原生JS和jQuery的对比使用

1.入口函数

```
js:
window.onload = function(){js代码}
```
实质就是一个事件，拥有事件的三要素，事件源，事件，事件处理程序。等到所有内容，以及我们的外部图片之类的文件加载完了之后，才会去执行。只能写一个入口函数。

```
jq:
$(function(){})
$(document).ready(function(){})
```
在html所有的标签都加在之后就会去执行，可以写多个。

2.获取元素

```
js:
 document.getElementsByTagName("a") 获取标签是a的,返回的是个集合
 document.getElementById("demo") 获取到ID是demo的对象
 document.getElementsByClassName("class") 获取到类名是一致的所有对象  有兼容性问题
 document.getElementsByTagName("*")  获得所有的标签，用来遍历  html5新增选择器
 document.querySelector(selector) 可以通过CSS选择器的语法找到DOM元素，返回第一个满足选择器条件的元素 一个dom对象
 document.querySelectorAll('.item');返回所有满足该条件的元素 一个元素类型是dom类型的数组
```

```
jq:
$("")
```

选择器	实例	选取
*	$("*")	所有元素
#id	$("#lastname")	id="lastname" 的元素
.class	$(".intro")	所有 class="intro" 的元素
element	$("p")	所有 <p> 元素
.class.class	$(".intro.demo")	所有 class="intro" 且 class="demo" 的元素
:first	$("p:first")	第一个 <p> 元素
:last	$("p:last")	最后一个 <p> 元素
:even	$("tr:even")	所有偶数 <tr> 元素
:odd	$("tr:odd")	所有奇数 <tr> 元素
:eq(*index*)	$("ul li:eq(3)")	列表中的第四个元素（index 从 0 开始）
:gt(*no*)	$("ul li:gt(3)")	列出 index 大于 3 的元素
:lt(*no*)	$("ul li:lt(3)")	列出 index 小于 3 的元素
:not(*selector*)	$("input:not(:empty)")	所有不为空的 input 元素
:header	$(":header")	所有标题元素 <h1> - <h6>
:animated		所有动画元素
:contains(*text*)	$(":contains('W3School')")	包含指定字符串的所有元素
:empty	$(":empty")	无子（元素）节点的所有元素
:hidden	$("p:hidden")	所有隐藏的 <p> 元素

选择器	实例	选取
:visible	$("table:visible")	所有可见的表格
s1,s2,s3	$("th,td,.intro")	所有带有匹配选择的元素
[*attribute*]	$("[href]")	所有带有 href 属性的元素
[*attribute=value*]	$("[href='#']")	所有 href 属性的值等于 "#" 的元素
[*attribute!=value*]	$("[href!='#']")	所有 href 属性的值不等于 "#" 的元素
[*attribute$=value*]	='.jpg']")	所有 href 属性的值包含以 ".jpg" 结尾的元素
:input	$(":input")	所有 <input> 元素
:text	$(":text")	所有 type="text" 的 <input> 元素
:password	$(":password")	所有 type="password" 的 <input> 元素
:radio	$(":radio")	所有 type="radio" 的 <input> 元素
:checkbox	$(":checkbox")	所有 type="checkbox" 的 <input> 元素
:submit	$(":submit")	所有 type="submit" 的 <input> 元素
:reset	$(":reset")	所有 type="reset" 的 <input> 元素
:button	$(":button")	所有 type="button" 的 <input> 元素
:image	$(":image")	所有 type="image" 的 <input> 元素
:file	$(":file")	所有 type="file" 的 <input> 元素
:enabled	$(":enabled")	所有激活的 input 元素
:disabled	$(":disabled")	所有禁用的 input 元素
:selected	$(":selected")	所有被选取的 input 元素
:checked	$(":checked")	所有被选中的 input 元素

3.相互转换

```
js==>jq 得到jQuery对象
var btn = document.querySelector('#btn');
$("btn")


jq==>js 得到DOM对象
$("#btn").get(0)
$("#btn")[0]
```

4.事件处理程序

```
js:
var btn = doncument.querySelector('#btn');
btn.onclick=function(){代码块}
btn.addEventListener('click',function(){代码块})

//事件
onclick 单击事件
onblur 失去焦点事件
onchange 当对象或选中区的内容改变时触发。
onmouseover 当用户将鼠标指针移动到对象内时触发。
onmouseout 当用户将鼠标指针移出对象边界时触发。
onsubmit 当表单将要被提交时触发
onload 加载事件
//补充
1、onmouseover、onmouseout：鼠标经过时自身触发事件，经过其子元素时也触发该事件；（父亲有的
```

