**浅谈前端移动端页面开发（布局篇）**

前言的一些碎碎念：最近一直在写移动端的页面，不过一直是用的别人造好的轮子，很多时候并没有想那是为什么，那是怎么样要那么写，就跟着别人的文档去了。本以为自己对移动端的那一丢丢理解，结果很多东西都特么有问题，所以，今天停下了手中的一些东西，来谈下移动端的布局方案吧

内容有些长，这也是我第一次写博客，不足之处还请严厉指出

**一. viewport**

1. 什么是viewport

简单来讲，viewport就是浏览器上，用来显示网页的那一部分区域了，也就是说，浏览器的实际宽度，是和我们手机的宽度不一样的，无论你的手机宽度是320px，还是640px，在手机浏览器内部的宽度，始终会是浏览器本身的viewport。如今的浏览器，都会给自己的本身提供一个viewport的默认值，可能是980px，或者是其他值。就以手机来说吧，目前，新版本的手机浏览器，绝大部分是以980px作为默认的viewport值的。我这里对新版本的不同平台下的浏览器做了测试，经过测试，iphone下的默认viewport为980px，安卓下的浏览器，目前主流的最新浏览器（比如chrome，还有很多国产的像qq，uc）的viewport也是980px了。

1. viewport是用来干什么的

viewport的默认值，一般来说是大于手机屏幕的。这样就可以做到当我们在浏览桌面端网页的时候，可以让桌面端端网页正常显示（我们普通页面设计的时候，一般页面的主区域是以960px来做的，所以980px这个值，可以做到桌面端网页的正常显示）。但是，其实我们手机的屏幕宽度是没有960px的，因此浏览器会出现横向滚动条。同时，即使是基于980的viewport，我们在移动端浏览我们的桌面页面的体验其实也并不好，所以，一般的，我们会专门给浏览器设计一个移动端的页面。

1. 对viewport的控制

如今可以绝大部分浏览器里（即主流的安卓浏览器和ios），都支持对viewport的一个控制了。一般的，我们会这么写。

viewport默认有6个属性

* + width: 设置viewport的宽度（即之前所提及到的，浏览器的宽度详），这里可以为一个整数，又或者是字符串"width-device"
  + initial-scale: 页面初始的缩放值，为数字，可以是小数
  + minimum-scale: 允许用户的最小缩放值，为数字，可以是小数
  + maximum-scale: 允许用户的最大缩放值，为数字，可以是小数
  + height: 设置viewport的高度（我们一般而言并不能用到）
  + user-scalable: 是否允许用户进行缩放，'no'为不允许，'yes'为允许

我们把这个标签是在head里面，像这样

<meta name="viewport" content="initial-scale=1">

这样就可以做到对viewport的控制了

**二. 关于我们的设备**

1. 三个需要了解的概念:
   * PPI: 可以理解为屏幕的显示密度
   * DPR: 设备物理像素和逻辑像素的对应关系，即物理像素/逻辑像素
   * Resolution: 就是我们常说的分辨率
2. 物理像素与逻辑像素

看了我们上面内容一的第一点之后，或许有些人会有些疑问，我的安卓手机，或者iphone6plus(目前应该仅限于这一款机型吧)，买回来的是1920x1080的或者其他更高的，比我之前所谓的那个viewport默认的980px要大。

这样的问题，也就是我之前所说的物理像素与逻辑像素的关系了（即DPR）。以1920x1080为例，1080为物理像素，而我们在viewport中，获取到的，比如"width-device"，是逻辑像素。所以之前viewport的默认值，所比对的大小，其实是逻辑像素的大小，而非物理像素的大小。

以iphone6为例，在不做任何缩放的条件下，iphone6的获取到的'width-device'为375px,为屏幕的逻辑像素。而购买时我们所知的750px，则为屏幕的物理像素。

1. [CSS](http://www.xiaoxiangzi.com/Programme/CSS/)的问题

有了上面第二点的一些基础，还是以iphone6为例，我们可以知道，其实我们所写的1px，在iphone6上为2px的物理像素。所以，最后的，给出一个结论。就是我们写的1px，在移动端，是逻辑像素的1px，而非物理像素的1px。

**三. 使用rem布局**

1. 简单说下rem

rem是根据页面的根元素的font-size的一个相对的单位，即

html{

font-size: 16px;

