**media媒体查询器用法总结**

**准备工作1：设置Meta标签**  
首先我们在使用Media的时候需要先设置下面这段代码，来兼容移动设备的展示效果：

**html 代码**



1

<name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">

**这段代码的几个参数解释：**  
**width = device-width：**宽度等于当前设备的宽度  
**initial-scale：** 初始的缩放比例（默认设置为1.0）  
**minimum-scale：**允许用户缩放到的最小比例（默认设置为1.0）  
**maximum-scale：**允许用户缩放到的最大比例（默认设置为1.0）  
**user-scalable：**用户是否可以手动缩放（默认设置为no，因为我们不希望用户放大缩小页面）  
  
**准备工作2：加载兼容文件JS**  
因为IE8既不支持HTML5也不支持CSS3 Media，所以我们需要加载两个JS文件，来保证我们的代码实现兼容效果：  
<!--[if lt IE 9]-->  
<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
<script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></script>  
<!--[endif]-->  
  
**准备工作3：设置IE渲染方式默认为最高(这部分可以选择添加也可以不添加)**  
现在有很多人的IE浏览器都升级到IE9以上了，所以这个时候就有又很多诡异的事情发生了，例如现在是IE9的浏览器，但是浏览器的文档模式却是IE8:  
  
  
为了防止这种情况，我们需要下面这段代码来让IE的文档模式永远都是最新的：

**html 代码**



1

<meta ;http-equiv="X-UA-Compatible" content="IE=edge">

不过我最近又发现了一个更给力的写法：

**html 代码**



1

<meta http-equiv="X-UA-Compatible" ;content="IE=edge, chrome=1">

怎么这段代码后面加了一个**chrome=1**，这个Google Chrome Frame（谷歌内嵌浏览器框架GCF），如果有的用户电脑里面装了这个chrome的插件，  
就可以让电脑里面的IE不管是哪个版本的都可以使用**Webkit**引擎及**V8**引擎进行排版及运算，无比给力，不过如果用户没装这个插件，那这段代码就会  
让IE以最高的文档模式展现效果。这段代码我还是建议你们用上，不过不用也是可以的。  
  
**进入CSS3 Media写法**  
我们先来看下下面这段代码，估计很多人在响应式的网站CSS很经常看到类似下面的这段代码：  
**@media screen and (max-width: 960px){        
body{background:#000;}  
}**  
  
这个应该算是一个**media**的一个标准写法，上面这段CSS代码意思是：当页面小于**960px**的时候执行它下面的CSS.这个应该没有太大疑问。  
  
应该有人会发现上面这段代码里面有个**screen**，他的意思是在告知设备在打印页面时使用衬线字体，在屏幕上显示时用无衬线字体。  
但是目前我发现很多网站都会直接省略**screen**,因为你的网站可能不需要考虑用户去打印时，你可以直接这样写：  
**@media(max-width:960px){  
body{background:#000;}  
}**  
  
**CSS2 Media用法**  
其实并不是只有CSS3才支持**Media**的用法，早在CSS2开始就已经支持**Media**，具体用法，就是在HTML页面的**heaad**标签中插入如下的一段代码：

**html 代码**



1

<link rel="stylesheet" type="text/css" media="screen" href="style.css">;

上面其实是CSS2实现的衬线用法，那CSS3的**Media**难道就只能支持上面这一个功能吗？答案当然不是，他还有很多用法。  
例如我们想知道现在的移动设备是不是纵向放置的显示屏，可以这样写：

**html 代码**



1

<link rel="stylesheet" type="text/css" media="screen and ;(orientation:portrait)" ;href="style.css">

我们把第一段的代码也用CSS2来实现，让它一样可以让页面宽度小于960的执行指定的样式文件：

**html 代码**



1

<link rel="stylesheet" type="text/css" media="screen and (max-width:960px)" href="style.css">

既然CSS2可以实现CSS的这个效果为什么不用这个方法呢，很多人应该会问，但是上面这个方法，最大的弊端是他会增加页面http的请求次数，增加了页面负担，我们用CSS3把样式都写在一个文件里面才是最佳的方法。  
  
  
**回归CSS3 Media**  
上面我们大概讲了下CSS2的媒体查询用法，现在我们重新回到CSS3的媒体查询，在第一段代码上面我用的是小于**960px**的尺寸的写法，那现在我们来实现等于**960px**尺寸的代码：  
**@media screen and (max-device-width:960px)  
{background:red;}  
}**  
  
然后就是当浏览器尺寸**960px**时候的代码了：  
**@media screen and (min-width:960px){   
body{background:orange;}  
}**  
  
  
我们还可以混合使用上面的用法：  
**@media screen and (min-width:960px) and (max-width:1200px){  
body{background:yellow;}  
}**  
上面的这段代码的意思是当页面宽度大于960px小于1200px的时候执行下面的CSS。  
  
  
**Media所有参数汇总**  
以上就是我们最常需要用到的媒体查询器的三个特性，大于，等于，小于的写法。媒体查询器的全部功能肯定不止这三个功能，下面是我总结的它的一些参数用法解释：  
**width:**浏览器可视宽度。  
**height:**浏览器可视高度。  
**device-width:**设备屏幕的宽度。  
**device-height:**设备屏幕的高度。  
**orientation:**检测设备目前处于横向还是纵向状态。  
**aspect-ratio:**检测浏览器可视宽度和高度的比例。(例如：aspect-ratio:16/9)  
**device-aspect-ratio:**检测设备的宽度和高度的比例。  
**color:**检测颜色的位数。（例如：min-color:32就会检测设备是否拥有32位颜色）  
**color-index:**检查设备颜色索引表中的颜色，他的值不能是负数。  
**monochrome:**检测单色楨缓冲区域中的每个像素的位数。（这个太高级，估计咱很少会用的到）  
**resolution:**检测屏幕或打印机的分辨率。(例如：min-resolution:300dpi或min-resolution:118dpcm)。  
**grid:**检测输出的设备是网格的还是位图设备。