# 移动适配页面实践与总结

作者：田翔 | 2016-08-31 14:38

在网易100分这个主要面向中小学生的产品之中，从移动平台获取的流量占据了差不多75%，由此，移动适配已经成为了项目之中重点，这两天开始着手网易100分的移动web页面适配。再此，发帖总结一下在适配的时候100分所采用的方案，以及开发中所遇到的大坑小坑各种坑。

## 布局

### lib-flexible

> 这是手淘团队开源的一个基于viewport与rem单位的H5适配开源库，原理非常的简单，通过手机dpr动态设置viewport的缩放以及html的font-size来兼容不同尺寸屏幕下页面的效果。

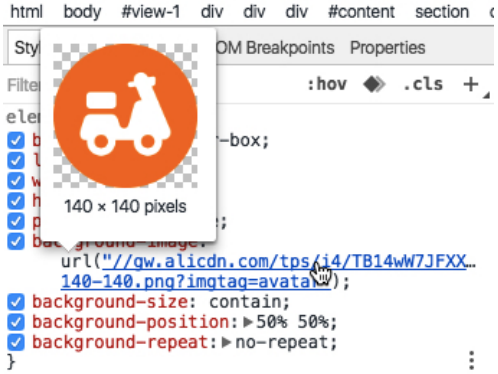
在100分的项目中，我们没有采取他默认的缩放加rem的方式，强制定义缩放比例为1，只使用了rem方案来适应不同的手机。查看lib-flexible代码，发现只要手动设置了meta标签，就会按照自己的缩放比例进行计算。

```
// flexible源代码第11行，判断缩放的方式
var metaEl = doc.querySelector('meta[name="viewport"]');
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);
        }
} else if (flexibleEl) {
        // .......
}
```

### 坑

> 在使用lib-flexible的时候，由于水土不服等多种原因，遇到了很多坑，如果有要采用同样方案的小伙伴，可以重点注意一下。

- 图标：如果没有自定义，android的dpr强行被设置为了1，也就是说，android和ios用了两套不同的方案，这会带来什么样的后果？刚才介绍过，该库是通过手机dpr动态设置viewport的缩放。那么，也就是说在android手机上面无法通过缩放来达到最好的效果，1倍图在dpr2，3下的显示效果视觉看到会来找你的。但是我们发现android中淘宝的图片是正常的，因为他首页的每张图片都是一个请求，并且类似于img,通过rem控制显示的大小。如图:




100分没有采取此方案，而是采取了简单粗暴的雪碧图方案，下文细说。

- iframe：在100分的项目中，强行设置了他的缩放比例，如果页面中有**iframe**的话，iframe同样会被缩放，

众所周知，URS登陆组件本质就是一个iframe,我们可以修改其中的样式来达到自己想要的效果，问题在于我们无法修改iframe上根html的font-size,因此缩放iframe时，无法自动变换iframe中各个属性的大小。如图：


