1. jQuery概述
2. 认识jQuery

集JavaScript、CSS、DOM、Ajax于一体的强大框架体系。它的宗旨是：以更少的代码，实现更多的功能。

1. jQuery基本功能
2. 访问和操作DOM元素
3. 什么是DOM元素（文档对象模型）

DOM的每一个页面都是DOM元素，通过传统的方法访问页面中的元素，就是访问DOM对象。

|  |
| --- |
| <**div id="divTmp"**>测试文本</**div**> <**div id="divOut"**></**div**> <**script**>  **var tDiv** =**document**.getElementById(**"divTmp"**);//获取DOM对象  **var oDiv**=**document**.getElementById(**"divOut"**);//获取DOM对象  **var cDiv**=**tDiv**.**innerHTML**;//获取DOM对象中的内容  **oDiv**.**innerHTML**=**cDiv**;//设置DOM对象中内容 </**script**> |

1. 什么是jQuery对象

通过本身自带的方法获取页面元素的对象，称为jQuery对象

|  |
| --- |
| <**div id="divTmp"**>测试文本</**div**> <**div id="divOut"**></**div**> <**script**>  **var tDiv**=**$**(**"#divTmp"**);//获取jQuery对象  **var oDiv**=**$**(**"#divOut"**);//获取jQuery对象  **var cDiv**=**tDiv**.**html**();//获取jQuery对象中的内容  **oDiv**.**html**(**cDiv**);//设置jQuery对象中的内容 </**script**> |

jQuery对象访问方法比DOM对象访问方法更简单，高效。

1. jQuery控制DOM对象

|  |
| --- |
| <**div**>  姓名：<**input type="text" id="text1"**><**br**/>  姓别：<**input id="Radio1" type="radio" name="rdoSex" value="男"**>男  <**input id="Radio2" type="radio" name="rdoSex" value="女"**>女  婚否：<**input id="Checkbox1" type="checkbox"**>  <**input type="button" id="button" value="提交"**"> </**div**> <**div id="divTip"**></**div**> <**script**>  **$**(**function** () {  **$**(**"#button"**).click(**function** () {  //获取输入框的值  **var** oTextValue=**$**(**"#text1"**).val();  //获取单选框按钮的值  **var** oRdoValue=**$**(**"#Radio1"**).**is**(**":checked"**)?**"男"**:**"女"**;  //获取复选框的值  **var** oChkValue=**$**(**"#Checkbox1"**).**is**(**":checked"**)?**"已婚"**:**"未婚"**;  //显示提示文本框元素和内容  **$**(**"#divTip"**).**html**(oTextValue+**"<br/>"**+oRdoValue+**"<br/>"**+oChkValue);  }) }) </**script**> |

1. jQuery操作DOM
2. 控制页面样式

点击改变内容和背景颜色

|  |
| --- |
| <**style**>  .**divDefault**{**background-color**: **yellow**;}  .**divClick**{**background-color**: **red**;} </**style**> <**div class="divDefault"**>点击前的样式</**div**> <**script**>  **$**(**function** () {  **$**(**".divDefault"**).click(**function** () {  **$**(**this**).toggleClass(**"divClick"**).**html**(**"点击后样式"**);  });  }) </**script**> |

1. 对页面事件的处理
2. 大量插件在页面中的运用
3. 与Ajax技术的完美结合
4. jQuery选择器
5. 基本选择器 id,class,标签,\*
6. id选择器
7. class选择器
8. 标签选择器
9. \*选择器
10. 层次选择器
11. 后代(ancestor descendant) 祖先的所有后代元素

|  |
| --- |
| <**div class="zx"**>  <**span class="fq"**>  <**span class="ez"**>子元素</**span**>  </**span**>  <**div class="xd"**>1</**div**>  <**div class="xd"**>2</**div**> </**div**> <**div class="xd"**>3</**div**> <**script**>  **$**(**function** () {*//匹配后代元素* **$**(**"div span"**).css(**"backgroundColor"**,**"red"**); });  </**script**> |

1. 父子(parent>child) 父元素下的所有子元素

|  |
| --- |
| **$**(**function** () {//匹配子元素  **$**(**"div>span"**).css(**"backgroundColor"**,**"yellow"**); }); |

1. 相邻(pref+naxt)紧跟在pref元素后的相邻元素

|  |
| --- |
| **$**(**function** () {//相邻  **$**(**".zx+div"**).css(**"backgroundColor"**,**"yellow"**);  //或者使用next，nextAll  // $(".zx").nextAll().css("backgroundColor","yellow") }); |

1. 兄弟(pref~siblings) pref元素之后的所有兄弟元素

|  |
| --- |
| **$**(**function** () {//兄弟  **$**(**".fq"**).siblings(**"div"**).css(**"backgroundColor"**,**"yellow"**); }); |

1. 过滤选择器
2. 简单过滤选择器 :first :last :not(selector) :even :eq(index)

|  |  |  |
| --- | --- | --- |
| 选择器 | 功能 | 返回值 |
| first()或:first | 获取第一个元素 | 单个元素 |
| last()或:last | 获取最后一个元素 | 单个元素 |
| :not(selector) | 获取除给定选择器外的所有元素 | 元素集合 |
| :even | 获取所有索引值为偶数的元素，索引号从0开始 | 元素集合 |
| :odd | 获取所有索引值为奇数的元素，索引号从0开始 | 元素集合 |
| :eq(index) | 获取指定索引值的元素，索引号从0开始 | 单个元素 |
| :gt(index) | 获取所有大于指定索引值的元素，索引号从0开始 | 元素集合 |
| :lt(index) | 获取所有小于于指定索引值的元素，索引号从0开始 | 元素集合 |
| :header | 获取所有标题类的元素，如h1,h2… | 元素集合 |
| :animated | 获取正在执行动画效果的元素 | 元素集合 |

