
 //选择所有的div和span元素,并修改它们的背景颜色  
 $("div,span").css("background","green");  
 //选择所有class属性为box的div元素,并修改它们的背景颜色  
 $("div.box").css("background","#654756");  
 //选择所有元素  
 $("\*").css("background","black");  
</script>  
</body>  
  
</html>

# 层次选择器

·查找子元素, 后代元素, 兄弟元素的选择器

·ancestor descendant

- 在给定的祖先元素下匹配所有指定的的后代元素

//选中ul下所有的的span  
$('ul span').css('background', 'red')

·parent>child

- 在给定的父元素下匹配所有指定的子元素

//选中ul下所有的子元素span  
$('ul>span').css('background', 'red')

·prev+next

- 匹配所有紧接在 prev 元素后的 next 元素

//选中class为box的下一个li  
$('.box+li').css('background', 'red')

·prev~siblings

- 匹配 prev 元素之后的所有 siblings 元素

//选中ul下的class为box的元素后面的所有兄弟元素  
$('ul .box~\*').css('background', 'red')

# 过滤选择器

·在原有选择器匹配的元素中进一步进行过滤的选择器

·选择第一个div

//选择第一个div  
$('div:first').css('background', 'red')

·选择最后一个class为box的元素

//选择最后一个class为box的元素  
$('.box:last').css('background', 'red')

·选择所有class属性不为box的div

//选择所有class属性不为box的div  
$('div:not(.box)').css('background', 'red')

·选择第二个和第三个li元素

//选择第二个和第三个li元素  
//多个过滤选择器是依次执行的  
$('li:gt(0):lt(2)').css('background', 'red') //索引起始位置发生变化，重新开始计算  
$('li:lt(3):gt(0)').css('background', 'red') //正确索引位置

·选择内容为BBBBB的li

//选择内容为BBBBB的li  
$('li:contains(BBBBB)').css('background', 'red')

·选择隐藏的li

//选择隐藏的li  
$('li:hidden ').show()

·选择有title属性的li元素

//选择有title属性的li元素  
$('li[title]').css('background', 'red')

·选择所有属性title为hello的li元素

//选择所有属性title为hello的li元素  
$('li[title=hello]').css('background', 'red')

<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="UTF-8">  
 <title>07\_过滤选择器</title>  
</head>  
  
<body>  
<div id="div1" class="box">class为box的div1</div>  
<div id="div2" class="box">class为box的div2</div>  
<div id="div3">div3</div>  
<span class="box">class为box的span</span>  
<br/>  
<ul>  
 <li>AAAAA</li>  
 <li title="hello">BBBBB</li>  
 <li class="box">CCCCC</li>  
 <li title="hello">DDDDDD</li>  
 <li title="two">BBBBB</li>  
 <li style="display:none">我本来是隐藏的</li>  
</ul>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 $(function () {  
 //选择第一个div  
 $('div:first').css('background', 'red')  
 //选择最后一个class为box的元素  
 // $('.box:last').css('background', 'red')  
 //选择所有class属性不为box的div  
 // $('div:not(.box)').css('background', 'red')  
 //选择第二个和第三个li元素  
 //多个过滤选择器是依次执行的  
 // $('li:gt(0):lt(2)').css('background', 'red') //索引起始位置发生变化，重新开始计算  
 // $('li:lt(3):gt(0)').css('background', 'red') //正确索引位置  
 //选择内容为BBBBB的li  
 // $('li:contains(BBBBB)').css('background', 'red')  
 //选择隐藏的li  
 // $('li:hidden ').show()  
 //选择有title属性的li元素  
 // $('li[title]').css('background', 'red')  
 //选择所有属性title为hello的li元素  
 // $('li[title=hello]').css('background', 'red')  
 })  
</script>  
</body>  
  
</html>

·id为data下，子元素为tbody下，子元素为tr，并且是**偶数**的元素

//id为data下，子元素为tbody下，子元素为tr，并且是偶数的元素  
$("#data>tbody>tr:odd").css('background', '#ccccff')

·id为data下，子元素为tbody下，子元素为tr，并且是**奇数**的元素

//id为data下，子元素为tbody下，子元素为tr，并且是奇数的元素  
$("#data>tbody>tr:even").css('background', '#ccccff')

### 表格隔行变色

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>01\_\_表格隔行变色</title>  
 <style>  
 div, span, p {  
 width: 140px;  
 height: 140px;  
 margin: 5px;  
 background: #aaa;  
 border: #000 1px solid;  
 float: left;  
 font-size: 17px;  
 font-family: Verdana;  
 }  
  
 div.mini {  
 width: 55px;  
 height: 55px;  
 background-color: #aaa;  
 font-size: 12px;  
 }  
  
 div.hide {  
 display: none;  
 }  
  
 #data {  
 width: 600px;  
 }  
  
 #data, td, th {  
 border-collapse: collapse;  
 border: 1px solid #aaaaaa;  
 }  
  
 th, td {  
 height: 28px;  
 }  
  
 #data thead {  
 background-color: #333399;  
 color: #ffffff;  
 }  
  
 .odd {  
 background-color: #ccccff;  
 }  
 </style>  
</head>  
<body>  
<table id="data">  
 <thead>  
 <tr>  
 <th>姓名</th>  
 <th>工资</th>  
 <th>入职时间</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td>Tom</td>  
 <td>$3500</td>  
 <td>2010-10-25</td>  
 <td><a href="javascript:void(0)">删除</a></td>  
 </tr>  
 <tr>  
 <td>Mary</td>  
 <td>$3400</td>  
 <td>2010-12-1</td>  
 <td><a href="javascript:void(0)">删除</a></td>  
 </tr>  
 <tr>  
 <td>King</td>  
 <td>$5900</td>  
 <td>2009-08-17</td>  
 <td><a href="javascript:void(0)">删除</a></td>  
 </tr>  
 <tr>  
 <td>Scott</td>  
 <td>$3800</td>  
 <td>2012-11-17</td>  
 <td><a href="javascript:void(0)">删除</a></td>  
 </tr>  
 <tr>  
 <td>Smith</td>  
 <td>$3100</td>  
 <td>2014-01-27</td>  
 <td><a href="javascript:void(0)">删除</a></td>  
 </tr>  
 <tr>  
 <td>Allen</td>  
 <td>$3700</td>  
 <td>2011-12-05</td>  
 <td><a href="javascript:void(0)">删除</a></td>  
 </tr>  
 </tbody>  
</table>  
  
<script src="jquery-1.10.1.js"></script>  
<script>  
 // $("#data>tbody>tr:odd").addClass("odd")  
 //id为data下，子元素为tbody下，子元素为tr，并且是奇数的元素  
 $("#data>tbody>tr:odd").css('background', '#ccccff')  
 // $("#data>tbody>tr:odd").attr('class', 'odd')  
</script>  
</body>  
</html>

# 表单选择器

·选择表单中type为input，有值为disabled的元素

//选择不可用的文本输入框  
$(':input:disabled').css('background', 'red')

·选择表单中type为checkbox，有值为checked的元素

//显示选择爱好 的个数  
***console***.log($(':checkbox:checked').length)

·选择表单中type值为submit的元素，并绑定点击监听事件

·选择select下，子元素为option，有值为selected的元素

$(':submit').click(function () {  
 //选择select下，子元素为option，有值为selected的元素  
 var city = $('select>option:selected').html();  
 var city1 = $('select').val() //得到的是选择的option的value  
 alert(city+city1)  
})

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>08\_表单选择器</title>  
</head>  
  
<body>  
<form>  
 用户名: <input type="text"/><br>  
 密 码: <input type="password"/><br>  
 爱 好:  
 <input type="checkbox" checked="checked"/>篮球  
 <input type="checkbox" checked="checked"/>足球  
 <input type="checkbox" checked="checked"/>羽毛球 <br>  
 性 别:  
 <input type="radio" name="sex" value='male'/>男  
 <input type="radio" name="sex" value='female'/>女<br>  
 邮 箱: <input type="text" name="email" disabled="disabled"/><br>  
 所在地:  
 <select>  
 <option value="1">北京</option>  
 <option value="2" selected="selected">天津</option>  
 <option value="3">河北</option>  
 </select><br>  
 <input type="submit" value="提交"/>  
</form>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 $(function () {  
 //选择不可用的文本输入框  
 // $(':input:disabled').css('background', 'red')  
 //显示选择爱好 的个数  
 // console.log($(':checkbox:checked').length)  
 //显示选择的城市名称  
 //选择表单中type值为submit的元素，并绑定点击监听事件  
 $(':submit').click(function () {  
 //选择select下，子元素为option，有值为selected的元素  
 var city = $('select>option:selected').html();  
 var city1 = $('select').val() //得到的是选择的option的value  
 alert(city+city1)  
 })  
 })  
</script>  
</body>  
</html>

# $的工具方法

·$.each()

- 遍历数组

-- 回调函数传入的是数组下标和本身的值

var ***arr*** = [2,4,7];  
$.each(***arr***,function (index,item) {  
 ***console***.log(index,item);  
})

- 遍历对象中的数据

-- 回调函数传入的是属性名和属性值

//创建一个对象  
var ***obj*** = {  
 name:"Tom",  
 age:"18",  
 setName:function (name) {  
 this.name = name;  
 }  
};  
//回调函数传入的是属性名和属性值  
$.each(***obj***,function (key,value) {  
 ***console***.log(key,value);  
})

·$.trim()

- 去除字符串两边的空格

//原生js方式去除两端的空格  
// console.log(str.trim());  
//jq方式去除两端的空格  
***console***.log($.trim(***str***));

·$.type(obj)

- 得到数据的类型

***console***.log($.type(***obj***));//object  
***console***.log($.type($));//function

·$.isArray(obj)

- 判断是否是数组

***console***.log($.isArray($("body")));//false  
***console***.log($.isArray([]));//true

·$.isFunction(obj)

- 判断是否是函数

***console***.log($.isFunction($));//true

·$.parseJSON(json)

- 解析json字符串转换为js对象/数组

//创建json对象  
var ***json*** = '{"name":"Tom","age":12}';  
//json对象-->js对象  
***console***.log($.parseJSON(***json***));  
//创建json数组  
***json*** = '[{"name":"Tom","age":12},{"name":"张三","age":18}]';  
//json数组-->js数组  
***console***.log($.parseJSON(***json***));

·json整体就2种类型:

- json对象 : {key1:value1 , key2:value2}

var ***json*** = '{"name":"Tom","age":12}';

- json数组: [value1, value2]

***json*** = '[{"name":"Tom","age":12},{"name":"张三","age":18}]';

- key只能是字符串

- value的类型

-- number

-- string

-- boolean

-- null

-- []

-- {}

# 多Tab点击切换

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>02\_多Tab点击切换</title>  
 <script src="jquery-1.10.1.js"></script>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 }  
  
 #tab li {  
 float: left;  
 list-style: none;  
 width: 80px;  
 height: 40px;  
 line-height: 40px;  
 cursor: pointer;  
 text-align: center;  
 }  
  
 #container {  
 position: relative;  
 }  
  
 #content1, #content2, #content3 {  
 width: 300px;  
 height: 100px;  
 padding: 30px;  
 position: absolute;  
 top: 40px;  
 left: 0;  
 }  
  
 #tab1, #content1 {  
 background-color: #ffcc00;  
 }  
  
 #tab2, #content2 {  
 background-color: #ff00cc;  
 }  
  
 #tab3, #content3 {  
 background-color: #00ccff;  
 }  
 </style>  
</head>  
<body>  
<h2>多Tab点击切换</h2>  
<ul id="tab">  
 <li id="tab1" value="1">10元套餐</li>  
 <li id="tab2" value="2">30元套餐</li>  
 <li id="tab3" value="3">50元包月</li>  
</ul>  
<div id="container">  
 <div id="content1">  
 10元套餐详情：<br/>&nbsp;每月套餐内拨打100分钟，超出部分2毛/分钟  
 </div>  
 <div id="content2" style="display: none">  
 30元套餐详情：<br/>&nbsp;每月套餐内拨打300分钟，超出部分1.5毛/分钟  
 </div>  
 <div id="content3" style="display: none">  
 50元包月详情：<br/>&nbsp;每月无限量随心打  
 </div>  
</div>  
  
<script>  
 // //方法一：先隐藏所有，再显示对应元素  
 // //获取container下，子元素为div的所有元素  
 // var $contents = $("#container>div");  
 // //给3个li加监听  
 // $("#tab>li").click(function () {//隐式遍历每个li，并为每个li加监听  
 // // console.log(this);//这里的this是点击的元素对象  
 // //隐藏获取container下的所有div子元素，将每个div的display设置为none  
 // $contents.css('display','none');  
 // //显示对应的内容  
 // //得到当前点击的li在兄弟元素中的下标  
 // var index = $(this).index();  
 // //显示指定的内容，两种方法都行，单链操作用原生代码效率比较高  
 // $contents[index].style.display = "block";  
 // // $($contents[index]).css('display','block');  
 // })  
  
 //方法一：先隐藏当前，再显示对应元素  
 //获取container下，子元素为div的所有元素  
 var ***$contents*** = $("#container>div");  
 //当前内容div显示的下标，初始值为0  
 var ***currIndex*** = 0;  
 //给3个li加监听  
 $("#tab>li").click(function () {//隐式遍历每个li，并为每个li加监听  
 // console.log(this);//这里的this是点击的元素对象  
 //隐藏当前已经显示的div内容  
 ***$contents***[***currIndex***].style.display = "none";  
 //显示对应的内容  
 //得到当前点击的li在兄弟元素中的下标  
 ***currIndex*** = $(this).index();  
 //显示指定的内容，两种方法都行，单链操作用原生代码效率比较高  
 ***$contents***[***currIndex***].style.display = "block";  
 // $($contents[index]).css('display','block');  
 })  
</script>  
</body>  
</html>

