再聊移动端页面的适配

作者：大漠 日期：2017-08-02 点击：43439 Layout 布局 mobile CSS

> 编辑推荐：3月31日前，点击注册激活 Coding.net 立赠30天付费会员，体验极速代码托管服务！

前端圈真乱，这话一点不假。但乱也乱的好处，乱则生变，有变化才有进步。今天还是老调重谈，聊聊移动端页面的适配。因为对于一枚前端而言，天天和页面打交道（H5页面），那么布局的活总是少不了，这也将面临不同终端的适配问题。不知道你是否和我一样，页面布局总是或多或少会有一些蛋疼的事情发生。如果是的话，建议你花点时间阅读完下面我扯蛋的东东。

Flexible承载的使命

Flexible到今天也有几年的历史了，解救了很多同学针对于H5页面布局的适配问题。而这套方案也相对而言是一个较为成熟的方案。简单的回忆一下，当初为了能让页面更好的适配各种不同的终端，通过Hack手段来根据设备的`dpr`值相应改变`<meta>`标签中`viewport`的值：

```
<!-- dpr = 1-->
<meta name="viewport" content="initial-scale=scale,maximum-sc
<!-- dpr = 2-->
<meta name="viewport" content="initial-scale=0.5,maximum-scal
<!-- dpr = 3-->
<meta name="viewport" content="initial-scale=0.3333333333,max
```

从而让页面达么缩放的效果，也变相的实现页面的适配功能。而其主要的思想有三点：

- 根据 `dpr` 的值来修改 `viewport` 实现 `1px` 的线
- 根据 `dpr` 的值来修改 `html` 的 `font-size`，从而使用 `rem` 实现等比缩放
- 使用Hack手段用 `rem` 模拟 `vw` 特性

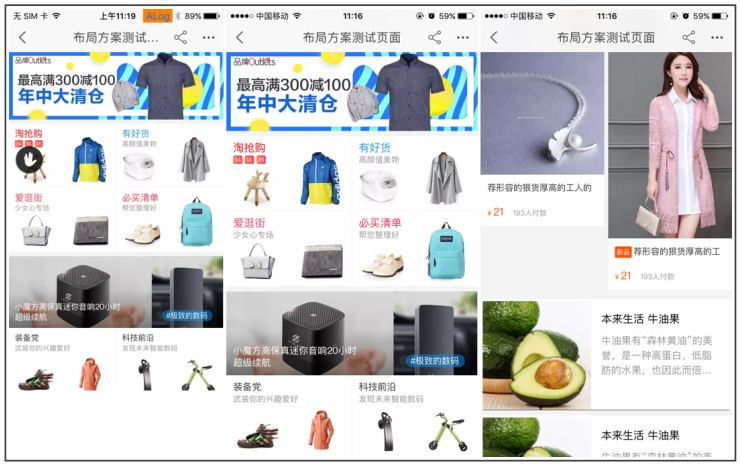
有关于Flexible方案实现适配，在2015年双十一之后做过这方面的技术文档分享，感兴趣的同学可以移步阅读《使用Flexible实现手淘H5页面的终端适配》一文。虽然Flexible解决了适配终端很多问题，但它并不是万能的，也不是最优秀的，他还是存在一些问题的，比如 `iframe` 的引用，有时候就把我们自己给埋进去了。针对其中的一些不足之处，有些同学对其进行过相关的改造，在网上搜索能找到相关的方案。

那么时代在变化，前端技术在不断的变化，试问：**Flexible还是最佳方案？Flexible还有存在的必要吗？** 最近一直在探讨这方面，这里先告诉大家**Flexible已经完成了他自身的历史使命，我们可以放下Flexible，拥抱新的变化。**接下来的内容，我将分享一下我最近自己探讨的新的适配方案，或许很多团队同学已经开始使用了，如果有不对之处，希望能得到大婶们的指正；如果您有更好的方案，希望能一起分享一起探讨。

