# 初始化

$(document).ready(function(){});

$(function(){});

$().ready(function(){});

1. js的window.onload事件是等到所有内容，以及我们的外部图片之类的文件加载完了之后，才回去执行

只执行一次。

1. jQuery的入口函数 是在 html所有标签都加载之后，就回去执行。写几次执行几次。

js创建对象：

1. var obj = {};
2. var obj1 = new Object();
3. var obj2 = Object.create();

1跟2的区别：

推荐使用第一个方式

第二种方式存在效率问题，因为要new对象，会涉及到原型查找的问题。

# 选择器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **基本选择器** | | | | |
| **符号** | 说明 | | 用法 | |
| **$(“#demo”)** | 选择id为demo的第一个元素 | | $(“#demo”).css(“background”,”red”) | |
| **$(“.liItem”)** | 选择所有类名（样式名）为liItem的元素 | | $(“.liItem”). css(“background”,”red”); | |
| **$(“div”)** | 选择所有标签名字为div的元素 | | $(“div”). css(“background”,”red”); | |
| **$(“\*”)** | 选择所有元素  少用或配合其他选择器来使用 | | $(“\*”). css(“background”,”red”) | |
| **$(“.liItem,div”)** | 选择多个指定的元素，这个地方是选择出了 .liItem元素和div元素 | | $(“.liItem,div”). css(“background”,”red”) | |
| **层级选择器** | | | | |
| **符号** | | 说明 | | 用法 |
| **空格** | | 后代选择器  选择所有的后代元素 | | $(“div span”). css(“background”,”red”); |
| **>** | | 子代选择器  选择所有的子代元素 | | $(“div > span”). css(“background”,”red”) |
| **+** | | 紧邻选择器  选择紧挨着的下一个元素 | | $(“div + p”). css(“background”,”red”) |
| **~** | | 兄弟选择器  选择后面的所有的兄弟元素 | | $(“div ~ p”). css(“background”,”red”) |
| **基本过滤选择器** | | | | |
| **符号** | | 说明 | | 用法 |
| **:eq(index)** | | index是从0开始的一个数字，选择序号为index的元素。选择第一个匹配的元素。 | | $(“li:eq(1)”). css(“background”,”red”) |
| **:gt(index)** | | Index 是从0开始的一个数字，选择序号大于index的元素 | | $(“li:gt(2)”). css(“background”,”red”) |
| **:lt(index)** | | Index是从0开始的一个数字，选择小于index 的元素 | | $(“li:lt(2)”). css(“background”,”red”) |
| **:odd** | | 选择所有序号为奇数行的元素 | | $(“li:odd”). css(“background”,”red”) |
| **:even** | | 选择所有序号为偶数的元素 | | $(“li:even”). css(“background”,”red”) |
| **:first** | | 选择匹配第一个元素 | | $(“li:first”). css(“background”,”red”) |
| **:last** | | 选择匹配的最后一个元素 | | $(“li:last”). css(“background”,”red”) |
| **:contains(text)** | | 选择所有包含指定文本的元素 | |  |
| **:empty()** | | 选择所有没有子元素的元素（包括文本） | |  |
| **:has(selector)** | | 选择元素其中至少包含指定选择器匹配的一种元素 | |  |
| **:hidden** | | 所有隐藏元素 | |  |
| **:visible** | | 所有课件元素 | |  |
| **:text :chexkbox** | | input的类型 | |  |
| **属性选择器** | | | | |
| **符号** | | 说明 | | 用法 |
| **$(“a[href]”)** | | 选择所有包含href属性的元素 | | $(“a[href]”). css(“background”,”red”) |
| **$(“a[href=’itcast’]”)** | | 选择href属性值为itcast的所有a标签 | | $(“a[href=’itcast’]”). css(“background”,”red”) |
| **$(“a[href!=’baidu’]”)** | | 选择所有href属性不等baidu的所有元素，包括没有href的元素 | | $(“a[href!=’baidu’]”). css(“background”,”red”) |
| **$(“a[href^=’web’]”)** | | 选择所有以web开头的元素 | | $(“a[href^=’web’]”). css(“background”,”red”) |
| **$(“a[href$=’cn’]”)** | | 选择所有以cn结尾的元素 | | $(“a[href$=’cn’]”). css(“background”,”red”) |
| **$(“a[href\*=’i’]”)** | | 选择所有包含i这个字符的元素，可以是中英文 | | $(“a[href\*=’i’]”). css(“background”,”red”) |
| **$(“a[href][title=’我’]”)** | | 选择所有符合指定属性规则的元素，都符合才会被选中。 | | $(“a[href][title=’我’]”). css(“background”,”red”) |

input的类型选择器$(“:radio”) $(“:text”) $(“button”)