|  |
| --- |
| <**style**>  .**getClass**{**background-color**: **yellow**;} </**style**>  <**div**>  <**h1**>基本过滤选择器</**h1**>  <**ul**>  <**li class="defClass"**>item 0</**li**>  <**li class="defClass"**>item 1</**li**>  <**li class="nofClass"**>item 2</**li**>  <**li class="defClass"**>item 3</**li**>  </**ul**>  <**span id="spnMove"**>span move</**span**> </**div**> <**script**>  **$**(**function** () {//给第一个li元素增加一个类名为getClass的样式  **$**(**"li:first"**).addClass(**"getClass"**);  });  **$**(**function** () {//给最后一个li元素增加一个类名为getClass的样式  **$**(**"li:last"**).addClass(**"getClass"**);  });  **$**(**function** () {//除类名为defClass的元素添加样式  **$**(**"li:not(.defClass)"**).addClass(**"getClass"**);  });  **$**(**function** () {**$**(**"li:even"**).addClass(**"getClass"**);});  **$**(**function** () {**$**(**"li:odd"**).addClass(**"getClass"**);});  **$**(**function** () {**$**(**"li:eq(1)"**).addClass(**"getClass"**); });  **$**(**function** () {**$**(**"li:gt(1)"**).addClass(**"getClass"**);});  **$**(**function** () {**$**(**":header"**).addClass(**"getClass"**);});  **$**(**function** () {  animateIt();  **$**(**"#spnMove:animated"**).addClass(**"getClass"**);  });  **function** animateIt(){  **$**(**"#spnMove"**).**slideToggle**(**"slow"**,animateIt);  } </**script**> |

1. 内容过滤选择器:contains(text) :empty :has(selector)

|  |  |  |
| --- | --- | --- |
| 选择器 | 功能 | 返回值 |
| :contains(text) | 获取包含给定文本的元素 | 元素集合 |
| :empty | 获取所有不包含子元素或者文本的空元素 | 元素集合 |
| :has(selector) | 获取含有选择器所匹配的元素的元素 | 元素集合 |
| :parent | 获取所有包含子元素或者文本的元素 | 元素集合 |

|  |
| --- |
| <**style**>  **div**{**width**:56**px**;**height**:65**px**;**border**:1**px solid**;**float**: **left**;} </**style**>  <**div**>ABCD</**div**> <**div**><**span**></**span**></**div**> <**div**>EFG</**div**> <**div**></**div**> <**script**>  **$**(**function** () {*//包含有文本A元素* **$**(**"div:contains('A')"**).css(**"backgroundColor"**,**"red"**);  }); </**script**>    **$**(**function** () {*//获取所有不包含子元素或者文本的空元素* **$**(**"div:empty"**).css(**"backgroundColor"**,**"yellow"**); });    **$**(**function** () {*//获取含有选择器所匹配的元素的元素* **$**(**"div:has(span)"**).css(**"backgroundColor"**,**"blue"**); });    **$**(**function** () {*//获取所有包含子元素或者文本的元素* **$**(**"div:parent"**).css(**"backgroundColor"**,**"pink"**); }) |

1. 可见性过滤选择器 :hidden :visible

|  |  |  |
| --- | --- | --- |
| 选择器 | 功能 | 返回值 |
| :hidden | 获取所有不可见元素，或者type为hidden的元素 | 元素集合 |
| :visible | 获取所有可见元素 | 元素集合 |

|  |
| --- |
| <**style**>  .**getClass**{**background-color**: **yellow**;} </**style**>  <**span style="display**: **none"**>hidden span</**span**> <**span**>visible span</**span**> <**div**>visible div</**div**> <**script**>  **$**(**function** () {//给所有可见的div元素添加样式  **$**(**"div:visible"**).addClass(**"getClass"**);  });  **$**(**function** () {//给不可见元素添加样式  **$**(**"span:hidden"**).show().addClass(**"getClass"**);  }) </**script**> |

1. 属性过滤选择器 [attribute=value]

|  |  |  |
| --- | --- | --- |
| 选择器 | 功能 | 返回值 |
| [attribute] | 获取包含给定属性的元素 | 元素集合 |
| [attribute=value] | 获取attribute属性的属性值为value的元素 | 元素集合 |
| [attribute!=value] | 获取不等于attribute属性的属性值为value的元素 | 元素集合 |
| [attribute^=value] | 获取attribute属性以属性值为value开始的元素 | 元素集合 |
| [attribute$=value] | 获取attribute属性以属性值为value结尾的元素 | 元素集合 |
| [attribute\*=value] | 获取attribute属性的属性值包含value的元素 | 元素集合 |
| [selector1][selector2][selectorN] | 获取满足多个条件的复合属性的元素 | 元素集合 |

|  |
| --- |
| <**style**>  **div**{**width**:100**px**;**height**:40**px**;**border**:1**px solid**;**float**: **left**;}  </**style**>  <**div id="div"**>ID</**div**> <**div title="A"**>title A</**div**> <**div id="divAB" title="AB"**>ID Title AB</**div**> <**div title="ABC"**>Title ABC</**div**>  <**script**>  **$**(**function** () {//设置所有含有id的div背景色为黄色  **$**(**"div[id]"**).css(**"backgroundColor"**,**"yellow"**);  }); </**script**>    **$**(**function** () {//设置所有属性为title且属性值为A的div的背景色为红色  **$**(**"div[title='A']"**).css(**"backgroundColor"**,**"red"**); });    **$**(**function** () {//设置所有属性为title且属性值不为A的元素的背景色为蓝色  **$**(**"div[title!='A']"**).css(**"backgroundColor"**,**"blue"**); });    **$**(**function** () {//设置所有属性为title且属性值以A开始的元素的背景色为橘色  **$**(**"div[title^='A']"**).css(**"backgroundColor"**,**"orange"**); });    **$**(**function** () {//设置所有属性为title且属性值以C结尾的元素的背景色为粉色  **$**(**"div[title$='C']"**).css(**"backgroundColor"**,**"pink"**); });    **$**(**function** () {//设置所有属性为title且属性值含有C的元素的背景色为粉色  **$**(**"div[title\*='B']"**).css(**"backgroundColor"**,**"pink"**); });    **$**(**function** () {//设置所有属性为title中id的值是“divAB”且属性值含有C的元素的背景色为粉色  **$**(**"div[id='divAB'][title\*='B']"**).css(**"backgroundColor"**,**"pink"**); }); |

