# jQuery框架

第3天课堂笔记（本课程共3天）

目录

[jQuery框架 1](#_Toc476951364)

[一、认识框架 2](#_Toc476951365)

[二、节点关系 3](#_Toc476951366)

[2.1 jQuery中的节点关系 3](#_Toc476951367)

[三、节点操作 5](#_Toc476951368)

[3.1 jQuery中的节点操作 5](#_Toc476951369)

[3.1.1 append就是往最后添加： 5](#_Toc476951370)

[3.1.2 appendTo 表示被动， 儿子appendTo父亲 5](#_Toc476951371)

[3.1.3 after 插入兄弟 before 插入兄弟 6](#_Toc476951372)

[3.1.4 insertBefore、insertAfter 6](#_Toc476951373)

[3.1.5 wrap、warpAll 6](#_Toc476951374)

[3.1.6 empty() 6](#_Toc476951375)

[3.1.7 remove() 7](#_Toc476951376)

[3.1.8 clone() 7](#_Toc476951377)

[四、jQuery中的Ajax 8](#_Toc476951378)

[4.1 $.get()方法 9](#_Toc476951379)

[4.2 $.post()方法 10](#_Toc476951380)

[4.3 $.ajax()方法 10](#_Toc476951381)

[4.4 serialize()方法 10](#_Toc476951382)

[4.5 jQuery中的JSONP跨域 10](#_Toc476951383)

[五、缓存问题解决 11](#_Toc476951384)

[5.1 随机数 11](#_Toc476951385)

[5.2 时间戳 12](#_Toc476951386)

# 一、认识框架

**计算机世界和物理化学不一样，计算机世界底层API少，逐渐一层一层的API丰富起来**。所谓的API就是你能够调用的方法、函数、语法。计算机的最底层是二进制，上面就是机器语言，再上面是汇编语言，再上面是高级语言。

汇编语言中，实际上就提供了不到100个API，比如汇编语言中求1~100的和：

|  |
| --- |
| 1. DSEG SEGMENT 2. RESULT DW ?;存放和 3. DSEG ENDS 4. CSEG SEGMENT 5. ASSUME CS:CSEG,DS:DSEG 6. START: 7. MOV AX,DSEG 8. MOV DS,AX 9. MOV AX,1;从1开始加 10. MOV RESULT,0;和的初值赋0 11. MOV CX,100;循环100次 12. NEXT: 13. ADD RESULT,AX;每次向和中累加AX 14. INC AX;然后AX加1 15. LOOP NEXT 16. MOV AX,4C00H;结束程序 17. INT 21H 18. CSEG ENDS 19. END START |

这里都是内存的变化，用内存的变化来编程。比如MOV表示把变量从内存一个地方移动到另一个地方。汇编语言非常晦涩，都是内存的变化。所以高级语言应运而生，C语言是最最著名的代表：

|  |
| --- |
| 1. int sum; 2. for(int i = 1 ; i <= 100 ; i++){ 3. sum += i; 4. } 5. prinf("%d",sum); |

c语言编译的过程，实际上就是把for这些字词翻译为汇编语言。大学里一门课程《编译原理》就是研究编译过程的。

上层的框架屏蔽了下层的语言的一些麻烦、不方便的东西，并且提供更方便的API。

[百度的七巧板](http://www.infoq.com/cn/presentations/lzx-baidu-tangram)

jQuery就是干这个事情的，把JS中的不方便封装起来，暴露的API都是非常简便的。

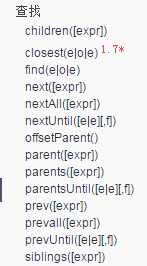
**jQuery的哲学就是DOM编程领域的霸主，操作DOM节点、绑定监听、运动、css样式、Ajax等等都有封装。**

工作上都是用jQuery，如果不用jQuery也是用类似的东西。没有人会不用轮子去开发页面效果。

JavaScript中Library表示“库”，如果这个库的功能很强大，甚至颠覆了传统编程的语法、行文习惯，我们就可以叫做“框架”。

# 二、节点关系

## 2.1 jQuery中的节点关系



感恩的心，太好用了！！注意，都是方法！！因为里面暗含了迭代器了。

● children()方法

所有亲儿子节点。就是儿子，孙子不在children()里面。

让box的所有儿子变红：

|  |
| --- |
| 1. $("#box").children().css("background-color","red"); |

儿子有很多，我们可以用选择器来表示怎么样的儿子：

|  |
| --- |
| 1. $("#box").children(**"h3"**).css("background-color","red"); |

还可以用筛选器：

|  |
| --- |
| 1. $("#box").children(**":odd"**).css("background-color","red"); |

jQuey屏蔽掉了浏览器兼容问题，不存在空文本节点也是儿子了，事实上children()返回的只能是节点，所有的文本都不算做是儿子了。

● find()