**$(this)[hasHighlight ? "removeClass" : "addClass"]("highlight")**

mouseover/mouseout事件，鼠标经过的时候会触发多次，每遇到一个子元素就会触发一次。

mouseenter/mouseleave事件，鼠标经过的时候只会触发一次

jQuery对象转换成DOM对象:

方式一：$(“#btn”)[0]

方式二：$(“#btn”).get(0)

DOM对象转换成jQuery对象：

var btn = document.getElementById(“btn”);

var $btn = $(btn);

jquery获取innerHTML $(“#id”).html()

jQuery的库冲突问题

通过调用jQuery.noConflict(); 将$控制权转移给其他库。

# DOM

1. 节点

1.查找节点 属性：$p.attr("title") 元素：$("ul li:eq(1)").text()

2.创建节点 元素（li）：文本（草莓）：属性（草莓）：$("ul").append($("<li tittle=’草莓’>草莓</li>"))

3.插入节点

append() 正常加 appendTo()逻辑反的，往里面加 prepend() prependTo()

after() insertAfter() 前正常，后逻辑反。 before() insertBefore()

4.删除节点

（1）remove var $li = $(“ul li:eq(1)”).remove(); 删除后，得到删除节点的引用,还可以操作添加给其他节点。也可以直接通过append等转移节点，可以达到删除效果。

$(“ul li:eq(1)”).remove(“li[title!=草莓]”); 传入选择器选择

（2）$(“ul li:eq(1)”).empty(); 删除内容 li还在

5.复制节点

$("ul li:eq(1)").clone().appendTo("ul"); 传入true，克隆节点事件。

6.替换节点

replaceWith() replaceAll() 前正常，后逻辑反。

7.包裹节点

wrap() 后包裹前 每个符合情况单独包裹

wrapAll() 符合元素的所有全部包裹一次

wrapInner() 包裹内容

1. 属性
2. 获取：$(“ul”).attr(“title”) 设置：$(“ul”).attr(“title”,”your title”) $(“ul”).attr({“title”:”your title”,”color”:”red”});

html() val() text() height() width() css()

1. 删除：$(“ul”).removeAttr(“title”)
2. 样式
   1. 获取：$(“ul”).attr(“class”) 设置：$(“ul”).attr(“class”,”high”)
   2. 追加：$(“ul”).addClass(“class”) 追加同一属性会覆盖 第一次调用与attr相同
   3. 移除：removeClass() 不传参数删除所有class属性，传参数删除参数
   4. 切换：$(“ul”).toggleClass(“class”)
   5. $(“ul”).hasClass(“class”) 相当于 (“ul”).is(“.another”)
3. HTML 文本 值
4. html() innerHTML 不能用于XML
5. text() innerText XML XHTML都有效 FireFox 不能用innerText 但是可用text()
6. val() 读写 下拉选择 $("select").val("22"); $("select").val(["22",”33”]);
7. 遍历节点
8. children()
9. next() 后面的同辈节点 prev()
10. siblings() 元素前后所有同辈节点
11. closest() 找最近的匹配元素，不匹配逐级向上查找。
12. find() filter() nextAll() prevAll() parent() parents()
13. CSS DOM
14. $(“li”).css(“color”) $(“li”).css(“color”,”red”) $(“li”).css({“color”:”red”,”tittle”:”title content”}) 直接写数字默认转像素

$(“li”).width() $(“li”).width(“400px”) $(“li”).width(400) $(“li”).width(“400em”)

1. offset() 获取元素的offset对象，该对象有left和top元素，左偏移和右偏移。
2. position() 获取元素相对于最近的一个position样式属性为relative absolute的祖父节点的相对偏移。
3. scrollTop() scrollLeft() 距顶端 距左端 的距离 传入值可设置滚动位置

# 事件和动画

## 事件

1. 事件绑定
2. $li.click(function(){})
3. $li.bind(“click”,function(){})
4. 合成事件
   1. $li.hover(fn1, fn2) 相当于mouseLeave mouseEnter
   2. $li.toggle(fn1,fn2) 模拟鼠标连续单击
5. 事件冒泡
   1. event.stopPropagation() event.preventDefault()

