## BOM

1. Bom概述

浏览器对象模型，提供独立于内容而与浏览器窗口进行交互的对象。其核心对象是window，最顶级的对象。

* window对象是浏览器最顶级的对象。
* 是js访问浏览器窗口的一个接口
* 是一个全局对象。一切全局变量、全局的函数都会变成window的属性和方法，但是调用时一般会省略window
* window下有一个特殊属性window.name，因此，不要定义一个变量名叫name

1. window对象常见事件

* 窗口加载事件window.onload：文档内容，包括样式表、图片、flash等完全加载完成后会触发该事件。将js代码封装进一个函数并于onload绑定后，使得js代码可以写在任何地方。其注册方式有传统方式与window.addEventListener方式

页面刷新、后腿回该页面、a标签的超链接均会触发onload事件，都是在火狐中后退是直接调用缓存，不会重新加载。采用另外一组代码可以解决问题：

window.addEvenListener(‘pageshow’, function(e){ //pageshow指页面加载

if (e.persisted){ } //若存在缓存，e.persisted为true，执行之后的语句

})

* 另一个窗口加载事件DOMContentLoaded，为DOM加载完毕后（不包括样式表、图片、flash等）触发。速度更快。兼容性问题：ie9以上才支持。

注册方式：document.addEventListener( ‘DOMConetentLoaded’ , function(){ } )

* 调整窗口大小事件window.resize，窗口大小发生变化就会触发
* window.innerWidth获取当前屏幕宽度，单位px

1. 定时器（函数方法）

* window.setTimeout( 调用的函数 , 延迟的毫秒数 )。定时器到期后会执行调用函数。时间可以省略，默认为0。 调用函数可以直接写匿名函数，也可以直接写函数名，甚至可以写成 ’ 函数名( ) ’

通过var time2=setTimeout ( )的方式为定时器命名

* 停止定时器 window.clearTimeout ( 定时器名字 )
* 重复调用定时器 window.setInterval( 调用的函数 , 延迟的毫秒数 ) 每隔一段时间调用一次。语法规范与色图Timeout一致。
* 停止定时器 window.clearInterval ( 定时器名字 )

1. this的指向问题

* this在函数定义时是确定不了的，只有在函数执行时才确定this指向的对象
* 一般情况下，this指向调用他的对象
* 全局作用域下和普通函数的this指向window。注意：**定时器的this永远指向window。**
* 方法调用中，this指向调用它的对象
* 构造函数中的this指向构造函数的实例

function fun(){ this } Fun=new fun() Fun中的this指向Fun，Fun即为fun构造函数的实例

1. js的执行机制

* js的特点是**单线程**，同一个时间只能做一件事。所有事件需要排队执行。
* 同步和异步
* 同步：前一件事结束后才开始下一个任务，即单线程
* 异步：允许多个任务同时进行，HTML5的新标准
* 同步任务：所有任务都是在主线程上执行，形成一条执行线
* 异步任务：通过回调函数callback实现
* 常见的异步任务有：
* 普通事件：click、resize等
* 资源加载：load、error等
* 定时器：setTimeout、setInterval等
* 执行路线

1. 先执行主线程执行栈
2. 遇到异步任务，交由异步进程处理判断是否满足函数条件，若满足，则写进任务队列中。若不满足，则等待满足条件的触发。
3. 执行完执行栈中的所有同步任务后，依次提取任务队列中的异步任务到执行栈，接着执行。
4. 执行栈执行完其内容后，会反复进入任务队列中获取满足条件的异步任务。执行栈与任务队列之间反复检查、提取事件的过程称为事件循环。Event Loop。
5. location对象

window **.** location属性用于获取或者**设置设置窗体的URL地址，并且可以用于解析URL**。因为该属性返回的是一个对象，因此也称为location对象。