1. 子元素过滤选择器 :nth-child(eq|even|odd|index)

|  |  |  |
| --- | --- | --- |
| 选择器 | 功能 | 返回值 |
| :nth-child(eq|even|odd|index) | 获取每个父元素下的特定位置元素，索引号从1开始 | 元素集合 |
| :first-child | 获取每个父元素下的第一个元素 | 元素集合 |
| :last-child | 获取每个父元素下的最后一个元素 | 元素集合 |
| :only-child | 获取每个父元素下的仅有一个元素 | 元素集合 |

|  |
| --- |
| <**style**>  .**getClass**{**background-color**: **yellow**;}  **li**{**width**:100**px**;} </**style**>  <**ul**>  <**li**>item 1-0</**li**>  <**li**>item 1-1</**li**> </**ul**> <**ul**>  <**li**>item 2-0</**li**>  <**li**>item 2-1</**li**> </**ul**> <**ul**>  <**li**>item 3-0</**li**> </**ul**>  <**script**>  **$**(**function** () {//给每个ul下的li的第二个li添加样式  **$**(**"li:nth-child(2)"**).addClass(**"getClass"**);  }); </**script**> |

1. 表单对象属性过滤选择器 :enable :checked :selected

|  |  |  |
| --- | --- | --- |
| 选择器 | 功能 | 返回值 |
| :enable | 获取表单中所有属性为可用的元素 | 元素集合 |
| :disable | 获取表单中所有属性为不可用的元素 | 元素集合 |
| :checked | 获取表单中所有被选中的元素 | 元素集合 |
| :selected | 获取表单中所有被选中option的元素 | 元素集合 |

1. 表单选择器:input :text :checkbox :button :submit

|  |  |  |
| --- | --- | --- |
| 选择器 | 功能 | 返回值 |
| :input | 获取所有input，textarea，select | 元素集合 |
| :text | 获取所有单行文本框 | 元素集合 |
| :password | 获取所有密码框 | 元素集合 |
| :radio | 获取所有单选按钮 | 元素集合 |
| :checkbox | 获取所有复选框 | 元素集合 |
| :submit | 获取所有提交按钮 | 元素集合 |
| :image | 获取所有图像域 | 元素集合 |
| :reset | 获取所有重置按钮 | 元素集合 |
| :button | 获取所有按钮 | 元素集合 |
| :file | 获取所有文件域 | 元素集合 |

|  |
| --- |
| <**input type="button"**>  <**script**>  **$**(**function** () { **$**(**":button"**).addClass(**"getClass"**);}); </**script**> |

1. jQuery操作DOM元素
2. DOM基础

文档对象：页面文档，创建一个页面并加载到web浏览器时，DOM模型则根据该页面的内容创建了一个文档文件。在页面文档中，通过树状模型来展示页面的元素和内容，其展示方式则是通过节点来实现。

1. 访问DOM元素

访问DOM元素主要包括对元素属性、内容、值、CSS的操作。

1. 元素属性操作
2. 获取元素属性attr(name)

语法格式：attr(name) name—属性名称

|  |
| --- |
| <**img src="images/borderimage.png" title="这是一幅画" alt=""**> <**div id="divAlt"**></**div**> <**script**>  **$**(**function** () {  **var** strAlt=**$**(**"img"**).attr(**"src"**);//获取属性值一  strAlt +=**"<br/><br/>"**+**$**(**"img"**).attr(**"title"**);//获取属性值二  **$**(**"#divAlt"**).**html**(strAlt);//显示在页面中  }) </**script**> |

1. 设置元素属性attr(key,value)

语法格式：attr(key,value) key—属性名称 value---属性值

或者：attr(key,function(index)) index—当前元素的索引号

函数的返回值作为属性值

|  |
| --- |
| <!--设置属性值--> <**img src="images/borderimage.png" alt=""**> <**span**>正在加载图片...</**span**> <**script**>  **$**(**function** () {  **$**(**"img"**).attr(**"src"**,**"images/flower-green.png"**);//设置src属性  **$**(**"img"**).attr(**"title"**,**"这是一幅画"**);  **$**(**"img"**).attr({**src**:**"images/flower-green.png"**,**title**:**"这是一幅画"**});//设置多个属性  }) </**script**> |

1. 删除元素的属性removeAttr(name)

语法格式：removeAttr(name) name—属性名称

1. 元素内容操作 html() text()

在jQuery中，操作元素内容的方法包括html()和text()。

区别：

|  |  |  |
| --- | --- | --- |
| 语法格式 | 参数说明 | 功能描述 |
| html() | 无参数 | 用于获取元素的HTML内容 |
| html(val) | val为元素的HTML内容 | 用于设置元素的HTML内容 |
| text() | 无参数 | 用于获取元素的文本内容 |
| text(val) | val为元素的文本内容 | 用于设置元素的文本内容 |

|  |
| --- |
| <**div id="div"**><**b**><**i**>示例</**i**></**b**></**div**> <**div id="divHtml"**></**div**> <**div id="divText"**></**div**> <**script**>  **$**(**function** () {  **var** strHtml=**$**(**"#div"**).**html**();//获取HTML内容  **var** strText=**$**(**"#div"**).**text**();//获取文本内容  **$**(**"#divHtml"**).**html**(strHtml);//设置HTML内容  **$**(**"#divText"**).**text**(strText);//设置文本内容  }) </**script**> |

1. 获取或设置元素值

val方法一般用于表单中获取或设置对象的值。

1. 获取元素值 val()

val()

1. 设置元素值 val(val)

val(val)，val(array)

还可以获取select标记中的多个选项值：

val().join(“.”)

