# 浏览器兼容性问题整理

11.高度不适应

高度不适应是当内层对象的高度发生变化时外层高度不能自动进行调节，特别是当内层对象使用margin 或paddign 时。

例：

#box {background-color:#eee; }

#box p {margin-top: 20px;margin-bottom: 20px; text-align:center; }

<div id="box">

<p>p对象中的内容</p>

</div>

解决方法：在P对象上下各加2个空的div对象CSS代码：.1{height:0px;overflow:hidden;}或者为DIV加上border属性。

12 .IE6下为什么图片下有空隙产生

解 决这个BUG的方法也有很多,可以是改变html的排版,或者设置img 为display:block 或者设置vertical-align 属性为 vertical-align:top | bottom |middle |text-bottom 都可以解决.

13.如何对齐文本与文本输入框

加上 vertical-align:middle;

<style type="text/css">

<!--

input {

width:200px;

height:30px;

border:1px solid red;

vertical-align:middle;

}

-->

</style>

14.web标准中定义id与class有什么区别吗

一.web标准中是不容许重复ID的,比如 div id="aa" 不容许重复2次,而class 定义的是类,理论上可以无限重复, 这样需要多次引用的定义便可以使用他.

二.属性的优先级问题

ID 的优先级要高于class,看上面的例子

三.方便JS等客户端脚本,如果在页面中要对某个对象进行脚本操作,那么可以给他定义一个ID,否则只能利用遍历页面元素加上指定特定属性来找到它,这是相对浪费时间资源,远远不如一个ID来得简单.

15. LI中内容超过长度后以省略号显示的方法

此方法适用与IE与OP浏览器

<style type="text/css">

<!--

li {

width:200px;

white-space:nowrap;

text-overflow:ellipsis;

-o-text-overflow:ellipsis;

overflow: hidden;

}

-->

</style>

16.为什么web标准中IE无法设置滚动条颜色了

解决办法是将body换成html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

<style type="text/css">

<!--

html {

scrollbar-face-color:#f6f6f6;

scrollbar-highlight-color:#fff;

scrollbar-shadow-color:#eeeeee;

scrollbar-3dlight-color:#eeeeee;

scrollbar-arrow-color:#000;

scrollbar-track-color:#fff;

scrollbar-darkshadow-color:#fff;

}

-->

</style>

17.为什么无法定义1px左右高度的容器

IE6下这个问题是因为默认的行高造成的,解决的方法也有很多,例如:overflow:hidden | zoom:0.08 | line-height:1px

18.怎么样才能让层显示在FLASH之上呢

解决的办法是给FLASH设置透明

<param name="wmode" value="transparent" />

19.怎样使一个层垂直居中于浏览器中

这里我们使用百分比绝对定位,与外补丁负值的方法,负值的大小为其自身宽度高度除以二

<style type="text/css">

<!--

div {

position:absolute;

top:50%;

lef:50%;

margin:-100px 0 0 -100px;

width:200px;

height:200px;

border:1px solid red;

}

-->

</style>

FF与IE

1. Div居中问题 powered by 25175.net

div设置 margin-left, margin-right 为 auto 时已经居中，IE 不行，IE需要设定body居中，首先在父级元素定义text-algin: center;这个的意思就是在父级元素内的内容居中。

2.链接(a标签)的边框与背景

a 链接加边框和背景色，需设置 display: block, 同时设置 float: left 保证不换行。参照 menubar, 给 a 和 menubar 设置高度是为了避免底边显示错位, 若不设 height, 可以在 menubar 中插入一个空格。

3.超链接访问过后hover样式就不出现的问题

被点击访问过的超链接样式不在具有hover和active了,很多人应该都遇到过这个问题,解决方法是改变CSS属性的排列顺序: L-V-H-A

Code:

<style type="text/css">

<!--

a:link {}

a:visited {}

a:hover {}

a:active {}

-->

</style>

4. 游标手指cursor

cursor: pointer 可以同时在 IE FF 中显示游标手指状， hand 仅 IE 可以

5.UL的padding与margin

ul标签在FF中默认是有padding值的,而在IE中只有margin默认有值,所以先定义 ul{margin:0;padding:0;}就能解决大部分问题

6. FORM标签

这 个标签在IE中,将会自动margin一些边距,而在FF中margin则是0,因此,如果想显示一致,所以最好在css中指定margin和 padding,针对上面两个问题,我的css中一般首先都使用这样的样式ul,form{margin:0;padding:0;}给定义死了,所以后 面就不会为这个头疼了.

7. BOX模型解释不一致问题

在FF和IE 中的BOX模型解释不一致导致相差2px解决方法：div{margin:30px!important;margin:28px;} 注意这两个 margin的顺序一定不能写反， important这个属性IE不能识别，但别的浏览器可以识别。所以在IE下其实解释成这样： div {maring:30px;margin:28px}重复定义的话按照最后一个来执行，所以不可以只写margin:xx px!important;

#box{ width:600px; //for ie6.0- w\idth:500px; //for ff+ie6.0}

#box{ width:600px!important //for ff width:600px; //for ff+ie6.0 width /\*\*/:500px; //for ie6.0-}

8.属性选择器(这个不能算是兼容,是隐藏css的一个bug)

p[id]{}div[id]{}

这个对于IE6.0和IE6.0以下的版本都隐藏,FF和OPera作用.属性选择器和子选择器还是有区别的,子选择器的范围从形式来说缩小了,属性选择器的范围比较大,如p[id]中,所有p标签中有id的都是同样式的.

