**jQuery**

**jQuery基本操作**

**jquery**中的window.onload

$(callBack);

$(docuement).ready(callBack)

$().ready(callBack)

**获取元素**

**$(‘选择器’)**

**jquery**获取到的元素都是集合，如果对它进行操作的话，操作的是整个集合。

获取到一个不存在的元素，会获取到一个空对象{}，我们要通过它的length属性去做判断，判断一个元素存不存在，不能直接判断它本身，要判断它的length属性。

**length** 获取到集合里元素的个数

.size() 获取到集合里元素的个数(但已经废弃了)

如果想让原生方法获取到的元素具有jquery的方法

用$转成jquery对象

例：var box=document.getElementById("box");

//box.css('background','green'); //报错

$(box).css('background','green');

**添加事件**

jquery中的事件是不需要加on

添加事件：元素.click(callBack)

jquery中事件里的this是指向原生获取的对象

在jquery事件里用this的话需要转一下**$(this)**

**链式操作**

能够进行链式操作是因为，一个方法调用完成后会返回一个对象

这个对象又可以去调用其它的方法

事件的话会按正常逻辑去执行

**jQuery方法**

**.eq() 指定元素**

指定的索引的一个元素。

参数是一个整数，整数是从前往后数，负数是从后往前。

.**css() 操作样式**

如果参数为1个，那代表获取这个属性的值 $('#box').css('width')

如果参数为2个，那代表修改这个属性的值 $('#box').css('width','30')

如果要修改多个属性，可以作为一个对象传进去 $('#box').css({width:200,height:300,background:'yellow'});

如果想要获取多个属性的值，可以把它们存到一个数组当中，传进去。得到的结果为一个对象 $('#box').css(['width','height','background','border'])

**.val()**

获取表单元素的value值

如果有参数的话。代表修改value值 $('#btn').val('这是一个新按钮');

**.addclass()**

添加class addClass('active')

**Removeclass**

删除class removeClass('active')

addClass('active').siblings().removeClass('active');

.**attr()**

属性操作

一个参数返回这个参数的值 $('input').eq(0).attr('type')

两个参数代表修改 $('input').eq(0).attr('id','btn2');

如果要设置多个属性的值，可以存到一个对象当中，把对象作为参数传进去 $('input').eq(0).attr({"type":'checkbox',"id":"chek","value":'选框'});

attr()取到的内容是字符串，用作判断条件时要注意

**.data()**

自定义属性，存的是数据

可以把一个对象当作参数传进去$('#btn').data({"name":'kaivon',"age":18});用的时候直接调用.data()来获取，获取到一个对象{name: "kaivon", age: 18}

如果存一个数据，两个参数中间要用逗号$('#btn').data("name",'kaivon')

**.html()** 相当于innerHTML

设置的时候，如果参数里有html标签，可以自动解析成真正的html标签

**.text()**

设置元素里面的文字

设置的时候，如果参数里有html标签，不会把html标签解析成真正的标签，会把它当作文字处理

**DOM方法**

**创建元素**

$('<li>white</li>')

**元素** **.parent()**

元素的父级

元素.parents() 元素的所有父级

**元素.siblings()**

除自身外所有的兄弟节点。可以传参，选着兄弟节点类型。

**元素集合.fist()**

第一个节点

取到的是第一层的节点

**元素集合.last()**

最后一个节点

取到的是第一层的节点

**元素集合.slice(start,end)**

start开始位置，包含它

end结束位置，但是不包含它

**父级.children()**

取到子节点，取到的是第一层的节点

**父级.find(‘选择器’)**

查找子孙节点，查到的节点没有层级限制，不包含自己

**元素** .**closest(‘选择器’)**

找到离他最近的节点

必需有嵌套的层，它一层一层往外找，直到找到最近的，满足参数的元素

它会从自己开始找

**父级.append(元素)**  把参数添加到父级的最后面

**父级.prepend(元素)** 把参数添加到父级的最前面

**元素.before(元素)**  把参数插入到元素的前面

**元素.after(元素)** 把参数插入到元素的后面

元素.appendTo(父级) 创建的元素添加到父级的最后面

元素.prependTo(父级) 创建的元素添加到父级的最前面

元素.insertBefore(被插入的元素) 把创建的元素插入到参数的前面

元素.insertAfter(被插入的元素) 把创建的元素插入到参数的后面

**元素.remove()**

$('li').first().remove();

删除元素

**元素.clone(boolean)**

参数默认为false，代表克隆的元素没有事件

参数为true，代表克隆的元素有事件

**元素**.**index()**

$('li').index('li.pink')

找到当前元素在它的兄弟节点当中的索引值

如果有参数代表，找到参数对应的对象在前面元素里的位置

**元素.prev()**

上一个节点

**元素.next()**

下一个节点

**包装对象**

**. wrap()**  在原来的标签外面嵌套一层标签（参数）

**. wrapAll()** 把所有的标签都拿过来，统一在它们的外面加一层标签（参数）

**. wrapInner()** 在原来的标签里面嵌套一层标签（参数）

**. unwrap()** 把标签的父级去掉

**. get(index)**

原生转jquery $()

jquery转原生.get(index)

