01. 移动端基础

1. 移动端样式重置

- 使用 normalize.css 去官网下载。
- 特殊样式重置：

```
/* css3盒模型 */
box-sizing:border-box;
-webkit-box-sizing:border-box;
/* 点击高亮我们需要清除，设置transparent完全透明 */
-webkit-tap-highlight-color:transparent;
/* 在移动端浏览器默认的外观在ios上加上了这个属性才能给按钮和输入框自定义样式 */
-webkit-appearance:none;
/* 禁用长按页面时的弹出菜单 */
img,a{
  -webkit-touch-callout:none;
}
```

2. 浏览器现状

> 移动端浏览器的现状：

1. UC浏览器，QQ浏览器，欧鹏浏览器，百度手机浏览器，360浏览器，谷歌浏览器，搜狗手机浏览器，猎豹浏览器，以及其它浏览器。

> PC端常见的浏览器：

1. 360浏览器，谷歌浏览器，火狐浏览器，QQ浏览器，百度浏览器，搜狗浏览器，IE浏览器。
2. 国内的UC和QQ，百度等手机浏览器都是根据webkit修改过来的内核，国内尚无自主研发的内核，就像国内的手机操作系统都是基于Android修改开发的一样。
3. 总结：兼容移动端主流浏览器，处理webkit内核浏览器即可（兼容webkit即可，兼容性还是比较可观的）。

3. 手机屏幕的现状

1. 移动端设备屏幕尺寸非常多，碎片化比较严重。
2. Android设备有多种分辨率：480x480，480x854，540x960，720x1280，1080x1920等，还有传说中的2K，4K屏幕。
3. 今年来iPhone的碎片化也加剧了，其设备的分辨率有：640x960，640x1136，750x1334，1242x2208等。
4. 作为开发者无需关注这些分辨率，因为我们常用的尺寸单位是像素。
5. 移动端常见的屏幕尺寸：https://material.io/devices。

4. 移动端调试方法

1. chrome DevTools（谷歌浏览器）的模拟手机调试。
2. 搭建本地web服务器，手机和服务器一个局域网内，通过手机访问服务器。
3. 使用外网服务器，直接IP或域名访问。

5. 移动端开发目前主要包括三类

1. 原生APP开发（iOS,Android，RN，uniapp，Flutter）不属于前端开发的内容之一，属于移动端开发工程师iOS,Android。
2. 小程序开发（原生小程序，uniapp，Taro等）**uniapp要学的一定。**
3. WEB页面（移动端的WEB页面，可以使用浏览器或者webview浏览）