# jQuery属性

·操作任意属性

- attr()

-- 专门操作属性值为非布尔值的属性

-- 读取元素的指定属性值

//读取第一个div的title属性  
***console***.log($('div:first').attr('title'));

-- 设置（修改）元素的指定属性值

//给所有的div设置name属性(name值为atguigu)  
$('div').attr('name', 'atguigu');

//给所有的div设置class='guiguClass'  
$('div').attr('class', 'guiguClass');

- removeAttr()

-- 移除元素的某个属性

//移除所有div的title属性  
$('div').removeAttr('title');

- prop()

-- 专门操作属性值为非布尔值的属性

-- 设置（修改）元素的指定属性值

//点击'全选'按钮实现全选  
 $('button:first').click(function () {  
 $(':checkbox').prop('checked', true)  
 })  
 //点击'全不选'按钮实现全不选  
 $('button:last').click(function () {  
 $(':checkbox').prop('checked', false)  
 })  
})

·操作class属性

- addClass()

-- 给元素添加class属性

//给所有的div添加class值(abc)  
$('div').addClass('abc');

- removeClass()

-- 移除元素中class属性中的某些值

//移除所有div的class的guiguClass值  
$('div').removeClass('guiguClass');

·操作HTML代码/文本/值

- html()

-- 获取元素的体文本

//得到最后一个li的标签体文本  
***console***.log($('li:last').html());

-- 设置（修改）元素的体文本

//设置第一个li的标签体为"<h1>mmmmmmmmm</h1>"  
$('li:first').html('<h1>mmmmmmmmm</h1>');

- val()

-- 获取元素的value值

//得到输入框中的value值  
***console***.log($(':text').val())

-- 设置（修改）元素的value值

//将输入框的值设置为atguigu  
$(':text').val('atguigu');

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>10\_属性</title>  
</head>  
<body>  
  
<div id="div1" class="box" title="one">class为box的div1</div>  
<div id="div2" class="box" title="two">class为box的div2</div>  
<div id="div3">div3</div>  
<span class="box">class为box的span</span>  
<br/>  
<ul>  
 <li>AAAAA</li>  
 <li title="hello" class="box2">BBBBB</li>  
 <li class="box">CCCCC</li>  
 <li title="hello">DDDDDD</li>  
 <li title="two"><span>BBBBB</span></li>  
</ul>  
  
<input type="text" name="username" value="guiguClass"/>  
<br>  
<input type="checkbox">  
<input type="checkbox">  
<br>  
<button>选中</button>  
<button>不选中</button>  
  
<!--  
1. 操作任意属性  
 attr()  
 removeAttr()  
 prop()  
2. 操作class属性  
 addClass()  
 removeClass()  
3. 操作HTML代码/文本/值  
 html()  
 val()  
-->  
  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 $(function () {  
 //读取第一个div的title属性  
 ***console***.log($('div:first').attr('title'));  
 //给所有的div设置name属性(name值为atguigu)  
 $('div').attr('name', 'atguigu');  
 //移除所有div的title属性  
 $('div').removeAttr('title');  
 //给所有的div设置class='guiguClass'  
 $('div').attr('class', 'guiguClass');  
 //给所有的div添加class值(abc)  
 $('div').addClass('abc');  
 //移除所有div的class的guiguClass值  
 $('div').removeClass('guiguClass');  
 //得到最后一个li的标签体文本  
 ***console***.log($('li:last').html());  
 //设置第一个li的标签体为"<h1>mmmmmmmmm</h1>"  
 $('li:first').html('<h1>mmmmmmmmm</h1>');  
 //得到输入框中的value值  
 ***console***.log($(':text').val())  
 //将输入框的值设置为atguigu  
 $(':text').val('atguigu');  
 //点击'全选'按钮实现全选  
 $('button:first').click(function () {  
 $(':checkbox').prop('checked', true)  
 })  
 //点击'全不选'按钮实现全不选  
 $('button:last').click(function () {  
 $(':checkbox').prop('checked', false)  
 })  
 })  
  
</script>  
</body>  
</html>

# jQuery操作css

·获取元素的某个css属性值

//得到第一个p标签的颜色  
***console***.log($('p:first').css('color'));

·设置元素的某个css属性值

//设置所有p标签的文本颜色为red  
$('p').css('color', 'red');

·设置元素的多个css属性值

//将css样式封装成对象，值可以不加px  
$('p:eq(1)').css({  
 color: '#ff0011',  
 background: 'blue',  
 width: 300,  
 height: 30  
})

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>11\_css</title>  
</head>  
<body>  
<p style="color: blue;">尚硅谷的后裔</p>  
<p style="color: green;">太阳的后裔</p>  
  
<!--  
设置css样式/读取css值  
 css()  
-->  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 /\*  
 1. 得到第一个p标签的颜色  
 2. 设置所有p标签的文本颜色为red  
 3. 设置第2个p的字体颜色(#ff0011),背景(blue),宽(300px), 高(30px)  
 \*/  
 $(function () {  
 //得到第一个p标签的颜色  
 ***console***.log($('p:first').css('color'));  
 //设置所有p标签的文本颜色为red  
 $('p').css('color', 'red');  
 //设置第2个p的字体颜色(#ff0011),背景(blue),宽(300px), 高(30px)  
 //将css样式封装成对象，值可以不加px  
 $('p:eq(1)').css({  
 color: '#ff0011',  
 background: 'blue',  
 width: 300,  
 })  
 })  
</script>  
</body>  
</html>

# offset

·获取元素在页面的相对偏移

·返回的对象包含两个整型属性

- left

- top

·该方法只对可见元素有效

var offset = $(".div1").offset();  
***console***.log("top:"+offset.top,"left:"+offset.left);

·设置元素在页面的相对偏移

//设置 div2 相对于页面的左上角的位置  
$(".div2").offset({  
 left:50,  
 top:100  
});

# position

·获取元素在父元素的相对偏移

·返回的对象包含两个整型属性

- left

- top

·该方法只对可见元素有效

//打印 div1 相对于父元素左上角的位置  
var position = $(".div1").position();  
***console***.log("top:"+position.top,"left:"+position.left);

# scrollTop

·读取滚动条的Y坐标

//得到div滚动条的坐标  
***console***.log($('div').scrollTop())  
//得到页面滚动条的坐标  
***console***.log($('body').scrollTop() + $('html').scrollTop())  
***console***.log($(***document***.body).scrollTop() + $(***document***.documentElement).scrollTop())//兼容chrome和IE

·设置滚动条的Y坐标

//让div的滚动条滚动到指定位置  
$('div').scrollTop(150)  
//让页面的滚动条滚动到指定位置  
$('body, html').scrollTop(200)

<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="UTF-8">  
 <title>13\_元素滚动</title>  
</head>  
<body style="height: 2000px;">  
<div style="border:1px solid black;width:100px;height:150px;overflow:auto">  
 This is some text. This is some text. This is some text. This is some text.  
 This is some text. This is some text. This is some text. This is some text.  
 This is some text. This is some text. This is some text. This is some text.  
 This is some text. This is some text. This is some text. This is some text.  
 This is some text. This is some text. This is some text. This is some text.  
 This is some text. This is some text. This is some text. This is some text.  
 his is some text.  
</div>  
<br>  
<br>  
<br>  
<button id="btn1">得到scrollTop</button>  
<button id="btn2">设置scrollTop</button>  
<script src="js/jquery-1.10.1.js"></script>  
<script>  
 $(function () {  
 // 1. 得到div或页面滚动条的坐标  
 $('#btn1').click(function () {  
 //得到div滚动条的坐标  
 ***console***.log($('div').scrollTop())  
 //得到页面滚动条的坐标  
 ***console***.log($('body').scrollTop() + $('html').scrollTop())  
 ***console***.log($(***document***.body).scrollTop() + $(***document***.documentElement).scrollTop())//兼容chrome和IE  
 })  
  
 // 2. 让div或页面的滚动条滚动到指定位置  
 $('#btn2').click(function () {  
 //让div的滚动条滚动到指定位置  
 $('div').scrollTop(150)  
 //让页面的滚动条滚动到指定位置  
 $('body, html').scrollTop(200)  
 })  
 })  
</script>  
</body>  
  
</html>

## 回到顶部

<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="UTF-8">  
 <title>03\_回到顶部</title>  
 <style>  
 #to\_top {  
 width: 30px;  
 height: 40px;  
 font: 14px/20px arial;  
 text-align: center;  
 background: #06c;  
 position: fixed;  
 cursor: pointer;  
 color: #fff;  
 left: 1250px;  
 top: 500px;  
 }  
 </style>  
</head>  
<body style="height: 2000px;">  
  
<div id="to\_top">返回顶部</div>  
  
<script src="jquery-1.10.1.js"></script>  
<script>  
 $("#to\_top").click(function () {  
 //瞬间回到顶部  
 // $("html,body").scrollTop(0);  
 //平滑滚动到顶部，总距离、总时间、间隔时间，使用循环定时器循环滚动，到达顶部，停止定时器  
 //获取html和body元素  
 var $page = $("html,body");  
 //总距离  
 var distance = $("html").scrollTop() + $("body").scrollTop();  
 //总时间  
 var times = 500;  
 //间隔时间  
 var intervalTimle = 50;  
 //每个间隔时间移动的距离  
 var itemDistance = distance/(times/intervalTimle);  
 var intervalId = setInterval(function () {  
 //每次滚动一段距离  
 distance -= itemDistance;  
 if (distance<=0){//如果distance小于0  
 distance = 0;//修正  
 //停止定时器  
 clearInterval(intervalId);  
 }  
 $page.scrollTop(distance);  
 },intervalTimle);  
 })  
</script>  
</body>  
  
</html>

# jQuery元素的尺寸

·内容尺寸

- height(): height

- width(): width

·内部尺寸

- innerHeight(): height+padding

- innerWidth(): width+padding

·外部尺寸

- outerHeight(false/true): height+padding+border 如果是true, 加上margin

- outerWidth(false/true): width+padding+border 如果是true, 加上margin

·width()

- 获取元素内容区宽度

·height()

- 获取元素内容区高度

// 内容尺寸  
***console***.log($div.width(), $div.height());//100 150

·innerWidth

- - 获取元素内部宽度

·innerHeight

-- 获取元素内部高度

***console***.log($div.innerWidth(), $div.innerHeight());//120 170

·outerWidth()

-- 获取元素可见宽度

·outerHeight()

-- 获取元素可见高度

***console***.log($div.outerWidth(), $div.outerHeight());//140 190

·outerWidth(true)

-- 获取元素可见宽度+margin宽度

·outerHeight(true)

-- 获取元素外部高度+margin高度

***console***.log($div.outerWidth(true), $div.outerHeight(true))//160 210

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>14\_元素的尺寸</title>  
</head>  
<style>  
 div {  
 width: 100px;  
 height: 150px;  
 background: red;  
 padding: 10px;  
 border: 10px #fbd850 solid;  
 margin: 10px;  
 }  
</style>  
</head>  
  
<body>  
<div>div</div>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script>  
 $(function () {  
 var $div = $('div')  
 // 内容尺寸  
 ***console***.log($div.width(), $div.height());//100 150  
 // 内部尺寸  
 ***console***.log($div.innerWidth(), $div.innerHeight());//120 170  
 //外部尺寸  
 ***console***.log($div.outerWidth(), $div.outerHeight());//140 190  
 ***console***.log($div.outerWidth(true), $div.outerHeight(true))//160 210  
 })  
</script>  
</body>  
  
</html>

# jQuery对象的过滤

·在jQuery对象中的元素对象数组中过滤出一部分元素来

- first()

-- 在jQuery对象中的元素对象数组中获取第一个元素

//在$lis中获取ul下li标签第一个  
$lis.first().css('background', 'red')

- last()

-- 在jQuery对象中的元素对象数组中获取最后一个元素

//在$lis中获取ul下li标签的最后一个  
$lis.last().css('background', 'red')

- eq(index|-index)

-- 在jQuery对象中的元素对象数组中获取指定索引的元素

//在$lis中获取ul下li标签的第二个  
$lis.eq(1).css('background', 'red')

- filter(selector)

-- 在jQuery对象中的元素对象数组中获取符合指定条件的元素

//在$lis中获取ul下li标签中title属性为hello的  
$lis.filter('[title=hello]').css('background', 'red')

//在$lis中获取ul下li标签中有title属性且title属性不为hello的  
$lis.filter('[title][title!=hello]').css('background', 'red')  
$lis.filter('[title!=hello]').filter('[title]').css('background', 'red')

- not(selector)

-- 在jQuery对象中的元素对象数组中获取不符合指定条件的元素

/在$lis中获取ul下li标签中title属性不为hello的  
$lis.not('[title=hello]').css('background', 'red')

- has(selector)

-- 在jQuery对象中的元素对象数组中获取包含特定后代的元素

//在$lis中获取ul下li标签中有span子标签的  
$lis.has('span').css('background', 'red')

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>15\_筛选\_过滤</title>  
</head>  
  