* URL 统一资源定位符，是互联网上标准资源的地址。互联网上每一个文件都有一个唯一的URL，包含的信息指出：**文件的位置、浏览器对它的处理方式**。



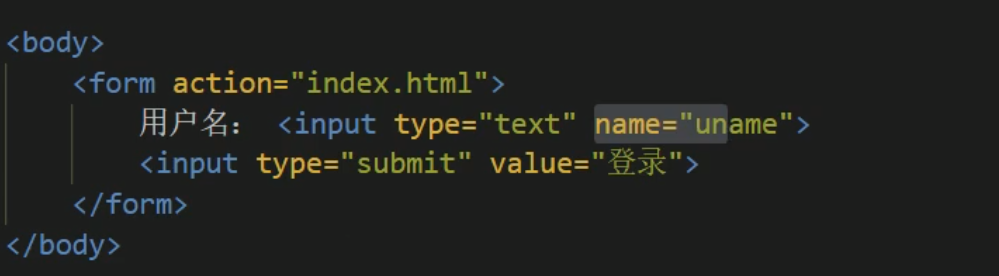
* location对象常见的属性



location.href=’目标地址’，可以使页面直接跳转到目标地址。

利用location.search可以实现页面间参数的传递。

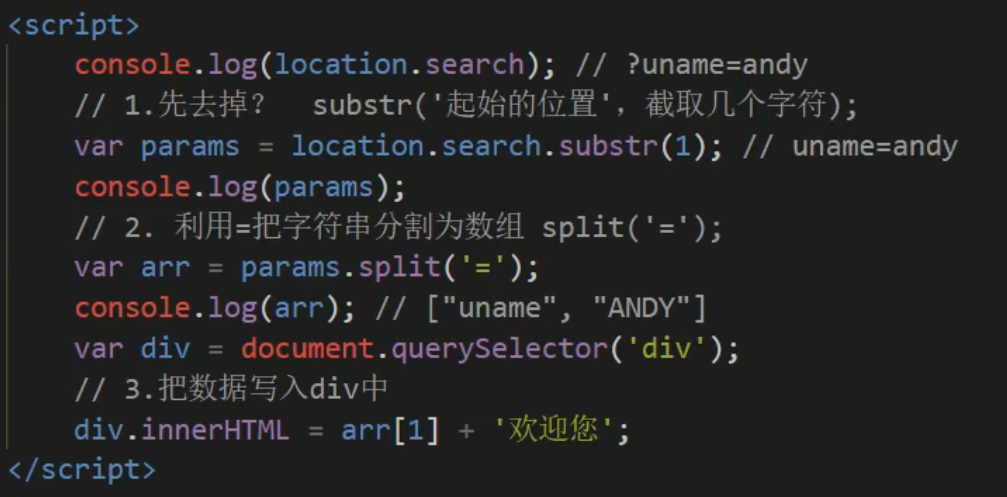
例子：页面1：



用户点击登录后，会跳转到index.html页面，同时url的search部分会变为

?uname=表单内容

页面2：

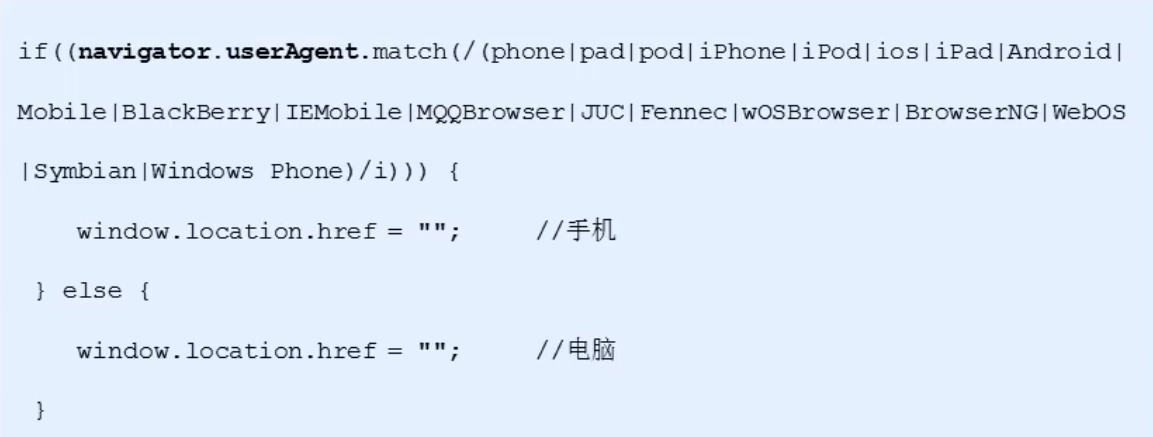


* location常见的方法
* location.assign( '目标地址’ )实现页面跳转，特点：会记录浏览器的历史
* location.replace( ‘目标地址’ )替换当前页面，不会记录历史
* location.reload ( ) 重新加载页面。默认为false，若为true，则为强制刷新，清除缓存。使用ctrl+F5也可以强制刷新。

1. navigator对象

包含有关浏览器的信息，常用属性有navigator.**userAgent**：可以返回由客户机发送给服务器的user-agent头部的值。

判断用户使用哪个终端打开页面，实现跳转：



1. history对象

包含用户在浏览器窗口中访问过的URL，实现与浏览器历史记录进行交互



1. 动画特效

* offset系列：元素偏移量

通过相关属性可以动态地得到元素的位置（偏移）、大小等。