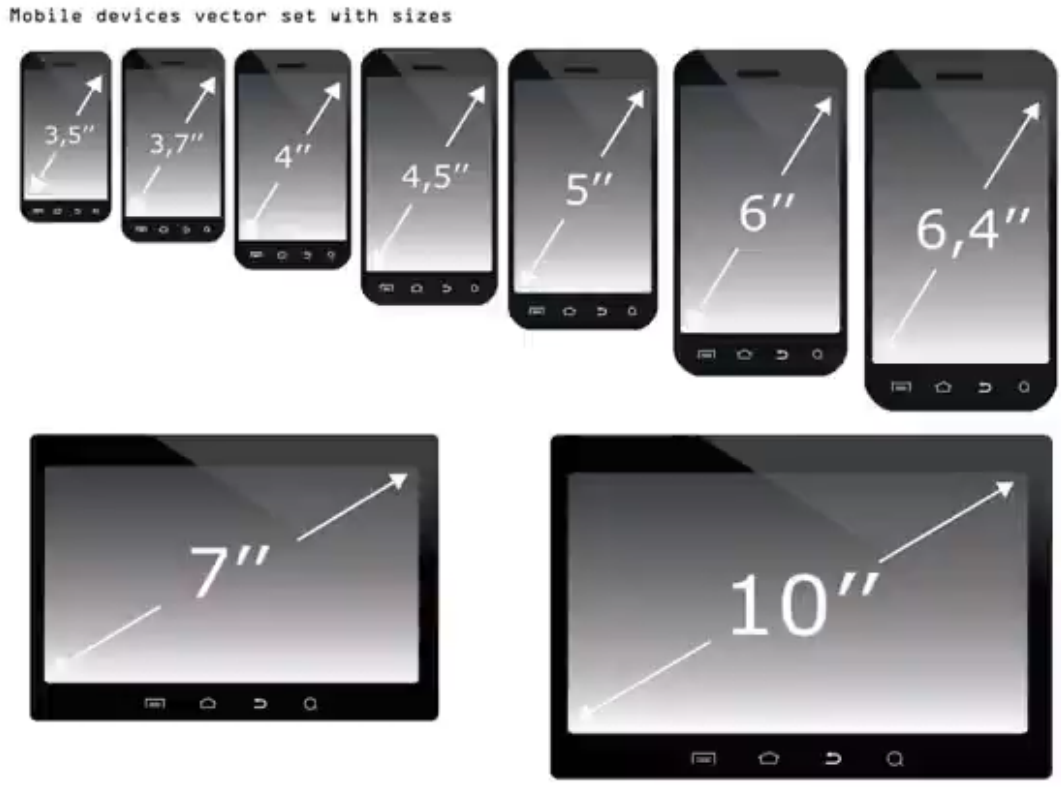
02.【重点】视口

1. 初识视口

1. 视口（viewport）：就是浏览器显示页面内容的屏幕区域**（在一个浏览器中，我们可以看到的区域就是视口）** 注意区分：浏览器工具栏它不是视口啊 。
2. 例如：fixed就是相对于视口进行定位的。
3. **在PC端的页面中，不需要对视口进行区分，因为布局视口和视觉视口是同一个视口。**
4. **但是在移动端，不太一样，你布局的视口和你可见的视口是不太一样的。**

- 这是因为 移动端的网页窗口往往比较小 ，我们可能会希望一个大的网页在移动端可以完整的显示。
- 所以在默认情况下， 移动端的布局视口是大于视觉视口的。

5. 视口可分为：**布局视口，视觉视口，理想视口。**



2. 布局视口

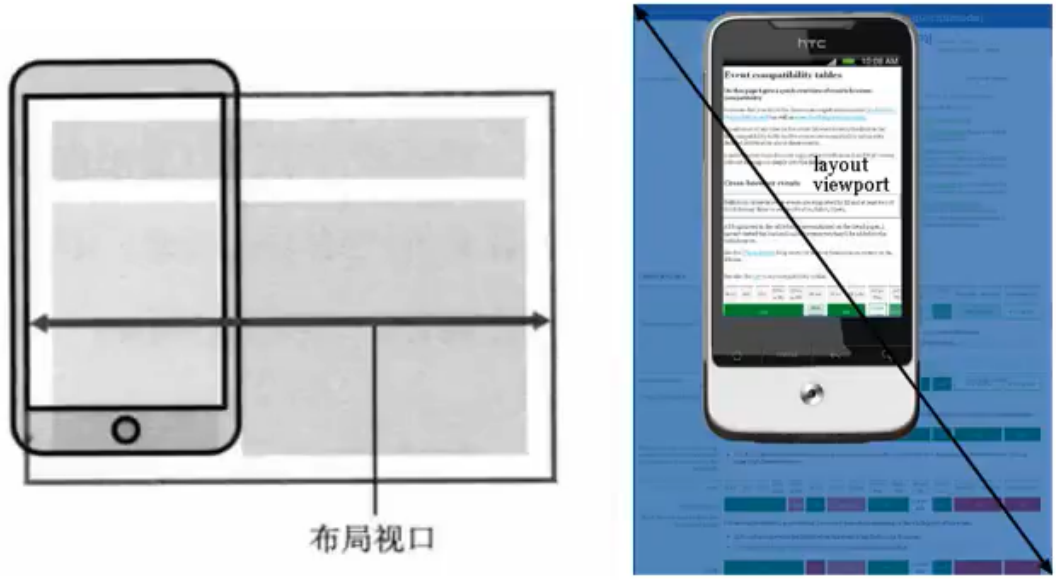
1. **布局视口：layout viewport。**
2. 默认情况下，一个PC端的网页在移动端会如何显示呢？

