[**JavaScript延迟加载lazyload的实现**](http://www.cnblogs.com/jenry/archive/2011/02/13/1953160.html)

懒加载技术(简称lazyload)并不是新技术, 它是js程序员对网页性能优化的一种方案.lazyload的核心是按需加载.在大型网站中都有lazyload的身影,例如谷歌的图片搜索页,迅雷首页，淘宝网,QQ空间等.因此掌握lazyload技术是个不错的选择,可惜jquery插件lazy load官网(http://www.appelsiini.net/projects/lazyload)称不支持新版浏览器。

涉及到图片，falsh资源 , iframe, 网页编辑器(类似FCK)等占用较大带宽，且这些模块暂且不在浏览器可视区内，因此可以使用lazyload在适当的时候加载该类资源。避免网页打开时加载过多资源，让用户等待太久。

lazyload的难点在如何在适当的时候加载用户需要的资源(这里用户需要的资源指该资源呈现在浏览器可视区域)。因此我们需要知道几点信息来确定目标是否已呈现在客户区，其中包括：

1. 可视区域相对于浏览器顶端位置
2. 待加载资源相对于浏览器顶端位置

在得到以上两点数据后，通过如下函数，便可得出某对象是否在浏览器可视区域了。

//返回浏览器的可视区域位置  
　　 function getClient(){  
 　　　　 var l,t,w,h;  
 　　　　 l = document.documentElement.scrollLeft || document.body.scrollLeft;  
 　　　　 t = document.documentElement.scrollTop || document.body.scrollTop;  
 　　　　 w = document.documentElement.clientWidth;  
 　　　　 h = document.documentElement.clientHeight;  
 　　　　 return {'left':l,'top':t,'width':w,'height':h} ;  
　　 }  
 　　//返回待加载资源位置  
　　 function getSubClient(p){  
　　　　 var l = 0,t = 0,w,h;  
 　　　　 w = p.offsetWidth ;  
 　　　　 h = p.offsetHeight;  
 　　　　while(p.offsetParent){  
 　　　　l += p.offsetLeft ;  
 　　　　 t += p.offsetTop ;  
 　　　　 p = p.offsetParent;  
 　　}  
 　　return {'left':l,'top':t,'width':w,'height':h } ;

其中函数 getClient()返回浏览器客户区区域信息，getSubClient()返回目标模块区域信息。此时确定目标模块是否出现在客户区实际上是确定如上两个矩形是否相交。

//判断两个矩形是否相交,返回一个布尔值  
 　　 function intens(rec1,rec2){  
 　　　　 var lc1,lc2,tc1,tc2,w1,h1;  
 　　　　 lc1 = rec1.left + rec1.width / 2;  
 　　　　 lc2 = rec2.left + rec2.width / 2;  
 　　　　tc1 = rec1.top + rec1.height / 2 ;  
 　　　　 tc2 = rec2.top + rec2.height / 2 ;  
 　　　　 w1 = (rec1.width + rec2.width) / 2 ;  
　　　　 h1 = (rec1.height + rec2.height) / 2;  
 　　　　 return Math.abs(lc1 - lc2) < w1 && Math.abs(tc1 - tc2) < h1 ;  
 　　 }

现在基本上可以实现延时加载了，接下来，我们在 window.onscroll 事件中编写一些代码监控目标区域是否呈现在客户区。

<div style = "width:100px; height:3000px"></div>  
　　<div id = "d1" style = "width:50px; height:50px; background:red;position:absolute; top:1000px">  
　　</div>  
　　 var d1 = document.getElementById("d1");  
　　 window.onscroll = function(){  
 　　 　　var prec1 = getClient();   
 　　　　var prec2 = getSubClient(d1);  
 　　　　　if(intens(prec1,prec2)){  
 　　 　　　　alert("true")  
　　 　 　　}  
 　　}

我们只需要在弹出窗口的地方加载我们需要的资源。

这里值得注意的是:目标对象呈现在客户区域时，会随着滚动而不断的弹出窗口。因此我们需要在弹出第一个窗口后取消对该区域的监测，这里用一个数组来收集需要监测的对象。还需要注意:因为onscroll事件和onresize事件都会改变游览器可视区域信息，因此在该类事件触发后需要重新计算目标对象是否在可视区域，这里用autocheck()函数实现。(迅雷首页的lazyload没有在onresize事件中重新计算目标对象是否在浏览器可视区域，因此如果先将浏览器窗口缩小到一定尺寸后滚动到需要加载图片的区域后点击最大化，图片加载不出来，呵呵，以后需要注意了)。

增加元素:<div id = "d2" style = "width:50px; height:50px; background:blue;position:absolute; top:2500px">

//比较某个子区域是否呈现在浏览器区域  
 　　function jiance(arr,prec1,callback){  
 　　　var prec2;  
 　　　for(var i = arr.length - 1 ; i >= 0 ;i--){  
 　　　　 if(arr[i]){  
 　　　　 prec2 = getSubClient(arr[i]);  
 　　　　 if(intens(prec1,prec2)){  
 　　　　　　callback(arr[i]);  
 　　　　　　 //加载资源后，删除监测  
 　　　　 　　 delete arr[i];  
 　　　　　　}  
 　　　　}  
 　　　}  
 　　}  
   
 　　//检测目标对象是否出现在客户区  
 　　function autocheck(){  
 　　　　 var prec1 = getClient();   
 　　　　jiance(arr,prec1,function(obj){  
　　　　　　//加载资源...  
 　　　　 alert(obj.innerHTML)  
 　　　　})  
 　　}  
 　　//子区域一   
 　　 var d1 = document.getElementById("d1");  
 　　 //子区域二  
 　　var d2 = document.getElementById("d2");  
　　 //需要按需加载区域集合  
 　　var arr = [d1,d2];  
 　　 window.onscroll = function(){  
　　　　 //重新计算  
 　　　　 autocheck();  
 　　}  
 　　window.onresize = function(){  
　　　　 //重新计算  
 　　　　 autocheck();  
 　　}

现在我们只需要在弹窗的地方加载我们需要的资源了