<body>  
<ul>  
 <li>AAAAA</li>  
 <li title="hello" class="box2">BBBBB</li>  
 <li class="box">CCCCC</li>  
 <li title="hello">DDDDDD</li>  
 <li title="two"><span>BBBBB</span></li>  
</ul>  
<li>eeeee</li>  
<li>EEEEE</li>  
<br>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
  
 $(function () {  
 //获取ul下所有li  
 var $lis = $('ul>li');  
 //在$lis中获取ul下li标签第一个  
 // $lis.first().css('background', 'red')  
 //在$lis中获取ul下li标签的最后一个  
 // $lis.last().css('background', 'red')  
 //在$lis中获取ul下li标签的第二个  
 // $lis.eq(1).css('background', 'red')  
 //在$lis中获取ul下li标签中title属性为hello的  
 // $lis.filter('[title=hello]').css('background', 'red')  
 //在$lis中获取ul下li标签中title属性不为hello的  
 // $lis.not('[title=hello]').css('background', 'red')  
 //在$lis中获取ul下li标签中有title属性且title属性不为hello的  
 $lis.filter('[title][title!=hello]').css('background', 'red')  
 // $lis.filter('[title!=hello]').filter('[title]').css('background', 'red')  
 //在$lis中获取ul下li标签中有span子标签的  
 // $lis.has('span').css('background', 'red')  
 })  
</script>  
</body>  
</html>

# jQuery对象的查找

·在已经匹配出的元素集合中根据选择器查找孩子/父母/兄弟标签

·查找的所有元素都不包括他自己

·children()

- 子元素中找符合条件的元素

//在$ul中获取ul标签的第2个span子标签  
$ul.children('span:eq(1)').css('background', 'red')

·find()

- 后代元素中找符合条件的元素

//在$ul中获取ul标签的第2个span后代标签  
$ul.find('span:eq(1)').css('background', 'red')

·parent()

- 找到父元素

//在$ul中获取ul标签的父标签  
$ul.parent().css('background', 'red')

·prevAll()

- 前面所有的兄弟元素

//id为cc的li标签的前面的所有li标签  
$ul.children('#cc').prevAll('li').css('background', 'red')

·nextAll()

- 后面所有的兄弟元素

//id为cc的li标签的前面的所有li标签  
$ul.children('#cc').nextAll('li').css('background', 'red')

·siblings() : 前后所有的兄弟元素

//id为cc的li标签的所有兄弟li标签  
$ul.children('#cc').siblings('li').css('background', 'red')

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>16\_筛选\_查找孩子-父母-兄弟标签</title>  
</head>  
<body>  
<div id="div1" class="box" title="one">class为box的div1</div>  
<div id="div2" class="box">class为box的div2</div>  
<div id="div3">div3</div>  
<span class="box">class为box的span</span>  
<br/>  
<div>  
 <ul>  
 <span>span文本1</span>  
 <li>AAAAA</li>  
 <li title="hello" class="box2">BBBBB</li>  
 <li class="box" id='cc'>CCCCC</li>  
 <li title="hello">DDDDDD</li>  
 <li title="two"><span>span文本2</span></li>  
 <span>span文本3</span>  
 </ul>  
 <span>span文本444</span><br>  
 <li>eeeee</li>  
 <li>EEEEE</li>  
 <br>  
</div>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
  
 $(function () {  
 var $ul = $('ul')  
 //在$ul中获取ul标签的第2个span子标签  
 // $ul.children('span:eq(1)').css('background', 'red')  
 //在$ul中获取ul标签的第2个span后代标签  
 // $ul.find('span:eq(1)').css('background', 'red')  
 //在$ul中获取ul标签的父标签  
 // $ul.parent().css('background', 'red')  
 //id为cc的li标签的前面的所有li标签  
 // $ul.children('#cc').prevAll('li').css('background', 'red')  
 //id为cc的li标签的所有兄弟li标签  
 $ul.children('#cc').siblings('li').css('background', 'red')  
 })  
  
</script>  
</body>  
</html>

## 爱好选择器

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
 <title>全选练习</title>  
</head>  
<body>  
  
<form>  
 你爱好的运动是？<input type="checkbox" id="checkedAllBox"/>全选/全不选  
  
 <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="checkedRevBtn" value="反　选"/>  
 <input type="button" id="sendBtn" value="提　交"/>  
</form>  
  
<script src="jquery-1.10.1.js"></script>  
<script type="text/javascript">  
 //找到全选/全不选  
 var ***$checkedAllBox*** = $("#checkedAllBox");  
 //找到所有name为items的标签  
 var ***$items*** = $(":checkbox[name=items]");  
 //点击'全选': 选中所有爱好  
 //为全选按钮添加单机绑定事件  
 $("#checkedAllBtn").click(function () {  
 //将$items中的所有元素的checked属性设置为true  
 ***$items***.prop("checked",true);  
 //更新'全选/全不选'为选中状态  
 ***$checkedAllBox***.prop("checked",true);  
 });  
 //点击'全不选': 所有爱好都不勾选  
 //为全不选按钮添加单机绑定事件  
 $("#checkedNoBtn").click(function () {  
 //将$items中的所有元素的checked属性设置为false  
 ***$items***.prop("checked",false);  
 //更新'全选/全不选'为不选中状态  
 ***$checkedAllBox***.prop("checked",false);  
 });  
 //点击'反选': 改变所有爱好的勾选状态  
 //为反选按钮添加单机绑定事件  
 $("#checkedRevBtn").click(function () {  
 //遍历$items的每一个元素  
 ***$items***.each(function () {  
 //将每个元素的checked属性取反  
 this.checked = !this.checked;  
 //更新'全选/全不选'的选中状态  
 //找到所有未选中的元素,如果个数0,则说明全选,'全选/全不选'为选中状态  
 ***$checkedAllBox***.prop("checked",***$items***.filter(":not(:checked)").length === 0);  
 });  
 });  
 //点击'提交': 提示所有勾选的爱好  
 $("#sendBtn").click(function () {  
 //找到选中的元素,并遍历选中的元素  
 ***$items***.filter(":checked").each(function () {  
 alert(this.value);  
 })  
 })  
 //点击'全选/全不选': 选中所有爱好, 或者全不选中  
 ***$checkedAllBox***.click(function () {  
 //将$items中的所有元素的checked属性设置为与$checkedAllBox的checked属性值相同  
 ***$items***.prop("checked",this.checked);  
 })  
 //点击某个爱好时, 必要时更新'全选/全不选'的选中状态  
 //使用隐式遍历为每个元素设置监听事件  
 ***$items***.click(function () {  
 //找到所有未选中的元素,如果个数0,则说明全选,'全选/全不选'为选中状态  
 ***$checkedAllBox***.prop("checked",***$items***.filter(":not(:checked)").length === 0);  
 })  
</script>  
</body>  
</html>

# jQuery文档增删改

·添加/替换元素

- append(content)

-- 向当前匹配的所有元素内部的最后插入指定内容

//向id为ul1的ul下添加一个span(最后)  
$('#ul1').append('<span>append()添加的span</span>')

- appendTo()

-- 向当前匹配的所有元素内部的最后插入指定内容

//向id为ul1的ul下添加一个span(最后)  
$('<span>appendTo()添加的span</span>').appendTo('#ul1')

- prepend(content)

-- 向当前匹配的所有元素内部的最前面插入指定内容

//向id为ul1的ul下添加一个span(最前)  
$('#ul1').prepend('<span>prepend()添加的span</span>')

- prependTo()

-- 向当前匹配的所有元素内部的最前面插入指定内容

//向id为ul1的ul下添加一个span(最前)  
$('<span>prependTo()添加的span</span>').prependTo('#ul1')

- before(content)

-- 将指定内容插入到当前所有匹配元素的前面

//在id为ul1的ul下的li(title为hello)的前面添加span  
$('#ul1>li[title=hello]').before('<span>before()添加的span</span>')

- after(content)

-- 将指定内容插入到当前所有匹配元素的后面替换节点

//在id为ul1的ul下的li(title为hello)的后面添加span  
$('#ul1>li[title=hello]').after('<span>after()添加的span</span>')

- replaceWith(content)

-- 用指定内容替换所有匹配的标签删除节点

//将在id为ul2的ul下的li(title为hello)全部替换为p  
$('#ul1>li[title=hello]').replaceWith('<p>replaceWith()替换的p</p>')

·删除元素

- empty()

-- 删除所有匹配元素的子元素

//移除id为ul2的ul下的所有li  
$('#ul2').empty()

- remove()

-- 删除所有匹配的元素

//移除id为ul2的ul下的所有li  
$('#ul2>li').remove()

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>17\_文档\_增删改</title>  
</head>  
<style type="text/css">  
 \* {  
 margin: 0px;  
 }  
  
 .div1 {  
 position: absolute;  
 width: 200px;  
 height: 200px;  
 top: 20px;  
 left: 10px;  
 background: blue;  
 }  
  
 .div2 {  
 position: absolute;  
 width: 100px;  
 height: 100px;  
 /\*top: 50px;\*/  
 background: red;  
 }  
  
 .div3 {  
 position: absolute;  
 top: 250px;  
 }  
</style>  
  
<body>  
<ul id="ul1">  
 <li>AAAAA</li>  
  
 <li title="hello">BBBBB</li>  
 <li class="box">CCCCC</li>  
  
 <li title="hello">DDDDDD</li>  
 <li title="two">EEEEE</li>  
 <li>FFFFF</li>  
</ul>  
<br>  
<br>  
<ul id="ul2">  
 <li>aaa</li>  
 <li title="hello">bbb</li>  
 <li class="box">ccc</li>  
 <li title="hello">ddd</li>  
 <li title="two">eee</li>  
</ul>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 $(function () {  
 //向id为ul1的ul下添加一个span(最后)  
 $('#ul1').append('<span>append()添加的span</span>')  
 $('<span>appendTo()添加的span</span>').appendTo('#ul1')  
 //向id为ul1的ul下添加一个span(最前)  
 $('#ul1').prepend('<span>prepend()添加的span</span>')  
 $('<span>prependTo()添加的span</span>').prependTo('#ul1')  
  
 //在id为ul1的ul下的li(title为hello)的前面添加span  
 $('#ul1>li[title=hello]').before('<span>before()添加的span</span>')  
  
 //在id为ul1的ul下的li(title为hello)的后面添加span  
 $('#ul1>li[title=hello]').after('<span>after()添加的span</span>')  
 //将在id为ul2的ul下的li(title为hello)全部替换为p  
 $('#ul1>li[title=hello]').replaceWith('<p>replaceWith()替换的p</p>')  
 //移除id为ul2的ul下的所有li  
 $('#ul2').empty()  
 $('#ul2>li').remove()  
 });  
</script>  
</body>  
</html>

## 添加删除记录

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  
<html>  
  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
 <title>添加删除记录练习</title>  
 <link rel="stylesheet" type="text/css" href="css.css"/>  
</head>  
<body>  
  
<table id="employeeTable">  
 <tr>  
 <th>Name</th>  
 <th>Email</th>  
 <th>Salary</th>  
 <th>&nbsp;</th>  
 </tr>  
 <tr>  
 <td>Tom</td>  
 <td>tom@tom.com</td>  
 <td>5000</td>  
 <td><a href="deleteEmp?id=001">Delete</a></td>  
 </tr>  
 <tr>  
 <td>Jerry</td>  
 <td>jerry@sohu.com</td>  
 <td>8000</td>  
 <td><a href="deleteEmp?id=002">Delete</a></td>  
 </tr>  
 <tr>  
 <td>Bob</td>  
 <td>bob@tom.com</td>  
 <td>10000</td>  
 <td><a href="deleteEmp?id=003">Delete</a></td>  
 </tr>  
  
</table>  
  
<div id="formDiv">  
  
 <h4>添加新员工</h4>  
  
 <table>  
 <tr>  
 <td class="word">name:</td>  
 <td class="inp">  
 <input type="text" name="empName" id="empName"/>  
 </td>  
 </tr>  
 <tr>  
 <td class="word">email:</td>  
 <td class="inp">  
 <input type="text" name="email" id="email"/>  
 </td>  
 </tr>  
 <tr>  
 <td class="word">salary:</td>  
 <td class="inp">  
 <input type="text" name="salary" id="salary"/>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2" align="center">  
 <button id="addEmpButton" value="abc">  
 Submit  
 </button>  
 </td>  
 </tr>  
 </table>  
</div>  
  
<script src="jquery-1.10.1.js" type="text/javascript" charset="utf-8"></script>  
<script type="text/javascript">  
 /\*  
 功能说明:  
 1. 点击'Submit', 根据输入的信息在表单中生成一行员工信息  
 2. 点击Delete链接, 提示删除当前行信息, 点击确定后删除信息  
 技术要点:  
 1. DOM查询  
 2. 绑定事件监听  
 3. DOM增删改  
 4. 取消事件的默认行为  
 \*/  
 //为submit按钮绑定一个监听事件  
 $("#addEmpButton").click(function () {  
 //收集输入的数据  
 var $empName = $("#empName");  
 var $email = $("#email");  
 var $salary = $("#salary");  
 var empName = $empName.val();  
 var email = $email.val();  
 var salary = $salary.val();  
  