* 位置：element . offsetTop以及element . offsetLeft 表示元素距离有定位的祖先元素最顶部以及最左部的距离，若父亲无定位，则参考系为body，单位px，返回**值没有单位**。记得没有bottom和right
* 大小：element.offsetWidth和element.offsetHeight。记得包含border、padding值。
* element.offsetParent返回**带定位的祖先元素**。

而节点操作中的parentNode返回的是亲爸爸，不一定带定位。

offset与style的区别

* style只能获取**行内样式表**的值，而offset都能获取
* offset获取的数值不带单位，而style返回值是带单位的字符串
* offset涉及大小的属性包含padding和border，而style不包含
* **offset是只读属性，而style可读写**

#拖拽案例

course：move；

* 鼠标按下mousedown，获得鼠标在盒子内的坐标
* 鼠标移动mousemove，鼠标在页面中的坐标减去鼠标在盒子内的坐标就是盒子在窗口内的坐标【开启绝对定位，使用top left定位】。（这个函数要写在mousedown内）
* 鼠标弹起mouseup，移除鼠标移动事件。
* client系列\_元素可视区

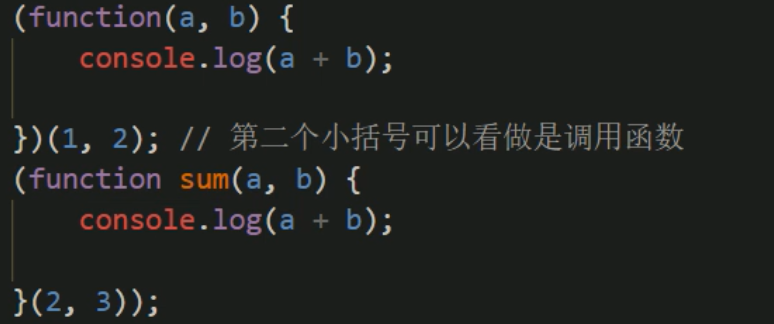
获取元素可视区的相关信息

* element.clientTop 元素**上边框**大小
* element.clientLeft 元素**左边框**大小
* element.clientWidth 返回宽度，包含padding和content，但是**不包含边框**
* element.clientHeight 返回高度，**不包含边框**

立即执行函数：不需要调用，直接执行

写法：( function ( 形参) { 内容 } )( 参数 ) ;