- 一般移动设备的浏览器都默认设置了一个布局视口，用于解决早期的PC端页面在手机上显示的问题。
- 第一：**它会按照宽度为980px来布局一个页面的盒子和内容。**
- 第二：**为了内容可以完整的显示在页面中，对整个页面进行缩小。**

3. 我们相对于980px布局的这个视口，称之为**布局视口（layout viewport）。**

- **布局视口的默认宽度是 980px。**

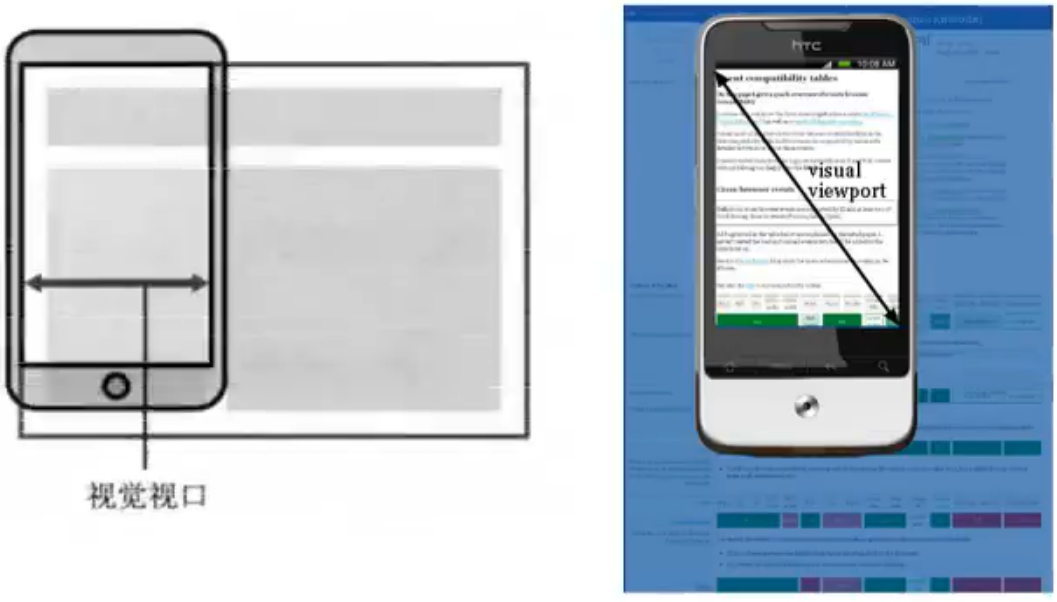
4. iOS，Android基本都是将这个视口分辨率设置为980px，所以PC上的网页大多都能在手机上呈现，只不过元素看上去很小，一般默认可以通过手动缩放网页。



3. 视觉视口

1. **视觉视口：visual viewport。**
2. 如果按默认情况下，布局视口980px显示内容，那么右侧有一部分区域就会无法显示，所以**手机端浏览器会默认对页面进行缩放以显示到用户的可见区域中**。
3. 那么显示在可视区域的这个视口，就是**视觉视口（visual viewport）。**
4. 我们可以通过缩放去操作视觉视口，但不会影响布局视口，布局视口仍保持原来的宽度。
5. 如果所有的网页都按照980px在移动端布局，那么最终页面都会被缩放显示。

- 事实上这种方式是**不利于我们进行移动端开发**的，我们希望的是**设置100px，那么显示的就是100px。**
- 如何做到这一点？通过**设置理想视口（ideal viewport）。**