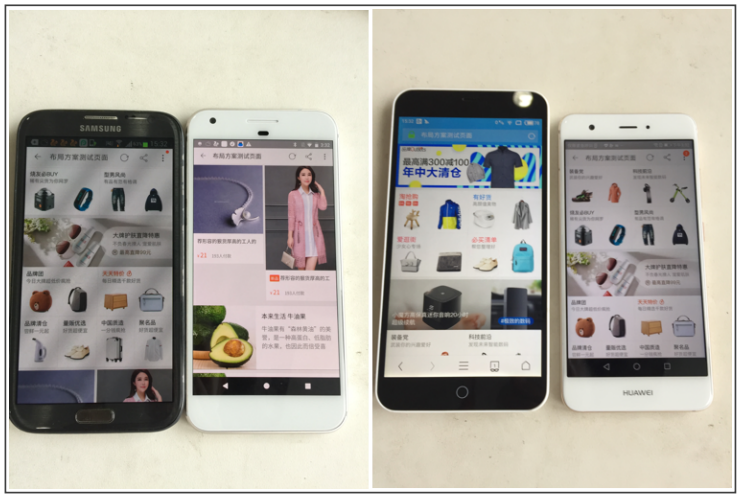
先上菜，再唠嗑

先上个二维码：

你可以使用手淘App、优酷APP、各终端自带的浏览器、UC浏览器、QQ浏览器、Safari浏览器和Chrome浏览器扫描上面的二维码，您看到相应的效果：



iPhone系列效果



部分Android效果

> **注：**如果扫上面的二维码没有任何效果，你可以点击这里，打开在线页面，重新生成你的设备能识别的二维码号 。

上面的Demo，测试了Top30的机型。目前未得到支持的：

品牌	型号	系统版本	分辨率	屏幕尺寸	手淘APP	优酷APP	原生浏览器	QQ浏览器	UC浏览器	Chrome浏览器
华为	Mate9	Android7.0	1080 x 1920	5英寸	Yes	Yes	No	Yes	Yes	Yes
华为	Mate7	Android4.2	1080 x 1920	5.2英寸	Yes	Yes	No	Yes	Yes	Yes

品牌	型号	系统版本	分辨率	屏幕尺寸	手淘APP	优酷APP	原生浏览器	QQ浏览器	UC浏览器	Chrome浏览器
魅族	Mx4 (M460 移动4G)	Android4.4.2	1152 x 1920	5.36英寸	Yes	No	No	Yes	Yes	Yes
Oppo	R7007	Android4.3	1280 x 720	5英寸	Yes	No	No	Yes	Yes	No
三星	N9008 (Galaxy Note3)	Android4.4.2	1080 x 1920	5.7英寸	Yes	No	Yes	Yes	Yes	Yes
华硕	ZenFone5(x86)	Android4.3	720 x 280	5英寸	No	No	No	Yes	No	No

Top30机型中不在列表中的，将看到的效果如上图所示。至于敢不敢用，这就得看亲了。必竟第一个吃螃蟹的人是需要一定的勇气！(^_^)

适配方案

前面给大家介绍了这个方案目前得到的支持情况以及效果。也扯了不少废话，接下来进入正题吧。

在移动端布局，我们需要面对两个最为重要的问题：

- 各终端下的适配问题
- Retina屏的细节处理

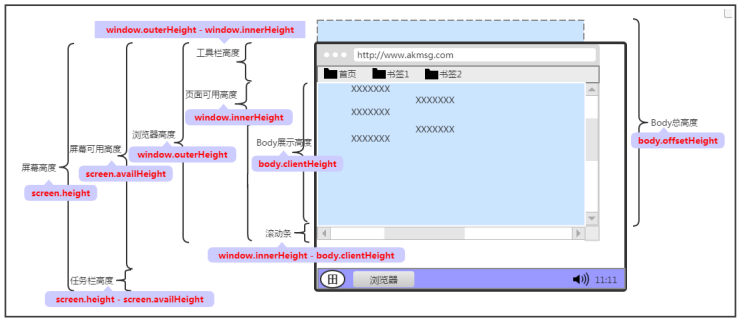
不同的终端，我们面对的屏幕分辨率、DPR、`1px`、`2x`图等一系列的问题。那么这个布局方案也是针对性的解决这些问题，只不过解决这些问题不再是使用Hack手段来处理，而是直接使用原生的CSS技术来处理的。

适配终端

首要解决的是适配终端。回想一下，以前的Flexible方案是通过JavaScript来模拟`vw`的特性，那么到今天为止，`vw`已经得到了众多浏览器的支持，也就是说，可以直接考虑将`vw`单位运用于我们的适配布局中。

