## 手机端浏览器及内核：

手机端系统主要有[苹果](https://baike.baidu.com/item/%E8%8B%B9%E6%9E%9C" \t "https://baike.baidu.com/item/%E6%89%8B%E6%9C%BA%E6%B5%8F%E8%A7%88%E5%99%A8/_blank)的IOS系统以及Android平台等。

近年来，移动互联网发展势头迅猛，手机浏览器战略地位凸显，众多实力雄厚的互联网企业纷纷加大在手机浏览器市场的布局，投入大量的资金和人力，抢占手机浏览器用户入口。

当下热门的浏览器有：

百度浏览器、Firefox浏览器、谷歌浏览器、UC浏览器、360浏览器、QQ浏览器、猎豹浏览器、Opera浏览器、搜狗浏览器、2345浏览器等。

国内用户首选品牌使用占比中，UC手机浏览器、QQ手机浏览器、360手机浏览器为用户手机浏览器平台首选前三位。

对于Android手机而言：使用率最高的就是Webkit内核，国内尚无自助研发的内核，都是基于开源内核Webkit进行二次开发的，并不是完全的自主内核

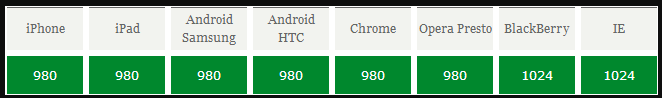
对于iPhone而言：由于系统封闭，不允许除系统自带浏览器内核以外的浏览器内核进入，因此各家浏览器的开发均为在Safari内核的基础上进行二次开发。

## viewport设置

### 1 什么是viewport

viewprot 指的是移动设备浏览器中放置页面的一个虚拟的窗口即用来显示网页的那一部分区域，相当于PC端浏览器可视区。

桌面浏览器中，正常的视口宽度就是整个浏览器的窗口宽度，会随着浏览器窗口的伸缩而缩放，默认情况下HTML标签会占满整个视口。

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。一般移动设备默认都是 viewprot 大于其可视区域，这样不会破坏没有针对移动设备优化的网页的布局，用户可以通过平移和缩放来看网页的其他部分，大部分移动设备默认的 viewport 为 980px,不同设备上的浏览器的默认viewport的宽度并不一样。

图示15-1 viewport的默认值图例

### 2 viewport的作用

在移动端开发中，常将视口抽象划分为布局视口、视觉视口和理想视口。

布局视口，移动设备默认的viewport即布局视口，宽度介于768px ~1024px之间，不进行缩放的话，比浏览器窗口要大。

视觉视口：视觉视口是用户正在看到的网页的区域，即屏幕宽度。

理想视口：布局视口对用户不友好，没有考虑到手机尺寸，所以苹果公司引入理想视口概念，那些基于理想视口而设计的网站，不需要用户手动缩放，也不需要出现横向滚动条，

网站的所有内容都可以正常的呈现给用户。

在PC端，视口指的是在PC端浏览器的可视区域；

而在移动端，它涉及3个视口：Layout Viewport（布局视口），Visual Viewport（视觉视口），Ideal Viewport（理想视口）。

### 3 对viewport的控制

移动端通常需要的是完美视口，所以就需要通过meta viewport对视口大小进行控制，

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的控制。

我们把这个标签设在head里面：

**示例：**

<head>

<meta charset="UTF-8">

<title>Title</title>

<meta name="viewport" content="initial-scale=1.0,

width=device-width, maximum-scale=1.0, user-scalable=no"/>

</head>

上面代码将视口的宽度设置为设备独立像素宽度，默认无缩放，并且也不允许用户缩放。

viewport属性解析：

width: 设置视口的宽度，可以是具体的像素值，也可以特定参数，例如device-width。

<meta name="viewport" content="width=640"> <!--视口宽度为640px-->

<meta name="viewport" content="width=device-width"> <!--视口宽度为设备屏幕的宽度（设备独立像素）-->

initial-scale: 页面初始的缩放值，为数字，可以是小数。

<meta name="viewport" content="width=device-width, initial-scale=1.0">

设置初始缩放为1.0，也就是不缩放。

* 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（通常不用）。
* user-scalable: 是否允许用户进行缩放，'no'为不允许，'yes'为允许。

