传智播客·前端与移动开发学院
http://web.itcast.cn

移动 Web

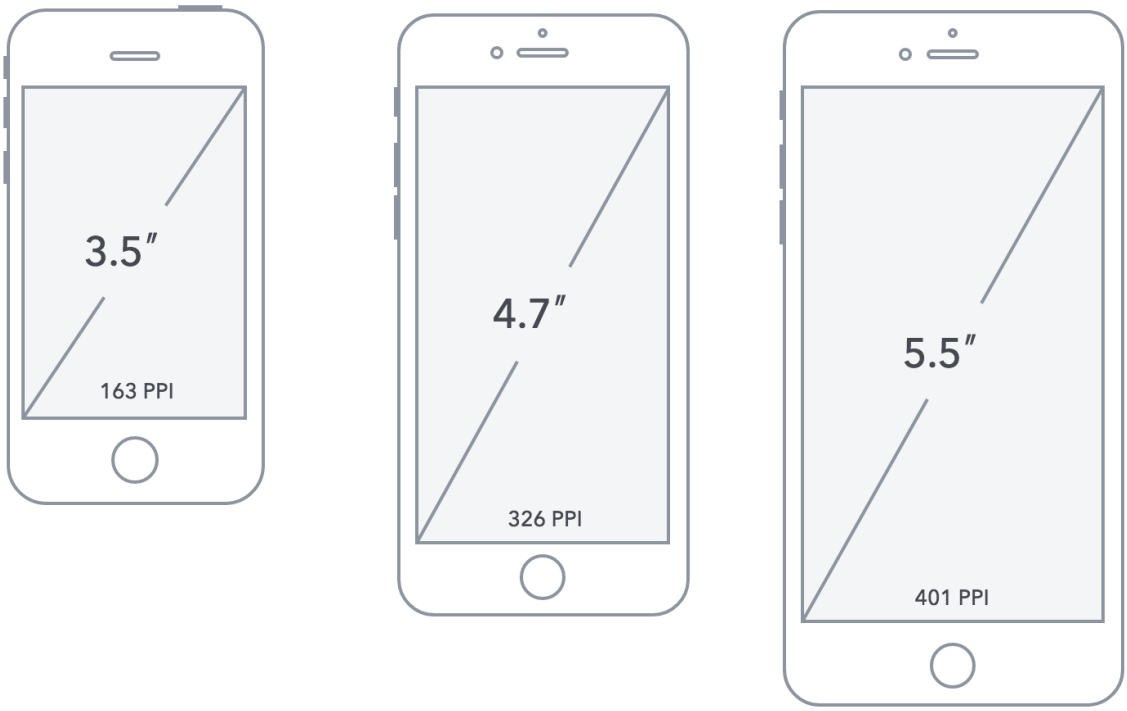
目录

第1章基础知识

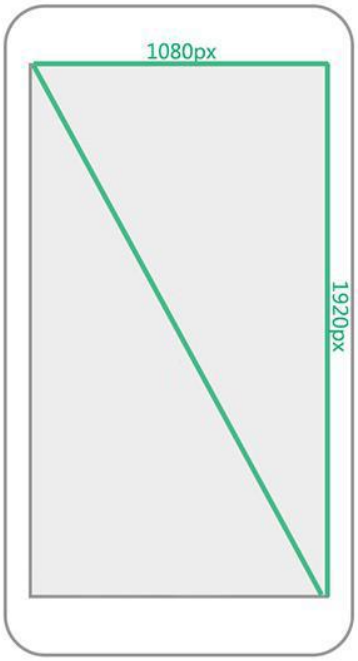
1.1 屏幕

移动设备与 PC 设备最大的差异在于屏幕，这主要体现在屏幕尺寸和屏幕分辨率两个方面。

通常我们所指的屏幕尺寸，实际上指的是屏幕对角线的长度（一般用英寸来度量）如下图所示



而分辨率则一般用像素来度量 px，表示屏幕水平和垂直方向的像素数，例如 1920*1080 指的是屏幕垂直方向和水平方向分别有 1920 和 1080 个像素点而构成，如下图所示



1.2 长度单位

在 Web 开发中可以使用 px（像素）、em、pt（点）、in（英寸）、cm（厘米）做为长度单位，我们最常用 px（像素）做为长度单位。

我们可以将上述的几种长度单位划分成相对长度单位和绝对长度单位。

如上图所示，iPhone3G/S 和 iPhone4/S 的屏幕尺寸都为 3.5 英寸（in）但是屏幕分辨率却分别为 480*320px、960*480px，由此我们可以得出英寸是一个绝对长度单位，而像素是一个相对长度单位（像素并没有固定的长度）。

1.3 像素密度

DPI（Dots Per Inch）是印刷行业中用来表示打印机每英寸可以喷的墨汁点数，计算机显示设备从打印机中借鉴了 DPI 的概念，由于计算机显示设备中的最小单位不是墨汁点而是像素，所以用 PPI（Pixels Per Inch）值来表示屏幕每英寸的像素数量，我们将 PPI、DPI 都称为像素密度，但 PPI 应用更广泛，DPI 在 Android 设备比较常见。

如下图所示，利用 *勾股定理* 我们可以计算得出 PPI

PPI 值的越大说明单位尺寸里所能容纳的像素数量就越多，所能展现画面的品质也就越精细，反之就越粗糙。

Retina 即视网膜屏幕，苹果注册的命名方式，意指具有较高 PPI（大于 320）的屏幕。

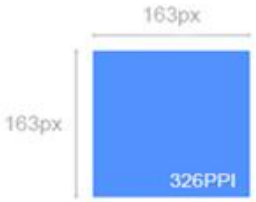
思考：在屏幕尺寸（英寸）固定时，PPI 和像素大小的关系？

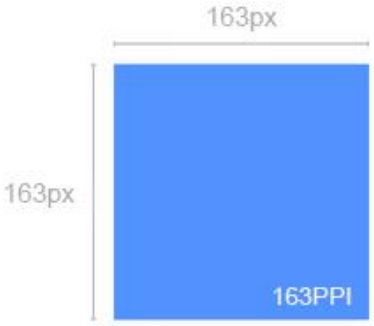
结论：屏幕尺寸固定时，当 PPI 越大，像素的实际大小就会越小，当 PPI 越小，像素实际大小就越大。