9.最狠的手段 - !important;

如果实在没有办法解决一些细节问题,可以用这个方法.FF对于”!important”会自动优先解析,然而IE则会忽略.如下

.tabd1{

background:url(/res/images/up/tab1.gif) no-repeat 0px 0px !important; /\*Style for FF\*/

background:url(/res/images/up/tab1.gif) no-repeat 1px 0px; /\* Style for IE \*/}

值得注意的是，一定要将xxxx !important 这句放置在另一句之上，上面已经提过

10.IE,FF的默认值问题

或 许你一直在抱怨为什么要专门为IE和FF写不同的CSS，为什么IE这样让人头疼，然后一边写css，一边咒骂那个可恶的M$ IE.其实对于css的标 准支持方面，IE并没有我们想象的那么可恶，关键在于IE和FF的默认值不一样而已，掌握了这个技巧，你会发现写出兼容FF和IE的css并不是那么困 难，或许对于简单的css，你完全可以不用”!important”这个东西了。

我们都知道，浏览器在显示网页的时候，都会根据网页的 css样式表来决定如何显示，但是我们在样式表中未必会将所有的元素都进行了具体的描述，当然也没有必要那么做，所以对于那些没有描述的属性，浏览器将采 用内置默认的方式来进行显示，譬如文字，如果你没有在css中指定颜色，那么浏览器将采用黑色或者系统颜色来显示，div或者其他元素的背景，如果在 css中没有被指定，浏览器则将其设置为白色或者透明，等等其他未定义的样式均如此。所以有很多东西出现FF和IE显示不一样的根本原因在于它们的默认显 示不一样，而这个默认样式该如何显示我知道在w3中有没有对应的标准来进行规定，因此对于这点也就别去怪罪IE了。

11.为什么FF下文本无法撑开容器的高度

标准浏览器中固定高度值的容器是不会象IE6里那样被撑开的,那我又想固定高度,又想能被撑开需要怎样设置呢？办法就是去掉height设置min-height:200px; 这里为了照顾不认识min-height的IE6 可以这样定义:

{

height:auto!important;

height:200px;

min-height:200px;

}

12.FireFox下如何使连续长字段自动换行

众所周知IE中直接使用 word-wrap:break-word 就可以了, FF中我们使用JS插入&#10;的方法来解决

<style type="text/css">

<!--

div {

width:300px;

word-wrap:break-word;

border:1px solid red;

}

-->

</style>

<div id="ff">aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</div>

<scrīpt type="text/javascrīpt">

/\* <![CDATA[ \*/

function toBreakWord(el, intLen){

var ōbj=document.getElementById(el);

var strContent=obj.innerHTML;

var strTemp="";

while(strContent.length>intLen){

strTemp+=strContent.substr(0,intLen)+"&#10;";

strContent=strContent.substr(intLen,strContent.length);

}

strTemp+="&#10;"+strContent;

obj.innerHTML=strTemp;

}

if(document.getElementById && !document.all) toBreakWord("ff", 37);

/\* ]]> \*/

</scrīpt>

13.为什么IE6下容器的宽度和FF解释不同呢

<?xml version="1.0" encoding="gb2312"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />

<style type="text/css">

<!--

div {

cursor:pointer;

width:200px;

height:200px;

border:10px solid red

}

-->

</style>

<div ōnclick="alert(this.offsetWidth)">让FireFox与IE兼容</div>

问 题的差别在于容器的整体宽度有没有将边框（border）的宽度算在其内,这里IE6解释为200PX ,而FF则解释为220PX,那究竟是怎么导致的 问题呢？大家把容器顶部的xml去掉就会发现原来问题出在这,顶部的申明触发了IE的qurks mode,关于qurks mode、 standards mode的相关知识

powered by 25175.netIE6,IE7,FF

IE7.0 出来了，对CSS的支持又有新问题。浏览器多了，网页兼容性更差了，疲于奔命的还是我们 ，为解决IE7.0的兼容问题，找来了下面这篇文章： 现在我大 部分都是用!important来hack，对于ie6和firefox测试可以正常显示，但是ie7对!important可以正确解释，会导致页面没 按要求显示！下面是三个浏览器的兼容性收集.

第一种，是CSS HACK的方法

height:20px; /\*For Firefox\*/

\*height:25px; /\*For IE7 & IE6\*/

\_height:20px; /\*For IE6\*/

注意顺序。

这样也属于CSS HACK，不过没有上面这样简洁。

#example { color: #333; } /\* Moz \*/

\* html #example { color: #666; } /\* IE6 \*/

\*+html #example { color: #999; } /\* IE7 \*/

第二种，是使用IE专用的条件注释

<!--其他浏览器 -->

<link rel="stylesheet" type="text/css" href="http://cgxcn.blog.163.com/blog/http://www.cnblogs.com/../css.css" />

<!--[if IE 7]>

<!-- 适合于IE7 -->

<link rel="stylesheet" type="text/css" href="http://cgxcn.blog.163.com/blog/http://www.cnblogs.com/../ie7.css" />

<![endif]-->

<!--[if lte IE 6]>

<!-- 适合于IE6及一下 -->

<link rel="stylesheet" type="text/css" href="http://cgxcn.blog.163.com/blog/http://www.cnblogs.com/../ie.css" />

<![endif]-->

第三种，css filter的办法，以下为经典从国外网站翻译过来的。.

新建一个css样式如下：

#item {

width: 200px;

height: 200px;

background: red;

}

新建一个div,并使用前面定义的css的样式：

<div id="item">some text here</div>

在body表现这里加入lang属性,中文为zh：

<body lang="en">

现在对div元素再定义一个样式：

\*:lang(en) #item{

background:green !important;

}

这样做是为了用!important覆盖原来的css样式,由于:lang选择器ie7.0并不支持,所以对这句话不会有任何作用,于是也达到了ie6.0下同样的效果,但是很不幸地的是,safari同样不支持此属性,所以需要加入以下css样式：

#item:empty {

background: green !important

}

:empty选择器为css3的规范,尽管safari并不支持此规范,但是还是会选择此元素,不管是否此元素存在,现在绿色会现在在除ie各版本以外的浏览器上。

对IE6和FF的兼容可以考虑以前的!important 个人比较喜欢用第一种，简洁，兼容性比较好

什么是浏览器HACK：当我们使用不同的浏览器（Firefox IE7 IE6）访问同一个网站，或者页面的时候，会出现一些不兼容的问题，有的显示出来正常，有的显示出来不正常，我们在编写CSS的时候会很恼火，刚修复了这个浏览器的问题，结果另外一个浏览器却出了新问题。而HACK就是一种办法，能让你在一个CSS里面独立的写支持不同浏览器的样式。这下就和谐了。呵呵！powered by 25175.net