所有后代元素。和children()不一样，children()返回的是自己的亲儿子元素列表，而find()返回的是自己的后代所有元素的列表。

|  |
| --- |
| 1. $("#box").find("p").css("background-color","red"); |

注意，和children()方法不一样，find()方法里面，必须写参数，表示后代的谁？

说白了find是寻找的意思，就是你要在后代寻找谁。

● parent()

亲爸爸。任何元素都只有一个亲爸爸。

|  |
| --- |
| 1. $("p").parent().css("background-color","red"); |

● parents()

这个元素的所有的祖先节点。

● siblings()

亲兄弟

|  |
| --- |
| 1. $(".xiaoming").siblings().css("background-color","red"); |

同样的，jQuery只返回节点元素，对于文本、注释都不视为兄弟。

可以加选择器、筛选器：

|  |
| --- |
| 1. $(".xiaoming").siblings(":odd").css("background-color","red"); |

排他，妈妈再也不用担心我写for循环了：

|  |
| --- |
| 1. $(this).addClass("cur").siblings().removeClass("cur"); |

我加cur，我的兄弟们去cur。

● prev()、next()、prevAll()、nextAll()

前一个兄弟、后一个兄弟、前所有兄弟、后所有兄弟。

总结：感恩！jQuery屏蔽了所有的不兼容，只选择nodeType=1的元素，不选择文本、注释等等。

并且不用for循环了，一次性得到所有东西了，原理我们后面的课程揭示。

# 三、节点操作

HTML节点我们原来最多最多就是改改HTML属性，比如src属性改改；或者改改css样式，比如.style或者.css()。

现在的问题是，我们要增加节点、删除节点、移动节点、替换节点。

## 3.1 jQuery中的节点操作



### 3.1.1 append就是往最后添加：

父亲.append(儿子)

|  |
| --- |
| 1. $("#box").append("<p>么么哒</p>"); |

jQuery节点操作非常简单，不用createElement的过程了。实际上内部就是用innerHTML来给你加进去的。

特别的，如果我们想先创建一个孤儿节点，给这个孤儿节点加样式、加监听、加动画，可以的，那么创建孤儿节点，也是用$()函数。 $()不仅仅能够选择页面上的节点，也能够创建新的孤儿节点：

|  |
| --- |
| 1. var $obj = **$(**"<p></p>"**)**; 2. $obj.css("background-color","red"); //jQuery对象就可以直接调用css方法 3. $("#box").append($obj); |

### 3.1.2 appendTo 表示被动， 儿子appendTo父亲

|  |
| --- |
| 1. $("<p>么么哒</p>").appendTo($("#box")); |

prepend 插入为第一个儿子， prependTo表示被动

|  |
| --- |
| 1. $("<p>么么哒啊！</p>").prependTo($("#box")); |

### 3.1.3 after 插入兄弟 before 插入兄弟

比如HTML结构：

|  |
| --- |
| 1. <div id="box"> 2. <p>哈哈</p> 3. <p>嘻嘻</p> 4. </div> |

执行代码：

|  |
| --- |
| 1. $("p").after("<h3>我是h3</h3>"); |

HTML将变为：

|  |
| --- |
| 1. <div id="box"> 2. <p>哈哈</p> 3. <h3>我是h3</h3> 4. <p>嘻嘻</p> 5. <h3>我是h3</h3> 6. </div> |

### 3.1.4 insertBefore、insertAfter

比如HTML结构：

|  |
| --- |
| 1. <div id="box"> 2. <p>000</p> 3. <p>111</p> 4. <p>222</p> 5. <p>333</p> 6. <p>444</p> 7. <p>555</p> 8. </div> |

代码：

|  |
| --- |
| 1. $("<p>么么哒</p>").insertBefore($("p")[2]); |

HTML结构将变为：

|  |
| --- |
| 1. <div id="box"> 2. <p>000</p> 3. <p>111</p> 4. <p>么么哒</p> 5. <p>222</p> 6. <p>333</p> 7. <p>444</p> 8. <p>555</p> 9. </div> |