众所周知，`vw`是基于Viewport视窗的长度单位，这里的视窗（Viewport）指的就是浏览器可视化的区域，而这个可视区

域是`window.innerWidth/window.innerHeight`的大小。用下图简单的来示意一下：



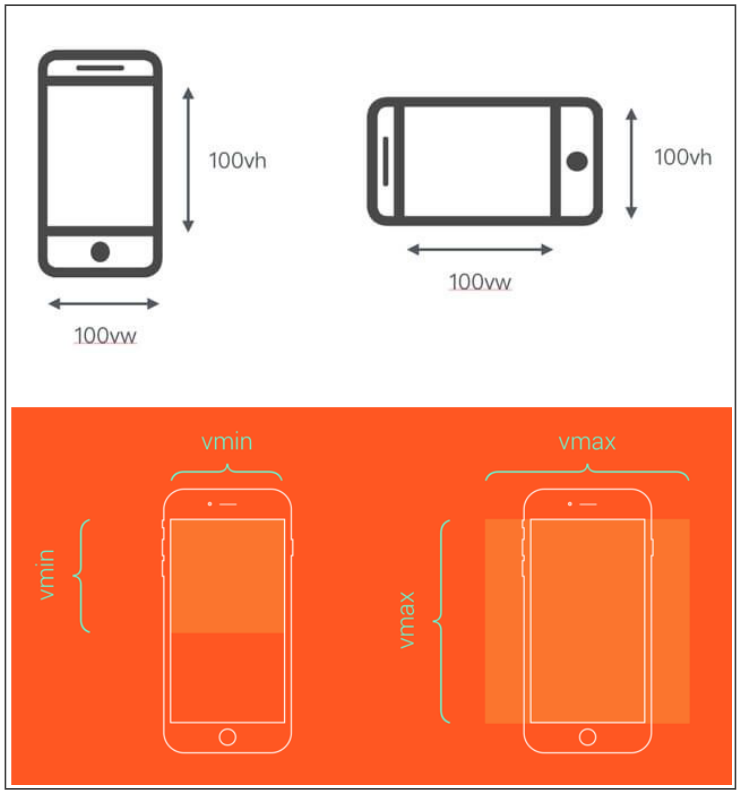
> 因为Viewport涉及到的知识点很多，要介绍清楚这方面的知识，都需要几篇文章来进行阐述。@PPK大神有两篇文章详细介绍了这方面的知识。中文可以移步这里进行阅读。

在CSS Values and Units Module Level 3中和Viewport相关的单位有四个，分别为`vw`、`vh`、`vmin`和`vmax`。

- `vw`：是Viewport's width的简写，`1vw`等于`window.innerWidth`的`1%`
- `vh`：和`vw`类似，是Viewport's height的简写，`1vh`等于`window.innerHeihgt`的`1%`
- `vmin`：`vmin`的值是当前`vw`和`vh`中较小的值
- `vmax`：`vmax`的值是当前`vw`和`vh`中较大的值

> `vmin`和`vmax`是根据Viewport中长度偏大的那个维度值计算出来的，如果`window.innerHeight > window.innerWidth`则`vmin`取百分之一的`window.innerWidth`，`vmax`取百分之一的`window.innerHeight`计算。

还是用一张图来示意吧，一图胜于千言万语：



所以在这个方案中大胆的使用 `vw` 来替代以前Flexible中的 `rem` 缩放方案。先来回归到我们的实际业务中来。目前出视觉设计稿，我们都是使用 `750px` 宽度的，从上面的原理来看，那么 `100vw = 750px` ，即 `1vw = 7.5px` 。那么我们可以根据设计图上的 `px` 值直接转换成对应的 `vw` 值。看到这里，很多同学开始感到崩溃，又要计算，能不能简便一点，能不能再简单一点，其实是可以的，我们可以使用PostCSS的插件postcss-px-to-viewport，让我们可以直接在代码中写 `px` ，比如：

```
[w-369]{
    width: 369px;
}

[w-369] h2 span {
    background: #FF5000;
    color: #fff;
    display: inline-block;
    border-radius: 4px;
    font-size: 20px;
    text-shadow: 0 2px 2px #FF5000;
    padding: 2px 5px;
    margin-right: 5px;
}
```