1.4 设备独立像素

随着技术发展，设备不断更新，出现了不同 PPI 的屏幕共存的状态（如 iPhone3G/S 为 163PPI，iPhone4/S 为 326PPI），像素不再是统一的度量单位，这会造成同样尺寸的图像在不同 PPI 设备上的显示大小不一样。

如下图，假设你设计了一个 163*163 的蓝色方块，在 PPI 为 163 的屏幕上，那这个方块看起来正好就是 1*1 寸大小，在 PPI 为 326 的屏幕上，这个方块看起来就只有 0.5*0.5 寸大小了。

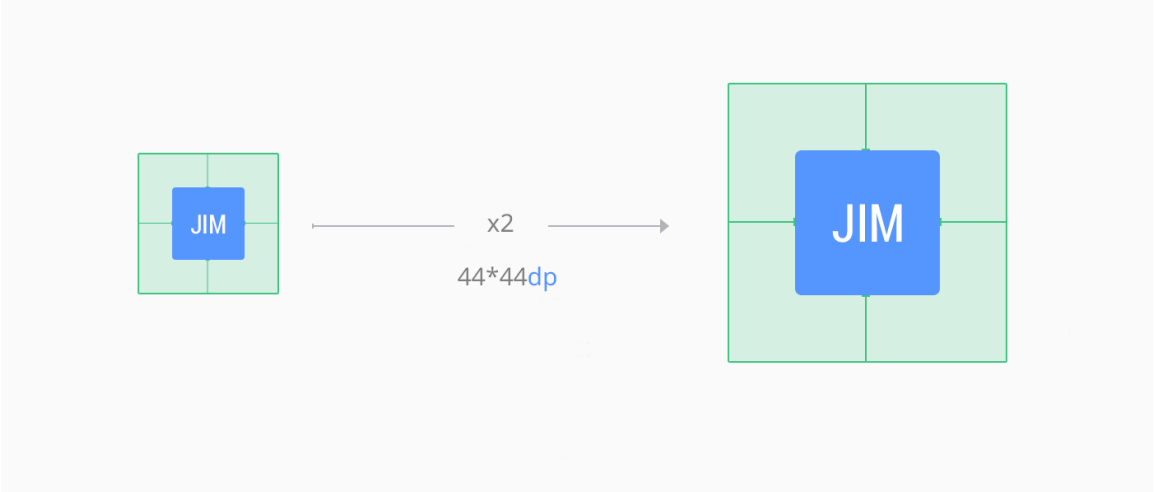




做为用户是不会关心这些细节的，他们只是希望在不同 PPI 的设备上看到的图像内容差不多大小，所以这时我们需要一个新的单位，这个新的单位能够

保证图像内容在不同的 PPI 设备看上去大小应该差不多，这就是独立像素，在 IOS 设备上叫 PT(Point)，Android 设备上叫 DIP(Device independent Pixel)或 DP。