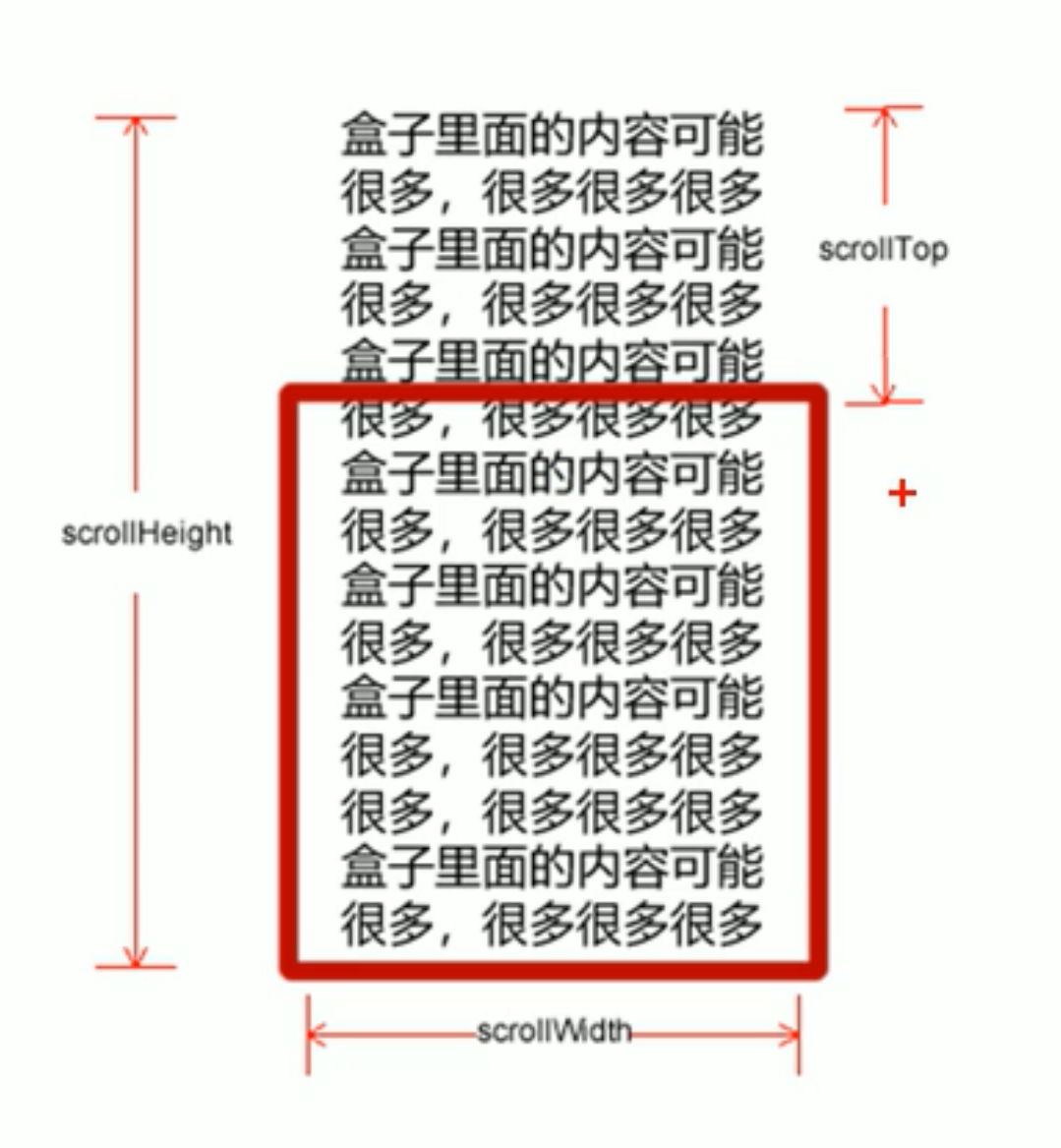
或者 ( function( 形参 ) { 内容 } ( 参数 ) ) ;