PostCSS编译之后就是我们所需要的带 `vw` 代码：

```
[w-369] {
    width: 49.2vw;
}
```

```
[w-369] h2 span {
    background: #ff5000;
    color: #fff;
    display: inline-block;
    border-radius: .53333vw;
    text-shadow: 0 0.26667vw 0.26667vw #ff5000;
    padding: .26667vw .66667vw;
}
[w-369] h2 span,
[w-369] i {
    font-size: 2.66667vw;
    margin-right: .66667vw;
}
```

在实际使用的时候，你可以对该插件进行相关的参数配置：

```
"postcss-px-to-viewport": {
    viewportWidth: 750,
    viewportHeight: 1334,
    unitPrecision: 5,
    viewportUnit: 'vw',
    selectorBlackList: [],
    minPixelValue: 1,
    mediaQuery: false
}
```

假设你的设计稿不是 `750px` 而是 `1125px` ，那么你就可以修改 `vewportWidth` 的值。有关于该插件的详细介绍，可以阅读其官方使用文档。

上面解决了 `px` 到 `vw` 的转换计算。那么在哪些地方可以使用 `vw` 来适配我们的页面。根据相关的测试：

- 容器适配，可以使用 `vw`
- 文本的适配，可以使用 `vw`
- 大于 `1px` 的边框、圆角、阴影都可以使用 `vw`
- 内距和外距，可以使用 `vw`

另外有一个细节需要特别的提出，比如我们有一个这样的设计：

如果我们直接使用：

```
[w-188-246] {
    width: 188px;
}
[w-187-246]{
    width: 187px
}
```

最终的效果会造成`[w-187-246]`容器的高度小于`[w-188-246]`容器的高度。这个时候我们就需要考虑到容器的长宽比缩放。这方面的方案很多，但我还是推荐工具化来处理，这里推荐@一丝 姐姐写的一个PostCSS插件postcss-aspect-ratio-mini。这个插件使用很简单，不需要做任何的配置，你只需要本地安装一下就OK。使用的时候如下：

```
[aspectratio] {
    position: relative;
}
[aspectratio]::before {
    content: '';
    display: block;
    width: 1px;
    margin-left: -1px;
    height: 0;
}

[aspectratio-content] {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    width: 100%;
    height: 100%;
}
[aspectratio][aspect-ratio="188/246"]{
    aspect-ratio: '188:246';
}
```

编译出来：

```
[aspectratio][aspect-ratio="188/246"]:before {
    padding-top: 130.85106382978725%;
}
```

这样就可以完美的实现长宽比的效果。有关于这方面的原理在这里不做过多阐述，感兴趣的话可以阅读早前整理的文章:

- CSS实现长宽比的几种方案
- 容器长宽比
- Web中如何实现纵横比
- 实现精准的流体排版原理

> 目前采用PostCSS插件只是一个过渡阶段，在将来我们可以直接在CSS中使用`aspect-ratio`属性来实现长

解决1px方案

前面提到过，对于`1px`是不建议将其转换成对应的`vw`单位的，但在Retina下，我们始终是需要面对如何解决`1px`的问题。在《再谈Retina下`1px`的解决方案》文章中提供了多种解决`1px`的方案。在这里的话，个人推荐另外一种解决`1px`的方案。依旧是使用PostCSS插件，解决`1px`可以使用postcss-write-svg。

使用postcss-write-svg你可以通过`border-image`或者`background-image`两种方式来处理。比如:

```
@svg 1px-border {
    height: 2px;
    @rect {
        fill: var(--color, black);
        width: 100%;
        height: 50%;
    }
}
.example {
    border: 1px solid transparent;
    border-image: svg(1px-border param(--color #00b1ff)) 2 2
}
```

这样PostCSS会自动帮你把CSS编译出来:

```
.example {
    border: 1px solid transparent;
```

```
    border-image: url("data:image/svg+xml;charset=utf-8,%3Csv
}
```

使用PostCSS的插件是不是比我们修改图片要来得简单与方便。

上面演示的是使用 `border-image` 方式，除此之外还可以使用 `background-image` 来实现。比如：

```
@svg square {
    @rect {
        fill: var(--color, black);
        width: 100%;
        height: 100%;
    }
}



.demo {
    background: white svg(square param(--color #00b1ff));
}
```