4. 理想视口

1. **理想视口：ideal viewport。**
2. 默认情况下的布局视口（layout viewport）并不适合我们进行布局。
3. 可以对布局视口（layout viewport）进行宽度和缩放设置，以满足正常在一个移动端窗口的布局

- 使网站在移动端有最理想的浏览和阅读宽度而设定的）。
- 理想视口，对设备来讲，使最理想的视口尺寸。

- 需要手动添加mate视口标签通知浏览器操作。

4. 这个时候就需要先设置meta视口标签，meta中的viewport。

- meta视口标签的主要目的：布局视口的宽度应该与理想视口的宽度一致，简单理解就是设备有多宽，我们布局的视口就有多宽。
- **提示：理想视口概念使乔布斯提出来的（iPhone手机）。** （理想视口就是布局视口等于视觉视口）。

5. meta视口标签

```
<!—
  1. width：设置布局视口的宽度，宽度是980px默认情况下（当我们对width进行更改的时候可以测试，布局视口宽度越大，缩小到移动端的时候元素就越小）。
     布局时候大于视觉视口，显示的内容就会被缩小。
  2. initial-scale视口的默认缩放比例1.0。
  3. user-scalable不允许用户自行缩放，但是这个属性有些浏览器不支持，设置了不让用户自行缩放，但无效（那么就又引用了其它的两个属性一个最大最小缩放比例）。
  4. maximum-scale最大允许的缩放比例1.0。
  5. minimum-scale最小允许的缩放比例1.0。
—>
<meta name="viewport" content="width=device-width,initial-scale=1.0,user-scalable=no,maximum-scale=1.0,minimum-scale=1.0" />
```

标准的viewport设置：

1. width视口宽度和设备保持一致。
2. initial-scale视口的默认缩放比例1.0。
3. user-scalable不允许用户自行缩放。
4. maximum-scale最大允许的缩放比例1.0。
5. minimum-scale最小允许的缩放比例1.0。

属性	解释说明
width	宽度设置的是viewport宽度，可以设置device-width特殊值
initial-scale	初始缩放比，大于0的数字
user-scalable	用户是否可以缩放，yes或者no（1或者0）
minimum-scale	最小缩放比，大于0的数字
maximum-scale	最大缩放比，大于0的数字

03.【重点】二倍图

1. 物理像素与物理像素比

1. 物理像素点指的是屏幕显示的最小颗粒，是物理真实存在的。这是厂商在出厂时就设置好的，比如苹果：6/7/8是750*1334。
2. 开发的时候1像素不一定等于1个物理像素的。
3. PC端页面，1个像素等于1个物理像素的，但移动端就不尽相同。
4. 一个像素能显示的物理像素点的个数，称为物理像素比或屏幕像素比。

案例：iPhone8为例子：750*1334的，页面宽度设置的时候width:375px屏幕占满，那么它的物理像素与物理像素比是1：2的关系（iPhone8的物理像素比是2）。

注意：

1. 物理像素就是我们说的分辨率，iPhone8的物理像素是750。
2. 在 iPhone8里面 1px开发像素 = 2个物理像素。

2. 物理像素&物理像素比

1. PC端和早前的手机屏幕/普通手机屏幕：1CSS像素 = 1 物理像素的。
2. Retina（视网膜屏幕）是一种显示技术，可以将把更多的物理像素点压缩至一块屏幕里，从而达到更高的分辨率，并提高屏幕显示的细腻程度。

- 视网膜屏幕刚开始是由摩托罗拉发明的，后面在苹果发扬光大（现在的手机大多数都是采用视网膜屏幕技术）。

3. 二倍图

1. 对于一张50px*50px的图片，在手机Retina屏幕中打开，按照刚才的物理像素比会放大倍数，这样会造成图片模糊。
2. 在标准的viewport设置中，使用倍图来提高图片质量，解决在高清设备中的模糊问题。
3. 通常使用二倍图，因为iPhone6\7\8的影响，但现在还存在3倍图4倍图的情况，这个看实际开发公司的需求。
4. 背景图片注意缩放问题。

```
  /* 二倍图使用方式 */
  /* 例如：原始图片100*100px */
  /* 手动将图片设置为50px，在移动端物理像素比1:2，最终图片还是100px，这样设置是不会影响图片的清晰度的 */
  img{
    width:50px;
    height:50px;
  }

  /* 背景图片的设置 */
  .box{
    /* 原始图片100*100px */
    background-size:50px 50px;
  }
```