 //生成对应的tr标签结构,并插入到#employeeTable的tbody中  
 /\*  
 <tr>  
 <td>Tom</td>  
 <td>tom@tom.com</td>  
 <td>5000</td>  
 <td><a href="deleteEmp?id=001">Delete</a></td>  
 </tr>  
 \*/  
 //生成对应的tr标签结构  
 var $addemp = $("<tr></tr>")  
 .append("<td>" + empName + "</td>")  
 .append("<td>" + email + "</td>")  
 .append("<td>" + salary + "</td>")  
 .append('<td><a href="deleteEmp?id=' + ***Date***.now() + '">Delete</a></td>');  
 //为新增的便签绑定点击监听  
 $addemp.find("a").click(clickDelete);  
 //插入到#employeeTable的tbody中  
 $addemp.appendTo("#employeeTable>tbody");  
 //清除输入的数据  
 $empName.val("");  
 $email.val("");  
 $salary.val("");  
 })  
 //点击Delete链接, 提示删除当前行信息, 点击确定后删除信息  
 //为Delete链接绑定一个监听事件  
 $("#employeeTable a").click(clickDelete);  
  
 //删除tr的方法  
 function clickDelete() {  
 //获取tr,先找父亲td,然后再找父亲tr  
 var $tr = $(this).parent().parent();  
 //找到tr的第一个孩子并获取文本  
 var name = $tr.children(":first").html();  
 if (confirm("确定删除"+name+"吗?")){  
 $tr.remove();  
 }  
 //取消浏览器默认行为  
 return false;  
 }  
</script>  
</body>  
</html>

# jQuery事件处理

## 事件绑定与解绑

·事件绑定(2种)：

- eventName(function(){})

-- 绑定对应事件名的监听, 例如：$('#div').click(function(){});

-- click()

--- 点击事件

//给class为out的div的点击事件绑定监听函数，打印'out clicked'(用两种方法绑定)  
$('.out').click(function () {  
 ***console***.log('out click1')  
})

-- mouseenter()

-- mouseleave()

--- 鼠标移入和鼠标移出事件

$('.inner')  
 .mouseenter(function () {  
 ***console***.log('进入...')  
 })  
 .mouseleave(function () {  
 ***console***.log('离开...')  
 })

-- hover()

--- 鼠标移入和鼠标移出事件

//给class为inner的div的鼠标移入和鼠标移出事件绑定监听函数  
$('.inner').hover(function () {  
 ***console***.log('进入...')  
}, function () {  
 ***console***.log('离开...')  
})

- on(eventName, funcion(){})

-- 通用的绑定事件监听, 例如：$('#div').on('click', function(){})

-- 点击事件

//给class为out的div的点击事件绑定监听函数，打印'out clicked'(用两种方法绑定)  
$('.out').on('click', function () {  
 ***console***.log('out clicked2')  
})

-- 鼠标移入移出事件

//给class为inner的div的鼠标移入和鼠标移出事件绑定监听函数  
$('.inner')  
 .on('mouseenter', function () {  
 ***console***.log('进入...')  
 })  
 .on('mouseleave', function () {  
 ***console***.log('离开...')  
 })

·优缺点

- eventName: 编码方便,有的事件监听不支持

- on: 编码不方便, 更通用

- 都可以同时绑定多个事件

·事件解绑

- off(eventName)

- 解除所有绑定事件

//点击btn1解除inner上的所有事件监听  
$('#btn1').click(function () {  
 $('.inner').off()  
})

- 解除指定绑定事件

//点击btn2解除inner上的mouseover事件  
$('#btn2').click(function () {  
 $('.inner').off('mouseover')  
})

·事件的坐标

- event.clientX, event.clientY 相对于视口的左上角

- event.pageX, event.pageY 相对于页面的左上角

- event.offsetX, event.offsetY 相对于事件元素左上角

//点击btn3得到事件坐标  
$('#btn3').click(function (event) {  
 ***console***.log(event.offsetX, event.offsetY)  
 ***console***.log(event.clientX, event.clientY)  
 ***console***.log(event.pageX, event.pageY)  
})

·事件相关处理

- 停止事件冒泡 : event.stopPropagation()

//点击.inner区域, 外部点击监听不响应  
$('.inner').click(function (event) {  
 ***console***.log('click inner')  
 // 停止事件冒泡  
 event.stopPropagation()  
})

- 阻止事件默认行为 : event.preventDefault()

//点击链接, 如果当前时间是偶数不跳转  
$('#test4').click(function (event) {  
 var time = ***Date***.now()  
 ***console***.log(time);  
 if(time%2===0) {  
 // 阻止事件默认行为  
 event.preventDefault()  
 }  
})

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>18\_事件绑定与解绑</title>  
</head>  
<style type="text/css">  
 \* {  
 margin: 0px;  
 }  
  
 .out {  
 position: absolute;  
 width: 200px;  
 height: 200px;  
 top: 20px;  
 left: 10px;  
 background: blue;  
 }  
  
 .inner {  
 position: absolute;  
 width: 100px;  
 height: 100px;  
 top: 50px;  
 background: red;  
 }  
  
 .divBtn {  
 position: absolute;  
 top: 250px;  
 }  
  
</style>  
  
<body style="height: 2000px;">  
  
<div class="out">  
 外部DIV  
 <div class="inner">内部div</div>  
</div>  
  
<div class='divBtn'>  
 <button id="btn1">取消绑定所有事件</button>  
 <button id="btn2">取消绑定mouseover事件</button>  
 <button id="btn3">测试事件坐标</button>  
 <a href="http://www.baidu.com" id="test4">百度一下</a>  
</div>  
<script src="js/jquery-1.10.1.js" type="text/javascript" charset="utf-8"></script>  
<script type="text/javascript">  
  
 $(function () {  
 //给class为out的div的点击事件绑定监听函数，打印'out clicked'(用两种方法绑定)  
 // $('.out').click(function () {  
 // console.log('out click1')  
 // })  
 $('.out').on('click', function () {  
 ***console***.log('out clicked2')  
 })  
  
 //给class为inner的div的鼠标移入和鼠标移出事件绑定监听函数  
 // $('.inner')  
 // .mouseenter(function () {  
 // console.log('进入...')  
 // })  
 // .mouseleave(function () {  
 // console.log('离开...')  
 // })  
 // $('.inner')  
 // .on('mouseenter', function () {  
 // console.log('进入...')  
 // })  
 // .on('mouseleave', function () {  
 // console.log('离开...')  
 // })  
 $('.inner').hover(function () {  
 ***console***.log('进入...')  
 }, function () {  
 ***console***.log('离开...')  
 })  
  
 //点击btn1解除inner上的所有事件监听  
 $('#btn1').click(function () {  
 $('.inner').off()  
 })  
  
 //点击btn2解除inner上的mouseover事件  
 $('#btn2').click(function () {  
 $('.inner').off('mouseover')  
 })  
  
 //点击btn3得到事件坐标  
 $('#btn3').click(function (event) {  
 ***console***.log(event.offsetX, event.offsetY)  
 ***console***.log(event.clientX, event.clientY)  
 ***console***.log(event.pageX, event.pageY)  
 })  
  
 //点击.inner区域, 外部点击监听不响应  
 $('.inner').click(function (event) {  
 ***console***.log('click inner')  
 // 停止事件冒泡  
 event.stopPropagation()  
 })  
  
 //点击链接, 如果当前时间是偶数不跳转  
 $('#test4').click(function (event) {  
 var time = ***Date***.now()  
 ***console***.log(time);  
 if(time%2===0) {  
 // 阻止事件默认行为  
 event.preventDefault()  
 }  
 })  
 })  
  
</script>  
</body>  
</html>

·区别mouseover与mouseenter?

- mouseover

-- 在移入子元素时也会触发, 对应mouseout

- mouseenter

-- 只在移入当前元素时才触发, 对应mouseleave

- hover()使用的就是mouseenter()和mouseleave()

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>19\_事件切换</title>  
</head>  
<style type="text/css">  
 \* {  
 margin: 0px;  
 }  
 .div1 {  
 position: absolute;  
 width: 200px;  
 height: 200px;  
 top: 50px;  
 left: 10px;  
 background: olive;  
 }  
 .div2 {  
 position: absolute;  
 width: 100px;  
 height: 100px;  
 top: 50px;  
 background: red;  
 }  
 .div3 {  
 position: absolute;  
 width: 200px;  
 height: 200px;  
 top: 50px;  
 left: 230px;  
 background: olive;  
 }  
 .div4 {  
 position: absolute;  
 width: 100px;  
 height: 100px;  
 top: 50px;  
 background: yellow;  
 }  
  
 .divText{  
 position: absolute;  
 top: 330px;  
 left: 10px;  
 }  
  
</style>  
<body>  
  
<div class="divText">  
 区分鼠标的事件  
</div>  
  
<div class="div1">  
 div1.....  
 <div class="div2">div2....</div>  
</div>  
  
<div class="div3">  
 div3.....  
 <div class="div4">div4....</div>  
</div>  
  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
  
 $('.div1').mouseover(function () {  
 ***console***.log('移入div1或其子元素')  
 }).mouseout(function () {  
 ***console***.log('移出div1或其子元素')  
 })  
  
 $('.div3').mouseenter(function () {  
 ***console***.log('移入div3元素')  
 }).mouseleave(function () {  
 ***console***.log('移出div3元素')  
 })  
  
 $('.div3').hover(function () {  
 ***console***.log('移入div33元素')  
 this.style.background = 'red'  
 }, function () {  
 ***console***.log('移出div33元素')  
 this.style.background = 'blue'  
 })  
</script>  
</body>  
</html>

## 事件委托

·绑定事件监听的问题: 新加的元素没有监听

·事件委托

- 将多个后代元素的事件监听委托给祖先元素处理

- 监听回调是加在了祖先元素上

- 当操作任何一个后代元素时, 事件会冒泡到祖先元素

- 祖先元素不会直接处理事件, 而是根据event.target得到发生事件的后代元素, 通过这个后代元素调用事件回调函数

·事件委托的2方

- 委托方: 业主 后代元素

- 被委托方: 中介 祖先元素

·使用事件委托的好处

- 添加新的后代元素, 自动有事件响应处理

- 减少事件监听的数量: n==>1

·jQuery的事件委托API

- 设置事件委托: $(parentSelector).delegate(childrenSelector, eventName, callback)

//事件委托  
$('ul').delegate('li', 'click', function () {  
 //这里的this是点击的li  
 this.style.background = 'red'  
})

- 移除事件委托: $(parentSelector).undelegate(eventName)

$('#btn2').click(function () {  
 // 移除事件委托  
 $('ul').undelegate()  
})

## 添加删除记录增强版

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  
<html>  
  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
 <title>添加删除记录练习</title>  
 <link rel="stylesheet" type="text/css" href="css.css"/>  
</head>  
<body>  
  
<table id="employeeTable">  
 <tr>  
 <th>Name</th>  
 <th>Email</th>  
 <th>Salary</th>  
 <th>&nbsp;</th>  
 </tr>  
 <tr>  
 <td>Tom</td>  
 <td>tom@tom.com</td>  
 <td>5000</td>  
 <td><a href="deleteEmp?id=001">Delete</a></td>  
 </tr>  
 <tr>  
 <td>Jerry</td>  
 <td>jerry@sohu.com</td>  
 <td>8000</td>  
 <td><a href="deleteEmp?id=002">Delete</a></td>  
 </tr>  
 <tr>  
 <td>Bob</td>  
 <td>bob@tom.com</td>  
 <td>10000</td>  
 <td><a href="deleteEmp?id=003">Delete</a></td>  
 </tr>  
  
</table>  
  
<div id="formDiv">  
  
 <h4>添加新员工</h4>  
  
 <table>  
 <tr>  
 <td class="word">name:</td>  
 <td class="inp">  
 <input type="text" name="empName" id="empName"/>  
 </td>  
 </tr>  
 <tr>  
 <td class="word">email:</td>  
 <td class="inp">  
 <input type="text" name="email" id="email"/>  
 </td>  
 </tr>  
 <tr>  
 <td class="word">salary:</td>  
 <td class="inp">  
 <input type="text" name="salary" id="salary"/>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2" align="center">  
 <button id="addEmpButton" value="abc">  
 Submit  
 </button>  
 </td>  
 </tr>  
 </table>  
</div>  
  
<script type="text/javascript" src="jquery-1.10.1.js"></script>  
<script type="text/javascript">  
 /\*  
 1. 添加  
 2. 删除  
 \*/  
 $('#addEmpButton').click(function () {  
 //1. 收集输入的数据  
 var $empName = $('#empName')  
 var $email = $('#email')  
 var $salary = $('#salary')  
 var empName = $empName.val()  
 var email = $email.val()  
 var salary = $salary.val()  
  
 //2. 生成对应的<tr>标签结构, 并插入#employeeTable的tbody中  
 /\*  
 <tr>  
 <td>Bob</td>  
 <td>bob@tom.com</td>  
 <td>10000</td>  
 <td><a href="deleteEmp?id=003">Delete</a></td>  
 </tr>  
 \*/  
 var $xxx = $('<tr></tr>')  
 .append('<td>'+empName+'</td>') // 拼串  
 .append('<td>'+email+'</td>')  
 .append('<td>'+salary+'</td>')  
 .append('<td><a href="deleteEmp?id="'+***Date***.now()+'>Delete</a></td>')  
 .appendTo('#employeeTable>tbody')  
  
 //3. 清除输入  
 $empName.val('')  
 $email.val('')  
 $salary.val('')  
 })  
  
 // 通过table实现对所有a的click事件委托  
 $('#employeeTable').delegate('a', 'click', clickDelete)  
  