1. 元素样式操作
2. 直接设置元素样式 css()

css(name,value)或者css({name1:value1,name2:value2,…})

name—样式名称

value—样式值

|  |
| --- |
| <**div id="div"**>示例</**div**> <**script**>  **$**(**function** () {  **$**(**"#div"**).css(**"background"**,**"red"**);  }) </**script**> |

1. 增加CSS类别 addClass()

addClass(class)或addClass(class0 class1…)

class ----类别名

|  |
| --- |
| <**style**>  .**getClass**{**background**: **yellow**;} </**style**> <**div id="div"**>示例</**div**> <**script**>  **$**(**function** () {  **$**(**"#div"**).addClass(**"getClass"**);  }) </**script**> |

1. 类别切换 toggleClass(class)

toggleClass(class)

class --- 类别名

其功能是当元素中含有class的CSS类别时，删除该类别，否则增加一个该名称为CSS类别

|  |
| --- |
| <**style**>  .**getClass**{**background**: **yellow**;}  </**style**> <**div id="div"**>示例</**div**> <**script**>  **$**(**function** () {  **$**(**"#div"**).click(**function** () {  **$**(**this**).toggleClass(**"getClass"**);  });  }) </**script**> |

1. 删除类别removeClass(class)

removeClass(class)或removeClass(class0,class1,…)或removeClass()

class --- 类别名

removeClass() --- 删除元素所有类别

1. 创建节点元素 $(html)

动态创建页面元素，语法：

$(html)

例子：$(“body”).append()

|  |
| --- |
| <**body**> <**script**>  **var div** = **$**(**"<div title='jQuery理念'>示例</div>"**); *//创建节点*  **$**(**"body"**).append(**div**); *//插入节点中* </**script**> </**body**> |

1. 插入节点
2. 内部插入节点方法append() prepend()

|  |  |  |
| --- | --- | --- |
| 语法格式 | 参数说明 | 功能描述 |
| append(content) | 插入到元素内的内容的后面的内容 | 向所选的元素内部的内容后面插入内容 |
| append(function()) | 通过函数返回插入到元素的内容后面的内容 | 向所选的元素内部的内容后面插入函数返回的内容 |
| appendTo(content) | 插入到元素的内容的后面的内容 | 把所选的元素插入到另一个指定的元素内的内容的后面 |
| prepend(content) | 插入目标元素内部的内容前面的内容 | 向所选的元素内部的内容前面插入内容 |
| prepend(function()) | 通过函数返回插入到元素的内容前面的内容 | 向所选的元素内部的内容前面插入函数返回的内容 |
| prependTo(content) | 插入到元素的内容的前面的内容 | 把所选的元素插入到另一个指定的元素内的内容的前面 |

|  |
| --- |
| <**style**>  **div**{**border**:1**px solid**;**float**: **left**;**width**:56**px**;**height**:56**px**;} </**style**>  <**div id="div"**>  <**span**>示例1</**span**> </**div**> <**div id="div2"**>示例2</**div**>  <**script**>  **$**(**function** () {  **$**(**"#div"**).append(**$**(**"#div2"**).**html**());*//向id为div的元素插入内容*  // $("#div").append(function () {//通过函数插入内容 // return $("#div2").html(); // });  })  </**script**>    **$**(**"#div"**).prepend(**$**(**"#div2"**).**html**());  *// $("#div").prepend(function () { // return $("#div2").html(); // })* |

|  |
| --- |
| **$**(**"span"**).**appendTo**(**$**(**"#div2"**));*//将span插入到id为div2的元素中*    **$**(**"span"**).**prependTo**(**$**(**"#div2"**)); |

1. 外部插入节点的方法after() before()

|  |  |  |
| --- | --- | --- |
| 语法格式 | 参数说明 | 功能描述 |
| after(content) | 插入元素外部后面的内容 | 向所选的元素外部后面插入内容 |
| after(function()) | 通过函数返回插入元素外部后面的内容 | 向所选的元素外部后面插入函数返回的内容 |
| before(content) | 插入元素外部后面的内容 | 向所选的元素外部前面插入内容 |
| before(function) | 通过函数返回插入元素外部前面的内容 | 向所选的元素外部前面插入函数返回的内容 |
| insertAfter(content) | 插入元素外部后面的内容 | 向所选的元素插入到另一元素的外面后面 |
| insertBefore(content) | 插入元素外部前面的内容 | 向所选的元素插入到另一元素的外面前面 |

|  |
| --- |
| <**style**>  **div**{**border**:1**px solid**;**width**:56**px**;**height**:56**px**;} </**style**>  <**div id="div"**>  <**span**>示例1</**span**> </**div**> <**div id="div2"**>示例2</**div**> <**div id="div3"**>示例3</**div**>  <**script**>  **$**(**function** () { **$**(**"#div"**).after(**$**(**"#div3"**))  }) </**script**>    **$**(**"#div span"**).insertBefore(**$**(**"#div3"**)); |

1. 复制节点 clone()

clone()其功能为复制匹配的DOM元素并且选中复制成功的元素，该方法仅是复制元素本身，被复制后的新元素不具备任何元素的行为，如需在复制时将该元素的全部行为也复制，可以通过方法clone(true)实现。

|  |
| --- |
| <**img src="images/borderimage.png"**> <**div id="div" style="width**:200**px**;**height**:100**px**;**border**:1**px solid"**></**div**> <**script**>  **$**(**function** () {  **$**(**"img"**).click(**function** () {  **$**(**this**).clone(**true**).**appendTo**(**"#div"**);  });  }) </**script**> |