4. 背景缩放background-size

1. background-size 属性规定背景图像的尺寸。
2. 语法：background-size: 背景图片宽度 背景图片高度。

- 单位：长度|百分比|cover|contain。

- cover把背景图像扩展至足够大，以使背景图像完全覆盖背景区域（背景图片不会现实完整，会存在背景图片溢出，盒子会铺满）。
- contain把图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域（高度和宽度是等比例拉伸，当宽度或者高度铺满盒子，就不会再进行拉伸了，可能有部分空白区域）。
- 百分比：是按父盒子的百分比进行缩放的。

5. 精灵图二倍缩放【重点】

1. 在firework里面把精灵图等比例缩放为原来的一半（只是缩小一半测量坐标位置，而不需要保存出来，保存将改变原图）。
   - firework类似于ps的一个软件。
2. 之后根据大小测量坐标。
3. 注意：代码里面background-size也要写：这里的宽高也是写精灵图大小的一半（background-size缩放的是整张图片，而不是精灵图上面的某一个小图标，一定要注意）。

04. 移动端开发选择

---

1. 移动端主流方案

1. **单独制作移动端页面（主流方式）**
   - 京东商城手机版
   - 淘宝触屏版
   - 苏宁易购手机版
   - 通常情况下，网址前面加m（mobile）可以打开移动端。通过判断设备，如果是移动端设备打开，则跳转到移动端页面。
2. **响应式页面兼容移动端（其次方案）**
   - 三星手机官网
   - 通过判断屏幕的宽度来改变样式，以适应不同端。
   - 缺点：制作麻烦，需要花很大精力去调兼容性问题。

2. 移动端样式重置

1. **移动端浏览器**
   - 移动端浏览器基本以webkit内核为主，因此我们就需要考虑webkit兼容性问题。
   - 可以放心使用H5和CSS3样式。
   - 同时我们浏览器的私有前缀我们只需要添加webkit即可。
2. **CSS初始化使用 normalize.css**

> 官网地址：http://necolas.github.io/normalize.css/

   - 移动端CSS初始化推荐使用normalize.css。
   - normalize.css：保护了有价值的默认值。
   - normalize.css：修复了浏览器的BUG。
   - normalize.css：是模块化的。
   - normalize.css：拥有详细的文档。
3. **特殊样式的处理**

> 特殊样式重置包含在normalize.css文件中，如果引用了normalize.css就不需要重置。

```
/* css3盒模型 */
box-sizing:border-box;
-webkit-box-sizing:border-box;
/* 点击高亮我们需要清除，设置transparent完全透明 */
-webkit-tap-highlight-color:transparent;
/* 在移动端浏览器默认的外观在ios上加上了这个属性才能给按钮和输入框自定义样式 */
-webkit-appearance:none;
/* 禁用长按页面时的弹出菜单 */
img,a{
  -webkit-touch-callout:none;
}
```

05. 移动端常见的布局

> 先进性技术选型，移动端技术选型。

1. 单独制作移动端页面（主流）

> 移动端布局：流式布局，rem+动态font-size布局，弹性布局，vw（viewport width）视口布局。

1. **流式布局（百分比布局）**
2. **rem+动态html的font-size** （px to rem 计算转换可以通过less计算，也可以安装插件进行计算）。