### 3.1.5 wrap、warpAll

在外面包裹

|  |
| --- |
| 1. $("a").wrap("<span></span>"); //每个a都用span包裹起来 2. $("a").wrapAll("<span></span>"); //所有a都用span包裹起来 |

### 3.1.6 empty()

|  |
| --- |
| 1. $("p").empty(); |

等价于

|  |
| --- |
| 1. $("p").html(“”); |

### 3.1.7 remove()

删除页面上所有p标签：

|  |
| --- |
| 1. $("p").remove(); |

### 3.1.8 clone()

克隆节点

克隆ul li里面的第1个，然后ul追加：

|  |
| --- |
| 1. $(“ul”).append($(“ul li”).eq(0).clone()); |

反过来：

|  |
| --- |
| 1. $(“ul li”).eq(0).clone().appendTo($(“ul”)); |

不要背API，常用的慢慢就记住了。要思考业务逻辑。

# 

# 四、jQuery中的Ajax

原生的Ajax这套“佛经”要非常熟悉，面试的时候笔试都有可能。

|  |
| --- |
| 1. var xhr = new XMLHttpRequest(); //不兼容，能力检测 2. xhr.onreadystatechange = function(){ 3. if(xhr.readyState == 4){ 4. if(xhr.status >= 200 && xhr.status < 300 || xhr.status == 304){ 5. //正面的HTTP状态码 6. alert(xhr.responseText); 7. } 8. } 9. } 10. xhr.open("post","check.php",true); 11. xhr.send("id=" + id + "&name=" + encodingURIComponent(name)); |

jQuery很重要，千万不要被忽悠，有些机构就宣称jQuery臃肿，很多公司不用，胡扯！！小公司自己如果有框架，也是模拟jQuery，没有时间成本培训员工，所以jQuery已经是事实上的标准了，大家都会jQuery，所以就当做面试的要求，进入公司之后就不用培训框架了。

大公司装×，自己的库，比如kissy、tangram，API都和jQuery很像。

组件开发中，组件可以脱离运动框架，可以自己内部有小的运动函数，是个例外。通常情况下，一个jQuery压缩包只有90k，您一个图片都200k。

把精力放在业务上。

## 4.1 $.get()方法

|  |
| --- |
| 1. **$.get**("a.json" , {"id":1,"name":"考拉"} , function(text){ 2. alert(typeof text); 3. }); |

jQuery默默的帮我们：

1） encodingURIComponent中文了；

2） 如果请求是404等错误状态，不会执行回调函数；

3） jQuery识别传回来的东西像JSON，已经帮我们转JSON了，

jQuery中get方法还可以这样使用两个参数：

|  |
| --- |
| 1. **$.get**("a.json?id=1&name=考拉" , function(text){ 2. alert(typeof text); 3. }); |

## 4.2 $.post()方法

|  |
| --- |
| 1. **$.post**("a.json",{"id":1,"name":"考拉"},function(data){ 2. alert(typeof data); 3. alert(data.content[2].id); 4. }); |

## 4.3 $.ajax()方法

配置信息非常多，写在json里面进行配置：

|  |
| --- |
| 1. $.ajax("b.json",{ 2. //请求类型 3. "type" : "get", 4. //传到服务器上的数据 5. "data" : { 6. "name":"xiaoming", 7. "age" :10 8. }, 9. //成功做的事情 10. "success" : function(data){ 11. alert(typeof data); 12. }, 13. //错误做的事情 14. "error" : function(XMLHttpRequest, textStatus, errorThrown){ 15. alert(errorThrown); 16. } 17. }); |

## 4.4 serialize()方法

用来表单序列化：

|  |
| --- |
| 1. var str = $("#biaodan").serialize(); |

注意能够调用srialize的元素，一般来说一定是form标签。

## 4.5 JSONP跨域

原理和我们上面写的一样，jQuery也帮我们创建了一个script标签，设置了src，发出了请求，定义了函数。

跨域使用$.ajax函数。

说说$.ajax函数，这是ajax最“全乎”的函数：

|  |
| --- |
| $.ajax("test.txt",{  "type" : "get",  "data" : {"a":1,"b":2},  "success" : function(data){    }  }); |

等价于：

|  |
| --- |
| $.get("test.txt",{"a":1,"b":2},function(data){  }); |

特别的，还能配置error，请求错误的时候做的事情。

|  |
| --- |
| $.ajax("t123123est.txt",{  "type" : "get",  "data" : {"a":1,"b":2},  "success" : function(data){  },  "error" : function(){  alert("系统错误");  }  }); |