1. 替换节点 replaceWith() replaceAll()
2. replaceWith()

replaceWith(content)

将所选的元素替换成指定的内容content

1. replaceAll()

replaceAll(selector)

将所选的元素替换成指定的selector的元素

|  |
| --- |
| <**span class="s1"**>示例1</**span**> <**span class="s2"**>示例2</**span**> <**script**>  **$**(**function** () {  **$**(**".s1"**).replaceWith(**"<span>示例3</span>"**);  **$**(**"<span>示例4</span>"**).**replaceAll**(**".s2"**);  }) </**script**> |

注：两者区别：.replaceAll()和.replaceWith()功能类似，主要是所选元素和替换元素的位置相反。

与html()区别：html()会替换指定元素内部的HTML，而replaceWith()会替换元素本身及其内部的HTML。

1. 包裹节点

相关方法

|  |  |  |
| --- | --- | --- |
| 语法格式 | 参数说明 | 功能描述 |
| wrap(html) | html为字符串代码，用于生成元素并包裹所选元素 | 把所选元素用其他字符串代码包裹起来 |
| wrap(elem) | 包裹所选元素的DOM元素 | 把所选元素用其他DOM元素包裹起来 |
| wrap(function) | 包裹结构的一个函数 | 把所选元素用函数返回的代码包裹起来 |
| unwrap() | 无参 | 移除所选元素的父元素或者包裹标记 |
| wrapAll(html) | html为字符串代码，用于生成元素并包裹所选元素 | 把所选元素用单个元素包裹起来 |
| wrapAll(elem) | 包裹所选元素的DOM元素 | 把所选元素用单个DOM元素包裹起来 |
| wrapInner(html) | html为字符串代码，用于生成元素并包裹所选元素 | 把所选元素的子内容（包括文本节点）用字符串代码包裹起来 |
| wrapInner(elem) | 包裹所选元素的DOM元素 | 把所选元素的子内容（包括文本节点）用DOM包裹起来 |
| wrapInner(fn) | 包裹结构的一个函数 | 把所选元素的子内容（包括文本节点）用函数返回的代码包裹起来 |

注：wrap(html)与wrapInner(html)较为常用，前者包裹外部元素，即包裹元素本身，后者包裹元素内部的文本字符。

|  |
| --- |
| <**p**>体育运动：<**span**>羽毛球</**span**></**p**> <**p**>类型图书：<**span**>网络、技术</**span**></**p**> <**script**>  **$**(**function** () { **$**(**"p"**).**wrap**(**"<b></b>"**); }) </**script**>    **$**(**"p"**).wrapInner(**"<i></i>"**); |

1. 遍历元素 each()

each(callback)

callback---function函数，该函数可以接受一个index，index为遍历元素的序号。

|  |
| --- |
| <**p**>  <**img src="images/flower-green.png" alt="1"**>  <**img src="images/borderimage.png" alt="2"**>  <**img src="images/sky.jpg" alt="3"**> </**p**> <**script**>  **$**(**function** () {  **$**(**"img"**).each(**function** (index) {  **this**.**title**=**"序号是："**+index+**"alt："**+**this**.**alt**;  });  }) </**script**> |

1. 删除元素 remove() empty()

remove(expr)

参数expr为可选，有参，则该参数为筛选元素的jQuery表达式，通过该表达式来获取指定的元素，并进行删除。

empty() 清空所选的页面元素或所有的后代元素

|  |
| --- |
| <**ul**>  <**li**>体育运动：<**span**>羽毛球</**span**></**li**>  <**li**>类型图书：<**span**>网络、技术1</**span**></**li**>  <**li title="t"**>类型图书：<**span**>网络、技术2</**span**></**li**>  <**li**>类型图书：<**span**>网络、技术3</**span**></**li**> </**ul**> <**script**>  **$**(**function** () {  **$**(**"ul li"**).remove(**"li[title=t]"**);*//删除指定属性的元素* **$**(**"ul li:eq(1)"**).remove();*//删除节点中第二个元素* }) </**script**> |

1. 综合案例

|  |
| --- |
| <**style**>  **table**{**width**:360**px**;**border-collapse**: **collapse**;}  **table tr th**,**td**{**border**:1**px solid #666**;**text-align**: **center**;}  **table tr th**{**background**: **#ccc**;**height**:32**px**;}  **table tr td img**{**border**:1**px solid #666**;**padding**:3**px**;**width**:42**px**;**height**:42**px**;**cursor**: **hand**;}  </**style**> </**head**> <**body**> <**table**>  <**tr**><**th**>1</**th**><**th**>2</**th**><**th**>3</**th**></**tr**>  <**tr id="0"**>  <**td**><**input id="checkbox1" type="checkbox" value="0"**></**td**>  <**td**>1001</**td**>  <**td**><**img src="images/flower-green.png"**></**td**>  </**tr**>  <**tr id="1"**>  <**td**><**input id="checkbox2" type="checkbox" value="1"**></**td**>  <**td**>1002</**td**>  <**td**><**img src="images/flower-green.png"**></**td**>  </**tr**>  <**tr id="2"**>  <**td**><**input id="checkbox3" type="checkbox" value="2"**></**td**>  <**td**>1003</**td**>  <**td**><**img src="images/flower-green.png"**></**td**>  </**tr**>  <**tr id="3"**>  <**td**><**input id="checkbox4" type="checkbox" value="3"**></**td**>  <**td**>1004</**td**>  <**td**><**img src="images/flower-green.png"**></**td**>  </**tr**> </**table**> <**table**>  <**tr**>  <**td**>  <**span**><**input type="checkbox" id="chkAll"**>全选</**span**>  <**span**><**input type="button" id="btnDel" value="删除" class="btn"**></**span**>  </**td**>  </**tr**> </**table**> <**script**>  **$**(**function** () {  */\* 全选复选框单击事件 \*/* **$**(**"#chkAll"**).click(**function** () {  **if**(**this**.**checked**){*//已勾选* **$**(**"table tr td input[type=checkbox]"**).attr(**"checked"**,**true**);*//设置属性* }**else**{  **$**(**"table tr td input[type=checkbox]"**).attr(**"checked"**,**false**);*//设置属性* }  });  */\* 删除按钮单击事件 \*/* **$**(**"#btnDel"**).click(**function** () {  **var** intL=**$**(**"table tr td input[type=checkbox]:checked:not('#chkAll')"**).**length**;*//获取全选框之外的所有选项* **if** (intL !==0){*//有选中* **$**(**"table tr td input[type=checkbox]:not('#chkAll')"**).each(**function** (index) {*//遍历* **if**(**this**.**checked**){  **$**(**"table tr[id="**+**this**.**value**+**"]"**).remove();  }  });  }  });  }); </**script**> |

