**多窗口和窗体**

一个web浏览器有多个标签，每一个标签页都是一个独立的浏览上下文（即，每一个独立的标签都是一个线程，有些时候，浏览器会进行线程的合并）。每一个标签页都有一个独立的window对象。而且相互之间不干扰。每个标签页中运行的脚本通常并不知道其他便签页的存在。更不会和其他标签页的window对象进行交互操作或者操作其文档内容。  
窗口和其他窗口并不是没有完全的关系。一个窗口或标签页中的脚本可以打开新的窗口或者标签页。当一个脚本这样做，多个窗口可以相互操作

**iframe**

已经处于半废弃的inframe标签。挺无奈的，已经快废了。  
不过学习还是学习一下吧  
iframe是用来嵌套多个文档的。由iframe所创建的嵌套浏览上下文是用自己的window对象表示的。每一个iframe都是一个独立的window对象。  
对于客户端js来说，窗口，标签页，iframe和框架都是浏览上下文。对于js来说都是window对象。和独立的不同之处在于，js脚本能够看到其祖先和子孙

**打开和关闭窗口**

使用window的open可以打开一个新的标签页  
window.open将会载入指定的url到新的或者已经存在的窗口中（取决于如何设置）。并返回代表那个窗口的window对象。  
open()第一个参数会在新窗口中显示文档的url，如果参数省，或为空，将会使用空白页面进行显示

about:blank

脚本无法通过简单的猜测窗口的名字来操纵窗口的web应用。只有设置了允许导航的页面才可以。  
即，当且仅当窗口包含的文档来自相同的源，或者这个脚本打开的哪个窗口。（或者递归打开的窗口），脚本才能通过名字指定存在的窗口，或者如果是一个窗口内嵌在另一个窗口里的窗体，那么在它们的脚本之间可以相互导航。使用

**\_top（顶级祖先窗口）**

和

**\_parent（直接父级窗口）**

来获取上下文。

**窗口名字**  
窗口名字，运行open()方法引用已存在的窗口。并同时可以作为标签a和标签form的taget的值，表示加载到哪

open第三个参数表明如何打开这个标签的，以及大小（一般弹窗广告喜欢这样做）

// 打开允许改变大小的浏览器的窗口，包含地址栏，工具栏和地址栏

var w = window.open("http://www.baidu.com/", "smallwin", "windth=200, height=350, statue=yes");

在新的窗口中运行

window.name

"smallwin"

第四个参数只在第二个参数命名的时候有用，是在当前（原先）窗口是替换掉历史记录，还是新建一个历史记录，默认是后者，因为要指定名称，才会在历史记录中储存

**窗口的返回值**

var w = window.open();

w.alert('about to visit https://www.google.com.hk/') // 在新的窗口调用alert()方法

w.location = "https://www.google.com" // 设置窗口的location，完成页面的跳转

w保存的是跳转网页的window对象的值

**关闭窗口**

如果已经使用open()打开窗口，同样可以用close()关闭其窗口

w.close();

undefined

这样可以将w开的窗口将其关闭  
如果要关闭本窗口可以调用

close()

这样就完成了页面的关闭  
即使一个窗口已经关闭了，那么其window对象仍旧保存在其父窗口，直到父窗口关闭，window对象彻底的消失。其closed的值为true，并且document的值为null，其方法也不能使用

**窗体之间的关系**

window对象的方法open()可以打开并创建一个新的window对象（即创建一个新的窗口）并且该窗口是具有opener属性，根据该属性，可以打开其原始窗口。  
即，通过这样两个窗口可以实现相互引用。或者调用对方的方法。  
如果一个窗口是顶级窗口，标签  
则

parent == self

true

self 属性返回窗体本身，表明parent指向其窗体  
如果一个窗体是在另外一个窗体之中，则parent.parent将会是顶级窗口。不管其嵌套了多少层，包括使用top属性指向的都是最顶层窗口。  
如果window本身就是顶层窗口，则top就是其窗口本身。其top就是parent

**交互窗口中的js**

每个窗口窗体都是js的执行上下文。以window对象作为全局对象，这样的话，一个窗口窗体中的代码可以应用到其他窗口或者窗体（并且同源策略没有进行阻止）。那么他们之间可以完成交互。