最近微软发布的IE7浏览器的兼容性确实给一些网页制作人员添加了一个沉重的负担，虽然IE7已经走向标准化，但还是有许多和FF不同的地方，所以需要用到IE7的HACK，有许多朋友问过IE7的HACK是什么，其实我也不知道。暂时还没找到IE7专用的HACK。除了前面那片文章，《针对firefox ie6 ie7的css样式》中的hack方式也是很好用的。

有一点逻辑思想的人都会知道可以用IE和FF的HACK结合起来使用，下面介绍三个HACK，例如：（适合新手，呵呵，高手就在这里路过吧。）

程序代码

第一个HACK，IE FF 所有浏览器 公用（其实也不算是HACK）

height:100px;

第二个HACK IE6专用

\_height:100px;

第三个HACK IE6 IE7公用

\*height:100px;

介绍完了这三个HACK了，下面我们再来看看如何在一个样式里分别给一个属性定义IE6 IE7 FF专用的HACK，看下面的代码，顺序不能错哦：

程序代码

height:100px;

\*height:120px;

\_height:150px;

下面我简单解释一下各浏览器怎样理解这三个属性：

在FF下，第2、3个属性FF不认识，所以它读的是 height:100px;

在IE7下，第三个属性IE7不认识，所以它读第1、2个属性，又因为第二个属性覆盖了第一个属性，所以IE7最终读出的是第2个属性 \*height:120px;

在IE6下，三个属性IE6都认识，所以三个属性都可以读取，又因为第三个属性覆盖掉前2个属性，所以IE6最终读取的是第三个属性。

1 针对firefox ie6 ie7的css样式

现在大部分都是用!important来hack，对于ie6和firefox测试可以正常显示，但是ie7对!important可以正确解释，会导致页面没按要求显示！找到一个针对IE7不错的hack方式就是使用“\*+html”，现在用IE7浏览一下，应该没有问题了现在写一个CSS可以这样：

#1 { color: #333; } /\* Moz \*/

\* html #1 { color: #666; } /\* IE6 \*/

\*+html #1 { color: #999; } /\* IE7 \*/

那么在firefox下字体颜色显示为#333，IE6下字体颜色显示为#666，IE7下字体颜色显示为#999。

2 css布局中的居中问题

主要的样式定义如下：

body {TEXT-ALIGN: center;}

#center { MARGIN-RIGHT: auto; MARGIN-LEFT: auto; }

说明：

首先在父级元素定义TEXT-ALIGN: center;这个的意思就是在父级元素内的内容居中；对于IE这样设定就已经可以了。

但在mozilla中不能居中。解决办法就是在子元素定义时候设定时再加上“MARGIN-RIGHT: auto;MARGIN-LEFT: auto; ”

需要说明的是，如果你想用这个方法使整个页面要居中，建议不要套在一个DIV里，你可以依次拆出多个div，只要在每个拆出的div里定义MARGIN-RIGHT: auto;MARGIN-LEFT: auto; 就可以了。

3 盒模型不同解释.

#box{

width:600px;

//for ie6.0- w\idth:500px;

//for ff+ie6.0

}

#box{

width:600px!important

//for ff

width:600px;

//for ff+ie6.0

width /\*\*/:500px;

//for ie6.0-

}

4 浮动ie产生的双倍距离

#box{ float:left; width:100px; margin:0 0 0 100px; //这种情况之下IE会产生200px的距离 display:inline; //使浮动忽略}

这里细说一下block,inline两个元素,Block元素的特点是:总是在新行上开始,高度,宽度,行高,边距都可以控制(块元素);Inline元素的特点是:和其他元素在同一行上,…不可控制(内嵌元素);

#box{ display:block; //可以为内嵌元素模拟为块元素 display:inline; //实现同一行排列的的效果 diplay:table;

5 IE与宽度和高度的问题

IE不认得min-这个定义，但实际上它把正常的width和height当作有min的情况来使。这样问题就大了，如果只用宽度和高度，正常的浏览器里这两个值就不会变，如果只用min-width和min-height的话，IE下面根本等于没有设置宽度和高度。比如要设置背景图片，这个宽度是比较重要的。要解决这个问题，可以这样：

#box{ width: 80px; height: 35px;}html>body #box{ width: auto; height: auto; min-width: 80px; min-height: 35px;}

6 页面的最小宽度

min-width是个非常方便的CSS命令，它可以指定元素最小也不能小于某个宽度，这样就能保证排版一直正确。但IE不认得这个，而它实际上把 width当做最小宽度来使。为了让这一命令在IE上也能用，可以把一个<div> 放到 <body> 标签下，然后为div指定一个类：

然后CSS这样设计：

#container{

min-width: 600px;

width:e?xpression(document.body.clientWidth < 600? “600px”: “auto” );

}

第一个min-width是正常的；但第2行的width使用了Javascript，这只有IE才认得，这也会让你的HTML文档不太正规。它实际上通过Javascript的判断来实现最小宽度。

7 清除浮动

.hackbox{

display:table;

//将对象作为块元素级的表格显示

}

或者

.hackbox{

clear:both;

}

或者加入:after（伪对象）,设置在对象后发生的内容，通常和content配合使用，IE不支持此伪对象，非Ie 浏览器支持，所以并不影响到IE/WIN浏览器。这种的最麻烦的

……#box:after{

content: “.”;

display: block;

height: 0;

clear: both;

visibility: hidden;

}

8 DIV浮动IE文本产生3象素的bug

左边对象浮动，右边采用外补丁的左边距来定位，右边对象内的文本会离左边有3px的间距.