上图中，因为父页面使用了缩放，导致iframe内容也同时做出了响应了缩放，但是因为iframe子页面无法设置font-size,因此显示非常奇怪。

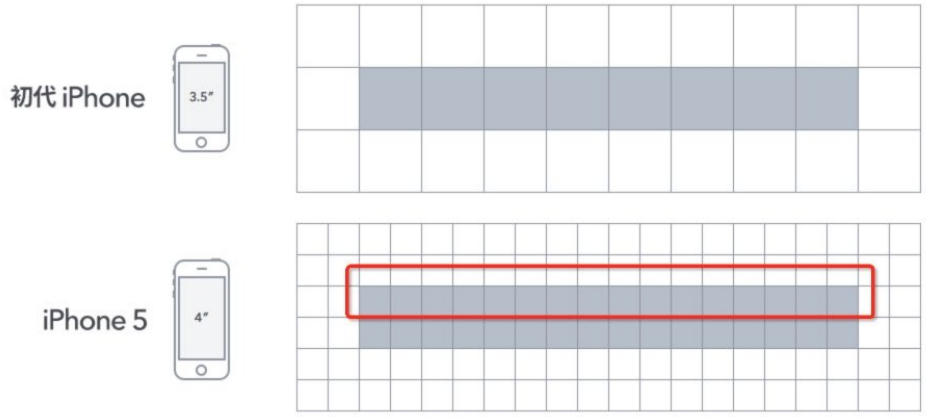
- 文字：在lib-flexible中，建议继续使用px作为文字的单位，因为缩放viewport的原因，需要对不同dpr下的手机设置不同的font-size,需要配合属性选择器使用。
- 通常视觉GG/MM们给出的都是视觉稿都以**640、750px**分辨率为标准，其中都是以像素单位来标注盒子模型的各个属性，如果基准**font-size**是**100px**的话，计算起来到还方便，但像lib-flexible他的根font-size设置的公式是：（手机分辨率/10），即750下font-size为**75px**,在这个数值下转换px和rem的话，会增加前端的计算量。因此，100分使用了scss中的宏来自动转换，如有需要的童鞋，可以直接拷贝使用：

```
$rem-baseline: 75px !default;
@function rem-separator($list) {
    @if function-exists("list-separator") == true {
        @return list-separator($list);
    }

    $test-list: ();
    @each $item in $list {
        $test-list: append($test-list, $item, space);
    }

    @return if($test-list == $list, space, comma);
}

@mixin rem-baseline($zoom: 100%) {
    font-size: $zoom / 16px * $rem-baseline;
}

@function rem-convert($to, $values...) {
    $result: ();
    $separator: rem-separator($values);
    @each $value in $values {
        @if type-of($value) == "number" and unit($value) == "rem" and $to == "px" {
            $result: append($result, $value / 1rem * $rem-baseline, $separator);
        } @else if type-of($value) == "number" and unit($value) == "px" and $to == "rem" {
            $result: append($result, $value / ($rem-baseline / 1rem), $separator);
```

```
        } @else if type-of($value) == "list" {
            $result: append($result, rem-convert($to, $value...), $separator);
        } @else {
            $result: append($result, $value, $separator);
        }
    }
    @return if(length($result) == 1, nth($result, 1), $result);
}

@function px2rem($values...) {
    @return rem-convert(rem, $values...);
}

@mixin px2rem($properties, $values...) {
    @if type-of($properties) == "map" {
        @each $property in map-keys($properties) {
            @include px2rem($property, map-get($properties, $property));
        }
    } @else {
        @each $property in $properties {
            #{$property}: rem-convert(rem, $values...);
        }
    }
}

// 使用方式
.login_box {
    @include px2rem(margin, 0 auto);
    @include px2rem(padding, 20px 0 15px);
}
```

刚才说到，100分项目采取了强制设置**initial-scale=1**的方式来按照视觉稿还原，这么做也有一些无法解决的问题。

- 1px问题，先看一张图：




这是在不同dpr手机下显示1像素的实际像素图。我们知道在2倍retina屏幕上相对于普通屏幕，1个像素需要4个物理像素来组成，这样才能达到高清的作用，如果视觉GG/MM在2倍稿子之中画出了1px的线，即在视觉640分辨率下，1px实际显示显示的是红线圈出来的灰色区域，如果在initial-scale=1情况下看，对应css应该为`0.5px`，对部分android机器而言，会将0.5px识别为0，因此这里一般有一个hack

```
transform:scaleX(0.5)或transform:scaleY(0.5)
```

这个hack也只是为了视觉而做出的无奈之举，解决这个问题的关键在于去**交流**

- 如果你的视觉设计师要求严格，对该界面把控非常严格的话，应该动态去使用scale的hack去改变。
- 如果部分区域实在没有必要**过度**调整，和视觉GG/MM交流一下更好。
- px最好用双数。

### flex + box-sizing

- 弹性盒子作为布局神器可以完成很多富有创造力的布局，只要记住常用的属性，用起来特别的方便。
- box-sizing，box-sizing作为css3的属性，给前端人员的尺寸测量带来了极大的便利性。没有用这个属性之前，我们设置宽度的时候，总是要去除**margin**,**padding**来计算，使用它后，不用再单独计算宽度和间距，全部根据**视觉稿**测量的为准。

起初采用此方案时，担心flex布局的兼容性，特意把flex的大部分属性进行了一次测试，发现一切安好。 测试系统：**IOS 8**、**IOS`` 9**、**android 4.2**、**android 4.4**、**android 6.0**。 有其他需求的话，可以自行用手机测试(只测试了常用的属性，欢迎提交其他属性测试) 测试地址

#### 坑

- 在华为的部分手机上发现flex不支持行内元素，必须改成块级元素才能被支持。
- 预留位置(目前适配页面不多，还没遇到其他问题, 后期补充)。

## 图片

### 雪碧图

为什么在100分项目中执迷于采用雪碧图方案呢，在图片方案选择中明明还有不少其他方案，如淘宝的每个图标就是一张大图的方案，图标采取base64方案在通过background-size缩放，字体图标方案。