举例说明就是 iPhone 3G（PPI 为 163）1dp = 1px，iPhone 4（PPI 为 326）1dp = 2px。

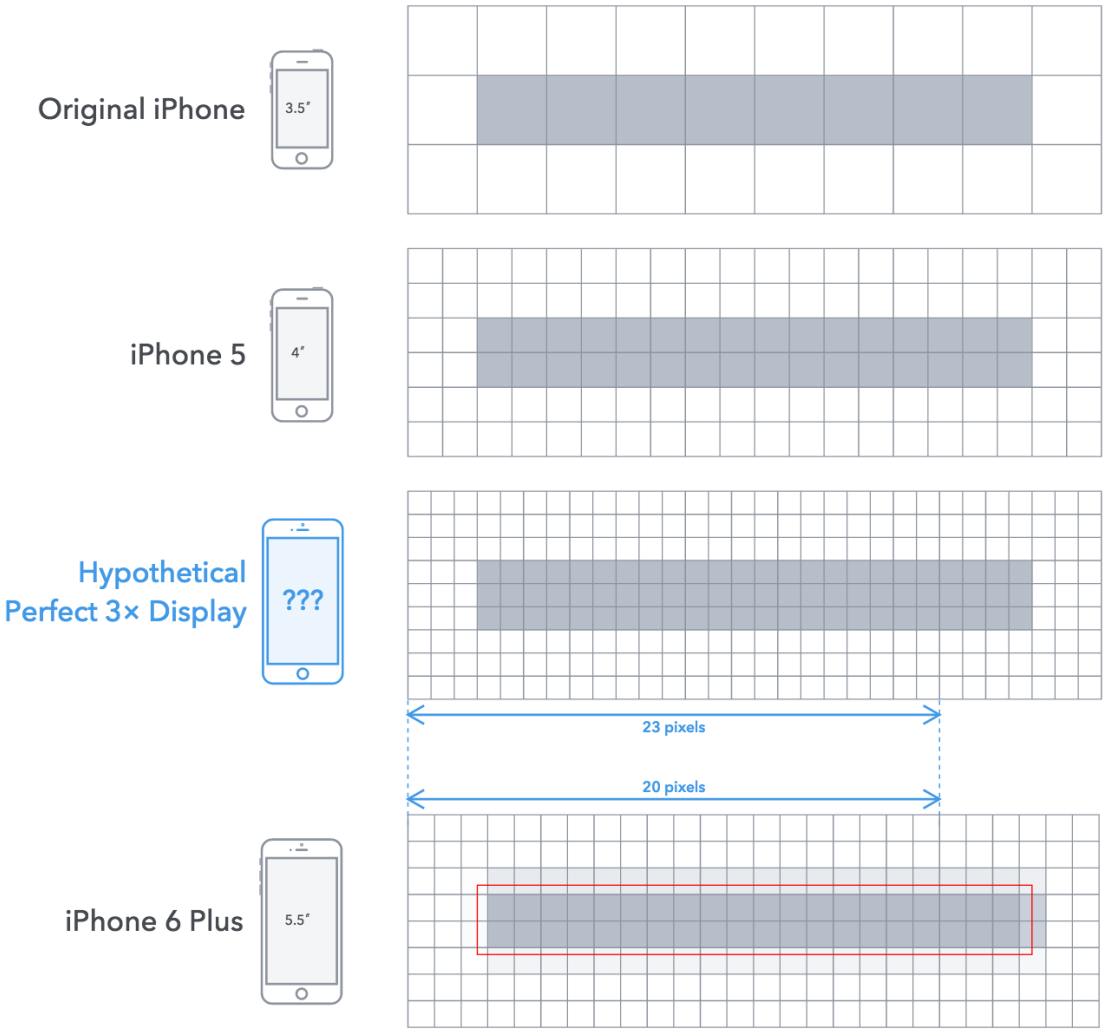


我们也不难发现，如果想要 iPhone 3G/S 和 iPhone 4/S 图像内容显示一致，可以把 iPhone 4/S 的尺寸放大一倍（它们是一个 2 倍(@2x)的关系），即在 iPhone3G/S 的上尺寸为 44*44px，在 iPhone4/S 上为 88*88px，我们要想实现这样的结果可以设置 44*44dp，这时在 iPhone3G/S 上代表 44*44px，在 iPhone4/S 上代表 88*88px，最终用可以看到的图像差不多大小。

通过上面例子我们不难发现 dp 同 px 是有一个对应（比例）关系的，这个对应（比例）关系是操作系统确定并处理，目的是确保不同 PPI 屏幕所能显示的图像大小是一致的，通过 window.devicePixelRatio 可以获得该比例值。

见代码示例 1-1.html

下图展示了 iPhone 不同型号间 dp 和 px 的比例关系



从上图我们得知 dp（或 pt）和 px 并不总是绝对的倍数关系（并不总能保证能够整除），而是 window.devicePixelRatio ~= 物理像素/独立像素，然而这其中的细节我们不必关心，因为操作系统会自动帮我们处理好（保证 1dp 在不同的设备上看上去大小差不多）。

1.5 像素

1、物理像素指的是屏幕渲染图像的最小单位，属于屏幕的物理属性，不可人为进行改变，其值大小决定了屏幕渲染图像的品质，我们以上所讨论的都指的是物理像素。

// 获取屏幕的物理像素尺寸

window.screen.width;

window.screen.height;

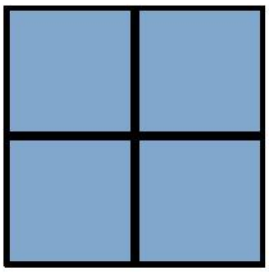
// 部分移动设备下获取会有错误，与移动开发无关，只需要了解

见代码示例 1-2.html

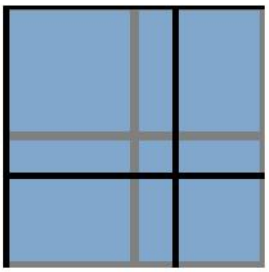
2、CSS 像素，与设备无关像素，指的是通过 CSS 进行网页布局时用到的单位，其默认值(PC 端)是和物理像素保持一致的（1 个单位的 CSS 像素等于 1 个单位的物理像素），但是我们可通缩放来改变其大小。

见代码示例 1-3.html

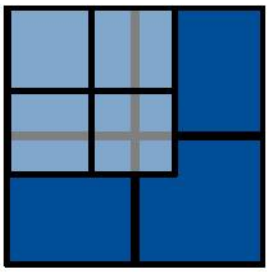
我们通过调整浏览器的缩放比例可以直观的理解 CSS 像素与物理像素之前的对应关系，如下图所示：



1个物理像素等于1个CSS像素



1个物理像素小于1个CSS像素



1个物理像素大于1个CSS像素

我们需要理解的是物理像素和 CSS 像素的一个关系，1 个物理像素并不总是等于一个 CSS 像素，通过调整浏览器缩放比例，可以有以上 3 种情况。

第2章远程调试

2.1 模拟调试

现代主流浏览器均支持移动开发模拟调试，通常按 F12 可以调起，其使用也比较简单，可以帮我们方便快捷定位问题。

2.2 真机调试

模拟调试可以满足大部分的开发调试任务，但是由于移动设备种类繁多，环境也十分复杂，模拟调试容易出现差错，所以真机调试变的非常必要。

有两种方法可以实现真机调试：

1、将做好的网页上传至服务器或者本地搭建服务器，然后移动设备通过网络来访问。（重点）

2、借助第三方的调试工具，如 weinre、debuggap、ghostlab(比较)等

真机调试必须保证移动设备同服务器间的网络是相通的。

第3章视口

视口（viewport）是用来约束网站中最顶级块元素<html>的，即它决定了<html>的大小。

3.1 PC 设备

在 PC 设备上 viewport 的大小取决于浏览器窗口的大小，以 CSS 像素做为度量单位。

通过以往 CSS 的知识，我们都能理解<html>的大小是会影响到我们的网页布局的，而 viewport 又决定了<html>的大小，所以 viewport 间接的决定并影响了我们网页的布局。

// 获取 viewport 的大小

document.documentElement.clientWidth;

document.documentElement.clientHeight;

见代码示例 3-1.html

下面我们通过一个代码实例来演示 PC 设备 viewport（浏览器窗口）是如何影响我们的网页布局的，见代码示例 3-2.html

当我们调整浏览器窗口时，4 个浮动的盒子换行显示了，原因是父盒子宽度不足以容纳 4 个子盒子，要解决这个问题可以给父盒子设置一个固定（较大）的宽度，如 1152px，见代码示例 3-3.html 这样我们可以解决盒子不换行的问题，也可以保证我们的网页内容可以正常的显示，但是出现了滚动条。