简写：return false; jquery不支持事件捕获

* 1. event.type()
  2. event.target()
  3. event.relatedTarget()
  4. event.pageX() event.pageY() 鼠标点击坐标
  5. event.which() 获取鼠标左中右键 1左,2中,3右
  6. event.metaKey() <ctrl>按键
  7. originalEvent() 指向原始事件对象
  8. $li.one(“click”,function(){}) 相当于bind 只有效一次，之后点击无作用

1. 移除事件
   1. $btn.unbind(“click”); 没有参数，删除所有绑定时间，函数作为第二个参数，只删除事件事件处理函数。只有一个参数，传入一个类型，只删除一个事件绑定。
2. 模拟操作
   1. $li.trigger(“click”) $li.click()
   2. 触发自定义事件 $li.bind(“myClick”,function(){}) $li.trigger(“myClick”)
   3. 传递数据 $li.bind(“myClick”,function(event,message1,message2){}) $li.trigger(“myClick”,[“参数一”,”参数二”])
   4. 执行默认操作

$(“input”).trigger(“focus”) 不仅会触发focus绑定的事件，也会使input本身绑定事件

$(“input”).triggerHandler(“focus”) 不触发浏览器默认操作，触发focus事件

1. 其他用法
   1. 绑定多个事件类型

$li.bind(“mouseover mouseout”,function(){})

* 1. 命名空间

$li1.bind(“click.plugin”,function(){})

$li2.bind(“mouseover.plugin”,function(){})

单击li1，会同时触发click和click.plugin事件。如果只是单击li2元素，则只触发click。后加感叹号只触发不包含在命名空间的事件。

$li3.unbind(“.plugin”) 整体解除绑定

* 1. 相同控件 不同命名空间的相同事件

$li1.bind(“click”,function(){});

$li1.bind(“click.plugin”,function(){});

$li2.bind(“click”,function(){

$li1.trigger(“click!”);

});

## jQuery中的动画

1. show () hide() 参数 slow 600ms normal 400ms fast 200ms 传入数字默认为ms 同时变高度 宽度 不透明度
2. fadeIn() fadeOut() 不透明度
3. slideUp() slideDown() slow 600ms normal 400ms fast 200ms
4. 自定义动画

$li.animate({left:”500px”},3000);

scrollTop

1. 累加、累减动画

$li.animate({left:”+=500px”},3000);

1. 多重动画 综合动画

$li.animate({left:”500px”,height:”200px”},3000); //同时执行多个动画

$li.animate({left:”500px”},3000)

.animate({height:”200px”},3000); //按顺序执行多个动画

1. 停止动画和判断是否处于动画状态

stop([clearQueue][,gotoEnd])

$li.stop(true,true)

$li.is(“:animated”)

1. toggle() slideToggle() fadeIn(600,0.2)
2. $li.animate({opacity:”show”},400) ; 相当于：$li.show(400) $li.fadeIn(400)

$li.animate({height:”show”},400) ; 相当于：$li.slideDown(400)

$li.animate({opacity:”0.6”},400) ; 相当于：$li.fadeTo(400,0.6)

1. $("#comment").animate({height: "+=50"}, 1000);
2. $comment.is(":animated") 判断处于动画状态 返回布尔值

例子：

$('tbody>tr').click(function(){

$(this).addClass("highlight")

.siblings().removeClass("highlight").end()

.find(':radio').attr("checked",true);

});

//$(':radio:checked').parent().parent().addClass("highlight");

//$(':radio:checked').parents('tr').addClass("highlight");

$('tbody>tr:has(":checked")').addClass("highlight");

$('th[class="h"]').click(function(){

$(this).parent().siblings('.child\_' + $(this).parent()[0].id).toggle(); //点击自动执行与取消

});

动态筛选

$("#filterName").keyup(function(){

$("table tbody tr").hide().filter(":contains("+ $(this).val() +")").show();

}).keyup();

$('#cssfile').attr("href", "css/" + this.id + ".css"); //jQuery

$('#cssfile')[0].href = ("css/" + this.id + ".css"); //js

var hasHighlight = $(this).hasClass('highlight');

$(this)[hasHighlight ? "removeClass" : "addClass"]("highlight")

.find(':checkbox').attr("checked", !hasHighlight);

$('tbody>tr:has(":checked")').addClass("highlight");

var index = $('div.tab\_menu>ul>li').index(this); //传入js对象