}

比如当我们在一个div中，如此写

div{

width: 2rem;

}

那么我们的width，是16\*2 = 32px

1. rem做到适配不同分辨率

这个是现在手机淘宝的移动端的解决方案，即使用rem的特性，来对页面进行布局。

下面举一个例子

假定设计稿的大小为750，那么我们则将整个图分成100份来看（下面的题外话会说明为什么会分成100份来看）

那么，我们现在就让根部元素的font-size为75px

html{

font-size: 75px;

}

那么，我们现在就可以比对设计稿，比如设计稿中，有一个div元素，宽度，高度都为75px,那么我们这样写即可

div{

height: 1rem;

width: 1rem;

}

可能看到这里，一些人还是不明白怎么用rem做到适配不同的分辨率，那么我们再来

现在，我们换设备了，不用这个设备是一个width为640的手机

那么这个时候，我们的rem单位就起到作用了。

我们的rem全是根据html的font-size来改变的，所以说，这个时候，我们只需要把html下的font-size改成64px。那么，我们之前的div，因为是根据html下的font-size动态变化的，那么。此时也就变成了宽度和高度都为64px的东西了。这样，就可以做到适配不同的屏幕分辨率了。（其实就是个等比缩放）

总结一下，我们的解决方案，其实就是 设计稿的像素/html的font-size = 用来代替px的rem。

这一个步骤，我们需要通过[JS](http://www.xiaoxiangzi.com/Programme/JavaScript/)来进行操作。

对于js的操作在下面会提到。

1. DPR的问题

视觉姐姐给了我们设计稿，并交由我们实现，那么，我们应该去认真的实现:-)（试想你做了一张图，而前端很多地方并没有按照你所想的，你所给的去做，而是私自改变了很多东西，你肯定会不高兴的）

那么1px会出现什么问题呢。

还记得我们第二大点讲的，我们的设备，是有物理像素和逻辑像素的。而假设我们的设计稿是750的，同时还是以iphone6为例，此时如果我们的viewport是这样的

<meta name="viewport" content="initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no">

之前说过，在不做任何缩放的条件下，iphone6获取到的viewport为375px。

然后我们的页面中有个div，他有一个边框值，如下

div{

height: 5rem;

widht:5rem;

border: 1px solid #000

}

此时我们写的1px，实际上是逻辑像素，而我们在iphone6上看到的是物理像素，于是这个时候，我们眼睛所看到的其实是2px（参考第二点第三个问题）

所以此时我们需要在viewport上做文章了，此时先明确，如果要获取到真正的1px，那么我们需要这么做，将viewport改为

<meta name="viewport" content="initial-scale=0.5, maximum-scale=0.5, minimum-scale=0.5, user-scalable=no">

即对屏幕做0.5倍的缩放。这样，我们就能得到实际的1px。

所以到这里，我们还要明确一点，viewport的meta标签，我们这里也只能通过js来动态生成。

同时，这样写，据说还可以避免比如inline的SVG等元素按照逻辑像素的渲染。避免了整个页面清晰度的打折（其实我并不能看出来）

1. 文字适配问题

最近深深纠结与rem与px做字体单位的问题，还是先分别谈下二者吧。

rem与px的特点：

* + 以rem作为字体单位:我们可以让页面整体的文字，也跟随着html的font-size来进行改变，这样，在不同的屏幕下，可以做到文字相对屏幕的比例是一样的。
  + 以px作为字体单位: 这个是目前很多网站还是依然采用的方法。因为以上面所写的，以rem作为字体单位。无论在任何屏幕下面，我们的文字都会根据屏幕做一个适应。试想这样一个场景。你买了一个大屏手机（5.7寸的），而别人用的是4寸的手机。以rem作为字体单位的话，那大屏手机看到的文字多少和小屏手机确实一样的了。这样来做，其实并不符合我们买大屏手机的期待。同时，以rem作为字体单位，可能会导致出现很多奇怪的字体大小（毕竟是根据html的font-size动态变化的嘛），同时这其中还涉及到了一个点阵尺寸的概念，这个在下面来讲。