1. jQuery中的事件与应用
2. 事件机制
3. 事件冒泡

|  |
| --- |
| <**div**>  <**input type="button" id="btn" value="点击"**> </**div**> <**div class="show"**></**div**> <**script**>  **$**(**function** () {  **var** count = 0;//记录执行次数  **$**(**"body,div, #btn"**).click(**function** (event) {//body、div、按钮都执行单击事件  count++;  **$**(**".show"**).show()  .**html**(**"hello world"**)  .append(**"<div>执行次数<b>"**+count+**"</b></div>"**);  event.stopPropagation();//阻止冒泡过程  });  }) </**script**>  点击按钮，点击一下只执行一次，而不是三次。如果不阻止冒泡，先执行点击按钮事件，再到div,然后是body，所以显示执行次数是3。 |

1. 页面载入事件ready()
2. ready()方法的原理

在jQuery脚本加载到页面时，会设置一个isReady的标记，用于监听页面加载的进度，当然遇到执行ready()方法时，通过查看isReady值是否被设置，如果被设置，说明页面并未加载完成，在这情况下，将未完成的部分用一个数组缓存起来，当全部加载完成后，在将未完成的部分通过缓存一一加载。

4中写法：

|  |
| --- |
| **$**(**document**).**ready**(**function** () {//do something}) |
|  |
| **$**(**function** () { //do something}) |
|  |
| **jQuery**(**document**).**ready**(**function** () {//do something}) |
|  |
| **jQuery**(**function** () {//do something}) |

1. 绑定事件 click() blind()

|  |
| --- |
| **$**(**"#btn"**).click(**function** () {  *//执行代码* }) |

blind(type,[data],fn)

type – 一个或多个类型字符串

data ---数据对象

fn ---绑定到选择元素的实践中的处理函数

|  |
| --- |
| **$**(**"#btn"**).bind(**"click"**,**function** () {  *//执行代码* }) |
| **或者** |
| **$**(**"#btn"**).bind({  focus:**function** () {  *//执行代码* },  change:**function** () {  *//执行代码* } }) |

1. 切换事件 hover() toggle()

切换事件就是有两个以上事件绑定于一个元素，在行为动作间进行切换。

hover()鼠标悬停事件

|  |
| --- |
| **$**(**"a"**).hover(**function** () {  *//* },**function** () {  *//* }) |

toggle()功能是依次调用函数

toggle(fn,fn2,[fn3,fn4])

当没有参数时，表示切换显示/隐藏

toggle(switch), switch为布尔值，true为显示元素，false为隐藏元素

toggle(speed,[callback]) 动画显示/隐藏，与show(speed,[callback])一样

|  |
| --- |
| <**img src="images/flower-green.png" style="border**: 1**px solid #666**;**padding**:3**px**;**"**> <**button**>11</**button**> <**script**>  **$**(**function** () {  **$**(**"button"**).click(**function**(){  **$**(**"img"**).toggle(  **function** () {**$**(**"img"**).attr(**"src"**,**"images/borderimage.png"**);**$**(**"img"**).css(**"display"**,**"block"**)},  **function** () {**$**(**"img"**).attr(**"src"**,**"images/flower-red.png"**);**$**(**"img"**).css(**"display"**,**"block"**)} );  }) </**script**>    **$**(**"button"**).click(**function** () {  **$**(**"img"**).toggle(); }); |

1. 移除事件 unbind([type],[fn])

unbind([type],[fn])

type --- 移除事件类型

如果unbind没有参数，则移除所有绑定的事件

1. 其它事件
2. one()为所选元素绑定一个仅触发一次的处理函数

one(type,[data],fn)

type --- 时间类型

data --- 可选参数

1. trigger()在所选元素上自动触发指定类型的事件

trigger(type,[data])

data --- 在触发事件时，传递给函数的附加参数

1. 表单应用

|  |
| --- |
| <**style**>  .**txt**{**background**: **yellow**; }  .**getClass**{**background**: **red**; }  .**blurClass**{**background**: **green**; }  </**style**> <**form id="form" action="#"**>  邮箱：<**input id="input" type="text" class="txt" title=""**>  <**span**></**span**> </**form**> <**script**>  **$**(**function** () {  **$**(**"#input"**).trigger(**"focus"**);//默认时自动获得焦点  **$**(**"#input"**).focus(**function** () {//文本框获取焦点事件  **$**(**this**).removeClass(**"txt"**)//删除类别样式  .addClass(**"getClass"**);//添加类别样式  **$**(**"span"**).**html**(**"请你输入邮箱地址"**);  });  **$**(**"#input"**).blur(**function** () {//文本框丢失焦点时间  **var** vtext = **$**(**"#input"**).val();//获取输入框的值  **if**(vtext.**length**===0){  **$**(**this**).removeClass(**"getClass"**)  .addClass(**"blurClass"**);  **$**(**"span"**).**html**(**"邮箱地址不能为空"**);  }**else**{  **$**(**"span"**).**html**(**""**);  }  });  }) </**script**> |