- （1）rem+动态html的font-size（动态的font-size使用媒体媒体查询控制），缺点就是使用媒体查询不断的判断视口的宽度。
- （2）rem+动态html的font-size（自己写一个js动态获取视口宽度）

```
// 获取html元素
const htmlEl = document.documentElement;
function setRemUnit(){
  // 获取html元素的宽度（也就是视口的宽度），并且计算html的font-size大小
  const unit = htmlEl.clientWidth / 10;
  // 设置font-size到html元素上
  htmlEl.style.fontSize = unit + 'px';
}
// 保证第一次进来时，可以设置font-size
setRemUnit();
window.addEventListener('resize',function(){
  setRemUnit();
});
window.addEventListener('pageshow',function(){
  if(e.persisted){
    setRemUnit();
  }
});
```

- （3）rem+动态html的font-size（使用lib-flexible库，这个库的原理和上面自己写的是一样的，在github上面可以下载）
- 注意：但是它已经很久没有更新了，从几年前最新更新的README.md中可以看到现在这个使用得还是比较少的。

3. **flex弹性布局（强烈推荐）**

4. **vw单位布局（也是不错的选择）**

2. 响应式页面兼容移动端（其次）

1. 媒体查询
2. Bootstrap

3. 流式布局（称百分比布局）

1. 流式布局，就是百分比布局，也称非固定像素布局。
2. 通过盒子的宽度设置成百分比来根据屏幕的宽度来进行缩进伸缩，不受固定像素的限制，内容像两侧填充。
3. 流式布局方式是移动WEB开发前些年使用的比较常见的布局方式**（现在很少用2022年）**。
4. **但因为不同的属性的百分比值，相对的可能是不同参照物，所以百分比布局往往很难统一。**
5. **所以百分比在移动端适配中使用非常少。**

> 百分比布局为了保证页面布局还需要两个属性。

- max-width 最大宽度（max-height 最大高度）
- min-width 最小宽度（min-height 最小高度）

06. rem-移动端适配布局

---

1. 问题来了

1. 页面布局文字能否随着屏幕大小变化而变化?
2. 流式布局和flex布局主要针对宽度布局，那高度布局如何设置呢?
3. 怎么样让屏幕发生变化的时候元素高度和宽度等比例缩放?
4. 使用rem+动态设置font-size来布局。

2. rem适配方案

1. 让一些不能等比自适应的元素，达到当设备尺寸发生改变的时候，等比例适配当前设备。
2. 使用媒体查询根据不同设备按照比例设置html的字体大小，然后页面元素使用rem做尺寸单位，当html字体大小变化元素尺寸也会发生变化，从而达到等比例缩放的适配。

3.【重要】rem基础

1. em指的是父元素字体大小，子元素相对父元素的字体大小。
2. **rem（root em）**是一个相对单位，类似于em，不同的是rem的基准是相对于html元素的字体大小。
3. 例如：根元素（html）设置font-size:12px，非根元素设置width:2rem，则换成px表示的就是24px。
4. rem的优点就是可以通过修改html里面的文字大小来改变页面中元素的大小可以整体控制。

4. rem适配方案技术方案

方案一：rem + 媒体查询 + less。

设计稿常见尺寸宽度：

设备	常见宽度
iPhone 4.5	640 px
iPhone 678	750 px
Android	常见的320 px, 360 px, 375 px, 384 px, 400 px, 414 px, 500 px, 720 px 大部分4.7~5寸的安卓设备为720 px

注意：一般情况下，我们以一套或者两套效果图适应大部分的屏幕，放弃极端屏幕或对齐优雅降级，牺牲一些效果。现在的基准以750 px为标准。

元素大小取值方法：

1. 最后的公式：页面元素的rem值 = 页面元素值(像素) / (屏幕宽度 / 划分的份数)；
2. 屏幕的宽度 / 划分的份数（就是html根元素font-size的大小，因为rem是根据html根元素来计算的相对单位）。
3. 或者：页面元素rem值 = 页面元素值（px） / html font-size字体大小。