#box{

float:left;

width:800px;}

#left{

float:left;

width:50%;}

#right{

width:50%;

}

\*html #left{

margin-right:-3px;

//这句是关键

}

HTML代码

<DIV id=box>

<DIV id=left></DIV>

<DIV id=right></DIV>

</DIV>

9 属性选择器(这个不能算是兼容,是隐藏css的一个bug)

p[id]{}div[id]{}

p[id]{}div[id]{}

这个对于IE6.0和IE6.0以下的版本都隐藏,FF和OPera作用

属性选择器和子选择器还是有区别的,子选择器的范围从形式来说缩小了,属性选择器的范围比较大,如p[id]中,所有p标签中有id的都是同样式的.

10 IE捉迷藏的问题

当div应用复杂的时候每个栏中又有一些链接，DIV等这个时候容易发生捉迷藏的问题。

有些内容显示不出来，当鼠标选择这个区域是发现内容确实在页面。

解决办法：对#layout使用line-height属性 或者给#layout使用固定高和宽。页面结构尽量简单。

11 高度不适应

高度不适应是当内层对象的高度发生变化时外层高度不能自动进行调节，特别是当内层对象使用

margin 或paddign 时。例：

<div id=”box”>

<p>p对象中的内容</p>

</div>

CSS：

#box {background-color:#eee; }

#box p {margin-top: 20px;margin-bottom: 20px; text-align:center; }

解决方法：在P对象上下各加2个空的div对象CSS代码：.1{height:0px;overflow:hidden;}或者为DIV加上border属性。

屏蔽IE浏览器（也就是IE下不显示）

\*:lang(zh) select {font:12px !important;} /\*FF,OP可见\*/

select:empty {font:12px !important;} /\*safari可见\*/

这里select是选择符，根据情况更换。第二句是MAC上safari浏览器独有的。

仅IE7识别

\*+html {…}

当面临需要只针对IE7做样式的时候就可以采用这个HACK。

IE6及IE6以下识别

\* html {…}

这个地方要特别注意很多地主都写了是IE6的HACK其实IE5.x同样可以识别这个HACK。其它浏览器不识别。

html/\*\*/ >body select {……}

这句与上一句的作用相同。

仅IE6不识别

select { display /\*IE6不识别\*/:none;}

这里主要是通过CSS注释分开一个属性与值，流释在冒号前。

仅IE6与IE5不识别

select/\*\*/ { display /\*IE6,IE5不识别\*/:none;}

这里与上面一句不同的是在选择符与花括号之间多了一个CSS注释。

仅IE5不识别

select/\*IE5不识别\*/ { display:none;}

这一句是在上一句中去掉了属性区的注释。只有IE5不识别

盒模型解决方法

selct {width:IE5.x宽度; voice-family :""}""; voice-family:inherit; width:正确宽度;}

盒模型的清除方法不是通过!important来处理的。这点要明确。

清除浮动

select:after {content:"."; display:block; height:0; clear:both; visibility:hidden;}

在Firefox中，当子级都为浮动时，那么父级的高度就无法完全的包住整个子级，那么这时用这个清除浮动的HACK来对父级做一次定义，那么就可以解决这个问题 。

截字省略号

select { -o-text-overflow:ellipsis; text-overflow:ellipsis; white-space:nowrap; overflow:hidden; }

这个是在越出长度后会自行的截掉多出部分的文字，并以省略号结尾，很好的一个技术。只是目前Firefox并不支持。

只有Opera识别

@media all and (min-width: 0px){ select {……} }

针对Opera浏览器做单独的设定。

以上都是写CSS中的一些HACK，建议遵循正确的标签嵌套(div ul li 嵌套结构关系)，这样可以减少你使用Hack的频率，不要进入理解误区，并不是一个页面就需要很多的hack来保持多浏览器兼容)，很多情况下也许一个Hack都不用也可以让浏览器工作得非常好，这些都是用来解决局部的兼容性问题，如果希望把兼容性的内容也分离出来，不妨试一下下面的几种过滤器。这些过滤器有的是写在CSS中通过过滤器导入特别的样式，也有的是写在HTML中的通过条件来链接或是导入需要的补丁样式。

IE5.x的过滤器，只有IE5.x可见

@media tty {

i{content:"";/\*" "\*/}} @import 'ie5win.css'; /\*";}

}/\* \*/

IE5/MAC的过滤器，一般用不着

/\*\*//\*/

@import "ie5mac.css";

/\*\*/

下面是IE的条件注释，个人觉得用条件注释调用相应 Hack是比较完美的多浏览器兼容的解决办法。把需要hack的地方单独放到一个文件里面，当浏览器版本符合的时候就可以调用那个被Hack的样式，这样不仅使用起来非常方便，而且对于制作这个CSS本身来讲，可以更严格的观察到是否有必要使用hack，很多情况下，当我本人写CSS如果把全部代码包括Hack都写到一个CSS文件的时候的时候会很随意，想怎么Hack就怎么Hack，而你独立出来写的时候，你就会不自觉的考虑是否有必要Hack，是先Hack CSS？还是先把主CSS里面的东西调整到尽可能的不需要Hack？当你仅用很少的Hack就让很多浏览器很乖很听话的时候，你是不是很有成就感呢？你知道怎么选择了吧～～呵呵

IE的if条件Hack 自己可以灵活使用参看这篇IE条件注释

Only IE

所有的IE可识别

只有IE5.0可以识别

Only IE 5.0+

IE5.0包换IE5.5都可以识别

仅IE6可识别

Only IE 7/-

IE6以及IE6以下的IE5.x都可识别

Only IE 7/-

仅IE7可识别

IE6、IE7、FF 与兼容性有关的特性

CSS HACK

代码:

所有浏览器 通用

height: 100px;

IE6 专用

\_height: 100px;powered by 25175.net

IE6 专用

\*height: 100px;

IE7 专用

\*+height: 100px;

IE7、FF 共用

height: 100px !important（优先级）;

一、CSS HACK

以下两种方法几乎能解决现今所有HACK.