注：所谓正常显示是指页面布局没有错乱。

总结：在 PC 端，我们通过调整浏览器窗口可以改变 viewport 的大小，为了保证网页布局不发生错乱，需要给元素设定较大固定宽度。

分析完 PC 端的情况，我们再来看一下移动端会是什么情况呢？

3.2 移动设备

移动设备屏幕普遍都是比较小的，但是大部分的网站又都是为 PC 设备来设计的，要想让移动设备也可以正常显示网页，移动设备不得不做一些处理，通

过上面的例子我们可以知道只要 viewport 足够大，就能保证原本为 PC 设备设计的网页也能在移动设备上正常显示，移动设备厂商也的确是这样来处理的。

在移动设备上 viewport 不再受限于浏览器的窗口，而是允许开发人员自由设置 viewport 的大小，通常浏览器会设置一个默认大小的 viewport，为了能够正常显示那些专为 PC 设计的网页，一般这个值的大小会大于屏幕的尺寸。

如下图为常见默认 viewport 大小（仅供参考）

iPhone	iPad	Android Samsung	Android HTC	Chrome	Opera Presto	BlackBerry	IE
980	980	980	980	980	980	1024	1024

从图中统计我们得知不同的移动厂商分别设置了一个默认的 viewport 的值，这个值保证大部分网页可以正常在移动设备下浏览。

下面我们通过一个小示例来验证上述结论，执行环境为 iPhone5s

在示例中我们设定.box{width: 490px;}了，发现两个盒子正好一行显示，见代码示例 3-4.html

设定了.box{width: 491px;}，则换行显示了，见代码示例 3-5.html

比较可以验证 iPhone5s 下 viewport 的默认宽为 980px，这样便可以保证原本为 PC 设计的网页，在移动端上也不会发生布局的错乱。

注：早期网页一般宽度设计成 960px。

我们再做另一个测试，见代码示例 3-6.html

在 iPhone5s 和部分 Android 中我们发现页面内容（文字、图片）被缩放了（变的非常小），而在部分安卓设备中则出现了滚动条。

产生缩放和滚动条的原因是什么呢？

要解释上面的原因，需要进一步对移动设备的 viewport 进行分析，移动设备上有 2 个 viewport（为了方便讲解人为定义的），分别是 layout viewport 和 ideal viewport。

1、layout viewport（布局视口）指的是我们可以进行网页布局区域的大小，同样是以 CSS 像素做为计量单位，可以通过下面方式获取

```
// 获取 layout viewport
document.documentElement.clientWidth;
document.documentElement.clientHeight;
```

通过前面介绍我们知道，如果要保证为 PC 设计的网页在移动设备上布局不发生错乱，移动设备会默认设置一个较大的 viewport（如 IOS 为 980px），这个 viewport 实际指的是 layout viewport。

见示例代码 3-7.html

2、ideal viewport（理想视口）设备屏幕区域，（以设备独立像素 PT、DP 做为单位）以 CSS 像素做为计量单位，其大小是不可能被改变，通过下面方式可以获取。

```
// 获取 ideal viewport 有两种情形
// 新设备
window.screen.width;
window.screen.height;
// 老设备
window.screen.width / window.devicePixelRatio;
window.screen.height / window.devicePixelRatio;
```

见代码示例 3-8.html

并不总是正确的，然而在实际开发我们一般无需获取这个值具体大小。

理解两个 viewport 后我们来解释为什么网页会被缩放或出现水平滚动条，其原因在于移动设备浏览器会默认设置一个 layout viewport，并且这个值会大于 ideal viewport，那么我们也知道 ideal viewport 就是屏幕区域，layout viewport 是我们布局网页的区域，那么最终 layout viewport 是要显示在 ideal viewport 里的，而 layout viewport 大于 ideal viewport 时，于是就出现滚动条了，那么为什么有的移动设备网页内容被缩放了呢？移动设备厂商认为将网页完整显示给用户才最合理，而不该出现滚动条，所以就将 layout viewport 进行了缩放，使其恰好完整显示在 ideal viewport（屏幕）里，其缩放比例为 ideal viewport / layout viewport。

3.3 移动浏览器

移动端开发主要是针对 IOS 和 Android 两个操作系统平台的，除此之外还有 Windows Phone。

移动端主要可以分成三大类，系统自带浏览器、应用内置浏览器、第三方浏览器

系统浏览器：指跟随移动设备操作系统一起安装的浏览器。

应用内置浏览器：通常在移动设备上都会安装一些 APP 例如 QQ、微信、微博、淘宝等，这些 APP 里往往会内置一个浏览器，我们称这个浏览器为应用内置浏览器（也叫 WebView），这个内置的浏览器一般功能比较简单，并且客户端开发人员可以更改这个浏览器的某些设置，在我们理实的开发里这个浏览器很重要。

第三方浏览器：指安装在手机的浏览器如 FireFox、Chrome、360 等等。

在 IOS 和 Android 操作系统上自带浏览器、应用内置浏览器都是基于 Webkit 内核的。

思考：移动端页面要达到什么效果才最合理？

第4章屏幕适配

经过分析我们得到，移动页面最理想的状态是，避免滚动条且不被默认缩放处理，我们可以通过设置<meta name="viewport" content="">来进行控制，并改变浏览器默认的 layout viewport 的宽度。

4.1 Viewport 详解

viewport 是由苹果公司为了解决移动设备浏览器渲染页面而提出的解决方案，后来被其它移动设备厂商采纳，其使用参数如下：

// 通过设置属性 content=""实现，中间以逗号分隔

// 例如<meta name="viewport" content="width=device-width">

width 设置 layout viewport 宽度，其取值可为数值或者 device-width。

见代码示例 4-1.html

height 设置 layout viewport 高度，其取值可为数值或者 device-height

见代码示例 4-2.html

initital-scale 设置页面的初始缩放值，为一个数字，可以带小数。

见代码示例 4-3.html

maximum-scale 允许用户的最大缩放值，为一个数字，可以带小数。

见代码示例 4-4.html

minimum-scale 允许用户的最小缩放值，为一个数字，可以带小数。

见代码示例 4-5.html

user-scalable 是否允许用户进行缩放，值为"no"或"yes"。

见代码示例 4-6.html

注：device-width 和 device-height 就是 ideal viewport 的宽高。

4.2 控制缩放

1、设置<meta name="viewport" content="initial-scale=1">，这时我们发现网页没有被浏览器设置缩放。见代码示例 4-3.html

