jQueryDom操作

DOM节点的创建

创建元素节点：$(“html结构”) : $("<div></div>")

**内容、属性、子节点、都可以直接加进去：**

（例）：$("<div class='right'><div class='aaron'>动态创建DIV元素节点</div></div>")

DOM节点的插入

**append(content)== appendChild：个匹配的元素追加内容；**

**appendTo(content)==A).append(B):把所有匹配的元素追加到另一个指定的元素中、元素集合中；**

**.after在某标签之后**

**.before在某标签之前**

before与after都是用来对相对选中元素外部增加相邻的兄弟节点

2个方法都是都可以接收HTML字符串，DOM 元素，元素数组，或者jQuery对象，用来插入到集合中每个匹配元素的前面或者后面

2个方法都支持多个参数传递after(div1,div2,....)

**相应的还可以在被选元素之前插入，jQuery提供的方法是prepend与prependTo**

$('.aaron1').prepend('<p>prepend增加的p元素</p>')，在要插入的元素之后写方法

$('<p>prependTo增加的p元素</p>').prependTo($('.aaron2'))，与上面相反，在要插入的元素之前写方法

**与内部插入处理一样，jQuery由于内容目标的位置不同，然增加了2个新的方法insertAfter与insertBefore**

$('<p style="color:red">测试insertBefore方法增加</p>', '<p style="color:red">多参数</p>').insertBefore($(".test1"))//这里test1**作为参数在后面，要插入的元素在前面**

$('<p style="color:red">测试insertAfter方法增加</p>', '<p style="color:red">多参数</p>').insertAfter($(".test2"))//这里test2**作为参数在后面，要插入的元素在前面**

DOM节点删除之empty()的基本用法

empty清空方法，但是**与删除又有点不一样，因为它只移除了 指定元素中的所有子节点**。

**remove()的有参用法和无参用法：**

通过remove方法**移除div及其内部所有元素**，**remove内部会自动操作事件销毁方法**

**remove可以传递一个选择器表达式用来过滤将被移除的匹配元素集合，可以选择性的删除指定的节点--例如：$("p").remove(":contains('3')")**

**empty方法与remove方法的区别**

**empty方法**

严格地讲，empty()方法并不是删除节点，而是清空节点，它能清空元素中的所有后代节点

empty不能删除自己本身这个节点

**remove方法**

该节点与该节点所包含的所有后代节点将同时被删除

提供传递一个筛选的表达式，删除指定合集中的元素

**保留数据的删除操作detach()**

if (!$("p").length) return; //去重

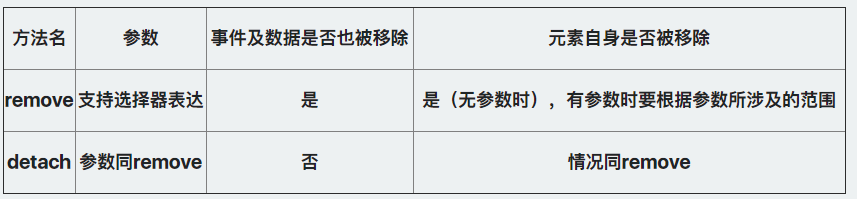
//通过detach方法删除元素

//只是页面不可见，但是这个节点还是保存在内存中

//数据与事件都不会丢失

p = $("p").detach()