1, !important (不是很推荐，用下面的一种感觉最安全)

随着IE7对!important的支持, !important 方法现在只针对IE6的HACK.(注意写法.记得该声明位置需要提前.)

代码:

<style>

#wrapper {

width: 100px!important; /\* IE7+FF \*/

width: 80px; /\* IE6 \*/

}

</style>

2, IE6/IE77对FireFox <from 针对firefox ie6 ie7的css样式>

\*+html 与 \*html 是IE特有的标签, firefox 暂不支持.而\*+html 又为 IE7特有标签.

代码:

<style>

#wrapper { width: 120px; } /\* FireFox \*/

\*html #wrapper { width: 80px;} /\* ie6 fixed \*/

\*+html #wrapper { width: 60px;} /\* ie7 fixed, 注意顺序 \*/

</style>

注意:

\*+html 对IE7的HACK 必须保证HTML顶部有如下声明：

代码:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

二、万能 float 闭合(非常重要!) 可以用这个解决多个div对齐时的间距不对，

关于 clear float 的原理可参见 [How To Clear Floats Without Structural Markup]

将以下代码加入Global CSS 中,给需要闭合的div加上 class=”clearfix” 即可,屡试不爽.

代码:

<style>

/\* Clear Fix \*/

.clearfix:after {

content:".";

display:block;

height:0;

clear:both;

visibility:hidden;

}

.clearfix {

display:inline-block;

}

/\* Hide from IE Mac \\*/

.clearfix {display:block;}

/\* End hide from IE Mac \*/

/\* end of clearfix \*/

</style>

三、其他兼容技巧(相当有用)

1, FF下给 div 设置 padding 后会导致 width 和 height 增加, 但IE不会.(可用!important解决)

2, 居中问题.

1).垂直居中.将 line-height 设置为 当前 div 相同的高度, 再通过 vertical-align: middle.( 注意内容不要换行.)

2).水平居中. margin: 0 auto;(当然不是万能)

3, 若需给 a 标签内内容加上 样式, 需要设置 display: block;(常见于导航标签)

4, FF 和 IE 对 BOX 理解的差异导致相差 2px 的还有设为 float的div在ie下 margin加倍等问题.

5, ul 标签在 FF 下面默认有 list-style 和 padding . 最好事先声明, 以避免不必要的麻烦. (常见于导航标签和内容列表)

6, 作为外部 wrapper 的 div 不要定死高度, 最好还加上 overflow: hidden.以达到高度自适应.

7, 关于手形光标. cursor: pointer. 而hand 只适用于 IE.贴上代码:

兼容代码:hack最推荐的模式。

/\* FF \*/

.submitbutton {

float:left;

width: 40px;

height: 57px;

margin-top: 24px;

margin-right: 12px;

}

/\* IE6 \*/

\*html .submitbutton {

margin-top: 21px;

}

/\* IE7 \*/

\*+html .submitbutton {

margin-top: 21px;

}

什么是浏览器HACK：当我们使用不同的浏览器（Firefox IE7 IE6）访问同一个网站，或者页面的时候，会出现一些不兼容的问题，有的显示出来正常，有的显示出来不正常，我们在编写CSS的时候会很恼火，刚修复了这个浏览器的问题，结果另外一个浏览器却出了新问题。而HACK就是一种办法，能让你在一个CSS里面独立的写支持不同浏览器的样式。这下就和谐了。呵呵！

最近微软发布的IE7浏览器的兼容性确实给一些网页制作人员添加了一个沉重的负担，虽然IE7已经走向标准化，但还是有许多和FF不同的地方，所以需要用到IE7的HACK，有许多朋友问过IE7的HACK是什么，其实我也不知道。暂时还没找到IE7专用的HACK。除了前面那片文章，《针对firefox ie6 ie7的css样式》中的hack方式也是很好用的。

有一点逻辑思想的人都会知道可以用IE和FF的HACK结合起来使用，下面介绍三个HACK，例如：（适合新手，呵呵，高手就在这里路过吧。）

程序代码

第一个HACK，IE FF 所有浏览器 公用（其实也不算是HACK）

height:100px;

第二个HACK IE6专用

\_height:100px;

第三个HACK IE6 IE7公用

\*height:100px;

介绍完了这三个HACK了，下面我们再来看看如何在一个样式里分别给一个属性定义IE6 IE7 FF专用的HACK，看下面的代码，顺序不能错哦：

程序代码

height:100px;

\*height:120px;

\_height:150px;

下面我简单解释一下各浏览器怎样理解这三个属性：

在FF下，第2、3个属性FF不认识，所以它读的是 height:100px;

在IE7下，第三个属性IE7不认识，所以它读第1、2个属性，又因为第二个属性覆盖了第一个属性，所以IE7最终读出的是第2个属性 \*height:120px;

在IE6下，三个属性IE6都认识，所以三个属性都可以读取，又因为第三个属性覆盖掉前2个属性，所以IE6最终读取的是第三个属性。

1 针对firefox ie6 ie7的css样式

现在大部分都是用!important来hack，对于ie6和firefox测试可以正常显示，但是ie7对!important可以正确解释，会导致页面没按要求显示！找到一个针对IE7不错的hack方式就是使用“\*+html”，现在用IE7浏览一下，应该没有问题了现在写一个CSS可以这样：

#1 { color: #333; } /\* Moz \*/

\* html #1 { color: #666; } /\* IE6 \*/

\*+html #1 { color: #999; } /\* IE7 \*/

那么在firefox下字体颜色显示为#333，IE6下字体颜色显示为#666，IE7下字体颜色显示为#999。

2 css布局中的居中问题

主要的样式定义如下：

body {TEXT-ALIGN: center;}

#center { MARGIN-RIGHT: auto; MARGIN-LEFT: auto; }

说明：

首先在父级元素定义TEXT-ALIGN: center;这个的意思就是在父级元素内的内容居中；对于IE这样设定就已经可以了。

但在mozilla中不能居中。解决办法就是在子元素定义时候设定时再加上“MARGIN-RIGHT: auto;MARGIN-LEFT: auto; ”

需要说明的是，如果你想用这个方法使整个页面要居中，建议不要套在一个DIV里，你可以依次拆出多个div，只要在每个拆出的div里定义MARGIN-RIGHT: auto;MARGIN-LEFT: auto; 就可以了。

3 盒模型不同解释.