编译出来就是：

```
xml;charset=utf-8,%3Csvg xmlns='http://www.w3.org/2000/svg'%3E
```

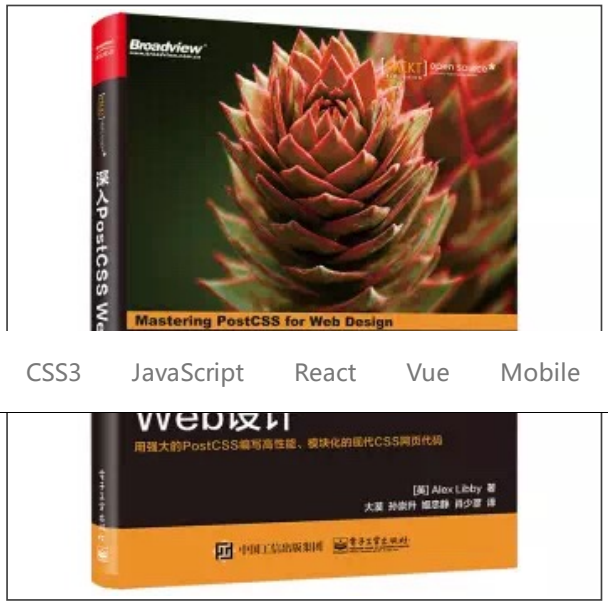
这个方案简单易用，是我所需要的。目前测试下来，基本能达到我所需要的需求。但有一点千万别忘了，记得在 `<head>` 中添加：

```
<meta name="viewport" content="width=device-width,initial-sca
```

上面阐述的是这个适配方案中所用到的技术点，简单的总结一下：

- 使用 `vw` 来实现页面的适配，并且通过PostCSS的插件postcss-px-to-viewport把 `px` 转换成 `vw` 。这样的好处是，我们在撸码的时候，不需要进行任何的计算，你只需要根据设计图写 `px` 单位
- 为了更好的实现长宽比，特别是针对于 `img` 、 `vedio` 和 `iframe` 元素，通过PostCSS插件postcss-aspect-ratio-mini来实现，在实际使用中，只需要把对应的宽和高写进去即可
- 为了解决 `1px` 的问题，使用PostCSS插件postcss-write-svg,自动生成 `border-image` 或者 `background-image` 的图片

这里使用了多个PostCSS的插件，其实现在有很多优秀的PostCSS插件能帮助我们解决很多问题。哪果你从未接触过有关于PostCSS相关的知识，建议你可以花点时间去学习一下，在W3cplus提供了一些有关于PostCSS相关的文章。如果你想系统的学习PostCSS相关的知识，推荐你购买《深入PostCSS Web设计》一书：



降级处理

最开始提到过，到目前为止，T30的机型中还有几款机型是不支持 `vw` 的适配方案。那么如果业务需要，应该怎么处理呢？有两种方式可以进行降级处理：

- **CSS Houdini**：通过CSS Houdini针对 `vw` 做处理，调用CSS Typed OM Level1 提供的 `CSSUnitValue` API。
- **CSS Polyfill**：通过相应的Polyfill做相应的处理，目前针对于 `vw` 单位的Polyfill主要有：vminpoly、Viewport Units Buggyfill、vunits.js和Modernizr。个人推荐采用Viewport Units Buggyfill

Viewport不足之处

采用 `vw` 来做适配处理并不是只有好处没有任何缺点。有一些细节之处还是存在一定的缺陷的。比如当容器使用 `vw` 单位， `margin` 采用 `px` 单位时，很容易造成整体宽度超过 `100vw` ，从而影响布局效果。对于类似这样的现象，我们可以采用相关的技术进行规避。比如将 `margin` 换成 `padding` ，并且配合 `box-sizing` 。只不过这不是最佳方案，随着将来浏览器或者应用自身的Webview对 `calc()` 函数的支持之后，碰到 `vw` 和 `px` 混合使用的时