2、设置<meta name="viewport" content="width=device-width">，这时我们发现网页也没有被浏览器设设置缩放。见代码示例 4-1.html

当我们设置 width=device-width，也达到了 initial-scale=1 的效果，得知其实 initial-scale = ideal viewport / layout viewport。

两种方式都可以控制缩放，开发中一般同时设置 width=device-width 和 initial-scale=1.0（为了解决一些兼容问题）参见移动前端开发之 viewport 深入理解，即<meta name="viewport" content="width=device-width, initial-scale=1.0">

4.3 避免滚动

我们知道，滚动条是 layout viewport 相对于 ideal viewport 的，所以只要设置 layout viewport 小于或等于 ideal viewport，即<meta name="viewport" content="width=device-width">。

经测试发现我们并没有完全的解决滚动条的问题，原因在于我们示例里的.box {width: 490px;}设置了一个绝对的宽度造成的，要解决这个问题我们可以设置一个百分比（100%）的宽度，见代码示例 4-7.html

4.4 适配方案

移动开发的核心是屏幕适配，然而并示有专门的规范进行约束，一般是对现有持术进行归纳而总结出适配方案，掌握了以上的技术细节后我们可以总结出以下几种适配方案：

4.4.1. 固定宽度

1、设置<meta name="viewport" content="width=device-width, initial-scale=1">

2、设置内容区域大小为 320px

3、设置内容区域水平居中显示

关于手机尺寸（ideal viewport）

手机型号	屏幕尺寸	物理像素	devicePixelRatio	ideal viewport
iPhone 3G	3.5吋	480*320	1	480*320
iPhone4	3.5吋	960*640	2	480*320
iPhone5	4.0吋	1136*640	2	568*320
iPhone6	4.7吋	1334*750	2	667*375
iPhone6 Plus	5.0吋	1920*1080	3	736*414

更多 ideal viewport 参考

通过汇总对比我们知道移动设备的屏幕尺寸虽然庞杂，但有几个主要尺寸，分别为 320px、360px，这三个尺寸占了绝大部分，并且以 320px 最多，所以我

们移动网页如果设计成 320px 宽，则可以保证在绝大多数设备里正常显示，此方案已经很少采用了。

见示例代码 4-8.html

4.4.2. 百分比宽度

1、设置<meta name="viewport" content="width=device-width, initial-scale=1">

2、 设置页面宽度为百分比

我们需要重新认识 CSS 里百分比的使用， 见代码示例 4-9.html

// 测试下列属性设置为百分比

width 参照父元素的宽度

height 参照父元素的高度

padding 参照父元素的宽度

border 不支持百分比设置

margin 参照父元素的宽度

我们发现这种方案最容易理解，但是在设置元素高度时有非常大的局限性。

4.4.3. rem 单位

1、设置<meta name="viewport" content="width=device-width, initial-scale=1">

2. 设置页面元素宽度单位为 rem 或 em

注：此方案比较灵活，我们的案例将采用这种方案

关于 em 和 rem

em 相对长度单位，其参照当前元素字号大小，如果当前元素未设置字号则会继承其祖先元素字号大小例如 .box {font-size: 16px;}则 1em = 16px .box {font-size: 32px;} 则 1em = 32px，0.5em = 16px

见代码示例 4-10.html

rem 相对长度单位，其参照根元素(html)字号大小例如 html {font-size: 16px;} 则 1rem = 16px html {font-size: 32px;} 则 1rem = 32px，0.5rem = 16px

见代码示例 4-11.html

4.4.4. 100%像素

1、设置网页宽度等于设备物理像素

2、设置初始化缩放比例（值为 1 / window.devicePixelRatio）

淘宝针对 iPhone 设备采用的这种方案

第5章媒体查询

设备终端的多样化，直接导致了网页的运行环境变的越来越复杂，为了能够保证我们的网页可以适应多个终端，不得不专门为某些特定的设备设计不同的展示风格，通过媒体查询可以检测当前网页运行在什么终端，可以有机会实现网页适应不同终端的展示风格。

5.1 媒体类型

将不同的终端设备划分成不同的类型，称为媒体类型。

值	描述
all	用于所有设备
print	用于打印机和打印预览
screen	用于电脑屏幕，平板电脑，智能手机等

5.2 媒体特性

每种媒体类型都具体各自不同的特性，根据不同媒体类型的媒体特性设置不同的展示风格。

值	描述
width	定义输出设备中的页面可见区域宽度
height	定义输出设备中的页面可见区域高度
min-width	定义输出设备中的页面最小可见区域宽度
min-height	定义输出设备中的页面最小可见区域高度
max-width	定义输出设备中的页面最大可见区域宽度
max-height	定义输出设备中的页面最大可见区域高度
device-width	定义输出设备的屏幕可见宽度
device-height	定义输出设备的屏幕可见高度
aspect-ratio	定义输出设备中的页面可见区域宽度与高度的比率
device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的比率

5.3 关键字

关键字将媒体类型或多个媒体特性连接到一起做为媒体查询的条件。

1、and 可以将多个媒体特性连接到一起，相当于“且”的意思。

2、not 排除某个媒体类型，相当于“非”的意思，可以省略。

3、only 指定某个特定的媒体类型，可以省略。

5.4 引入方式

1、link 方法，见代码示例 5-1.html

<link href="./5-1.css" media="only screen and (max-width: 320px)">

2、@media 方法（写在 CSS 里），见代码示例 5-2.html

```
/* 屏幕尺寸小于等于320px */
@media only screen and (max-width: 320px) {
    body {
        background-color: pink;
    }
}
```