#box{

width:600px;

//for ie6.0- w\idth:500px;

//for ff+ie6.0

}

#box{

width:600px!important

//for ff

width:600px;

//for ff+ie6.0

width /\*\*/:500px;

//for ie6.0-

}

4 浮动ie产生的双倍距离

#box{ float:left; width:100px; margin:0 0 0 100px; //这种情况之下IE会产生200px的距离 display:inline; //使浮动忽略}

这里细说一下block,inline两个元素,Block元素的特点是:总是在新行上开始,高度,宽度,行高,边距都可以控制(块元素);Inline元素的特点是:和其他元素在同一行上,…不可控制(内嵌元素);

#box{ display:block; //可以为内嵌元素模拟为块元素 display:inline; //实现同一行排列的的效果 diplay:table;

5 IE与宽度和高度的问题

IE不认得min-这个定义，但实际上它把正常的width和height当作有min的情况来使。这样问题就大了，如果只用宽度和高度，正常的浏览器里这两个值就不会变，如果只用min-width和min-height的话，IE下面根本等于没有设置宽度和高度。比如要设置背景图片，这个宽度是比较重要的。要解决这个问题，可以这样：

#box{ width: 80px; height: 35px;}html>body #box{ width: auto; height: auto; min-width: 80px; min-height: 35px;}

6 页面的最小宽度

min-width是个非常方便的CSS命令，它可以指定元素最小也不能小于某个宽度，这样就能保证排版一直正确。但IE不认得这个，而它实际上把 width当做最小宽度来使。为了让这一命令在IE上也能用，可以把一个<div> 放到 <body> 标签下，然后为div指定一个类：

然后CSS这样设计：

#container{

min-width: 600px;

width:e?xpression(document.body.clientWidth < 600? “600px”: “auto” );

}

第一个min-width是正常的；但第2行的width使用了Javascript，这只有IE才认得，这也会让你的HTML文档不太正规。它实际上通过Javascript的判断来实现最小宽度。

7 清除浮动

.hackbox{

display:table;

//将对象作为块元素级的表格显示

}

或者

.hackbox{

clear:both;

}

或者加入:after（伪对象）,设置在对象后发生的内容，通常和content配合使用，IE不支持此伪对象，非Ie 浏览器支持，所以并不影响到IE/WIN浏览器。这种的最麻烦的

……#box:after{

content: “.”;

display: block;

height: 0;

clear: both;

visibility: hidden;

}

8 DIV浮动IE文本产生3象素的bug

左边对象浮动，右边采用外补丁的左边距来定位，右边对象内的文本会离左边有3px的间距.

#box{

float:left;

width:800px;}

#left{

float:left;

width:50%;}

#right{

width:50%;

}

\*html #left{

margin-right:-3px;

//这句是关键

}

HTML代码

<DIV id=box>

<DIV id=left></DIV>

<DIV id=right></DIV>

</DIV>

9 属性选择器(这个不能算是兼容,是隐藏css的一个bug)

p[id]{}div[id]{}

p[id]{}div[id]{}

这个对于IE6.0和IE6.0以下的版本都隐藏,FF和OPera作用

属性选择器和子选择器还是有区别的,子选择器的范围从形式来说缩小了,属性选择器的范围比较大,如p[id]中,所有p标签中有id的都是同样式的.

10 IE捉迷藏的问题

当div应用复杂的时候每个栏中又有一些链接，DIV等这个时候容易发生捉迷藏的问题。

有些内容显示不出来，当鼠标选择这个区域是发现内容确实在页面。

解决办法：对#layout使用line-height属性 或者给#layout使用固定高和宽。页面结构尽量简单。

11 高度不适应

高度不适应是当内层对象的高度发生变化时外层高度不能自动进行调节，特别是当内层对象使用

margin 或paddign 时。例：

<div id=”box”>

<p>p对象中的内容</p>

</div>

CSS：

#box {background-color:#eee; }

#box p {margin-top: 20px;margin-bottom: 20px; text-align:center; }

解决方法：在P对象上下各加2个空的div对象CSS代码：.1{height:0px;overflow:hidden;}或者为DIV加上border属性。

屏蔽IE浏览器（也就是IE下不显示）

\*:lang(zh) select {font:12px !important;} /\*FF,OP可见\*/

select:empty {font:12px !important;} /\*safari可见\*/

这里select是选择符，根据情况更换。第二句是MAC上safari浏览器独有的。

仅IE7识别

\*+html {…}

当面临需要只针对IE7做样式的时候就可以采用这个HACK。

IE6及IE6以下识别

\* html {…}

这个地方要特别注意很多地主都写了是IE6的HACK其实IE5.x同样可以识别这个HACK。其它浏览器不识别。

html/\*\*/ >body select {……}

这句与上一句的作用相同。

仅IE6不识别

select { display /\*IE6不识别\*/:none;}

这里主要是通过CSS注释分开一个属性与值，流释在冒号前。

仅IE6与IE5不识别

select/\*\*/ { display /\*IE6,IE5不识别\*/:none;}

这里与上面一句不同的是在选择符与花括号之间多了一个CSS注释。

仅IE5不识别

select/\*IE5不识别\*/ { display:none;}

这一句是在上一句中去掉了属性区的注释。只有IE5不识别

盒模型解决方法

selct {width:IE5.x宽度; voice-family :""}""; voice-family:inherit; width:正确宽度;}

盒模型的清除方法不是通过!important来处理的。这点要明确。

清除浮动

select:after {content:"."; display:block; height:0; clear:both; visibility:hidden;}