1. 列表应用
2. 网页选项卡的应用
3. jQuery的动画与特效
4. 显示与隐藏 show() hide() 或切换方法toggle()

动画：

show(speed,[callback])

speed --- 动画执行速度，默认值有slow(o.6秒)，normal(0.4秒)，fast(0.2秒)

如果不用默认值，可以写数字，单位是毫秒

callback --- 动画完成时执行的回调函数

hide(speed,[callback])

1. 滑动 slideDown slideUp或slideToggle()
2. slideDown和slideUp 仅改变元素高度

slideDown(speed,[callback])

slideUp(speed,[callback])

|  |
| --- |
| <**button**>隐藏图片</**button**> <**div**>  <**img src="images/sky.jpg"**>  <**div id="div2"**></**div**> </**div**> <**script**>  **$**(**function** () {  **var** buttonShow = **false**;  **$**(**"button"**).click(**function** () {//点击button  **if**(!buttonShow){  //图片高度向上减小，执行完成后，回调一个函数  **$**(**"img"**).**slideUp**(3000,**function** () {  **$**(**"#div2"**).**html**(**"隐藏成功"**)  });  **$**(**this**).**html**(**"显示图片"**);  buttonShow = **true**;  }**else**{  **$**(**"#div2"**).**html**(**""**);  **$**(**"img"**).**slideDown**(3000);  **$**(**this**).**html**(**"隐藏图片"**);  buttonShow = **false**;  }  });  }) </**script**> |

1. slideToggle()

|  |
| --- |
| **$**(**"button"**).click(**function** () {  **$**(**"img"**).**slideToggle**(3000,**function** () {  *//* }) }); |

1. 淡入淡出 fadeIn() fadeOut()，fadeTo()
2. fadeIn ,fadeOut

通过改变元素的透明度，实现淡入淡出效果

fadeIn(speed,[callback])

fadeOut(speed,[callback])

1. fadeTo

将所选元素的不透明度以动画效果的速度调整到指定的不透明值。

fadeTo(speed,[callback])

|  |
| --- |
| <**button**>淡入</**button**> <**button**>淡出</**button**> <**div**>  <**img src="images/sky.jpg"**>  <**div id="div2"**></**div**> </**div**> <**script**>  **$**(**function** () {  **$**(**"button:eq(0)"**).click(**function** () {  **$**(**"img"**).**fadeIn**(3000,**function** () {  **$**(**"#div2"**).**html**(**"淡入"**);  })  });  **$**(**"button:eq(1)"**).click(**function** () {  **$**(**"img"**).**fadeOut**(3000,**function** () {  **$**(**"#div2"**).**html**(**"淡出"**);  })  });  }) </**script**> |

1. 自定义动画 animate() stop() delay()
2. animate(parames,[duration],[easing],[callback])

params --- 用于制作动画效果的属性样式和值的集合。

duration --- 完成时间，默认值有slow,normal,fast。可自定义数字

easing --- 动画插件使用，有linear,swing

|  |
| --- |
| <**button**>点击变大</**button**> <**script**>  **$**(**function** () {  **$**(**"button"**).click(**function** () {  **$**(**this**).animate(  {**width**:**"20%"**,**height**:**"70px"**},  3000,  **function** () {  **$**(**this**).css(**"border"**,**"1px solid"**)  .**html**(**"变大了"**);  });  });  }) </**script**> |

1. stop()停止动画

stop([clearQueue],[gotoEnd])

clearQueue --- 布尔值，表示是否停止正在执行的动画

gotoEnd --- 也是布尔值，表示是否立即完成正在执行的动画

1. delay()延时动画的执行

delay(duration,[ queueName])

duration ---延时时间值，单位毫秒

queueName ---队列名，即动画队列

1. 动画效果
2. Ajax在jQuery中的应用
3. 加载异步数据 load()
4. load()

在传统的JavaScript中，使用XMLHttpRequest对象异步加载数据，在jQuery中，使用load方法可以轻松实现获取异步数据的功能。

load(url,[data],[callback])

url --- 被加载的页面地址

data --- 发送到服务器的数据，其格式为key/value

callback --- 表示加载完成后，返回加载页的回调函数

|  |
| --- |
| <**div class="divFrame"**></**div**> <**button**>加载</**button**> <**script**>  **$**(**function** () {  **$**(**"button"**).click(**function** () {  **$**(**".divFrame"**).**load**(**"../综合案例01.html"**);  });  }) </**script**> |

1. 全局函数getJSON()

获取json格式数据

格式为：

$.getJSON(url,[data],[calback])

url ---请求的地址

|  |
| --- |
| [{  **"id"**:1,  **"name"**:**"图书"** },{  **"id"**:2,  **"name"**:**"音乐"** }] |

|  |
| --- |
| <**div class="divFrame"**></**div**> <**button**>加载</**button**> <**script**>  **$**(**function** () {  **$**(**"button"**).click(**function** () {  **$**.getJSON(**"../data.json"**,**function** (data) {  **$**(**".divFrame"**).empty();//清空内容  **var** str = **""**;  **$**.each(data,**function** (index,info) {//遍历数据  str += **"id:"**+info[**"id"**]+**"<br>"**;  str += **"name:"**+info[**"name"**]+**"<hr>"**;  });  **$**(**".divFrame"**).**html**(str);  });  });  }) </**script**> |

1. 全局函数getScript()

获取.js文件的内容

在页面中获取js文件有多种方法，如：

|  |
| --- |
| <**script type="text/javascript" src="../js/jquery.min.js"**></**script**> |

|  |
| --- |
| $(<**script type="text/javascript" src="../js/jquery.min.js"/**>).appendTo(**"head"**) |

getScript调用格式如下：

$.getScript(url,[callback])

url ----js文件地址

1. 异步加载XML文件 $.get()

