**基础核心**

学习要点：

1.代码风格

2.加载模式

3.对象互换

4.多个库之间的冲突

本节课我们简单的介绍一下 jQuery一些核心的问题，这些问题都粗略的为大家介绍了

jQuery的大致使用模式，为后续课程展开提供了帮助。对于 JavaScript课程已经学完的同学，这些概念会非常的清晰，而对于 JavaScript薄弱的同学可能会有一些模糊，但不必太担心，后续会慢慢展开。而对于完全没有 JavaScript基础的同学，就无法学习了。

一．代码风格

在jQuery程序中，不管是页面元素的选择、内置的功能函数，都是美元符号“$”来起始的。而这个“$”就是jQuery当中最重要且独有的对象：jQuery对象，所以我们在页面元

素选择或执行功能函数的时候可以这么写：

$(function () {});//执行一个匿名函数

$(‘#box’);//进行执行的ID元素选择

$(‘#box’).css(‘color’, ‘red’);//执行功能函数

由于$本身就是jQuery对象的缩写形式，那么也就是说上面的三段代码也可以写成如下

形式：

jQuery(function () {});

jQuery(‘#box’);

jQuery(‘#box’).css(‘color’, ‘red’);

在执行功能函数的时候，我们发现.css()这个功能函数并不是直接被“$”或jQuery对象调

用执行的，而是先获取元素后，返回某个对象再调用.css()这个功能函数。那么也就是说，

这个返回的对象其实也就是jQuery对象。

$().css(‘color’, ’red’); //理论上合法，但实际上缺少元素而报错

值得一提的是，执行了.css()这个功能函数后，最终返回的还是jQuery对象，那么也就

是说，jQuery的代码模式是采用的连缀方式，可以不停的连续调用功能函数。

$('#box').css('color', 'red').css('font-size', '50px'); //连缀

jQuery中代码注释和JavaScript是保持一致的，有两种最常用的注释：单行使用“//...”，

多行使用“/\* ... \*/”。

//$('#box').css('color', 'red');

二．加载模式

我们在之前的代码一直在使用$(function () {});这段代码进行首尾包裹，那么为什么必须

要包裹这段代码呢？原因是我们 jQuery库文件是在body元素之前加载的，我们必须等待所有的DOM元素加载后，延迟支持DOM操作，否则就无法获取到。

在延迟等待加载，JavaScript提供了一个事件为load，方法如下：

window.onload = function () {};//JavaScript等待加载

$(document).ready(function () {});//jQuery等待加载

load和ready区别

|  |  |  |
| --- | --- | --- |
|  | window.onload | $(document).ready() |
| 执行时机 | 必须等待网页全部加载完毕（包括图片等），然后再执行包裹代码 | 只需要等待网页中的DOM结构加载完毕，就能执行包裹的代码 |
| 执行次数 | 只能执行一次，如果第二次，那么  第一次的执行会被覆盖 | 可以执行多次，第N次都不会被上  一次覆盖 |
| 执行简写 | 无 | $(function(){}); |

在实际应用中，我们都很少直接去使用window.onload，因为他需要等待图片之类的大

型元素加载完毕后才能执行JS代码。所以，最头疼的就是网速较慢的情况下，页面已经全

面展开，图片还在缓慢加载，这时页面上任何的 JS交互功能全部处在假死状态。并且只能

执行单次在多次开发和团队开发中会带来困难。

面试:$(this)和this有什么不一样?

一个是Jquery的封装对象，一个是JS原生的对象

三．对象互换

jQuery对象虽然是 jQuery库独有的对象，但它也是通过 JavaScript进行封装而来的。我们可以直接输出来得到它的信息。

alert($);//jQuery对象方法内部

alert($());//jQuery对象返回的对象，还是 jQuery

alert($('#box'));//包裹 ID元素返回对象，还是 jQuery

从上面三组代码我们发现：只要使用了包裹后，最终返回的都是 jQuery对象。这样的

好处显而易见，就是可以连缀处理。但有时，我们也需要返回原生的 DOM对象，比如：

alert(document.getElementById('box'));//[object HTMLDivElement]

jQuery想要达到获取原生的 DOM对象，可以这么处理：

alert($('#box').get(0));//ID元素的第一个原生 DOM

从上面 get(0)，这里的索引看出，jQuery是可以进行批量处理 DOM的，这样可以在很

多需要循环遍历的处理上更加得心应手。

四．多个库之间的冲突

当一个项目中引入多个第三方库的时候，由于没有命名空间的约束（命名空间就好比同

一个目录下的文件夹一样，名字相同就会产生冲突），库与库之间发生冲突在所难免。

那么，既然有冲突的问题，为什么要使用多个库呢？原因是 jQuery只不过是 DOM操

作为主的库，方便我们日常 Web开发。但有时，我们的项目有更多特殊的功能需要引入其

他的库，比如用户界面 UI方面的库，游戏引擎方面的库等等一系列。

而很多库，都使用“$”作为基准起始符，如果想和 jQuery共容有两种方法：

1.将 jQuery库在 Base库之前引入，那么“$”的所有权就归 Base库所有，而 jQuery可

以直接用 jQuery对象调用，或者创建一个“$$”符给 jQuery使用。

var $$ = jQuery;//创建一个$$的 jQuery对象

2.如果将 jQuery库在 第三方库之后引入，那么“$”的所有权就归 jQuery库所有，而第三方库将会冲突而失去作用。这里，jQuery提供了一个方法：

jQuery.noConflict();

var $$ = jQuery;

$(function () {//将$符所有权剔除

alert($('#box').ge(0));

alert($$('#box').width());

});