 /\*  
 点击删除的回调函数  
 \*/  
 function clickDelete () {  
 var $tr = $(this).parent().parent()  
 var name = $tr.children(':first').html()  
 if(confirm('确定删除'+name+'吗?')) {  
 $tr.remove()  
 }  
  
 return false  
 }  
  
</script>  
</body>  
</html>

# jQuery平滑翻页轮播图

<!DOCTYPE html>  
<html>  
  
<head>  
 <meta charset="UTF-8">  
 <title>焦点轮播图</title>  
 <style type="text/css">  
 /\*去除内边距,没有链接下划线\*/  
 \* {  
 margin: 0;  
 padding: 0;  
 text-decoration: none;  
 }  
  
 /\*让<body有20px的内边距\*/  
 body {  
 padding: 20px;  
 }  
  
 /\*最外围的div\*/  
 #container {  
 width: 600px;  
 height: 400px;  
 overflow: hidden;  
 position: relative; /\*相对定位\*/  
 margin: 0 auto;  
 }  
  
 /\*包含所有图片的<div>\*/  
  
 #list {  
 width: 4200px; /\*7张图片的宽: 7\*600 \*/  
 height: 400px;  
 position: absolute; /\*绝对定位\*/  
 z-index: 1;  
  
 }  
  
 /\*所有的图片<img>\*/  
 #list img {  
 float: left; /\*浮在左侧\*/  
 }  
  
 /\*包含所有圆点按钮的<div>\*/  
 #pointsDiv {  
 position: absolute;  
 height: 10px;  
 width: 100px;  
 z-index: 2;  
 bottom: 20px;  
 left: 250px;  
 }  
  
 /\*所有的圆点<span>\*/  
  
 #pointsDiv span {  
 cursor: pointer;  
 float: left;  
 border: 1px solid #fff;  
 width: 10px;  
 height: 10px;  
 border-radius: 50%;  
 background: #333;  
 margin-right: 5px;  
 }  
  
 /\*第一个<span>\*/  
  
 #pointsDiv .on {  
 background: orangered;  
 }  
  
 /\*切换图标<a>\*/  
  
 .arrow {  
 cursor: pointer;  
 display: none;  
 line-height: 39px;  
 text-align: center;  
 font-size: 36px;  
 font-weight: bold;  
 width: 40px;  
 height: 40px;  
 position: absolute;  
 z-index: 2;  
 top: 180px;  
 background-color: RGBA(0, 0, 0, 0.3);  
 color: #fff;  
 }  
  
 /\*鼠标移到切换图标上时\*/  
 .arrow:hover {  
 background-color: RGBA(0, 0, 0, 0.7);  
 }  
  
 /\*鼠标移到整个div区域时\*/  
 #container:hover .arrow {  
 display: block; /\*显示\*/  
 }  
  
 /\*上一个切换图标的左外边距\*/  
 #prev {  
 left: 20px;  
 }  
  
 /\*下一个切换图标的右外边距\*/  
 #next {  
 right: 20px;  
 }  
 </style>  
</head>  
  
<body>  
  
<div id="container">  
 <div id="list" style="left: -600px;">  
 <img src="img/5.jpg" alt="5"/>  
 <img src="img/1.jpg" alt="1"/>  
 <img src="img/2.jpg" alt="2"/>  
 <img src="img/3.jpg" alt="3"/>  
 <img src="img/4.jpg" alt="4"/>  
 <img src="img/5.jpg" alt="5"/>  
 <img src="img/1.jpg" alt="1"/>  
 </div>  
 <div id="pointsDiv">  
 <span index="1" class="on"></span>  
 <span index="2"></span>  
 <span index="3"></span>  
 <span index="4"></span>  
 <span index="5"></span>  
 </div>  
 <a href="javascript:;" id="prev" class="arrow">&lt;</a>  
 <a href="javascript:;" id="next" class="arrow">&gt;</a>  
</div>  
  
<script src="./js/jquery.1.10.2.js"></script>  
<script type="text/javascript" src="js/app.js"></script>  
</body>  
</html>

/\*  
 功能说明:  
 1. 点击向右(左)的图标, 平滑切换到下(上)一页  
 2. 无限循环切换: 第一页的上一页为最后页, 最后一页的下一页是第一页  
 3. 每隔3s自动滑动到下一页  
 4. 当鼠标进入图片区域时, 自动切换停止, 当鼠标离开后,又开始自动切换  
 5. 切换页面时, 下面的圆点也同步更新  
 6. 点击圆点图标切换到对应的页  
  
 bug: 快速点击下一页时, 有问题  
 \*/  
//绑定文档加载完成的监听  
$(function () {  
 //准备工作  
 //找到轮播内容包裹器  
 var $container = $("#container");  
 //找到图片内容  
 var $list = $("#list");  
 //找到切换按钮内容中的每个按钮  
 var $points = $("#pointsDiv>span");  
 //找到向左和向右的箭头  
 var $prev = $("#prev");  
 var $next = $("#next");  
 //每次翻页的偏移量  
 var PAGE\_WIDTH = 600;  
 //翻页的持续时间  
 var TIME = 400;  
 //单次移动的间隔时间  
 var ITRM\_TIME = 20;  
 //图片的数量  
 var imgCount = $points.length;  
 //保存当前图片下标  
 var index = 0;  
 //判断是否正在翻页,默认为false  
 var moving = false;  
  
 //点击向右(左)的图标, 平滑切换到下(上)一页  
 //为向右图标绑定点击事件  
 $next.click(function () {  
 //平滑翻到下一页  
 nextPage(true);  
 });  
 $prev.click(function () {  
 //平滑翻到上一页  
 nextPage(false);  
 });  
  
 //每隔3s自动滑动到下一页  
 var intervalId = setInterval(function () {  
 nextPage(true);  
 },1000);  
  
 //当鼠标进入图片区域时, 自动切换停止, 当鼠标离开后,又开始自动切换  
 $container.hover(function () {//鼠标移入时  
 //清除定时器  
 clearInterval(intervalId);  
 },function () {//鼠标移出时  
 //重新启动定时器  
 //每隔3s自动滑动到下一页  
 intervalId = setInterval(function () {  
 nextPage(true);  
 },1000);  
 });  
  
 //点击圆点图标切换到对应的页  
 //为所有圆点添加点击监听  
 $points.click(function () {  
 //获取点击的是第几个圆点  
 var targetIndex = $(this).index();  
 if (targetIndex!=index){//如果点击的圆点不为当前圆点  
 //切换到指定图片  
 nextPage(targetIndex);  
 }  
 });  
  
 //定义一个平滑翻页函数,如果是传入true,则翻到下一页,如果是false则翻到上一页  
 // 如果传入的是一个数值,则切换到指定下标的图片  
 function nextPage(next) {  
 //如果正在翻页,直接结束该方法  
 if (moving){  
 return;  
 }  
 //设为正在翻页  
 moving = true;  
 //总偏移量,如果是翻到下一页,则偏移量增加-600,否则增加600  
 var offset = 0;  
 //如果传入的是布尔值,则总偏移量为一页,可能是上一页或下一页  
 if (typeof next === "boolean"){  
 offset = next ? -PAGE\_WIDTH : PAGE\_WIDTH;  
 }else {//如果传入的是数值,则总偏移量为(next-index)\*PAGE\_WIDTH  
 //如果next-index为正数,则向右移动  
 //如果next-index为负数,则向左移动  
 offset = -(next-index)\*PAGE\_WIDTH;  
 }  
 //单次移动的偏移量  
 var itemoffset = offset/(TIME/ITRM\_TIME);  
 //获取图片元素在父元素的相对偏移  
 var currLeft = $list.position().left;  
 //目标偏移量  
 var targetLeft = currLeft + offset;  
  
 //启动循环定时器不断更新$list的lift,到达目标处时停止定时器  
 var intervalId = setInterval(function () {  
 //下一次移动的偏移量  
 currLeft += itemoffset  
  
 //无限循环切换: 第一页的上一页为最后页, 最后一页的下一页是第一页  
 //如果到达目标偏移量,则停止计时器  
 if(currLeft === targetLeft){  
 //清除定时器  
 clearInterval(intervalId);  
 if (currLeft <= -(imgCount+1)\*PAGE\_WIDTH){//如果目标偏移量为第7张图片的位置,说明切换到第一张,把偏移量瞬间切换为第一张图  
 currLeft = -PAGE\_WIDTH;  
 }else if(currLeft === 0){ //如果目标偏移量为0,说明切换到最后一张,把偏移量瞬间切换为第五张图  
 currLeft = -imgCount \* PAGE\_WIDTH;  
 }  
 //设为未翻页  
 moving = false;  
 }  
  
 //修改图片元素的偏移量  
 $list.css("left",currLeft);  
 },ITRM\_TIME);  
  
 //切换页面时, 下面的圆点也同步更新  
 //传入next,如果是下一页,则圆点是下一个,否则是上一个  
 updatePoints(next);  
 }  
  
 //更新圆点,传入next,如果是下一页,则圆点是下一个,否则是上一个  
 //如果传入的是数值,则切换到指定下标的圆点  
 function updatePoints(next) {  
 //计算出目标圆点的下标targetIndex[0,imgCount-1]  
 var targetIndex = 0;  
 //如果传入的是布尔值,则目标圆点为一个,可能是上一个或下一页=个  
 if (typeof next === "boolean"){  
 if (next){//如果切换到下一张图片  
 targetIndex = index + 1;  
 if (targetIndex === imgCount){//如果targetIndex等于图片数量  
 //将下标置为0  
 targetIndex = 0;  
 }  
 }else {//如果切换到上一张图片  
 targetIndex = index - 1;  
 if (targetIndex === -1){//如果targetIndex小于图片数量  
 //将下标置为imgCount-1  
 targetIndex = imgCount-1;  
 }  
 }  
 }else {//如果传入的是数值,则targetIndex为next  
 targetIndex = next;  
 }  
  
 //找到当前index的<span>,并将它的class移除  
 $points.eq(index).removeClass("on");  
 // $points[index].className = '';  
  
 //给目标圆点添加class="on"  
 $points.eq(targetIndex).addClass("on");  
 // $points[index].className = 'on';  
  
 //更新index的值为targetIndex  
 index = targetIndex;  
 }  
  
})

# jQuery内置动画

## 淡入淡出

·不断改变元素的透明度（opacity）来实现的

·fadeIn()

- 带动画的显示

- 第一个参数传入动画速度

-- fast: 200ms

-- normal: 400ms

-- slow: 600ms

-- 指定数值，如1000

- 第二个参数传入动画执行完成的方法

- 这两个参数可传可不传

- 其它两个方法也有这两个参数

//2. 点击 btn3 实现 慢慢淡入  
$('#btn2').click(function () {  
 $('.div1').fadeIn()  
})

·fadeOut()

- 带动画的隐藏

- 可以传入动画速度

//1. 点击 btn1 实现 慢慢淡出  
$('#btn1').click(function () {  
 // $('.div1').fadeOut()//正常淡出  
 // $('.div1').fadeOut('slow')//缓慢淡出  
 // $('.div1').fadeOut(1000)//指定时间淡出  
 $('.div1').fadeOut('fast',function () {  
 alert("动画完成了");  
 })//快速淡出  
})

·fadeToggle()

- 带动画切换显示/隐藏

//3. 点击 btn3 实现 淡出/淡入切换  
$('#btn3').click(function () {  
 $('.div1').fadeToggle(function () {  
 alert('动画结束了')  
 })  
})

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>21\_淡入淡出</title>  
</head>  
<style type="text/css">  
 \* {  
 margin: 0px;  
 }  
  
 .div1 {  
 position: absolute;  
 width: 200px;  
 height: 200px;  
 top: 50px;  
 left: 10px;  
 background: red;  
 }  
</style>  
  
<body>  
<button id="btn1">慢慢淡出</button>  
<button id="btn2">慢慢淡入</button>  
<button id="btn3">淡出/淡入切换</button>  
  
<div class="div1">  
</div>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 $(function () {  
 //1. 点击 btn1 实现 慢慢淡出  
 $('#btn1').click(function () {  
 // $('.div1').fadeOut()//正常淡出  
 // $('.div1').fadeOut('slow')//缓慢淡出  
 // $('.div1').fadeOut(1000)//指定时间淡出  
 $('.div1').fadeOut('fast',function () {  
 alert("动画完成了");  
 })//快速淡出  
 })  
  
 //2. 点击 btn3 实现 慢慢淡入  
 $('#btn2').click(function () {  
 $('.div1').fadeIn()  
 })  
  
 //3. 点击 btn3 实现 淡出/淡入切换  
 $('#btn3').click(function () {  
 $('.div1').fadeToggle(function () {  
 alert('动画结束了')  
 })  
 })  
 })  
</script>  
</body>  
</html>

## 滑动

·不断改变元素的高度实现

·slideDown()

- 带动画的展开

- 第一个参数传入动画速度

-- fast: 200ms

-- normal: 400ms

-- slow: 600ms

-- 指定数值，如1000

- 第二个参数传入动画执行完成的方法

- 这两个参数可传可不传

- 其它两个方法也有这两个参数

//2. 点击 btn3 实现 向下滑动  
$('#btn2').click(function () {  
 $('div').slideDown()  
})

·slideUp()

- 带动画的收缩

//1. 点击 btn1 实现 向上滑动  
$('#btn1').click(function () {  
 $('div').slideUp(1000,function () {  
 alert("动画完成了");  
 })  
})