$.get(url,[data],[callback],[type])

url --- 等待加载的数据地址

type --- 返回的数据格式，如html，xml，js，json,text等

1. 请求服务器数据
2. $.get()请求数据

$.get()方法实现XML文档的加载，还可以实现数据的请求。它的不足在于不适合传递数据量较大的数据，请求的数据会保存在浏览器中缓存中，有一定的安全风险。而post方法向服务器请求数据，不存在这个方面不足。

1. $.post()请求数据

$.post(url,[data],[callback],[type])

与$.get()方法参数意义一样。

1. serialize() 简化传参方式

$.post与$.get()向服务器传递参数时，其中的参数是通过名称属性逐个搜索输入字段方式进行传输，如果表单输入字段过多，这两方法叫麻烦，且不利于扩展。为了解决这个问题，jQuery引入serialize方法。

serialize()

将选择的DOM元素转化成能随AJAX传递的字符串，即序列化所选的DOM元素。

注：serialize本身有很多不足

1. $.ajax()方法
2. $.ajax([options])

option --- 为方法中的请求设置，格式为key/value

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| url | String | 发送请求的地址（默认当前页面） |
| type | String | 数据请求方式（post或get）,默认get |
| data | String或object | 发送到服务器的数据，不是字符串的转为字符串格式，如果是get方式，字符串将附在url的后面 |
| dataType | String | 服务器返回的数据类型，其类型有：html,script,text,xml,json |
| beforeSend | Function | 用于发送请求前修改XMLHttpRequest对象 |
| complete | Function | 请求完成后的回调函数，无论请求成功与否都会调用，其中有两个参数，XMLHttpRequest对象和strStatus，用于描述成功请求类型的字符串。 |
| success | Function | 请求成功后的回调函数 |
| error | Function | 请求失败后的回调函数 |
| timeout | Number | 请求超时的时间，单位毫秒 |
| global | Boolean | 是否响应全局事件，默认是true |
| async | Boolean | 是否为异步请求，默认是true |
| cache | Boolean | 是否进行页面缓存，true表示进行缓存 |

1. $.ajaxSetup()设置全局Ajax

$.ajaxSetup([option])

option --- 对象

1. Ajax中的全局事件

使用$.ajax()时，参数global默认是true，即所有在执行的Ajax的数据请求，都绑定了全局事件。

1. ajaxComplete(callback) Ajax请求完成时执行函数
2. ajaxError(callback) Ajax请求发生错误时执行函数
3. ajaxSend(callback) Ajax请求发送前执行函数
4. ajaxStart(callback) Ajax请求开始时执行函数
5. ajaxStop(callback) Ajax请求结束时执行函数
6. ajaxSuccess(callback) Ajax请求成功时执行函数
7. jQuery常用插件
8. jQuery插件概述
9. 验证插件validate
10. 表单插件form
11. Cookie插件cookie
12. 搜索插件AutoComplete
13. 图片灯箱插件notesforlightbox
14. 右键菜单插件contextmenu
15. 图片放大镜插件jqzoom
16. 自定义jQuery插件
17. jQueryUI插件
18. jQueryUI

jQueryUI侧重于用户界面的体验

1. jQueryUI交互性插件
2. jQueryUI微型插件
3. jQuery实用工具函数
4. 什么是工具函数

直接依附于jQuery对象，针对jQuery对象本身定义的方法，即全局性的函数，称为工具函数，或Utilities函数。

一般格式：$.函数名()或jQuery.函数()

1. 工具函数分类
2. 浏览器的检测$.browser() $.support.boxModel
3. 浏览器类型检测

获取浏览器的名称或版本信息，可以通过$.browser()对象属性获取

1. 浏览器特征检测

检测浏览器是否支持标准的W3C盒子模型，$.support.boxModel

1. 数组和对象的操作
2. 遍历$.each(obj,fn(para1,para2))
3. 数据筛选 $.grep(array,fn(elementOfArray,indexInArray),)

$.grep(array,fn(elementOfArray,indexInArray),[invert])

array – 要筛选的原数组

elementOfArray --- 数组中的元素

indexInArray --- 元素下标

invert --- 布尔值，是否根据fn的规则去反向结果，默认为false

1. 数据变更 $.map()

$.map(array,fn(elementOfArray,indexInArray))

array – 要变更的原数组

elementOfArray --- 数组中的元素

indexInArray --- 元素下标

1. 数据搜索 $.inArray()

$.inArray(value,array)

value – 要搜索的对象

array --- 搜索对象的数组

1. 字符串操作 $.trim()

$.trm(str) 删除空格

1. 测试操作
2. $.isArray(obj)
3. $.isFunction(obj)
4. $.isEmptyObject(obj)
5. $.isPlainObject(obj)
6. $.contains(obj)
7. URL操作 $.param()

$.param(obj,[traditional])

obj --- 需要进行序列化的对象

traditional ---- 是否使用普通方式浅层序列化

1. 工具函数扩展 $.extend()
2. 其他工具函数-----$.proxy(fn,scope)

$.proxy(fn,scope)或$.proxy(scope,name)

1. jQuery性能优化与最佳实践
2. 优化选择器执行的速度
3. 优先使用id与标签选择器
4. 避免访问语法

|  |
| --- |
| //id号修饰ID号  **$**(**"#div1 #div2"**)  //避免使用tag或class来修改ID号  **$**(**".div1 #div2"**) **$**(**"div #div2"**)  //尽量使用tag访问元素的属性  **$**(**"div[title='tmp']"**) |

1. 使用jQuery对象缓存，即先用变量保存

|  |
| --- |
| **var** obj = **$**(**"#div"**);//先使用变量进行保存 obj.bind(**"click"**,**function** () {alert(**"hello"**);}) |

1. 给选择器一个上下文
2. 处理选择器中的不规范元素标志
3. 优化事件中的冒泡现象
4. 使用data方法缓存数据
5. 解决jQuery库与其他库的冲突
6. 使用子查询优化选择器性能
7. 减少对DOM元素直接操作