字体大小引发的系列问题：

* + 字体大小：我们平时也看过，很多网站，是不以奇数作为字体大小的。我稍微查了些东西，在知乎上的现在网页设计中的为什么少有人用 11px、13px、15px 等奇数的字体？问题下，有一些比较好的解答，我就不再多说（我也并不能比这个问题说的更多），总的来说，其实就是偶数宽度的字体能够显得均衡，以及一个点阵的问题。不过因为要谈及点阵，所以我拿上面回答中的一个内容举例。
    - 倘若一个字体，只提供了12px，14px，16px的点阵。那么当你写13px，15px，17px的时候。就并没有其字体大小所对应的点阵。那么这样就造成了一个问题。他们会使用其相邻的点阵，比如对应使用了12px，14px，16px的点阵，而导致一个问题，文字占用的大小确实改变，但点阵却并没有改变。

文字适配的解决方案：

上面说了这么多，我们总要有一套解决方案吧

对于一些标题性的文字，我们依然可以用rem。让他随着屏幕来进行缩放，因为标题性文字一般较大，而较大的文字，点阵对其影响就越小。这样，即使出现奇怪的尺寸，也能够让字体得到很好的渲染。

对于一些正文内容的文字（即站在使用者的角度，你不希望他进行缩放的文字）。我们采用px来进行处理。

**四.安卓与ios不得不说的问题（解决篇）**

1. 在 三.使用rem布局 里面，我们给出了各种情况的解决方案，并且，在我举例的时候，热衷于使用iphone来举例，但其实，上面的所有问题，不是仅仅iphone会出现的问题，安卓也是一样。但是，如果你已经看完了上面，那么这里，才是真正给出我们解决方案的地方，并且，这个解决方案并不完善。
2. 谈谈iphone的r屏与安卓的各种屏

rem布局也好，用viewport进行缩放也罢，文字的适配问题也是，都是基于我们想对各个不同的设备所进行的匹配。这套方案很好，然而也有其兼顾不到的地方。即安卓和ios的屏幕的一些问题，当然，细的东西我们不谈，我们只谈dpr。

* + 先谈iphone
    - 其实iphone为开发者考虑到了很多东西，为了让开发者便于开发，在6plus出现之前，iphone的dpr始终也就是2（即前面所谈的物理像素/逻辑像素＝2），即使是6plus出现了，iphone到底其实也就只有2，3这两个dpr。我们很容易对其做到兼顾。
  + 再谈安卓
    - 安卓并没有对自己的屏幕叫做r屏，但是其原理和iphone的r屏可以说是一样。r屏做的是什么，把两个（三个）物理像素，丢到了一个逻辑像素里面，让屏幕展现的更清晰（当然，这是我片面的理解，不过我觉得大体来说并没有错，我们也不用去深入探讨r屏还有什么东西，我也并不懂）。而安卓也是一样，他也同样把n个物理像素丢到了一个逻辑像素里面。而这里的n，也就是dpr值（所以当我看到好多人问安卓为什么不采用r屏的时候，我真的也是……醉了？）。而安卓的dpr值，并不像iphone那样，就只有两个值。安卓的dpr是千奇百怪的，可能是1.5，2，3，4，2.5等等的都有。（甚至我还看到了1.7之类的，安卓的各个设备商，玩的真尼玛high啊。怎么高兴怎么来。）

所以，对安卓的屏幕的dpr的处理，其实是很头疼的，因为，他和我们对字体的处理，有了很大的冲突。这个在下面提及

1. 首先看看手淘的解决方案
   * rem布局

用js获取到页面的宽度，然后对其进行宽度/10的处理，再将其写到html的font-size中。手淘的flexible.js里面的这一部分，并为了方便看懂做了些改写。大体就是这样的

function refreshRem(){

var docEl = window.document.documentElement;

var width = docEl.documentElement.getBoundingClientRect().width;

var rootSize = width/10;

docEl.style.fontSize = rootSize + 'px';