## 移动端web页面适配方案

随着互联网技术的快速发展，加之智能设备的迅速普及,传统网站都已经逐渐向移动端转移和扩展,移动端web开发技术的掌握也显得尤为重要。

在这里我们说的移动端Web主要运行在手机端的web页面，如WAP页, 因为具备HTML5特性也叫H5页面等。

由于手机屏幕尺寸、分辨率不同，或者需要考虑横竖屏问题，为了使得web页面在不同移动设备上具有尽可能统一或合理的展示效果，需要在开发过程中使用合理的适配方案。

现在我们来看看下面的几种常见的适配方案。

### 1宽度自适应适配（了解）

垂直方向用定值，水平方向用百分比、弹性布局及浮动、定位等布局技巧的综合运用创建可扩展的流式布局，让页面有能力随着屏幕宽度变化，页面也会跟着变化。

这种布局可以一定程度上满足自适应的需求，但不能在每种尺寸下都获得良好的视觉效果比如在大屏幕的手机下显示效果会变成有些页面元素宽度被拉的很长，但是高度还是和原来一样。

缺点是：更适用于对横向拉伸的设计元素不适用于复杂的活动页，对设计稿有一定要求存在局限性。

### 2. rem适配

#### 2.1. rem单位

rem( font size of the root element)是指相对于根元素的字体大小的单位,即根据html元素的font-size来计算大小。

大多数浏览器的默认字号是16px，因此默认1rem=16px。

在布局过程中只要使用 rem 设置元素的尺寸，在改变rem的基准值html的字体大小的时候，页面上的元素将等比例缩放。

#### 2.2. rem布局实现方法

1）设置页面的viewport

2）确立好基准值，常规布局，使用rem单位取代px

3）换算不同设备尺寸下1rem的值，即改变不同尺寸下html 的font-size

换算中比较常用的是css媒体查询区间查询定制和js动态计算控制。

##### rem + 媒体查询

以750为基准设置html的font-size为100px，通过媒体查询分别设置每个屏幕的根字体。

代码示例：

@media only screen and (min-width: 720px) {

html {

font-size: 96px !important;

}

}

@media only screen and (min-width: 750px) {

html {

font-size: 100px !important;

}

}

在此我们以100为例，其它字号大小同理。

此种方法基本能够满足常规适配需求，缺点是由于设备尺寸繁多需要设置密集的断点，其次

由于区配的区间性对不同设备的适配不是非常精准；不在查询设置范围之内，会导致无法适配。

##### js动态计算并设置html的fontsize值

基准值以750px设置1rem=100px为例，首先得到将要适配的设备尺寸，与750换算比例

从而得出不同设备尺寸下1rem的大小。

<script>

//获取html宽度（设备宽度）  
 var deviceWidth = document.documentElement.clientWidth;

// 750可以根据实际设计稿的宽度进行修改  
 if(deviceWidth>750){  
 deviceWidth = 750;  
 }  
 var fs = (deviceWidth\*100)/750;  
 document.documentElement.style.fontSize = fs + 'px';  
</script>

根据上面的js代码，如果页面宽度低于750px,那么页面中html的font-size也会按照（当前页面宽度/750）的比例变化。这样，页面中所有应用了rem的作尺寸单位的元素都会随着页面变化而等比例缩放，从而达到适配。

## 移动端重置样式特殊

1、去除android下a/button/input标签被点击时产生的边框 & 去除ios下a标签被点击时产生的半透明灰色背景。

-webkit-tap-highlight-color: rgba(0,0,0,0);

2、按钮在ios下都是圆角。

-webkit-appearance: none; //button与input都会有个默认背景

border-radius: 0; //input有个默认圆角

3、修改placeholder的样式。

input::-webkit-input-placeholder{

color:#000; //默认的样式

}

input:focus::-webkit-input-placeholder{

color:#f00; //点击后的样式

}