**remove与detach**区别：



**remove：移除节点**

无参数，移除自身整个节点以及该节点的内部的所有节点，包括节点上事件与数据

有参数，移除筛选出的节点以及该节点的内部的所有节点，包括节点上事件与数据

**detach：移除节点**

移除的处理与remove一致

与remove()不同的是，所有绑定的事件、附加的数据等都会保留下来

例如：$("p").detach()这一句会移除对象，仅仅是显示效果没有了。但是内存中还是存在的。当你append之后，又重新回到了文档流中。就又显示出来了。

DOM节点的复制与替换

**.clone()方法** 深度 复制所有匹配的元素集合，包括所有匹配元素、匹配元素的下级元素、文字节点。

**如果节点有事件或者数据之类的其他处理，我们需要通过clone(ture)传递一个布尔值ture用来指定，这样不仅仅只是克隆单纯的节点结构，还要把附带的事件与数据给一并克隆了**

**替换replaceWith()和replaceAll()**

**.replaceWith( newContent )：用提供的内容替换集合中所有匹配的元素并且返回被删除元素的集合**

**.replaceAll()和.replaceWith()功能类似，但是目标和源相反，用上述的HTML结构，我们用replaceAll处理**

**包裹wrap()方法**

**.wrap( wrappingElement )：在集合中匹配的每个元素周围包裹一个HTML结构**

**.wrap( function ) ：一个回调函数，返回用于包裹匹配元素的 HTML 内容或 jQuery 对象**

**包裹unwrap()方法**

**作用与wrap方法是相反的。将匹配元素集合的父级元素删除，保留自身（和兄弟元素，如果存在）在原来的位置。**

**wrapAll()方法**

**.wrapAll( wrappingElement )：给集合中匹配的元素增加一个外面包裹HTML结构**

**wrapInner()方法**

**.wrapInner( wrappingElement )：给集合中匹配的元素的内部，增加包裹的HTML结构**

jQuery如何获取元素的内容

**js中获取元素内容的方式**

**表单 value**

**普通 innerText innerHTML**

**以下是jQuery中获取元素内容的方式**

**表单 val()**

**普通 text() html()**

**val() text() html() 使用时，如果不传参数则为获取，如果传入了字符串参数，则为赋值**！！！

jQuery遍历之each()

**.each() 方法就是一个for循环的迭代器，它会迭代jQuery对象合集中的每一个DOM元素。每次回调函数执行时，会传递当前循环次数作为参数(从0开始计数**

**$("li").each(function(index, element) {**

**$(this).css('color','red')**

**})**

**$.each($("div"), function(index, ele)){**

**}**

校验插件：

**优点：简单的校验使用插件更加便捷**

**缺点：提供的校验规则有限，不完整，校验规则的触发方式比较bug，**已存在的校验还存在瑕疵比如说邮箱的校验！

**校验规则：（常用的）**

**required="true" 非空**

**rangelength="[6, 10]" 字符串长度**

**minlength="6" 最小长度**

**equalTo="#pwd"（id名） 验证密码是否一样**

**email="email" 邮箱判断**

**dateISO="true" 日期判断**

**然后表单执行$(“form”). validate();**

bootstrap

**简洁、直观、强悍的前端开发框架，让web开发更加迅速、简单！**

**核心的功能：全局的样式、栅格系统、组件**

**（1）、全局样式(bootstrap的样式分为两种：标签选择器为系统标签设置了基础样式，提供了许多的类来设置样式，我们使用样式时，只需要为标签添加类名即可！)**

**(2)、栅格系统(响应式布局)**

**总共屏幕会分为12份！**

在大屏幕下 **col-lg-份数** 一行显示6个

在中等屏幕下 **col-md-份数**  一行显示4个

在小屏幕下 **col-sm-份数**  一行显示3个

在超小屏幕下 **col-xs-份数** 一行显示两个

**栅格系统的最外层使用的类为container-fluid**

**内部承载列的类的为行 类为row**

**列的类有四个 col-lg- col-md- col-sm- col-xs-**

如果针对每一个屏幕尺寸都设置了列的类，那么就按照设定的值来显示

如果只针对某一个屏幕尺寸设置了列的类，那么当前这个屏幕尺寸与比其大的屏幕尺寸都生效++++但是比其小的屏幕尺寸不起作用

栅格系统下列与列之间可以使用列偏移的类：

col-lg-offset-份数

col-md-offset-份数

col-sm-offset-份数

col-xs-offset-份数

**偏移也有传递性，设置在屏幕尺寸小的屏幕下的列偏移，会传递到大屏幕尺寸下！**

传上不传下

**组件:** **使用时直接拷贝代码即可.**

**前端的重点**

**1、表格table**

**2、表单form 元素：input select...option textarea**

**3、js的语法、jQuery的语法，js的dom操作代码。（创建、删除、赋值、获取元素）**

**4、ajax！！！**