}

* + dpr的配置

首先，在引入flexible.js之前，我们可以对dpr进行手动的配置，即使用自定义的meta标签来配置dpr（看清楚是flexible，而非viewport）

<meta name="flexible" content="initial-dpr=2,maximum-dpr=3" />

iniital-dpr是把dpr强制设定为给定的值，而maximum-dpr则是给出一个最大的dpr限制，然后对其和系统的dpr做一个比较。

然后依然为了方便阅读我把flexble.js这一部分的代码抽象出来，

var doc = window.document

var metaEl = doc.querySelector('meta[name="viewport"]');

var flexibleEl = doc.querySelector('meta[name="flexible"]');

var dpr = 0;

var scale = 0; //缩放比例

//在meta标签中，已经有了viewport，则使用已有的viewport,并根据meta标签，对dpr进行设置

if (metaEl) {

console.warn('将根据已有的meta标签来设置缩放比例');

var match = metaEl.getAttribute('content').match(/initial\-scale=([\d\.]+)/);

if (match) {

scale = parseFloat(match[1]);

dpr = parseInt(1 / scale);

}

//如果在meta标签中，我们手动配置了flexible，则使用里面的内容

} else if (flexibleEl) {

var content = flexibleEl.getAttribute('content');

if (content) {

var initialDpr = content.match(/initial\-dpr=([\d\.]+)/);

var maximumDpr = content.match(/maximum\-dpr=([\d\.]+)/);

if (initialDpr) {

dpr = parseFloat(initialDpr[1]);

scale = parseFloat((1 / dpr).toFixed(2));

}

if (maximumDpr) {  
dpr = parseFloat(maximumDpr[1]);  
scale = parseFloat((1 / dpr).toFixed(2));  
}  
}  
}

这样,我们通过flexible的分析与获取，对dpr进行了书写。不过其实这里，是有个问题的。即在书写maximum的的情况下，其实根本没有像文档中给我们的说法一样，做一个比较，而是做了和initialDpr一样的一个处理。不过这里也不对其做一个探讨了。

然后，这套解决方案，然后当我们在meta标签里面并没有对viewport以及flexible两个的任意一个进行书写的时候，他也是会自动获取一个dpr值的

if (!dpr && !scale) {

var isAndroid = window.navigator.appVersion.match(/android/gi);

var isIPhone = window.navigator.appVersion.match(/iphone/gi);、

//devicePixelRatio这个属性是可以获取到设备的dpr的

var devicePixelRatio = win.devicePixelRatio;

if (isIPhone) {

// iOS下，对于2和3的屏，用2倍的方案，其余的用1倍方案

if (devicePixelRatio >= 3 && (!dpr || dpr >= 3)) {

dpr = 3;

} else if (devicePixelRatio >= 2 && (!dpr || dpr >= 2)){

dpr = 2;

} else {

dpr = 1;

}

} else {

// 其他设备下，仍旧使用1倍的方案

dpr = 1;

}

scale = 1 / dpr;

}

这里我们可以看到，手机淘宝并没有对安卓的dpr进行一个适配，原因之后再讲。

然后到了这里，我们获取到了我们需要的dpr值，并根据dpr值获取到了我们所需要的缩放值（即scale）

然后我们要做的，就是在并没有viewport的meta标签对情况下自己动态将这个标签写进我们的header，形式是这样的

<meta name="viewport" content="initial-scale=scale, maximum-scale=scale, minimum-scale=scale, user-scalable=no">

这样，dpr的配置，也就完成了，当然，安卓设备并没有对dpr进行一个配置（上面的动态生成就不给出js了）

* + 文字的解决方案

由于手淘暂时并没有对安卓做一个处理，所以，这里，只是对iphone做了一个处理

即在html上，加入了一个自定义属性，data-dpr。

<html data-dpr='dpr'></html>

还是以750的设计稿为例（即iphone6）

假如设计稿上某a标签是32px，那么，我们要这么写

a{

font-size: 16px

}

/\*iphone6\*/

[data-dpr='2'] a{

font-size: 32px

}

/\*iphone6plus\*/

[data-dpr='3'] a{

font-size: 32px

}

1. 现在的一些问题

正如我们看到的，手淘目前的方案里面，是没有考虑到安卓dpr的问题的。即，这套方案，只对于iphone的r屏做了一个处理，而对于安卓，并没有做dpr的处理。我们来分析下原因吧（个人拙见）。

我们希望字体能够以px来展现，同时，我们也希望我们的东西能对dpr做一个适配。对于ios，这自然是可行的，即采用了data-dpr的自定义属性来调整文字。4到6写一套字体大小，6p写一套字体大小，然后在对dpr为1的屏幕写一套字体大小。其实这种写法还是很恶心，不过基于对dpr的适配，这样写也算是个解决方案了。