如何实现JSONP？

|  |
| --- |
| $.ajax("test.txt",{  //JSON跨域的时候要写一个dataType，注意不是type而是dataType  "dataType" : "jsonp",  //要定义的函数名字，因为JSONP不缺执行，缺定义  "jsonpCallback" : "fun",  //信息回来之后执行的事情  "success" : function(data){  alert(data);  }  }); |

此时如果面试的时候，人家问，这些jQuery代码底层发生了什么，要会回答：

1. 创建了一个函数叫做fun，指向了我么你的success函数
2. 创建了一个script标签，然后设置了src为test.txt，上树，瞬间下树。

# 五、缓存问题解决

Ajax缓存极其严重，如果两次get、post都是通往同一个URL的，并且携带的参数一样，即使服务器返回200的状态码，也会被当做304那样有缓存。这样就会表现出内容更改不及时，后台的文件已经更改了，但是前台由于有缓存就不会变化。

篮球比赛图文直播就是Ajax做的，你想想看如果有缓存，后台已经比赛进行到了10分钟了，你这里还在开球呢。

## 5.1 随机数

只要更改参数，就不会有缓存！所以我们受此启发， 可以为我们的URL后面添加不影响页面使用的随机数。

|  |
| --- |
| 1. xhr.open("get","a.txt?z=" + Math.random(),true); |

甚至可以不写k：

|  |
| --- |
| 1. xhr.open("get","a.txt?" + Math.random(),true); |

jQuery中：

|  |
| --- |
| 1. $.get(“a.txt”,{“z”:Math.random()},true); |

## 5.2 时间戳

Date是系统内置的构造函数，new Date()就能产生一个日期时间对象，输出之后就是一个当前系统时间。

Date.parse(日期时间对象)就能产生一个时间戳，是从1970年1月1日到现在这一时刻的毫秒数。

|  |
| --- |
| 1. Date.parse(new Date()); |

|  |
| --- |
| 1. xhr.open("get","a.txt?z=" + Date.parse(new Date()),true); |

# 六、jQuery插件开发

## 1.什么是jQuery插件

jQuery框架本身给我们提供了一些方法供使用，但是方法的数目是有限的，其不能任意满足我们对各种功能的需求。那么我们自己可以来给jquery框架开发，扩展一些额外功能的方法。

开发、扩展jquery框架额外方法的过程就是『插件开发』

## 2.两种形式丰富方法

2.1 给$.fn(给jQuery对象)丰富

$.fn成员 = 值;

$.fn.extend(json对象);

* 1. 给jquery(给$对象)丰富

$.成员 = 值;

$.extend(json对象);

## 3.开发图片自动切换插件

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <script src="js/jquery-1.12.3.min.js"></script>  
 <script src="js/jquery-picchange.js"></script>  
 <script>  
// //实现picbofang()方法  
// $.picbofang = function(){  
//  
// var changepic = function() {  
// //获得当前图片的序号  
// var picid = $('img').attr('src').substr(7, 1);  
//// console.log(picid);  
// var nextid = ++picid;  
// if (nextid == 5) {  
// nextid = 0;  
// }  
// $("img").attr('src', "images/" + nextid + ".jpg");  
//  
// };  
// setInterval(changepic, 1000);  
// };  
  
 //页面加载完毕调用一个插件方法，使得图片依次播放  
 $(**function**(){  
 $.picbofang();  
 });  
 </script>  
</head>  
<body>  
<h2>插件开发之图片切换</h2>  
<img src="images/0.jpg" alt="" width="300" height="300">  
</body>  
</html>

jquery-picchange.js

*/\*\*  
 \* Created by maxwell on 17/3/29.  
 \*/  
/\*\*  
 date:2017/3/29 10:12  
 author: maxwelldu  
 example:  
 html\_code:  
 <h2>插件开发之图片切换</h2>  
 <img src="images/0.jpg" alt="" width="300" height="300">  
 jquery\_code:  
 $(function(){  
 $.picbofang();  
 });  
  
 \*/*//实现图片轮播效果  
$.picbofang = **function**(){  
  
 **var** changepic = **function**() {  
 //获得当前图片的序号  
 **var** picid = $('img').attr('src').substr(7, 1);  
// console.log(picid);  
 **var** nextid = ++picid;  
 **if** (nextid == 5) {  
 nextid = 0;  
 }  
 $("img").attr('src', "images/" + nextid + ".jpg");  
  
 };  
 setInterval(changepic, 1000);  
};