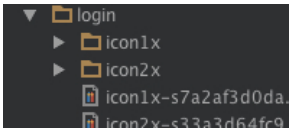
- 字体图标方案应该是最好的解决方案了，能非常契合的和rem布局配合，能完成不同场景下显示效果一致的问题。难度在于，做字体**工作量**巨大,让视觉去做这么多矢量图，在转成svg与字体文件，视觉不愿意去做。
- 淘宝每个图标一张大图，在任何dpr的机型之中，都对应的同一张图片，好处了通过rem的方式，在任何尺寸下都能完美的还原视觉稿，但同时给对用户流量造成了不必要的开支，即在dpr1的pad上面，无论如何都是请求最高分辨率的图片。同时，每张图片一个请求不符合减少HTTP请求这一条优化原则。因此没有采用。
- base64图片少还好，如果图片多的话，尤其是在于retina这种需要双倍图的情况下，会大大增加css文件的大小（gzip压缩base64字符也并不明显），如果css过大，还不如增加一个请求来请求图片资源。如果用了自动化工具还好，一旦没有使用自动化工具，代码混乱且难以维护。这种方式比较适合图片小而且不经常更换的场景，因此该方案不能作为主体方案。

说了这么多其他方式，说说100分采取的传统雪碧图方案，相对于传统雪碧图的繁琐。在100分项目中，采取的通过compass的方式进行动态生成，并且为了兼容retina屏幕，使用了2套不同的图片进行生成，在通过媒体查询的方式确定当前手机需要访问什么样的图片资源。在利用scss强大的mixin，完成自己想要的功能。 相关代码和项目结构如下：

```
//生成compass雪碧图
@function generate-sprite($path){
    @return sprite-map($path, $spacing: 10px, $layout: vertical);
}
// 获取某一张图的background-position
@mixin sprite-background($name,$normal,$retina:null) {
    background-repeat: no-repeat;
```

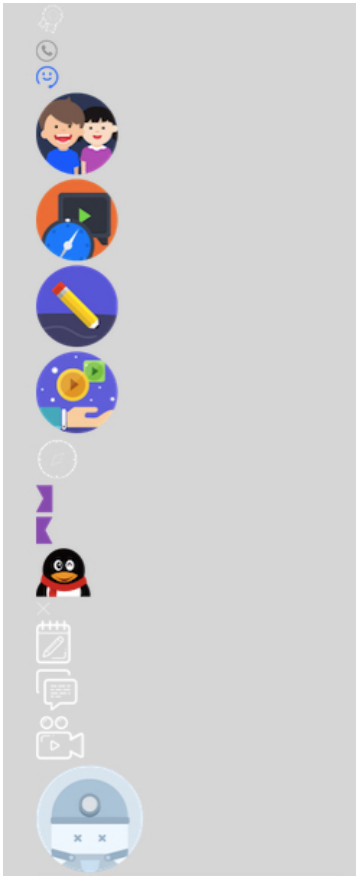
```
    background-image: sprite-url($normal);
    background-position: sprite-position($normal, $name);
    height: image-height(sprite-file($normal, $name));
    width: image-width(sprite-file($normal, $name));

    // retina屏幕样式
    @if ($retina != null){
        @media (min--moz-device-pixel-ratio: 1.3),
        (-o-min-device-pixel-ratio: 2.6/2),
        (-webkit-min-device-pixel-ratio: 1.3),
        (min-device-pixel-ratio: 1.3),
        (min-resolution: 1.3dppx) {
            background-image: sprite-url($retina);
            background-position: 0 round(nth(sprite-position($retina, $name), 2) / 2);
            height: round(image-height(sprite-file($retina, $name)) / 2);
            width: round(image-width(sprite-file($retina, $name)) /2);
            $double-width: ceil(image-width(sprite-path($retina)) / 2);
            $auto-height: auto;
            @include background-size($double-width $auto-height);
        }
    }
}

//调用方式, 写起来非常的简单
/* 设置不同倍数的雪碧图*/
$sprites: generate-sprite("mobile/mob/member/login/icon1x/*.png");
$sprites-retina: generate-sprite("mobile/mob/member/login/icon2x/*.png");

.icon-logo {
    @include sprite-background(logo, $sprites, $sprites-retina);
}
```

项目结构：



根据不同dpr取不同的图片

生成的图片样式如下:



觉得不好看浪费空间的话可以设置排列方式。反正不用自己量，who care?

该方案灵活简单，雪碧图交给compass生成，增加修改删除测量都是全自动化的，非常的易于使用以及后期维护，同时也兼容了高清图的需求。有一个缺点就是不能100%还原视觉稿，在任何手机下看到的图标大小都是固定的。具体采取以上什么方式，还是要根据当前开发环境进行分析。

## 字体

> 安利一个简单的字体图标生成网站，可以自己上传矢量图生成图标，也可以使用网站提供的图标，贴心的是还会生成该图标使用的css。地址

## 总结

总的来说，移动适配还不存在万能的技术方案，无论是字体、间距、布局、图片都还需要根据自己项目的环境以及复杂度来进行选型。

## 参考资料


1. 使用Flexible实现手淘H5页面的终端适配
2. 使用Sass和Compass制作雪碧图


标签： HTML 移动适配