不过同样的解决方案到安卓就不行了，安卓的dpr有时候会很乱（比如现在在goole的手机测试里面可以看到，安卓的dpr，lg的某些设备还采用了1.7那样的奇怪dpr）。而当1.7dpr这种不规范的数字出现的时候，我们就不能采用之前的解决方案了，比如

[data-dpr='1.7'] a{

font-size: 25px

}

这样的东西是不可能去写的，那万一还有2.25，2.5之类的呢？我们都要拿去匹配么？

其实现在，因为我们通过devicePixelRatio可以获取到安卓的dpr值，即可以做到对安卓设备的dpr一个匹配。但是，文字如果采用px的话，确实是很难做到匹配的。

即总结一下，就是说，对于安卓的dpr匹配，目前来说，是没有什么问题的，但是，对于dpr匹配之后的字体，那肯定是有问题的。

常见的dpr下的字体，我们依然可以解决，但是不常见的dpr，我们确实很难做到对dpr的解决。那如何解决这些问题呢。目前以我本人这个不太灵光的脑袋，确实也不晓得该如何进行一个处理了，起码做不到很好的解决。

不过，还是丢上些个人的观点吧。

在之前的对dpr的判断中，是根据了设备进行判断，即安卓不对dpr进行改变，仅对ios的设备进行改变。那么，我们其实可不可以以dpr的值来做一个处理呢？即像这样写

if (!dpr && !scale) {

//devicePixelRatio这个属性是可以获取到设备的dpr的

var devicePixelRatio = win.devicePixelRatio;

//判断dpr是否为整数

var isRegularDpr = devicePixelRatio.toString().match(/^[1-9]\d\*$/g)

if (isRegularDpr) {

// 对于是整数的dpr，对dpr进行操作

if (devicePixelRatio >= 3 && (!dpr || dpr >= 3)) {

dpr = 3;

} else if (devicePixelRatio >= 2 && (!dpr || dpr >= 2)){

dpr = 2;

} else {

dpr = 1;

}

} else {

// 对于其他的dpr，人采用dpr为1的方案

dpr = 1;

}

scale = 1 / dpr;

}

我们对这里做了一点点修改，即来判断dpr是否是规则的，也就是是否是我们常见的1，2，3等，然后，我们只对规则的dpr，来进行一个字体的处理。这样，iphone依然还是用之前的匹配方案。而其实目前安卓，很多的设备还是比较常见的dpr了，所以我们这里，将之前对设备的判断，转变成对dpr是否是整数的一个判断。其他地方不变，可以解决对安卓dpr的部分匹配。

同样，开发的时候，如果并不在乎字体的问题的话，大可以直接使用rem。那样是可以做到dpr和文字都适配的问题。不过正如我们讲到字体的时候所说的，使用rem是很多用户不希望的（大屏机还是和小屏机看到一样多的内容），同时，还有点阵的问题。

好，东西写到这里，也将近到了尾声。第一次写这么长的东西，感觉好累啊=\_=。嗯还有篇2000字的检讨要写，默默匿了去写检讨了。

**参考**

手机淘宝的flexible设计与实现

**题外话：**

1. iphone6plus很有趣的地方

iphone6plus照理来说的，其实际dpr是2.87左右的，不过，为了方便开发者来开发，iphone6plus对其做了一个调整，将dpr调整为3，然后在对屏幕进行了一个缩放。这样做，自然是方便了开发者前去开发，然而，这样做，也有了一些性能上的损失。（iphone为开发者考虑的还是挺周全的，看看隔壁安卓，dpr怎么爽怎么来，都特么自己玩自己的）

1. 有意思的vh和vw

vh，vw目前还存在很大程度的兼容性问题，所以还并没有采用。

vh，vw有什么特点呢

这两个元素分别会把屏幕上的可视高度（说通俗点就是你手机屏幕那个框框头装起的东西），宽度，分成100份来看，比如先前我们用rem来处理的地方，我们需要在html元素下写上font-size: 75px,然后再在div下写上width:1rem。而有了vh，vw之后，我们如此处理html的font-size就好。

html {

font-size: 10vw;

}

这样写，省去了一部js操作的步骤。