东西，儿子也有）
2、onmouseenter、onmouseleave：鼠标经过时自身触发事件，经过其子元素时不触发该事件。（父亲的东西就是父亲的，不归儿子所有）
这四个事件两两配对使用，onmouseover、onmouseout一对，onmouseenter、onmouseleave一对，不能混合使用。

```
jq:
$("#btn").click(function(){代码块})
```

	事件	说明
鼠标事件	click	单击
	dblclick	双击
	mouseenter	当鼠标指针穿过元素时，会发生 mouseenter 事件。
	mouseleave	当鼠标指针离开元素时，会发生 mouseleave 事件
	hover	hover()方法用于模拟光标悬停事件。
	mouseup()	当在元素上松开鼠标按钮时，会发生 mouseup 事件。
	mousedown()	当鼠标指针移动到元素上方，并按下鼠标按键时，会发生 mousedown 事件
键盘事件	keypress	键被按下
	keydown	键按下的过程
	keyup	键被松开
表单事件	submit	当提交表单时，会发生 submit 事件。该事件只适用于 <form> 元素。
	change	当元素的值改变时发生 change 事件（仅适用于表单字段）。
	focus	当元素获得焦点时（当通过鼠标点击选中元素或通过 tab 键定位到元素时），发生 focus 事件
	blur	当元素失去焦点时发生 blur 事件。
文档/窗口事件	load	load() 方法添加事件处理程序到 load 事件。//jQuery 版本 1.8 中被废弃，在 3.0 版本被移除。
	resize	当调整浏览器窗口大小时，发生 resize 事件。
	scroll	当用户滚动指定的元素时，会发生 scroll 事件。
	unload	当用户离开页面时，会发生 unload 事件。//jQuery 版本 1.8 中被废弃，在 3.0 版本被移除。

5.设置类

```
js:
document.querySelector('#btn').className="active";
document.querySelector('#btn').classList.add("active")
可以同时设置多个类名。
//补充:
document.querySelector('#xx').setAtrribute('className','class2');
document.querySelector('#xx').classList.add(".xxx");添加一个或多个类
document.querySelector('#xx').classList.remove(".xxx");移除一个或多个类
document.querySelector('#xx').classList.toggle(".xxx");存在就删除,没有就添加
document.querySelector('#xx').classList.contains('className'); 是否有该类
```

```
jq:
addClass() ==> $(".XX").addClass("xxx"); 添加一个或多个类
removeClass() ==> $(".XX").removeClass("xxx");移除一个或多个类
toggleClass() ==> $(".XX").toggleClass("xxx");存在就删除,没有就添加
hasClass() ==> $(".XX").hasClass("xxx");检查是否存在这个类名
css() ==> $(".XX").css("color","red");
读操作：获取匹配元素集合中第一个元素的指定样式值(一个或多个)、读取多个样式值的操作是在jQuery v1.9才加入的
写操作：为匹配元素集合中的每一个元素设置一个或多个CSS属性的值、传入的参数可以是单个的键值对、
```

也可以是PlainObject指定的多个值
attr() ==> $(".XX").attr("class","xxx");attr()方法设置或返回被选元素的属性值、根据该方法不同的参数、其工作方式也有所差异
prop() ==> $(".xx").prop("class","xxx")

6.设置内容，html和text

```
js:
document.querySelector('#xx').innerHTML = "<h1>内容</h1>"
document.querySelector('#xx').innerText = "内容"
document.querySelector('#xx').value = "" 获取表单元素


jq:
$("#XX").html()
1.普通元素内容html()(相当与原生的innerHTML)
$("#XX").html()//获取元素内容
$("#XX").html("内容")//设置元素的内容
1.普通元素内容text()(相当与原生的innerText)
$("#XX").text()//获取元素内容
$("#XX").text("内容")//设置元素的内容
1.获取表单内容val()(相当与原生的value)/*val() 方法返回或设置被选元素的值。
元素的值是通过 value 属性设置的。该方法大多用于 input 元素。如果该方法未设置参数，则返回被选元素的当前值*/
$("#XX").val()//获取表单的值
$("#XX").val("内容")//设置表单的值
```

7.属性

```
js:(一般用于自定义属性)
document.querySelector('#xx').getAttribute('属性名')    获取元素属性
document.querySelector('#xx').setAttrbute('属性名', value)    设置元素属性
document.querySelector('#xx').removeAttrbute('属性名')  移除属性 用于操作自定义属性


jq:
attr() ==> $(".XX").attr("属性名","属性值");attr()方法设置或返回被选元素的属性值、根据该方法不同的参数、其工作方式也有所差异
prop() ==> $(".xx").prop("属性名","属性值")
```

