移动端适配

移动端适配

移动设备分辨率

前言

主流的移动端屏幕以及一些概念

viewport视口

手机的各种屏幕参数

移动端适配

rem适配方案

vw单位适配

移动设备分辨率

前言

一直以来，我都觉得前端要做移动端的适配都是比较麻烦的事情，现在花一点时间总结一下移动端适配的各种方案，这总结有可能不能很全面，希望以后陆续可以补充完整。

主流的移动端屏幕以及一些概念

viewport视口

我们做前端移动适配的时候，三个视口的概念是必须要有所了解的，这是最基本的概念。简单来说，移动设备上的viewport就是设备的屏幕上能用来显示我们的网页的那一块区域，但viewport又不局限于浏览器可视区域的大小，它可能比浏览器的可视区域要大，也可能比浏览器的可视区域要小。在默认情况下，一般来讲，移动设备上的viewport都是要大于浏览器可视区域的，这是因为考虑到移动设备的分辨率相对于桌面电脑来说都比较小，所以为了能在移动设备上正常显示那些传统的为桌面浏览器设计的网站，移动设备上的浏览器都会把自己默认的viewport设为980px或1024px（也可能是其它值，这个是由设备自己决定的），但带来的后果就是浏览器会出现横向滚动条，因为浏览器可视区域的宽度是比这个默认的viewport的宽度要小的。

名称 说明

visual viewport 显示在屏幕上的网页区域，可以简单的认为是手持设备物理屏幕的可视区域

layout viewport 就是你写样式的时候页面的布局区域，简单地理解为你布局完之后的一整个网页的大小

ideal viewport 最理想的视口，这个随设备不同而改变，最理想的视口就是移动设备的屏幕大小

光说可能不能理解清晰，引用几张图片吧

最后这个理想视口是根据不同的手机有所不同的，可以通过http://viewportsizes.com/查询到各种手机的理想视口。

手机的各种屏幕参数

关于手机的各种屏幕参数，在前端页面布局的时候一般采用的都是i6的设计图也就是使用750px宽的设计图进行布局然后对其他手机的屏幕进行适配。

以下是iPhone手机的一些具体参数：

设备 屏幕尺寸 分辨率（pt） Reader 物理分辨率（px） PPI(DPI)

i4 3.5寸 320\*480 @1x 320\*480 163

i5 4寸 320\*568 @2x 640\*960 326

i6 4.7寸 375\*667 @2x 750\*1334 326

i6 plus 5.5寸 414\*736 @3x 1242\*2208 401

屏幕尺寸：这个就没什么好说的啦，实打实的手机屏幕大小

分辨率(pt)：pt可以简单理解为一个物理长度单位，这个是绝对大小，是印刷行业常用的单位，等于1/72英寸。

Reader:是px与pt的的比值，也就是我们常说的几倍屏。

物理分辨率（px）:物理点，屏幕上能够显示的最小单位。

PPI（DPI）:每英寸下有几个px。

移动端适配

目的：

移动端适配的目的就是使编辑的页面在不同尺寸的移动设备上，页面保持比较合理的尺寸比例，以及基本的样式，让用户在视觉上以及使用体验上不会有太大的偏差。

rem适配方案

rem单位相信很多人都有接触过，rem按我的理解是一个相对单位，是相对于页面默认的font-size大小进行适应的。比如说，当页面默认font-size是16px时，1个rem就相当于16px的大小，我们可以通过动态改变font-size的方式来让页面适应不同的手机屏幕大小。改变font-size的方式大致有两种，一种是通过媒体查询media query，另一种是通过js获取屏幕宽度后根据不同屏幕宽度改变font-size。

js动态改变font-size关键代码：

<!-- css -->

<!-- 在meta里插入viewport，一般移动端适配都需要加这一行 -->

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

1

2

3

window.onload=function () {

document.documentElement.style.fontSize=document.documentElement.clientWidth/3.75+'px'

}//根据设计图来设置除数，比如说是i6就除以7.5那font-size就是50px,1rem=100rpx;

1

2

3

通过上面的设置，你就可以使用rem单位了，在i6规格的设计图下，100px=1rem的方式进行换算，进行页面的布局。通过js实现动态改变font-size来实现移动端适配的方法是一种简单实用的方法，实现效果也是蛮出色的。

另外一种通过media query的就暴力得多了，在css里面直接加入下面代码就可以实现简单的适配。

@media screen and (min-width: 320px) {html{font-size:50px;}}

@media screen and (min-width: 360px) {html{font-size:56.25px;}}

@media screen and (min-width: 375px) {html{font-size:58.59375px;}}

@media screen and (min-width: 400px) {html{font-size:62.5px;}}

@media screen and (min-width: 414px) {html{font-size:64.6875px;}}

@media screen and (min-width: 440px) {html{font-size:68.75px;}}

@media screen and (min-width: 480px) {html{font-size:75px;}}

@media screen and (min-width: 520px) {html{font-size:81.25px;}}

@media screen and (min-width: 560px) {html{font-size:87.5px;}}

@media screen and (min-width: 600px) {html{font-size:93.75px;}}

@media screen and (min-width: 640px) {html{font-size:100px;}}

@media screen and (min-width: 680px) {html{font-size:106.25px;}}

@media screen and (min-width: 720px) {html{font-size:112.5px;}}

@media screen and (min-width: 760px) {html{font-size:118.75px;}}

@media screen and (min-width: 800px) {html{font-size:125px;}}

@media screen and (min-width: 960px) {html{font-size:150px;}}

vw单位适配

vw单位是CSS3才新加入使用的单位，是相对于屏幕进行换算的，是用于移动端适配的利器。100VW就是占满屏幕宽度以此来适配各种不同的屏幕大小。但是，这个vw单位在使用起来的时候还是挺麻烦的，因为要对设计图的尺寸进行换算，而且这种换算成vw会导致页面的实际效果与设计图会有1px到2px的偏差。在换算方面我们可以使用postcss下的一个插件postcss-pixel-to-viewport,这个插件可以让我们在编写CSS代码的时候直接按照设计图给的尺寸直接写px，然后用插件将px转成vw即刻。

配置：

//postcss.config.js

const postcss\_px\_to\_viewport = require("postcss-px-to-viewport")

module.exports = {

plugins: [

postcss\_px\_to\_viewport({

viewportWidth: 375,

unitPrecision: 5,

viewportUnit: 'vw',

selectorBlackList: [],

minPixelValue: 1,

mediaQuery: false

})

]