5.5 常用特性

1、width / height 完全等于 layout viewport，见代码示例 5-3.html

2、max-width / max-height 小于等于 layout viewport，见代码示例 5-4.html

3、min-width / min-height 大于等于 layout viewport，见代码示例 5-5.html

4、device-width / device-height 完全等于 ideal viewport，见代码示例 5-6.html

5、orientation: portrait | landscape 肖像/全景模式，见代码示例 5-7.html

.....还有其它

第6章CSS 预处理器

CSS 预处理器是一种语言，用来为 CSS 增加一些编程的的特性，无需考虑浏览器的兼容性问题，并且你可以在 CSS 中使用变量、简单的程序逻辑、函数等等在编程语言中的一些基本技巧，可以让你的 CSS 更简洁，适应性更强，代码更直观等诸多好处。

常见的 CSS 预处理器有：LESS、SASS、Stylus 等

其使用方法大致相同，我们在这里以 LESS 为例进行学习

6.1 LESS

LESS 是动态的样式表语言，通过简洁明了的语法定义，使编写 CSS 的工作变得非常简单，本质上，LESS 包含一套自定义的语法及一个解析器。

6.1.1. 安装

1、安装 Nodejs 环境 Node Package Manager (验证 node -v npm -v)

2、打开控制台（cmd），执行 npm install -g less (验证 lessc -v)

3、命令行编译 lessc path/xxx.less path/xxx.css

6.1.2. 编译

浏览器只能识别 CSS，LESS 只是用来提升 CSS 可维护性的一个工具，所最终需要将 LESS 编译成 CSS，然而通过命令行编译效率比较低下，一般都会借助于编辑器来完成编译，以 sublime_text 为例，sublime_text 默认并不支持 LESS 的编译操作，需要安装插件实现。

1、执行 npm install -g less-plugin-clean-css（使用 sublime_text 才用）

2、ctrl+shit+p 打开命令面板

3、输入 install package 然后回车

4、安装 LESS、lessc、Less2Css 三个插件

5、alt+s 快捷键即可实现编译

6.1.3. 语法

1、变量

格式：@变量名：值，定义完成后可以重复使用

```
@color: red; // 定义变量

.nav {
    color: @color; // 使用变量
}

.header {
    color: @color; // 使用变量
}
```

见代码示例 6-1.less

2、混合

我们可以像使用函数一样来使用 CSS

```
.box-sizing () { // 定义一个“函数”
    box-sizing: border-box;
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    -ms-box-sizing: border-box;
    -o-box-sizing: border-box;
}

.box {
    .box-sizing; // 调用一个函数
}
```

见代码示例 6-2.less

3、嵌套

嵌套可以非常方便的管理我们的 CSS 层级关系

```less
.header {
    height: 40px; // 定义给header的属性

    nav {
        overflow: hidden; // 定义给nav的属性

        a { // 定义给a的属性
            display: block;
            width: 100px;
            height: 40px;
            float: left;
        }
    }
}
```

见代码示例 6-3.html

6.1.4. 浏览器中使用

了解了 LESS 基本语法后，可以用 LESS 写编写 CSS 代码了，但是需要实时的将 LESS 编译成 CSS 浏览器才能识别，利用编辑器能够编译，但是效率相对较低。

我们可以引入一个 less.js 文件，实现实时的解析，而不必每次修改都要编译，最后完成所有开发任务后，再通过编辑器编译成 css 文件。

1、下载然后引入 less.js

2、引入 xx.less 文件，如：

<link rel="stylesheet/less" type="text/css" href="styles.less" />

注意：rel 属性必须指定成 stylesheet/less，并且 styles.less 要先于 less.js 引入。

必须以服务器方式访问，webstrom 自带服务器功能也可以使用 ghostlab 调试工具的服务器。

见代码示例 6-1.html

第7章触屏事件

7.1.1. 事件类型

touchstart: 手指触摸屏幕时触发

touchmove: 手指在屏幕上移动时触发

touchend: 手指离开屏幕时触发

见示例代码 7-1.html

7.1.2. TouchEvent 对象

touches: 位于屏幕上的所有手指的列表

targetTouches: 位于该元素上的所有手指的列表

changedTouches：touchstart 时包含刚与触摸屏接触的触点，touchend 时包含离开触摸屏的触点

见代码示例 7-2.html

注：没有对比出 touches 同 targetTouches 的差异，推荐使用 targetTouches

7.1.3. Touch 对象

clientX/Y	手指相对于 layout viewport 的水平/垂直像素距离
pageX/Y	手指相对于 layout viewport 的水平/垂直像素距离（含滚动）
screenX/Y	手指相对于 layout viewport 的水平/垂直像素距离（含滚动）