**.width()**

元素的width值

如果有参数，代表设置宽度

. **height()**

元素的height值

如果有参数，代表设置高度

**innerWidth()**

width+padding

**innerHeight()**

height+padding

**outerWidth()**

width+padding+border

如果有参数即参数是true(参数是布尔值，默认没有是false)，代表width+padding+border+margin

**outerHeight()**

height+padding+border

如果有参数即参数是true(参数是布尔值，默认没有是false)，代表width+padding+border+margin

jquery可以获取到隐藏元素的属性（用的方法是getComputedStyle）,原生的方法获取不到隐藏元素的属性

**可视区的尺寸**

$(window).width()

$(window).height()

**页面内容的尺寸**

$(document).width()

$(document).height()

**滚动条的距离**

scrollTop()

scrollLeft()

如果有参数，代表设置滚动条的距离，有版本兼容问题

原生滚动条的距离

document.body.scrollTop || document.documentElement.scrollTop

window.pageYoffset

设置滚动条的距离

window.scrollTo(x,y)

.**offset()**

.offset().left

.offset().top

相对于文档的位置 与原生js的getBoundingclientRect不同

**. position()**

.position().left

.position().top

相对于有定位父级的位置

没有定位的父级的话是相对于可视区的位置

与offsetLeft/offsetTop的区别

原生offset是包含margin值

position不包含margin值

原生offset如果没有定位父级，相对于body取值

position如果没有定位父级，相对于可视区取值

offsetParent

找到最近的有定位的父级，如果没有找到，返回可视区的距

**事件**

jquery中的事件

都是绑定事件，不会覆盖

**添加事件**

1. 元素.事件(function(){});
2. 元素.on(‘事件 事件 事件’，function(){}) $('#box').on('click mouseover',function(){console.log(3);});
3. 自定义事件

$('#box').on('click.kaivon',function(){

console.log(4);

});

调用自定义事件 $('#box').trigger('kaivon');（这是直接调用，还可以用事件调用）。

**移除事件**

**元素.off(**事件名|有名称的事件名**)**

$(this).off('.drag'); 通过事件的标识符去清除对应的事件

**调用事件**

**.Trigger**(事件名|自定义事件名|有名称的事件名) $('div').trigger('kaivon'); 页面加载完成之触发自定义事件

$('div').click(function(){$('div').trigger('miaov');}) div点击后触发自定义事件

$('#box').trigger('click');调用所有的click事件

**事件对象**

ev.clientX|Y 到可视区左侧或顶部的距离

ev.pageX|Y 到文档左侧或顶部的距离

ev.keyCode 键盘按下的键的编码

ev.which 与ev.keyCode类似，但不存在兼容性问题，推荐使用。

**阻止事件冒泡与默认事件**

**return false** 整合了阻止事件冒泡与默认事件。

**事件委托$(ev.target)**

在jquery中ev.target获取到的是原生的元素，如果在jqury中使用的话，要转为**$(ev.target)**

**检测数据类型方法**

以上这个方法是jquery中的方法，但是原生js可以使用

**$.type()** 判断数据类型

**$.isFunction()** 判断参数是不是一个函数

**$.isNumeric()** 判断参数是不是一个数字

**$.isArray()**  判断参数是不是一个数组

**$.isWindow()** 判断参数是不是window对象

**$.isEmptyObject()** 判断参数是不是一个空对象

**$.isPlainObject()** 判断参数是不是一个纯对象

**运动**

**.show**(duration时间,easing运动效果,complete回调函数) 必须要有时间参数

**.hide**(duration时间,easing运动效果,complete回调函数) 必须要有时间参数

**.toggle**(duration,easing,complete) 在以上两种效果之间的切换

easing： linear 匀速（默认的）swing 缓冲

complete 运动完成后触发的函数

**.fadeIn**(duration,easing,complete) 淡入 //默认有时间参数

**.fadeOut**(duration,easing,complete) 淡出 fadeOut运动完成后，会display为none

.slideDown(duration,easing,complete) 往下滑 //默认有时间参数

.slideUp(duration,easing,complete) 往上滑

.slideToggle(duration,easing,complete) 两种效果之间的切换

**.animate(**properties,duration,easing,complete)

properties 要运动的属性，它是一个对象, 属性值里只有加号，代表目标值为加号后面的值,属性值里有+=，代表点击一次就加符号后面的值, 属性值为toggle，代表从0到设置的属性值之间切换

duration 运动的时间（默认为400）

easing 运动方式（默认为linear）

complete 回调函数

**delay(time)** 延迟，参数为延迟的时间

**stop()** 停止运动，只停止当前的运动，点击的时候正在运动哪个属性，就把哪个属性停止

**stop(true)** 停止所有的运动

**stop(true,true)** 当前次的运动会跳到目标点，后面的运动也不会走了

**.finish();**效果与stop(true,true)效果一样