·slideToggle()

- 带动画的切换展开/收缩

//3. 点击 btn3 实现 向上/向下切换  
$('#btn3').click(function () {  
 $('div').slideToggle(2000)  
})

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>22\_滑动</title>  
</head>  
<style type="text/css">  
 \* {  
 margin: 0px;  
 }  
  
 .div1 {  
 position: absolute;  
 width: 200px;  
 height: 200px;  
 top: 50px;  
 left: 10px;  
 background: red;  
 }  
</style>  
<body>  
<button id="btn1">慢慢收缩</button>  
<button id="btn2">慢慢展开</button>  
<button id="btn3">收缩/展开切换</button>  
  
<div class="div1">  
</div>  
  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
  
 $(function () {  
 //1. 点击 btn1 实现 向上滑动  
 $('#btn1').click(function () {  
 $('div').slideUp(1000,function () {  
 alert("动画完成了");  
 })  
 })  
  
 //2. 点击 btn3 实现 向下滑动  
 $('#btn2').click(function () {  
 $('div').slideDown()  
 })  
  
 //3. 点击 btn3 实现 向上/向下切换  
 $('#btn3').click(function () {  
 $('div').slideToggle(2000)  
 })  
 })  
  
</script>  
</body>  
</html>

## 显示与隐藏

·显示或隐藏某个元素，默认没有动画

·改变宽高和透明度，左上角是固定的

·show()

- (不)带动画的显示

- 如果传入参数则会带动画

- 第一个参数传入动画速度

-- fast: 200ms

-- normal: 400ms

-- slow: 600ms

-- 指定数值，如1000

- 第二个参数传入动画执行完成的方法

- 这两个参数可传可不传

- 其它两个方法也有这两个参数

//1. 点击btn1, 立即显示  
$('#btn1').click(function () {  
 $('div').show()  
})

//2. 点击btn2, 慢慢显示  
$('#btn2').click(function () {  
 $('div').show(1000,function () {  
 alert("动画完成");  
 })  
})

·hide()

- (不)带动画的隐藏

//3. 点击btn3, 慢慢隐藏  
$('#btn3').click(function () {  
 $('div').hide(2000)  
})

·toggle()

- (不)带动画的切换显示/隐藏

//4. 点击btn4, 切换显示/隐藏  
$('#btn4').click(function () {  
 $('div').toggle()  
})

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>23\_显示与隐藏</title>  
</head>  
<style type="text/css">  
 \* {  
 margin: 0px;  
 }  
  
 .div1 {  
 position: absolute;  
 width: 200px;  
 height: 200px;  
 top: 50px;  
 left: 10px;  
 background: red;  
 display: none;  
 }  
</style>  
<body>  
<button id="btn1">瞬间显示</button>  
<button id="btn2">慢慢显示</button>  
<button id="btn3">慢慢隐藏</button>  
<button id="btn4">显示隐藏切换</button>  
  
<div class="div1">  
</div>  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
 $(function () {  
 //1. 点击btn1, 立即显示  
 $('#btn1').click(function () {  
 $('div').show()  
 })  
  
 //2. 点击btn2, 慢慢显示  
 $('#btn2').click(function () {  
 $('div').show(1000,function () {  
 alert("动画完成");  
 })  
 })  
  
 //3. 点击btn3, 慢慢隐藏  
 $('#btn3').click(function () {  
 $('div').hide(2000)  
 })  
  
 //4. 点击btn4, 切换显示/隐藏  
 $('#btn4').click(function () {  
 $('div').toggle()  
 })  
 })  
</script>  
</body>  
</html>

## 自定义动画

·jQuery动画本质

- 在指定时间内不断改变元素样式值来实现的

·animate()

- 自定义动画效果的动画

- 第一个参数为css样式对象

- 第二个参数为动画时长

- 第三个参数为动画完成后的回调函数

- 第二个和第三个函数可传可不传

//宽/高都扩大为200px,宽度和高度同时变化  
***$div1*** .animate({  
 width: 200,  
 height: '200px'  
 }, 1000,function () {  
 alert("动画完成");  
 });

//宽先扩为200px, 高后扩为200px  
***$div1*** .animate({  
 width: 200  
 }, 2000)  
 .animate({  
 height: 200  
 }, 2000)

//移动到(100, 20)处  
***$div1*** .animate({  
 left: 100,  
 top: 20  
 }, 1000)

//移动距离为(-100, -20)  
***$div1***.animate({  
 left: '-=100',  
 top: '-=20'  
}, 1000)

·stop()

- 停止动画

$('#btn4').click(function () {  
 ***$div1***.stop() // 只停止当前运行的动画(后面其它动画还会运行)  
 // $div1.stop(true) // 停止所有动画  
 // $div1.stop(true, true)  
})

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>24\_自定义动画</title>  
 <style type="text/css">  
 \* {  
 margin: 0px;  
 }  
  
 .div1 {  
 position: absolute;  
 width: 100px;  
 height: 100px;  
 top: 50px;  
 left: 300px;  
 background: red;  
 }  
 </style>  
</head>  
<body>  
<button id="btn1">逐渐扩大</button>  
<button id="btn2">移动到指定位置</button>  
<button id="btn3">移动指定距离</button>  
<button id="btn4">停止动画</button>  
  
<div class="div1">  
 爱在西元前，学在尚硅谷  
</div>  
  
<script src="js/jquery-1.10.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
  
 var ***$div1*** = $('.div1')  
  
 $('#btn1').click(function () {  
 //宽/高都扩大为200px,宽度和高度同时变化  
 // $div1  
 // .animate({  
 // width: 200,  
 // height: '200px'  
 // }, 1000,function () {  
 // alert("动画完成");  
 // });  
  
 //宽先扩为200px, 高后扩为200px  
 ***$div1*** .animate({  
 width: 200  
 }, 2000)  
 .animate({  
 height: 200  
 }, 2000)  
 })  
   
 $('#btn2').click(function () {  
  
 // 1).移动到(500, 100)处  
 /\*$div1  
 .animate({  
 left: 500,  
 top: 100  
 }, 1000)\*/  
 // 2).移动到(100, 20)处  
 ***$div1*** .animate({  
 left: 100,  
 top: 20  
 }, 1000)  
 })  
 $('#btn3').click(function () {  
 // 1). 移动距离为(100, 50)  
 /\*$div1.animate({  
 left: '+=100',  
 top: '+=50'  
 }, 1000)\*/  
  
 //移动距离为(-100, -20)  
 ***$div1***.animate({  
 left: '-=100',  
 top: '-=20'  
 }, 1000)  
 })  
  
 /\*  
 4. 停止动画  
 \*/  
 $('#btn4').click(function () {  
 ***$div1***.stop() // 只停止当前运行的动画(后面其它动画还会运行)  
 // $div1.stop(true) // 停止所有动画  
 // $div1.stop(true, true)  
 })  
</script>  
</body>  
</html>

## 导航条下拉菜单动画

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>04\_导航动态显示效果</title>  
 <script src="jquery-1.10.1.js"></script>  
 <style rel="stylesheet">  
 \* {  
 margin: 0;  
 padding: 0;  
 word-wrap: break-word;  
 word-break: break-all;  
 }  
  
 body {  
 background: #FFF;  
 color: #333;  
 font: 12px/1.6em Helvetica, Arial, sans-serif;  
 }  
  
 a {  
 color: #0287CA;  
 text-decoration: none;  
 }  
  
 a:hover {  
 text-decoration: underline;  
 }  
  
 ul, li {  
 list-style: none;  
 }  
  
 img {  
 border: none;  
 }  
  
 h1, h2, h3, h4, h5, h6 {  
 font-size: 1em;  
 }  
  
 html {  
 overflow: -moz-scrollbars-vertical; /\* 在Firefox下始终显示滚动条\*/  
 }  
  
 #navigation {  
 width: 784px;  
 padding: 8px;  
 margin: 8px auto;  
 background: #3B5998;  
 height: 18px;  
 }  
  
 #navigation ul li {  
 float: left;  
 margin-right: 14px;  
 position: relative;  
 z-index: 100;  
 }  
  
 #navigation ul li a {  
 display: block;  
 padding: 0 8px;  
 background: #EEEEEE;  
 font-weight: 700;  
 }  
  
 #navigation ul li a:hover {  
 background: none;  
 color: #fff;  
 }  
  
 #navigation ul li ul {  
 background-color: #88C366;  
 position: absolute;  
 width: 80px;  
 overflow: hidden;  
 display: none;  
 }  
  
 #navigation ul li ul li {  
 border-bottom: 1px solid #BBB;  
 text-align: left;  
 width: 100%;  
 }  
 </style>  
</head>  
<body>  
<div id="navigation">  
 <ul>  
 <li><a href="#">首 页</a></li>  
 <li>  
 <a href="#">衬 衫</a>  
 <ul>  
 <li><a href="#">短袖衬衫</a></li>  
 <li><a href="#">长袖衬衫</a></li>  
 <li><a href="#">无袖衬衫</a></li>  
 </ul>  
 </li>  
 <li>  
 <a href="#">卫 衣</a>  
 <ul>  
 <li><a href="#">开襟卫衣</a></li>  
 <li><a href="#">套头卫衣</a></li>  
 </ul>  
 </li>  
 <li>  
 <a href="#">裤 子</a>  
 <ul>  
 <li><a href="#">休闲裤</a></li>  
 <li><a href="#">卡其裤</a></li>  
 <li><a href="#">牛仔裤</a></li>  
 <li><a href="#">短裤</a></li>  
 </ul>  
 </li>  
 <li><a href="#">联系我们</a></li>  
 </ul>  
</div>  
  
<script>  
 //找到navigation的子元素下ul的子元素li元素,并且该li元素有ul子元素  
 var ***$li*** = $("#navigation>ul>li:has(ul)");  
 //为$li添加鼠标移入移出监听事件  
 ***$li***.hover(function () {//移入时  
 //动画展开  
 //查找移入li的ul子元素  
 var $ul = $(this).children("ul");  
 //先停止当前动画  
 $ul.stop();  
 //为$ul添加展开动画  
 $ul.slideDown();  
 },function () {//移出时  
 //动画收缩  
 //查找移入li的ul子元素  
 var $ul = $(this).children("ul");  
 //先停止当前动画  
 $ul.stop();  
 //为$ul添加展开动画  
 $ul.slideUp();  
 })  
</script>  
</body>  
</html>

# 多库共存

·问题

- 如果有2个库都有$, 就存在冲突

·解决

- jQuery库可以释放$的使用权, 让另一个库可以正常使用, 此时jQuery库只能使用jQuery了

API

- jQuery.noConflict()

-- 释放$的使用权,让另一个库可以正常使用

<script src="js/myLib.js" type="text/javascript"></script>  
<script src="js/jquery-1.10.1.js" type="text/javascript" charset="utf-8"></script>  
<script type="text/javascript">  
 //释放$的使用权,让另一个库可以正常使用  
 ***jQuery***.noConflict()  
  
 //使用的是myLib.js中的$  
 $()  
  
 //使用jQuery  
 jQuery(function () {  
 jQuery('body')  
 })  
</script>

## 区别window.onload与 $(document).ready()

·window.onload

- 包括页面的图片加载完后才会回调(晚)

- 只能有一个监听回调

//页面和其他数据(图片)加载完才执行  
***window***.onload = function () {  
 ***console***.log('onload', ***$logo***.width())//190  
}

·$(document).ready()

- 等同于: $(function(){})

- 页面加载完就回调(早)

- 可以有多个监听回调

//在 $(function(){}) 中 打印 img 的宽度  
//页面加载完才执行  
$(function () {  
 ***console***.log('ready', ***$logo***.width())//0  
})  
$(function () {  
 ***console***.log('$2()', ***$logo***.width())  
})

·浏览器会对图片等数据进行自动缓存

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>window.onload与$(document).ready()</title>  
</head>  
<body>  
<h1>测试window.onload与$(document).ready()</h1>  
<img id="logo" src="https://gss0.bdstatic.com/5bVWsj\_p\_tVS5dKfpU\_Y\_D3/res/r/image/2017-05-19/6fec71d56242b74eb24b4ac80b817eac.png">  
  
<script type="text/javascript" src="js/jquery-1.10.1.js"></script>  
<script type="text/javascript">  
 //获取logo元素  
 var ***$logo*** = $('#logo')  
 //直接打印img的宽度，观察其值  
 //立即执行  
 ***console***.log("直接" + ***$logo***.width()) // 0  
 //在 $(function(){}) 中 打印 img 的宽度  
 //页面加载完才执行  
 $(function () {  
 ***console***.log('ready', ***$logo***.width())//0  
 })  
 $(function () {  
 ***console***.log('$2()', ***$logo***.width())  
 })  
 // $(function () {})完整的写法  
 // $(document).ready(function () {  
 //  
 // })  
  
 // window.onload = function () {  
 // console.log('onload1()')  
 // }  
 //页面和其他数据(图片)加载完才执行  
 ***window***.onload = function () {  
 ***console***.log('onload', ***$logo***.width())//190  
 }  
  
 //当前元素加载完成时才执行  
 ***$logo***.on('load', function () {  
 ***console***.log('imgload', this.width)//190  
 })  
</script>  
</body>  
</html>

# jQuery插件

## 自定义jQuery插件

·扩展jQuery的工具方法

- $.extend(object)