（未设置 viewport 时，screenX/Y 在 Webview 中不正确）

target	手指最初与屏幕接触时的元素

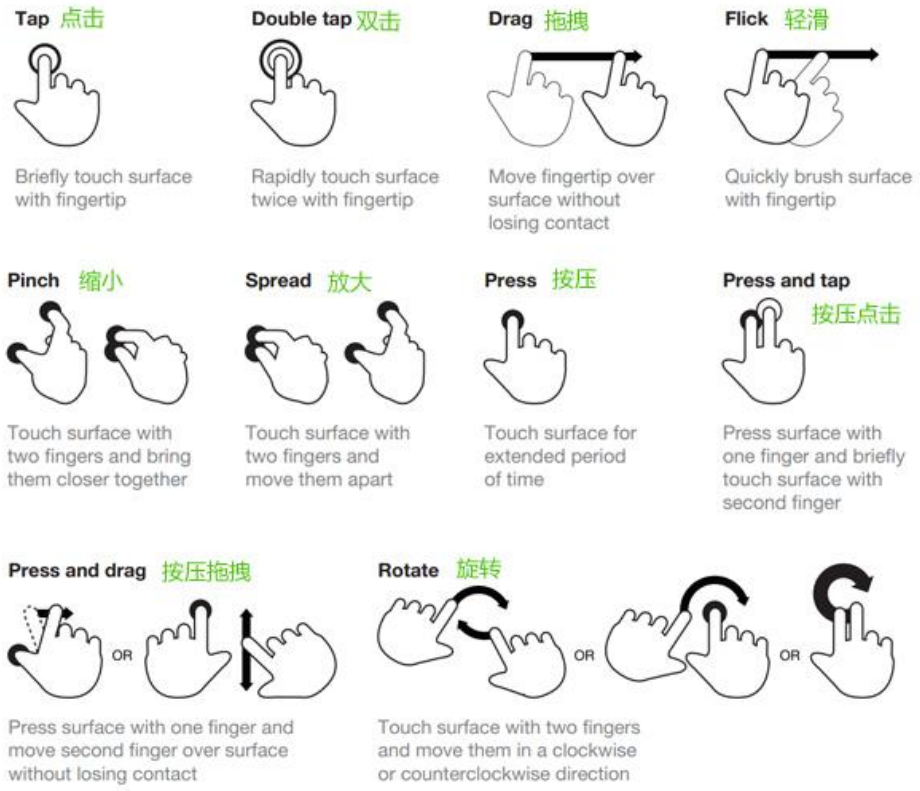
移动开发通常会设置<meta name="viewport" content="width=device-width, initial-scale=1">，这时这三对坐标值是完全一样的。

见代码示例 7-3.html

7.1.4. click 延时

早期移动设备浏览器网页时内容非常小，为了增强用户体验，苹果公司专门为移动设备设计了双击放大的功能，确保用户可以非常方便的放大网页内容，但是当用户单击一个按钮时，移动设备会延时（约 300ms）执行，判断用单是否要双击。用触屏事件可以解决这个问题，见代码示例 7-4.html

7.1.5. 手势封装



利用触屏事件简易封装手势，主要用途是熟悉触屏事件的使用

1、tap 检测接触和离开屏幕的距离来实现，见代码示例 7-5.html

2、drag 跟踪手指移动位置，进而设置元素定位坐标，见代码示例 7-6.html

3、swipe 判断手指滑动的方向，见代码示例 7-7.html

7.1.6. zepto.js

zeptojs 为我们封装了常的触屏事件，需要 touch 模块支持，默认没有构建此模，我们可以自定义构建。

1、安装 Nodejs 环境

2、下载 zepto.js

3、解压缩

4、cmd 命令行进入解压缩后的目录

5、执行 npm install 命令

6、编辑 make 文件，添加自定义模块并保存，如下图

```
42  modules = (env['MODULES'] || 'zepto event ajax form ie').split(' ')
```

7、然后执行命令 npm run-script dist

8、查看目录 dist 即构建好的 zepto.js

见代码示例 7-8.html

第8章移动端类库

8.1.1. iScroll.js

一个可以实现客户端原生滚动效果的类库。

1、下载 iScroll

2、build 目录下提供了不同版本的 iScroll，可根据情况选择使用

3、html 要求有 3 层结构如下图

```
<!-- 符合三层的结果 -->
<!-- 最外层必须要有一个明确的高度，否则无效 -->
<div class="wrapper">
    <!-- 滚动内容的父结点，必须要有 -->
    <ul>
        <!-- 这里是内容区域，这里的总高度要大于wrapper -->
        <li>很多很多的内容，但并不一定必须是一个列表</li>
        <li>很多很多的内容，也可以是正常的网页内容</li>
        <li>很多很多的内容，这里写列表只是方便演示</li>
    </ul>
</div>
```

4、获取 wrapper 这个最外层结点，然后实例化，如下图

```
<!-- 别忘了引入插件 -->
<script src="./js/iscroll.js"></script>
<script>

    // 获取DOM元素
    var wrapper = document.querySelector('.wrapper');

    // 实例化
    var scroller = new IScroll(wrapper, {
        // 这里可以配置一些参数
    });

</script>
```

见示例代码 iscroll.html

8.1.2. swipe.js

1、下载 swipe.js

2、html 结构要求有三层结构，如下图

```
<div class="swipe">
    <ul class="swipe-wrapper">
        <li>需要轮播的无素</li>
        <li>这里可以是一第图</li>
        <li>或者其它什么元素都可以</li>
    </ul>
</div>
```

4、需要一些基础 CSS 样式，这些样式要对应到 html 结构上，如下图

```
<style>
    /* 要注意以下的CSS样式，不要被自己的样式覆盖了，*/
    /* 否则可以出现错误 */

    /* 最外层 */
    .swipe {
        overflow: hidden;
        visibility: hidden;
        position: relative;
    }
    /* 第二层 */
    .swipe-wrap {
        overflow: hidden;
        position: relative;
    }
    /* 第三层 */
    .swipe-wrap > li {
        float:left;
        width:100%;
        position: relative;
    }
</style>
```

5、获取 swipe 元素，然后实例化，如下图

```
<!-- 别忘了引入插件 -->
<script src="./js/swipe.js"></script>
<script>
    // 获取DOM元素
    var ele = document.querySelector('.swipe');

    // 实例化
    var slider = new Swipe(ele, {
        // config
    });
</script>
```

见示例代码 swipe.html

8.1.3. swiper.js

1、下载 swiper.js，

2、其中文网站非常详细介绍了其使用方法

8.1.4. fastclick.js

在移动设备上为了提升 click 的响应速度，我们选择了使用 Zepto 事件封装的 tap 来进行模拟，但是这会带来一个副作用，这个副作用就是“点透”，我们通过一个例子来解释“点透”，见示例代码 fastclick.html（自行查阅点透现象发生的原因）

从上可以看出 Zepto.js 有不完善的地方，并且我们有时也希望我们的移动版页面在 PC 端上也可用，但是 PC 端是不支持 touch 事件的，这时我们面临的问题是即提升 click 在移动设备上的响应速度，又不能使用 Zepto.js 的 tap 事件，这时 fastclick 可以解决这个问题。

