## DOM（二）

10.1.2

2.文档信息

Js通过Document表示整个文档，在浏览器中，document对象是HTMLDocument的一个实例，而HTMLDocument继承自Document对象。document对象是window对象的一个属性。因此可以作为全局访问。

Document类型可以表示HTML页面或者其他基于XML的文档，不过最常见的还是HTMLDocument实例的document对象（不是Documet区分大小写）。

documentElment属性用来直接饮用html元素。也可document.childNodes[0]。明显前者更便利更专业。

还有一些使用频率比较高的标签，可以直接从document对象上直接引用。

比如document.body。 document.doctype;

但是doctype的实用性不强，而且各个浏览器对document.doctype的支持不一致。所以这个属性用处很有限。

也可以获取title，通过document.title。可以用这个方法修改标题。

可以通过document.URL（完整的URL即地址栏中的URL）;

Document.domain则只包含域名。



Document.referrer属性则保存着链接到当前页面的那个页面的URL。在没有来源网页的情况下，referrer属性可能是一个空字符串。

1. 查找元素

Document.getElements.....

Document.getElementsByname....

其他的没啥说的，值得注意的是:

当文档中有多个id相同的元素，通过id获取元素则只能获取第一个元素。

在这几个getElements...之中，有的是返回的NodeList类型，而有的是返回的是HTMLCollection。我实在是不能分清什么时候返回NodeList，什么时候返回HTMLCollection。如果真的需要区分的时候，就先做一个实验，在控制台打印方法获得的集合是什么类型的然后在进行下一步判断。

NodeList和HTMLCollection非常相似，想要分请必须明白他俩之间的微妙区别，都是一个动态的类数组。都支持通过item()和中括号来获取元素。只不过HTMLCollection拥有一个NodeList没有的方法，namedItem（“name..”）（Tip：如果由于很多name相同的元素，namedItem()方法只能获取到第一项）可以通过name的值获取元素。也可以通过中括号语法[‘name..’]的方式获取具体的成员（NodeList不行）。

1. 特殊集合

还有一些特殊方法也是因为常用，所以提供了快捷方式，可以直接通过document.xxx直接引用。

以下方法获取的都是HTMLCollection。

document.anchors 文档中所有具有name属性的a元素（锚点）。

document.applets 所有applets元素，这个元素不再推荐使用，（基本上浏览器都不支持applets标签，H5已经拒绝了这个标签改用object标签）所以这个属性也没啥用了。

document.forms 包含文档中的所有form元素，相当于document.getElementsByTagName(‘form’);

document.images 所有的img元素

document.links 包含所有的带有href的a元素。

1. 文档写入

主要讨论四个方法 write() writeln() open() close();

Write()方法人尽皆知不在赘述，

writeln和write相似，只不过会自动在后面加上一个换行符“/n”，但是在浏览器中显示的样子就是一个小空格而已，并没有换行。

效果如下：



open()和close()方法用于打开和关闭文档流。

Tip：书中所说的是，在页面加载期间用不上打开和关闭输出流，在页面加载完毕之后则会重写整个页面。但是经过我的实验，当script标签具有defer属性的时候，中途需要document.write或者writeln就必须打开输出流，利用document.open然后还要关闭document.close，不打开将会曝出警告，提示应该打开文档流，不关闭则就会出现页面一直加载转圈的情况。但是把script标签（无论是嵌入还是外联）写在html文档中任何一个位置（包裹中间或者是文档最后HTML标签后都不会出现这个问题），而defer则是等着解析完了执行，这也就是说defer是一个特殊的延迟，既不是相当于把script标签写在文档最后面，也不是等着整个页面解析完成之后解析，总之在defer中不允许对输入流进行操作，这里只是一个猜想，网上的资料也没有太好能解释这个的，书中对此的解释也几乎没有。

后续补充：网上的多数解释是，defer属于一个异步执行，详细见大神博客（感觉靠谱的，解释的详细的，看完差不多懂了）：

<https://www.cnblogs.com/jiasm/p/7683930.html>

基本上总结一下就是加载（下载）是在解析的过程进行的，属于解析的一部分，普通的script标签的话，渲染页面，下载，执行只能一步一步的进行，不能同时进行（下载可以一次下载好几个，但是只能是下载不能做其他的）。

而defer和async的下载甚至执行可以和页面渲染一起进行。

Defer支持让渲染页面和下载异步，下载完，也就是页面解析完成之后就可以进行按顺序执行。

Async支持让渲染页面和下载异步，不过谁先下载完就暂停页面渲染，直接执行。

所有执行过程都不能和渲染同时进行，无论是异步或者不是异步。也就是说异步只是允许下载和渲染同时进行，不允许执行和渲染同时执行，但是下载可以和执行同时进行。

实验方式：

很简单，只需要在一个外部js中写上document.write(“xxx”);然后把他放在文档最前，中间，再放到文档最后，然后在改为defer放到最前。就能看出结果。如果是嵌入的js而不是外部文件，则没有没有任何一种情况需要open和close。

需要的时候控制台会抛出警告（不是报错）:



Failed to execute 'write' on 'Document': It isn't possible to write into a document from an asynchronously-loaded external script unless it is explicitly opened.

发现错误这种错误可以选择open和close解决，或者是选择创建DOM动态添加的方式。