8.节点

```
js:
document.createElement("p");//创建一个p标签
ele.appendChild(newNode);//生JS向子节点列表的末尾添加新的子节点
ele.insertBefore(newNode, targetNode);//原生JS在节点的已有子节点之前插入一个新的子节点
ele.parentNode.removeChild(ele);//移除节点


jq:
$('<p></p>');//创建一个p标签
$('#xx').append('<p>Hello World.</p>');//在匹配元素子节点列表结尾添加内容
$('#xx').appendTo('<p>Hello World.</p>');//把匹配元素添加到目标元素子节点列表结尾
$('#xx').prepend('<p>Hello World.</p>');//在匹配元素子节点列表开头添加内容
$('#xx').prependTo('<p>Hello World.</p>');//把匹配元素添加到目标元素子节点列表开头
$('#xx').before('<p>Hello World.</p>');//在目标元素前添加
$('#xx').after('<p>Hello World.</p>');//在目标元素后添加
//$('#xx').remove();//删除节点
$("ul").remove(); //可以删除匹配的元素 自杀
$("ul").empty(); // 可以删除匹配的元素里面的子节点 孩子
$("ul").html(""); // 可以删除匹配的元素里面的子节点 孩子
```

9.数组操作

```
js:
数组元素的添加
arrayObj. push();// 将一个或多个新元素添加到数组结尾，并返回数组新长度
arrayObj.unshift();// 将一个或多个新元素添加到数组开始，数组中的元素自动后移，返回数组新长度
arrayObj.splice();//将一个或多个新元素插入到数组的指定位置，插入位置的元素自动后移，返回""。
数组元素的删除
```

arrayObj.pop(); //移除最后一个元素并返回该元素值
arrayObj.shift(); //移除最前一个元素并返回该元素值，数组中元素自动前移
arrayObj.splice(deletePos,deleteCount); //删除从指定位置deletePos开始的指定数量deleteCount的元素，数组形式返回所移除的元素
数组的截取和合并
arrayObj.slice(start, [end]); //以数组的形式返回数组的一部分，注意不包括 end 对应的元素，如果省略 end 将复制 start 之后的所有元素
arrayObj.concat(); //将多个数组（也可以是字符串，或者是数组和字符串的混合）连接为一个数组，返回连接好的新的数组
数组的拷贝
arrayObj.slice(0); //返回数组的拷贝数组，注意是一个新的数组，不是指向
arrayObj.concat(); //返回数组的拷贝数组，注意是一个新的数组，不是指向
数组元素的排序
arrayObj.reverse(); //反转元素（最前的排到最后、最后的排到最前），返回数组地址
arrayObj.sort(); //对数组元素排序，返回数组地址
数组元素的字符串化
arrayObj.join(','); //返回字符串，这个字符串将数组的每一个元素值连接在一起，中间用 ,隔开。

toLocaleString 、toString 、valueOf：可以看作是join的特殊用法，不常用

jq:
jq对数组的封装
$.each(object,[callback])//通用例遍方法，可用于例遍对象和数组。不同于例遍 jQuery 对象的 $().each() 方法，此方法可用于例遍任何对象。回调函数拥有两个参数：第一个为对象的成员或数组的索引，第二个为对应变量或内容。如果需要退出 each 循环可使回调函数返回 false，其它返回值将被忽略。
object:需要例遍的对象或数组。
callback:每个成员/元素执行的回调函数。

一、遍历
$.each(arr, callback(key, val));
1、回调函数拥有两个参数：第一个为对象的成员或数组的索引， 第二个为对应变量或内容
2、如果需要退出 each 循环，可使回调函数返回 false，用return false， 其它返回值将被忽略.

二、筛选
$.grep(arr, callback, invert)
1、此函数至少传递两个参数，第三个参数为true或false,对过滤函数返回值取反
2、默认invert为false, 即过滤函数返回true为保留元素. 如果设置invert为true, 则过滤函数返回true为删除元素.
3、过滤函数必须返回 true 以保留元素或 false 以删除元素.

三、转换
$.map(arr,callback(key,val))
1、将一个数组中的元素转换到另一个数组中。
2、和each函数差不多， 区别就是回调函数可以改变当前元素.返回null则删除当前元素.

四、合并
$.merge(arr1,arr2)
1、arr1后面加上arr2后返回arr1

五、判断
$.inArray(val,arr)
1、判断val是否在arr里面
2、确定第一个参数在数组中的位置, 从0开始计数(如果没有找到则返回 -1).
3、indexOf()返回字符串的首次出现位置,而$.inArray()返回的是传入参数在数组中的位置,同样的,如果找到的,返回的是一个大于或等于0的值,若未找到则返回-1.