在Firefox中，当子级都为浮动时，那么父级的高度就无法完全的包住整个子级，那么这时用这个清除浮动的HACK来对父级做一次定义，那么就可以解决这个问题 。

截字省略号

select { -o-text-overflow:ellipsis; text-overflow:ellipsis; white-space:nowrap; overflow:hidden; }

这个是在越出长度后会自行的截掉多出部分的文字，并以省略号结尾，很好的一个技术。只是目前Firefox并不支持。

只有Opera识别

@media all and (min-width: 0px){ select {……} }

针对Opera浏览器做单独的设定。

以上都是写CSS中的一些HACK，建议遵循正确的标签嵌套(div ul li 嵌套结构关系)，这样可以减少你使用Hack的频率，不要进入理解误区，并不是一个页面就需要很多的hack来保持多浏览器兼容)，很多情况下也许一个Hack都不用也可以让浏览器工作得非常好，这些都是用来解决局部的兼容性问题，如果希望把兼容性的内容也分离出来，不妨试一下下面的几种过滤器。这些过滤器有的是写在CSS中通过过滤器导入特别的样式，也有的是写在HTML中的通过条件来链接或是导入需要的补丁样式。

IE5.x的过滤器，只有IE5.x可见

@media tty {

i{content:"";/\*" "\*/}} @import 'ie5win.css'; /\*";}

}/\* \*/

IE5/MAC的过滤器，一般用不着

/\*\*//\*/

@import "ie5mac.css";

/\*\*/

下面是IE的条件注释，个人觉得用条件注释调用相应 Hack是比较完美的多浏览器兼容的解决办法。把需要hack的地方单独放到一个文件里面，当浏览器版本符合的时候就可以调用那个被Hack的样式，这样不仅使用起来非常方便，而且对于制作这个CSS本身来讲，可以更严格的观察到是否有必要使用hack，很多情况下，当我本人写CSS如果把全部代码包括Hack都写到一个CSS文件的时候的时候会很随意，想怎么Hack就怎么Hack，而你独立出来写的时候，你就会不自觉的考虑是否有必要Hack，是先Hack CSS？还是先把主CSS里面的东西调整到尽可能的不需要Hack？当你仅用很少的Hack就让很多浏览器很乖很听话的时候，你是不是很有成就感呢？你知道怎么选择了吧～～呵呵

IE的if条件Hack 自己可以灵活使用参看这篇IE条件注释

Only IE

所有的IE可识别

只有IE5.0可以识别

Only IE 5.0+

IE5.0包换IE5.5都可以识别

仅IE6可识别

Only IE 7/-

IE6以及IE6以下的IE5.x都可识别

Only IE 7/-

仅IE7可识别

Css 当中有许多的东西不不按照某些规律来的话，会让你很心烦，虽然你可以通过很多的hack，很多的!important 来控制它，但是你会发现长此以往你会很不甘心，看看许多优秀的网站，他们的CSS让IE6,Ie7,Firefox,甚至Safari,Opera运行起来完美无缺是不是很羡慕？而他们看似复杂的模版下面使用的Hack 少得可怜。其实你要知道IE 和 Firefox 并不不是那么的不和谐，我们找到一定的方法，是完全可以让他们和谐共处的。不要你认为发现了hack的办法，你就掌握了一切，我们并不是Hack的奴隶。

div ul li 的嵌套顺序

今天只讲一个规则。就是<div><ul><li>的三角关系。我的经验就是<div>在最外面，里面是<ul>，然后再是<li>，当然<li>里面又可以嵌套<div>什么的，但是并不建议你嵌套很多东西。当你符合这样的规则的时候，那些倒霉的，不听话的间隙就不会在里面出现了，当你仅仅是<div>里面放<li>，而不用<ul>的时候，你会发现你的间隙十分难控制，一般情况下，IE6和IE7会凭空多一些间距。但很多情况你来到下一行，间隙就没了，但是前面的内容又空了很大一块，出现这种情况虽然你可以改变IE的Margin，然后调整Firefox下面的Padding，以便使得两者显示起来得效果很相似，但是你得CSS将变得臭长无比，你不得不多考虑更多可能出现这种问题补救措施，虽然你知道千篇一律来Hack它们，但是你会烦得要命。

具体嵌套写法

遵循上面得嵌套方式，<div><ul><li></li></ul></div> 然后在CSS 里面告诉 ul {Margin:0px;Padding:0px;list-style:none;}，其中list-style:none是不让<li>标记的最前方显示圆点或者数字等目录类型的标记，因为IE和Firefox显示出来默认效果有些不一样。因此这样不需要做任何手脚，你的IE6、和IE7、Firefox显示出来的东西(外距，间距，高度，宽度)就几乎没什么区别了，也许细心的你会在某一个时刻发现一、两个象素的差别，但那已经很完美了，不需要你通过调整大片的CSS来控制它们的显示了，你愿意，你可以仅仅hack一两个地方，而且通常这种hack可以适应各种地方，不需要你重复在不同的地方调试不同的Hack方式–减轻你的烦恼。你可以ul.class1, ul.class2, ul.class3 {xxx:xxxx}的方式方便的整理出你要hack的地方，而统一Hack。尝试一下吧，再也不要乱嵌套了，虽然在Div+CSS的方式下你几乎可以想怎么嵌套就怎么嵌套，但是按照上面的规律你将轻松很多，从而事半功倍！

<script type="text/javascript">

/\*

\*\*兼容IE,Firefox 遵循W3C标准写法的图片无缝滚动代码（支持左移/右移功能）(本示例为上下两行图片进行双行滚动)powered by 25175.net