立即执行函数中的变量都是局部变量

* scroll系列\_滚动属性，**事件scroll**

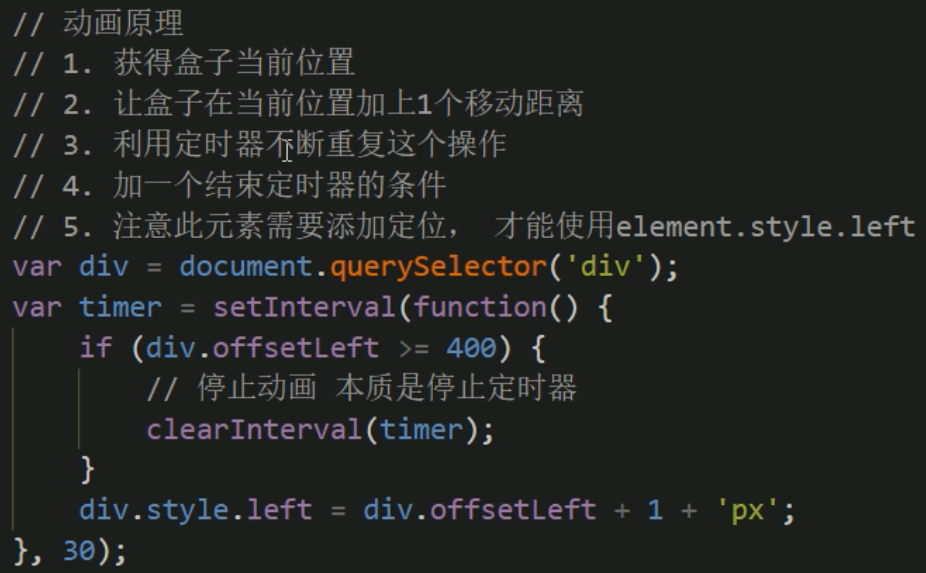
动态获得元素大小和滚动距离

* element.scrollTop 返回被卷去的上侧距离
* element.scrollLeft 返回被卷去的左侧距离
* element.scrollWidth 返回自身宽度，**不包含边框，包含padding**
* element.scrollHeight 返回自身高度，**不包含边框，包含padding。**
* **指的是内容的高度大小，可能超过盒子大小。这是与client系列的最大区别**
* window.pageYOffset 获得**整个页面**被卷去的顶部距离

window.pageXOffset 获得**整个页面**被卷去的左边距离

与scroll系列的区别是scroll返回的是**元素**被卷去的距离

**记住页面运动的事件源是 document**

* **window.scroll( x , y ) 滚动窗口至文档中特定位置，x y不带单位，默认为px**
* mouseenter与mouseover
* mouseover 鼠标移动到元素上触发，**经过子盒子也会再触发一次**，原因是该事件会冒泡，经过子盒子时元素未绑定mouseover，则会向上冒泡直到找到绑定该事件的祖先元素，触发事件。
* mouseenter 鼠标移动到元素上触发，只有**经过自身才会触发**，不会冒泡
* mouseleave 鼠标离开时触发，同样**不会冒泡**
* 动画函数封装
* 实现原理：通过定时器setInterval ( ) 不断移动盒子，注意元素需要**定位**
* 

offsetLeft只能获取不能赋值，因此将位置赋值给定位的left

* 函数的封装：将实现语句封装到function之内。将定时器函数作为调用者的一个属性，实现给不同调用者添加不同定时器的操作，从而节省空间。



clearInterval清除以前的定时器，防止事件触发多次而加快速度

* 缓动动画：使元素运动速度有所变化，一般是让速度慢慢停下来

核心：使每次移动的距离变小，速度从而发生变化

算法：每次移动的距离=（目标值-现在的位置）\*常数k

* 回调函数：函数可以作为一个参数，传到另一个函数里。当另一个函数执行完后，再执行传进去的函数，这个过程称为回调。
* 节流阀：防止事件连续触发导致速度加快

目的：当上一个函数动画执行完毕，再执行下一个函数，阻止事件连续触发

核心：利用回调函数添加一个变量来控制，锁住函数和解锁函数

申明一个全局变量，var flag=true；

if( flag ){ flag=false; //关闭阀门

执行函数; }

再动画函数末尾加入一个包含 ’ flag=true; ’的回调函数，开启阀门。