- 扩展

// 扩展jQuery的工具方法 : $.extend(object)  
// min(a, b) : 返回较小的值  
// max(c, d) : 返回较大的值  
// leftTrim() : 去掉字符串左边的空格  
// rightTrim() : 去掉字符串右边的空格  
//扩展$   
$.extend({  
 min: function (a, b) {  
 return (a < b) ? a : b  
 },  
 max: function (a, b) {  
 return (a > b) ? a : b  
 },  
 leftTrim: function (strToBeTrimed) {  
 //^是开头,\s是空格,+是一个或多个,替换为空串  
 return strToBeTrimed.replace(/^\s+/, '')  
 },  
 rightTrim: function (strToBeTrimed) {  
 //\s是空格,+是一个或多个,$是结尾,替换为空串  
 return strToBeTrimed.replace(/\s+$/, '')  
 }  
})

- 使用

//使用jQuery的工具方法  
// 得到最大最小值  
var ***a*** = 1  
var ***b*** = 2  
var ***result\_min*** = $.min(***a***, ***b***)  
var ***result\_max*** = $.max(***a***, ***b***)  
***console***.log(***result\_min***)  
***console***.log(***result\_max***)  
  
// 左trim 右trim  
var ***str*** = ' 悟空 '  
***console***.log('|' + ***str*** + '|')  
var ***resultStrLeft*** = $.leftTrim(***str***)  
***console***.log('|' + ***resultStrLeft*** + '|')  
var ***resultStrRight*** = $.rightTrim(***str***)  
***console***.log('|' + ***resultStrRight*** + '|')

·扩展jQuery对象的方法

- $.fn.extend(object)

- 扩展

//扩展jQuery对象的方法  
//扩展 $('#id').XXXXX  
//$.fn.extend(object)  
// checkAll() : 全选  
// unCheckAll() : 全不选  
// reverseCheck() : 全反选  
$.fn.extend({  
 checkAll: function () {  
 // console.log('checkAll')  
 this.prop('checked', true);  
 },  
 unCheckAll: function () {  
 this.prop('checked', false)  
 },  
 reverseCheck: function () {  
 this.each(function () {//这里的this是jQuery对象  
 this.checked = !this.checked;//这里的this是dom元素  
 })  
 }  
})

- 使用

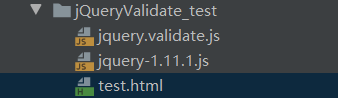
//全选  
$('#checkedAllBtn').click(function () {  
 $(':checkbox').checkAll()  
})  
  
//全不选  
$('#checkedNoBtn').click(function () {  
 $(':checkbox').unCheckAll()  
})  
  
//反选  
$('#reverseCheckedBtn').click(function () {  
 $(':checkbox').reverseCheck()  
})

## jquery-validation

·表单验证插件

·到jQuery官网下载该插件

·引入到项目



<script type="text/javascript" src="jquery-1.11.1.js"></script>  
<script type="text/javascript" src="jquery.validate.js"></script>

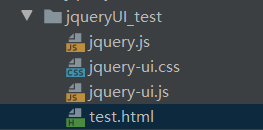
·使用

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <style>  
 /\*修改提示文本的样式\*/  
 .error {  
 color: #F00;  
 }  
 </style>  
</head>  
<body>  
<!--  
 required: "必填字段",  
 remote: "请修改此字段",  
 email: "请输入有效的电子邮件地址。",  
 url: "请输入有效的URL。",  
 date: "请输入有效日期。",  
 dateISO: "请输入有效日期（ISO）。",  
 number: "请输入有效数字。",  
 digits: "只能输入数字。",  
 equalTo: "请重新输入相同的值。",  
 maxlength: $.validator.format( "请输入不超过x个字符。" ),  
 minlength: $.validator.format( "请至少输入x个字符。" ),  
 rangelength: $.validator.format( "请输入n~m个字符。" ),  
 range: $.validator.format( "请输入一个介于n和m之间的值。" ),  
 max: $.validator.format( "请输入一个小于或等于x的值。" ),  
 min: $.validator.format( "请输入一个大于或等于x的值。" ),  
 step: $.validator.format( "请输入x的倍数。" )  
-->  
<form id="myForm" action="xxx">  
 <p>用户名(必须的, 最小2位): <input name="name" type="text" required minlength="2"></p>  
 <p>密码(必须的, 6到8位): <input id="password" name="pwd" type="password" required minlength="6" maxlength="8"></p>  
 <p>确认密码(与密码相同): <input name="pwd2" type="password" equalTo="#password"></p>  
 <input class="submit" type="submit" value="提交">  
</form>  
  
<script type="text/javascript" src="jquery-1.11.1.js"></script>  
<script type="text/javascript" src="jquery.validate.js"></script>  
<script type="text/javascript">  
 /\*  
 声明式验证  
 \*/  
 $(function () {  
 $('#myForm').validate({  
 messages: {  
 name: {  
 required: '用户名不能为空',  
 minlength: '用户名的最小长度是2'  
 },  
 pwd: {  
 required: '密码不能为空',  
 minlength: '密码的最小长度是6',  
 maxlength: '密码的最大长度是8'  
 },  
 pwd2: {  
 equalTo: '确认密码必须与密码相同'  
 }  
 }  
 })  
 })  
</script>  
</body>  
</html>

## jquery UI

·到jQuery官网下载该插件

·引入到项目



<link rel="stylesheet" href="jquery-ui.css">

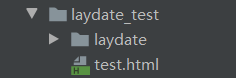
<script type="text/javascript" src="jquery.js"></script>  
<script type="text/javascript" src="jquery-ui.js"></script>

·使用

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>测试jQuery-UI的基本使用</title>  
 <link rel="stylesheet" href="jquery-ui.css">  
 <style type="text/css">  
 /\*修改原有的样式\*/  
 .xiugai div{  
 background-color: #0D8BBD;  
 }  
 </style>  
</head>  
<body>  
  
<!--1. Accordion: 手风琴效果-->  
<h2>1. Accordion: 手风琴效果</h2>  
<div id="accordion" class="xiugai">  
 <h3>First</h3>  
 <div>Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.</div>  
 <h3>Second</h3>  
 <div>Phasellus mattis tincidunt nibh.</div>  
 <h3>Third</h3>  
 <div>Nam dui erat, auctor a, dignissim quis.</div>  
</div>  
  
<!-- 2. Autocomplete: 自动完成搜索输入框 -->  
<h2>2. Autocomplete: 自动完成搜索输入框</h2>  
<div>  
 <input id="autocomplete" title="type &quot;a&quot;">  
</div>  
  
<!--3. Tabs: 选项卡-->  
<h2>3. Tabs: 选项卡</h2>  
<div id="tabs">  
 <ul>  
 <li><a href="#tabs-1">First</a></li>  
 <li><a href="#tabs-2">Second</a></li>  
 <li><a href="#tabs-3">Third</a></li>  
 </ul>  
 <div id="tabs-1">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</div>  
 <div id="tabs-2">Phasellus mattis tincidunt nibh. Cras orci urna, blandit id, pretium vel, aliquet ornare, felis. Maecenas scelerisque sem non nisl. Fusce sed lorem in enim dictum bibendum.</div>  
 <div id="tabs-3">Nam dui erat, auctor a, dignissim quis, sollicitudin eu, felis. Pellentesque nisi urna, interdum eget, sagittis et, consequat vestibulum, lacus. Mauris porttitor ullamcorper augue.</div>  
</div>  
  
  
<script type="text/javascript" src="jquery.js"></script>  
<script type="text/javascript" src="jquery-ui.js"></script>  
<script type="text/javascript">  
  
 // 1. Accordion: 手风琴效果  
 $('#accordion').accordion()  
  
 // 2. Autocomplete: 自动完成搜索输入框  
 var ***availableTags*** = [  
 "ActionScript",  
 "AppleScript",  
 "Asp",  
 "BASIC",  
 "C",  
 "C++",  
 "Clojure",  
 "COBOL",  
 "ColdFusion",  
 "Erlang",  
 "Fortran",  
 "Groovy",  
 "Haskell",  
 "Java",  
 "JavaScript",  
 "Lisp",  
 "Perl",  
 "PHP",  
 "Python",  
 "Ruby",  
 "Scala",  
 "Scheme"  
 ]  
 $( "#autocomplete" ).autocomplete({  
 source: ***availableTags*** })  
  
 // 3. Tabs: 选项卡  
 $('#tabs').tabs()  
</script>  
  
</body>  
</html>

## laydete

·引入到项目



<script src="laydate/laydate.js"></script>

·使用

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>laydate测试</title>  
 <style>  
 li {  
 list-style: none;  
 }  
 </style>  
 <script src="laydate/laydate.js"></script>  
</head>  
<body>  
  
<input placeholder="请输入日期" class="laydate-icon" onclick="laydate()">  
<br>  
<br>  
  
<input class="laydate-icon" id="demo" value="2014-6-25">  
<script type="text/javascript">  
 laydate({  
 //为id为demo的元素加监听  
 elem: '#demo'  
 })  
</script>  
<br>  
<br>  
  
开始日：<li class="laydate-icon" id="start" style="width:200px; margin-right:10px;"></li>  
结束日：<li class="laydate-icon" id="end" style="width:200px;"></li>  
<script>  
 //配置对象  
 var ***start*** = {  
 //为某个元素加监听  
 elem: '#start',  
 //设置日期格式  
 format: 'YYYY/MM/DD hh:mm:ss',  
 min: laydate.now(), //设定最小日期为当前日期  
 max: '2099-06-16 23:59:59', //最大日期  
 istime: true,  
 istoday: false,  
 choose: function (datas) {  
 ***end***.min = datas; //开始日选好后，重置结束日的最小日期  
 ***end***.start = datas //将结束日的初始值设定为开始日  
 }  
 }  
 var ***end*** = {  
 elem: '#end',  
 format: 'YYYY/MM/DD hh:mm:ss',  
 min: laydate.now(),  
 max: '2099-06-16 23:59:59',  
 istime: true,  
 istoday: false,  
 choose: function (datas) {  
 ***start***.max = datas; //结束日选好后，重置开始日的最大日期  
 }  
 }  
 laydate(***start***)  
 laydate(***end***)  
</script>  
</body>  
</html>

# 京东仿写

·有bug先检查id选择器有没有加#号

·jQuery对象需要通过attr()获取和修改标签属性值

·jQuery对象需要通过css()获取和修改css属性值

·只有图片加载完成时，才能获取图片的大小

·所有源码和图片请在百度云查看

链接：https://pan.baidu.com/s/1ncrdFwjR\_Y731VdIDubu4Q

提取码：523k

/\*  
 1. 鼠标移入显示,移出隐藏  
 目标: 手机京东, 客户服务, 网站导航, 我的京东, 去购物车结算, 全部商品  
 2. 鼠标移动切换二级导航菜单的切换显示和隐藏  
 3. 输入搜索关键字, 列表显示匹配的结果  
 4. 点击显示或者隐藏更多的分享图标  
 5. 鼠标移入移出切换地址的显示隐藏  
 6. 点击切换地址tab  
  
 7. 鼠标移入移出切换显示迷你购物车  
 8. 点击切换产品选项 (商品详情等显示出来)  
  
 9. 点击向右/左, 移动当前展示商品的小图片  
 10. 当鼠标悬停在某个小图上,在上方显示对应的中图  
 11. 当鼠标在中图上移动时, 显示对应大图的附近部分区域  
 \*/  
  
$(function () {  
 //鼠标移入显示,移出隐藏  
 //目标: 手机京东, 客户服务, 网站导航, 我的京东, 去购物车结算, 全部商品  
 //设计思想:  
 //将需要进行移入移出监听的元素的name属性值设为相同,然后找到他们设置hover监听  
 //将需要进行移入移出监听的元素id属性值设为唯一值  
 //将要显示/隐藏元素的id值为 移入/移出元素的id+"\_items"  
 //找到移入移出监听的元素id属性值  
 //拼接字符串 移入/移出元素的id+"\_items"  
 //通过拼接的字符串找到要显示/隐藏元素  
 //将要显示/隐藏元素调用show()或hide()方法  
 showhide()  
 //鼠标移入显示,移出隐藏所需内容的函数  
 function showhide() {  
 /\*  
 对哪些元素绑定什么监听?  
 对哪些元素进行什么DOM操作  
 \*/  
 //找出要进行移入移出监听的元素,并给它们设置hover监听  
 $("[name = show\_hide]").hover(function () {//鼠标移入时  
 //要显示/隐藏元素的id值为 移入元素的id+"\_items"  
 //找到移入元素的id值,并+"\_items"  
 var id = this.id + "\_items";  
 // alert(id);  
 //找到要显示/隐藏的元素,并显示该元素  
 $("#"+id).show();  
 },function () {//鼠标移出时  
 //要显示/隐藏元素的id值为 移入元素的id+"\_items"  
 //找到移入元素的id值,并+"\_items"  
 var id = this.id + "\_items";  
 //找到要显示/隐藏的元素,并隐藏该元素  
 $("#"+id).hide();  
 })  
 }  
  