\*\*

\*\*实现思路：一个设定宽度并且隐藏超出它宽度的内容的容器demo，里面放demo1和demo2，demo1是滚动内容,demo2为demo1的直接克隆，

\*\*通过不断改变demo1的scrollTop或者scrollLeft达到滚动的目的，当滚动至demo1与demo2的交界处时直接跳回初始位置，

\*\*因为demo1与demo2一样，所以分不出跳动的瞬间，从而达到“无缝”滚动的目的。

\*/

var speed1=0 //图片左移的速度

var MyMar1 //setInterverval方法返回的间隔ID,些ID为全局的调用次数(用于第一行图片)

var MyMar2 //setInterverval方法返回的间隔ID,些ID为全局的调用次数(用于第二行图片)

var lefthit=0 //记录左移按钮点击次数

var righthit=0 ////记录右移按钮点击次数

//调用右移方法前,先让左移停止（clearInterval清除setInterval对方法的调用）

function clearleft(){

clearInterval(MyMar1)

clearInterval(MyMar2)

}

//图片左移的方法

function toleft(){

//判断是否是第一次按下左移按钮（图片往左开始移动的时候，不会在调用此方法。只有当按了右移按钮时，才会把lefthit重设为0）

if(lefthit==0){

speed1=30;//图片左移速度时间毫秒

//让的内容demo2与demo1内容一样

document.getElementById("demo2").innerHTML=document.getElementById("demo1").innerHTML

//上行图片移动的方法

function Marquee1(){

//当demo滚动至demo2的交界处。与就是与demo2重合时。offsetWidth与scrollLeft刚好相等

if(document.getElementById("demo2").offsetWidth-document.getElementById("demo").scrollLeft<=0){

//demo跳回初始位置

document.getElementById("demo").scrollLeft-=document.getElementById("demo1").offsetWidth

}

else{

//demo开始重新移动

document.getElementById("demo").scrollLeft++

}

}

//setInterval让Marquee1方法每隔多少秒执行一次，并把返回的调用次数ID

MyMar1=setInterval(Marquee1,speed1)

//当鼠标移动demo上面时。调用clearInterval清除MyMar1.这样图片便不会在移动了。移到图片时，第一行和第二行的图片都会停止

document.getElementById("demo").onmouseover=function()

{clearInterval(MyMar1);clearInterval(MyMar2)}

//当鼠标移除demo时,重新setInterval让方法执行，这样图片会继续移动

document.getElementById("demo").onmouseout=function()

{MyMar1=setInterval(Marquee1,speed1);clearInterval(MyMar2);MyMar2=setInterval(Marquee2,speed1)}

//下面的方法与上面的方法一样，用于第二行的图片向左移动

document.getElementById("demo\_c").innerHTML=document.getElementById("demo\_b").innerHTML

function Marquee2(){

if(document.getElementById("demo\_c").offsetWidth-document.getElementById("demo\_a").scrollLeft<=0){

document.getElementById("demo\_a").scrollLeft-=document.getElementById("demo\_b").offsetWidth}

else{

document.getElementById("demo\_a").scrollLeft++

}

}

MyMar2=setInterval(Marquee2,speed1)

document.getElementById("demo\_a").onmouseover=function()

{clearInterval(MyMar2);clearInterval(MyMar1)}

document.getElementById("demo\_a").onmouseout=function()

{MyMar2=setInterval(Marquee2,speed1); document.getElementById("demo").onmouseout()}

lefthit=1 //设置左移按钮，当按了一次左移后(不管按多少次)，在没按右移按钮之前，toleft方法里的的代码不会再被执行

righthit=0 //把右移按钮设为0,这样当下次点了右移按钮的时候，以便toright方法被调用

}

}

var MyMar3 //setInterverval方法返回的间隔ID,些ID为全局的调用次数(用于第一行图片)

var MyMar4 //setInterverval方法返回的间隔ID,些ID为全局的调用次数(用于第二行图片)

var speed2=0 //图片右移的速度

//调用移左方法前,先让右移停止（clearInterval清除setInterval对方法的调用）

function clearright(){

clearInterval(MyMar3)

clearInterval(MyMar4)

}

//图片右移方法(与图片左移方法思路一样)

function toright(){

//判断是否是第一次按下右移按钮（图片往右开始移动的时候，不会在调用此方法。只有当按了左移按钮时，才会把righthit重设为0）

if(righthit==0){

speed2=30;

document.getElementById("demo2").innerHTML=document.getElementById("demo1").innerHTML

function Marquee\_a(){

if(document.getElementById("demo").scrollLeft<=0){

document.getElementById("demo").scrollLeft=document.getElementById("demo2").offsetWidth

}

else{

document.getElementById("demo").scrollLeft--

}

}

MyMar3=setInterval(Marquee\_a,speed2)

document.getElementById("demo").onmouseover=function()

{clearInterval(MyMar3);document.getElementById("demo\_a").onmouseover()}

document.getElementById("demo").onmouseout=function()

{MyMar3=setInterval(Marquee\_a,speed2);clearInterval(MyMar4);MyMar4=setInterval(Marquee\_b,speed2)}

document.getElementById("demo\_c").innerHTML=document.getElementById("demo\_b").innerHTML

function Marquee\_b(){

if(document.getElementById("demo\_a").scrollLeft<=0){

document.getElementById("demo\_a").scrollLeft=document.getElementById("demo\_c").offsetWidth

}

else{

document.getElementById("demo\_a").scrollLeft--

}

}

MyMar4=setInterval(Marquee\_b,speed2)

document.getElementById("demo\_a").onmouseover=function()

{clearInterval(MyMar4);clearInterval(MyMar3)}

document.getElementById("demo\_a").onmouseout=function()

{MyMar4=setInterval(Marquee\_b,speed2);document.getElementById("demo").onmouseout()}

righthit=1 //设置右移按钮，当按了一次右移按钮后(不管按多少次)，在没按左移按钮之前，toright方法里的的代码不会再被执行

lefthit=0 //把左移按钮设为0,这样当下次点了左移按钮的时候，以便toleft方法被调用

}

}

</script>