如图实际应用的场景，当点击半闭按钮时，如果下面有 click 事件或链接则会被触发。

1、下载 fastclick

2、引入 lib 目录下的 fastclick.js

2、调用方法即可，如下图

```
<!-- 引入库文件 -->
<script src="./js/fastclick.js"></script>
<script>

    // 此事件代表html结构加载完成时触发
    // 类似jQuery里的$(function () { // code });
    document.addEventListener('DOMContentLoaded', function() {
        // 可以在body节点上调用法，代表整个页面的click
        // 响应速度都会得到提升
        FastClick.attach(document.body);

        // 也可以在某一特定的DOM节点上调用
        // 代表只有这个DOM节点的子节点上的click
        // 响应速度得到提升

        // FastClick.attach(document.querySelector('.box'));
    });

    $('.box').on('click', function () {
        // 添加上面代码后我们就正常使用click就可以
        // 在移动设备300ms的延时将得到提升
    });

</script>
```

第9章网页布局

9.1 布局方式

1、固定宽度布局：为网页设置一个固定的宽度，通常以 px 做为长度单位，常见于 PC 端网页。

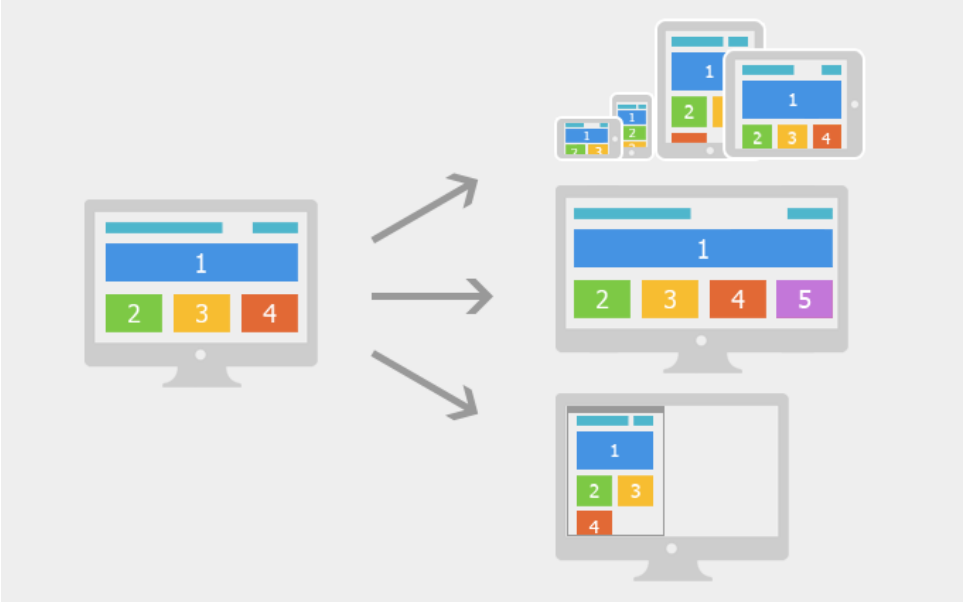
2、流式布局：为网页设置一个相对的宽度，通常以百分比做为长度单位。

3、栅格化布局：将网页宽度人为的划分成均等的长度，然后排版布局时则以这些均等的长度做为度量单位，通常利用百分比做为长度单位来划分成均等的长度。

4、响应式布局：通过检测设备信息，决定网页布局方式，即用户如果采用不同的设备访问同一个网页，有可能会看到不一样的内容，一般情况下是检测设备屏幕的宽度来实现。

9.2 响应式布局

Responsive design，意在实现不同屏幕分辨率的终端上浏览网页的不同展示方式。通过响应式设计能使网站在手机和平板电脑上有更好的浏览阅读体验。



如上图所示，屏幕尺寸不一样展示给用户的网页内容也不一样，我们利用媒体查询可以检测到屏幕的尺寸（主要检测宽度），并设置不同的 CSS 样式，就可以实现响应式的布局。

我们利用响应式布局可以满足不同尺寸的终端设备非常完美的展现网页内容，使得用户体验得到了很大的提升，但是为了实现这一目的我们不得不利用媒体查询写很多冗余的代码，使整体网页的体积变大，应用在移动设备上就会带来严重的性能问题。

响应式布局常用于企业的官网、博客、新闻资讯类型网站，这些网站以浏览内容为主，没有复杂的交互。

一般我们会对常见的设备尺寸进行划分后，再分别确定为不同的尺寸的设备设计专门的布局方式，如下图所示

类型	布局宽度
大屏幕	大于等于1200px
默认	大于等于980px
平板	大于等于768px
手机到平板	小于等于767px
手机	小于等于480px

以上是我们对常见的尺寸进行分类后的结果，下图是与之对应的媒体查询条件。

```
/* 大屏幕 */
@media (min-width: 1200px) { ... }

/* 平板电脑和小屏电脑之间的分辨率 */
@media (min-width: 768px) and (max-width: 979px) { ... }

/* 横向放置的手机和竖向放置的平板之间的分辨率 */
@media (max-width: 767px) { ... }

/* 横向放置的手机及分辨率更小的设备 */
@media (max-width: 480px) { ... }
```

以微金所为例我们来实现一个响应式布局。

第10章CSS 框架

随着 Web 应用变的越来越复杂，在大量的开发过程中我们发现有许多功能模块非常相似，比如轮播图、分页、选项卡、导航栏等，开发中往往会把这些具有通用性的功能模块进行一系列封装，使之成为一个个组件应用到项目中，可以极大的节约开发成本，将这些通用的组件缩合到一起就形成了前端框架。

10.1 Bootstrap

简洁、直观、强悍的前端开发框架，让 web 开发更迅速、简单。

来自 Twitter，粉丝众多，是目前最受欢迎的前端框架。

开始使用吧!

10.2 Amaze UI

Amaze ~ 妹子 UI，国人开发，后起之秀！

开始使用吧!

10.3 Framework7

Framework7 是一个开源免费的框架可以用来开发混合移动应用（原生和 HTML 混合）或者开发 iOS & Android 风格的 WEB APP。

开始使用吧！