 //鼠标移动切换二级导航菜单的切换显示和隐藏  
 //设计思想:  
 //将需要进行移入移出监听的元素的class属性值设为相同,然后找到他们设置hover监听  
 //将要显示/隐藏元素的class属性值设为相同  
 //通过class找到要显示/隐藏元素  
 //然后调用show()或hide()方法  
 hoverSubMenu();  
 function hoverSubMenu() {  
 //找出要进行移入移出监听的元素,并给它们设置hover监听  
 // $("#category\_items>div").hover(function () {  
 // console.log(this+"移入");  
 // },function () {  
 // console.log(this+"移出");  
 // })  
 $(".cate\_item").hover(function () {  
 // console.log(this+"移入");  
 //找到要显示/隐藏的元素,并显示该元素  
 // $(this).children(":last").show();  
 $(this).children(".sub\_cate\_box").show();  
 },function () {  
 // console.log(this+"移出");  
 //找到要显示/隐藏的元素,并隐藏该元素  
 // $(this).children(":last").hide();  
 $(this).children(".sub\_cate\_box").hide();  
 })  
 }  
  
 //输入搜索关键字, 列表显示匹配的结果  
 //找到输入框,并设置获取焦点和键盘响应事件,还有失去焦点事件  
 search();  
 function search() {  
 var $txtSearch = $("#txtSearch");  
 $txtSearch.on("keyup focus",function () {  
 //找到输入框,并设置获取焦点和键盘响应事件  
 //如果输入框有文本才显示列表  
 //获取输入框文本  
 var txt = this.value.trim();  
 if (txt){//如果输入框文本不为空  
 //显示列表  
 $("#search\_helper").show();  
 }  
 });  
 //失去焦点事件  
 $txtSearch.blur(function () {  
 //隐藏列表  
 $("#search\_helper").hide();  
 })  
 }  
  
 //点击显示或者隐藏更多的分享图标  
 //为更多图标绑定点击事件  
 //要改变的元素:父元素宽度width,要显示的两个a标签的display,更多图标的class  
 shareMore();  
 function shareMore() {  
 var isOpen = false;//标识当前分享更多的状态,初始值为关闭  
 var $shareMore = $("#shareMore");//获取更多图标按钮  
 var $parent = $shareMore.parent();//获取图标按钮的父元素  
 //获取要显示的两个a标签  
 var $as = $shareMore.prevAll("a:lt(2)");//图标按钮前面的兄弟元素,并且是a便签,下标小于2,prevAll是从下往上找的,离$shareMore越近的元素,下标越小  
 //获取更多图标按钮的子元素  
 var $b = $shareMore.children();  
  
 //为更多图标绑定点击事件  
 $shareMore.click(function () {  
 if (isOpen){//如果是打开的  
 //更改父元素宽度width  
 $parent.css("width",155);  
 //隐藏两个a标签  
 $as.hide();  
 //移除更多图标的class中的的backword  
 $b.removeClass("backword");  
 }else {//如果是关闭的  
 //更改父元素宽度width  
 $parent.css("width",200);  
 //显示两个a标签  
 $as.show();  
 //增加更多图标的class  
 $b.addClass("backword");  
 }  
 //更改标识当前分享更多的状态  
 isOpen = !isOpen;  
 })  
 }  
  
 //鼠标移入移出切换地址的显示隐藏  
 //为配送地址栏绑定hover事件  
 //为关闭按钮绑定点击事件  
 //要更改的元素,地址列表,关闭按钮  
 address();  
 function address() {  
 //找到配送地址选项栏  
 var $store\_select = $("#store\_select");  
 //为配送地址栏绑定hover事件  
 $store\_select.hover(function () {  
 //找到store\_select的子元素,并且下标大于0,显示这两个元素  
 $(this).children(":gt(0)").show();  
 },function () {  
 //找到store\_select的子元素,并且下标大于0,隐藏这两个元素  
 $(this).children(":gt(0)").hide();  
 });  
 //找到配送地址选项栏的最后一个子元素,绑定点击监听事件  
 $store\_select.children(":last").click(function () {  
 //找到store\_select的子元素,并且下标大于0,隐藏这两个元素  
 // $(this).parent().children(":gt(0)").hide();  
 $store\_select.children(":gt(0)").hide();  
 });  
 }  
  
 //点击切换地址tab  
 //为三个tab按钮绑定点击事件  
 //更改li子元素的class属性值  
 clickTab();  
 function clickTab() {  
 //找到三个tab按钮  
 var $lis = $("#store\_tabs>li");  
 // 为三个tab按钮绑定点击事件  
 $lis.click(function () {  
 //清除所有li的class为hover的属性值  
 $lis.removeClass("hover");  
 //给点击的li加上hover样式  
 this.className = "hover";  
 })  
 }  
  
 //鼠标移入移出切换显示迷你购物车  
 //为加入购物车按钮绑定hover  
 //为加入购物车按钮添加minicart样式  
 //将其子元素的div元素设置为显示状态  
 hoverMiniCart()  
 function hoverMiniCart() {  
 //为加入购物车按钮绑定hover  
 $("#minicart").hover(function () {  
 //为加入购物车按钮添加minicart样式  
 this.className = "minicart";  
 //将其子元素的div元素设置为显示状态  
 $(this).children("div").show();  
 },function () {  
 //为加入购物车按钮移除minicart样式  
 this.className = "";  
 //将其子元素的div元素设置为隐藏状态  
 $(this).children("div").hide();  
 })  
 }  
  
 //点击切换产品选项 (商品详情等显示出来)  
 //获取5个li产品选项,绑定点击事件  
 //清除所有li的class样式  
 //更改点击li的class样式  
 //隐藏所有内容  
 //显示点击li的指定内容  
 clickProductTabs();  
 function clickProductTabs() {  
 //获取5个li产品选项  
 var $lis = $("#product\_detail>ul>li");  
 //为5个li产品选项绑定点击事件  
 $lis.click(function () {  
 //清除所有li的class样式  
 $lis.removeClass("current");  
 //为当前点击li添加class样式  
 this.className = 'current';  
 //找到所有的内容  
 var $contents = $("#product\_detail>div:gt(0)");  
 //隐藏所有内容  
 $contents.hide();  
 //获取当前点击li的下标  
 var index = $(this).index();  
 //找到点击li的指定内容并显示  
 $contents.eq(index).show();  
 //原生方法  
 // $contents[index].display = "block";  
 })  
 }  
  
 //点击向右/左, 移动当前展示商品的小图片  
 //为向左/右按钮绑定点击监听事件  
 //改变左/右按钮的class样式  
 //改变存放图片的ul中的left属性  
 moveMiniImg();  
 function moveMiniImg() {  
 //找到向左/右按钮的a标签  
 var $as = $("#preview>h1>a");  
 //获取向左按钮  
 var $backward = $as.first();  
 //获取向右按钮  
 var $forward = $as.last();  
 //获取存放图片的ul  
 var $ul = $("#icon\_list");  
 //最多可以显示的图片数量  
 var SHOW\_COUNT = 5;  
 //获取总图片数量  
 var imgCount = $ul.children("li").length;  
 var moveCount = 0;//移动的次数,向右增大,向左减小  
 var liwidth = $ul.children(":first").width();//li(图片)的宽度  
  
 //初始化更新右边按钮是否可点击  
 if (imgCount>SHOW\_COUNT){//如果图片数量大于显示数量,可点击  
 //将向右按钮的class属性值修改为forward  
 $forward.attr("class","forward");  
 }  
  
 //为向右按钮绑定点击监听事件  
 $forward.click(function () {  
 //判断是否需要移动,如果不需要则直接结束  
 if (moveCount === imgCount - SHOW\_COUNT){//如果向右移动的数量等于可移动图片数量,说明已经在最后  
 return;//直接结束,不移动  
 }  
 //移动次数加1  
 moveCount++;  
 //将向左按钮的class属性值修改为backward  
 $backward.attr("class","backward");  
 if (moveCount === imgCount - SHOW\_COUNT){//如果向右移动的数量等于可移动图片数量,说明移动到最后  
 //将向右按钮的class属性值修改为forward\_disabled不可点击  
 $forward.attr("class","forward\_disabled");  
 }  
 //改变存放图片的ul中的left属性  
 $ul.css("left",-moveCount \* liwidth);  
 // $ul.css({  
 // left: -moveCount \* liwidth  
 // })  
 });  
  
 //为向左按钮绑定点击监听事件  
 $backward.click(function () {  
 //判断是否需要移动,如果不需要则直接结束  
 if (moveCount === 0){//如果向左移动的数量等于0,说明已经在最左边  
 return;//直接结束,不移动  
 }  
 //移动次数减1  
 moveCount--;  
 //将向右按钮的class属性值修改为forward  
 $forward.attr("class","forward");  
 if (moveCount === 0){//如果向左移动的数量等于0,说明已经在最左边  
 //将向左按钮的class属性值修改为backward\_disabled不可点击  
 $backward.attr("class","backward\_disable");  
 }  
 //改变存放图片的ul中的left属性  
 $ul.css("left",-moveCount \* liwidth);  
 // $ul.css({  
 // left: -moveCount \* liwidth  
 // })  
 })  
 }  
  
 //当鼠标悬停在某个小图上,在上方显示对应的中图  
 //找到每个小图li,并添加hover监听  
 //当悬停某个小图时,添加class属性,边框变为红色  
 //将上边图片修改为对应的图  
 hoverMiniImg()  
 function hoverMiniImg() {  
 //找到每个小图li,并添加hover监听  
 $("#icon\_list>li").hover(function () {  
 //当悬停某个小图时,找到对应的img标签  
 var $img = $(this).children("img");  
 //添加class属性,边框变为红色  
 // this.children()[0].className = "hoveredThumb";  
 $img.addClass("hoveredThumb");  
 //在上方显示对应的中图  
 //获取小图的src  
 var ssrc = $img.attr("src");  
 //将src中的.jpg替换为-m.jpg  
 var msrc = ssrc.replace(".jpg","-m.jpg");  
 //找到存放中图的img,将src修改为对应的图  
 $("#mediumImg").attr("src",msrc);  
 },function () {  
 //找到对应的img标签,并移除class属性,边框变为红色  
 $(this).children("img").removeClass("hoveredThumb");  
 })  
 }  
  
 //当鼠标在中图上移动时, 显示对应大图的附近部分区域  
 //为中图片maskTop绑定hover事件  
 //显示小黄块  
 //随鼠标移动改变小黄块的位置  
 //显示右边大图的容器  
 //改变加载图的显示  
 //改变大图的位置偏移量  
 bigImg();  
 function bigImg() {  
 var $mediumImg = $("#mediumImg");//中图  
 var $mask = $("#mask");//小黄快  
 var $maskTop = $("#maskTop");//中图背景  
 var $largeImgContainer = $("#largeImgContainer");//大图容器  
 var $loading = $("#loading");//正在加载图  
 var $largeImg = $("#largeImg");//大图  
 var maskWidth = $mask.width();//小黄快的宽度  
 var maskHeight = $mask.height();//小黄快的高度  
 var maskTopWidth = $maskTop.width();//maskTop的宽度  
 var maskTopHeight = $maskTop.height();//maskTop的高度  
  
 //为中图片maskTop绑定hover事件  
 $maskTop.hover(function () {  
 $mask.show();//显示小黄快  
  
 //动态加载对应的大图  
 //获取大图的src值,先获取中图的src值,再将-m.替换成-l.  
 var src = $mediumImg.attr("src").replace("-m.","-l.");  
 //设置大图的src值  
 $largeImg.attr("src",src);  
 //显示大图容器  
 $largeImgContainer.show();  
 //当大图加载完成时执行  
 $largeImg.on("load",function () {  
 //得到大图的宽高  
 var largeWidth = $largeImg.width();  
 var largeHeight = $largeImg.height();  
  
 //为$largeImgContainer设置大小  
 $largeImgContainer.css({  
 width:largeWidth/2,  
 height:largeHeight/2  
 })  
  
 //显示大图  
 $largeImg.show();  
 //隐藏加载图片进度条  
 $loading.hide();  
  
 //绑定鼠标移动监听  
 $maskTop.mousemove(function (event) {//在移动过程中被反复调用  
  
 //移动小黄快  
 //偏移量  
 var left = 0;  
 var top = 0;  
 //鼠标相对于父元素的坐标  
 var eventLeft = event.offsetX;  
 var eventTop = event.offsetY;  
 //计算出小黄块的偏移量  
 left = eventLeft - maskWidth/2;  
 top = eventTop - maskHeight/2;  
 //left的范围为[0,maskTopWidth - maskWidth]  
 //Top的范围为[0,maskTopHeight - maskHeight]  
 //如果超过此范围则小黄块移到边界不动  
 if (left<0){  
 left = 0;  
 }else if (left > maskTopWidth - maskWidth){  
 left = maskTopWidth - maskWidth;  
 }  
 if (top<0){  
 top = 0;  
 }else if (top > maskTopHeight - maskHeight){  
 top = maskTopHeight - maskHeight;  
 }  
 //小黄快随鼠标移动  
 $mask.css({  
 left:left,  
 top:top  
 });  
  
 //移动大图  
 //计算出大图的偏移量  
 //通过大图与中图的比例,得出大图的偏移量  
 //当小黄快向右移动时,大图实际上是向左移动  
 //当小黄快向下移动时,大图实际上是向上移动  
 //也就是小黄快和大图的偏移量是相反的  
 // 要将对应内容显示,偏移量必然为-  
 left = -left \* largeWidth/maskTopWidth;  
 top = -top \* largeHeight/maskTopHeight;  
 //大图随鼠标移动  
 $largeImg.css({  
 left:left,  
 top:top  
 })  
 });  
 });  
  
  
 },function () {  
 $mask.hide();//隐藏小黄快  
 $largeImgContainer.hide();//隐藏大图容器  
 $largeImg.hide();//隐藏大图  
 })  
  
 }  
})