候，可以结合 `calc()` 函数一起使用，这样就可以完美的解决。

另外一点， `px` 转换成 `vw` 单位，多少还会存在一定的像素差，毕竟很多时候无法完全整除。

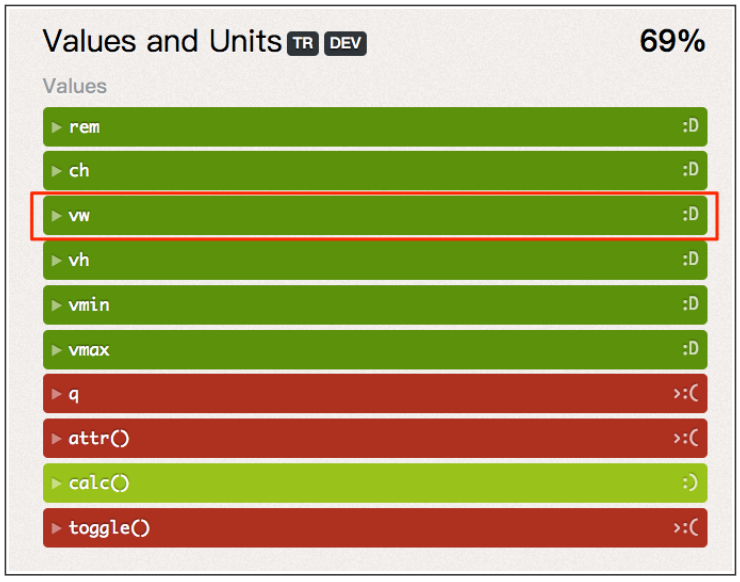
到目前为止，我发现的两个不足之处。或许在后面的使用当中，还会碰到一些其他不为人之的坑。事实也是如此，不管任何方案，踩得坑越多，该方案也越来越强大。希望喜欢这个适配方案的同学和我一起踩坑，让其更为完善。

如何判断自己的应用是否支持

虽然该文的示例，进行了多方面的测试。但很多同学还是会担

描下面的二维码：

当页面跑完测试之后，找到对应的**Values and Units**列表项：



如果 vw 栏是绿色代表你的设备或应用支持该方案；反之则不支持。另外你也可以经常关注css3test相关的更新，后面将会根据相关的规范更新测试代码，让你能快速掌握哪些属性可以大胆使用。

总结

H5页面的适配方案总是令人蛋疼的，事实上页面的布局总是令人蛋疼的。但技术是不断革新的，我们可以随着保持对新技术的关注，尝试这些新特性运用到实际项目中，只有这样，我们解决问题的方案才会越来越完善。

到写这篇文章为止，虽然还有那么一两款机型不支持 vw ，但并不影响我们去使用。只有不断去尝试，才会有进步。在此，

或者发邮件给我一起讨论。

大漠

常用昵称“大漠”，W3CPlus创始人，目前就职于手淘。对HTML5、CSS3和Sass等前端脚本语言有非常深入的认识和丰富的实践经验，尤其专注对CSS3的研究，是国内最早研究和使用CSS3技术的一批人。CSS3、Sass和Drupal中国布道者。2014年出版《图解CSS3：核心技术与案例实战》。

上一篇: 深入PostCSS Web设计 | 下一篇: 为什么是display:contents而不是CSS Grid的subgrid

W3cplus是一个致力于推广国内前端行业的技术博客。它以探索为己任，不断活跃在行业技术最前沿，努力提供高质量前端技术博文；其文章范围广泛，主要以CSS3、HTML5、JavaScript、Vue、React、Mobile、动画等教程、译文和案例为主。

W3cplus提供相关广告展示与招聘发布

常用昵称“大漠”，W3CPlus创始人，目前就职于淘宝。对HTML5、CSS3和CSS处理器等前端脚本语言有非常深入的认识和丰富的实践经验，尤其专注对CSS3和动画的研究，是国内最早研究和使用CSS3和CSS处理器技术的一批人。现在主要在探讨学习JavaScript、React和Vue相关技术知识。CSS3、CSS处理器和Drupal中国布

本书是国内著名的Web前端专家历时2载的心血之作，根据最新的CSS3撰写，融入了作者在CSS领域多年的使用经验，旨在将本书打造成为CSS3领域最权威和实用的专业著作，供没有经验的读者系统学
供有经验的读者参考备查。

本书理论知识系统全面，详细讲解了选择器、伸缩布局盒模型、渐变、过渡、动画

mail.alremiao@gmail.com