动态设置html标签font-size大小：

1. 假设：设计稿是 750px。
2. 假设：我们将整个屏幕划分为15等分（划分标准不一，可以是20等分也可以是10等分或者 15等分）。
3. 每一份作为html字体大小，这里就是50（750px / 15 = 50）。
4. **用页面元素的大小除以，不同的html字体大小会发现它们比例还是相同的。**
5. 比如：我们以750为标准设计稿。
6. 一个100*100像素的页面元素，在750屏幕下，就是100/50(100除以50)转换成rem是2rem * 2rem 比例是1:1。
7. 比如：320屏幕下，html字体大小为21.33，则2rem = 42.66px此时的宽和高都是42.66但是宽和高的比例还是1:1。
8. 但是已经能实现不同屏幕下，页面元素盒子等比例缩放的效果。

rem + 媒体查询 + less缺点：

1. 通过媒体查询来设置不同尺寸范围内的屏幕html的font-size尺寸。
2. 缺点1：我们需要针对不同的屏幕编写大量的媒体查询。
3. 缺点2：如果动态改变尺寸，内容不会实时的进行更新（因为屏幕尺寸它是由一个区间的，媒体查询是设置的区间，不像js那样是实时获取屏幕尺寸的）。

方案二：rem + flexible.js。

> 简洁高效的rem适配方案flexible.js。

- rem + flexible.js（flexible.js和自己写的是一个原理）

1. 手机淘宝团队出的简洁高效移动适配库。
2. 再也不用写不同屏幕的的媒体查询，因为在js里面做了处理。
3. 它的原理是把当前设备划分为10等份，但是不同设备下，比例还是一致的。
4. 我们要做的，就是确定好我们当前设备的html文字大小就可以了。
5. 案例分析：

- 比如当前设计稿是750 px ，那么我们只需要把html文字大小设置成75px（750px/10）就可以了。
- 里面页面元素rem值：页面元素的px（像素值）/ 75 ，剩余的让flexible.js来计算。

3. **总结：**

- 两种方案现在都是存在的。
- 方案2更简单。

07. 媒体查询-移动端适配布局

1. 媒体查询

1. 什么是媒体查询？媒体查询（Media Query）是CSS3新语法。
2. 使用 @media查询，可以针对不同的媒体类型定义不同的样式。
3. **@media可以针对不同的屏幕尺寸设置不同的样式。**
4. 当你重置浏览器大小的过程中，页面也会根据浏览器的宽度和高度重新渲染。
5. 目前针对苹果手机，Android手机，平板等设备都用得到多媒体查询。

2. 媒体查询语法

```
@media mediatype and|not|only(media feature){
  /* CSS-Code; */
}
```

1. 用@media开头注意@符号。
2. mediatype 媒体类型。
3. 关键字 and not only。
4. media feature 媒体特性必须有小括号包含。

3. 媒体特性

1. 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，暂且了解三个，注意它们要加在小括号里面。

4. 媒体查询类型

> mediatype查询类型，将不同的终端设备划分到不同的类型，称为媒体类型。

值	解释说明
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**
speech	应用于屏幕阅读器等发声设备

5. 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：排除某个媒体的类型，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。

6. 媒体查询案例

```
/* 1. 这句话的意思是：在我们屏幕上并且最大宽度是800像素，设置我们想要的样式。
   2. max-width小于等于800。
   3. 媒体查询可以根据不同的屏幕尺寸在改变不同的样式。
*/
@media screen and(max-width:800px){
  body{
    background-color:pink;
  }
}

@media screen and(max-width:500px){
  body{
    background-color:purple;
  }
}
```

注意：为了防止混乱，媒体查询我们要按照从小到大或者从大到小的顺序来书写，但是最好还是从小到大，这样代码更简洁。

7. 媒体查询+rem实现元素动态大小变化

1. rem单位是跟着html来走的，有了rem页面元素可以设置不同大小尺寸。
2. 媒体查询可以根据不同设备宽度来修改样式。
3. 媒体查询+rem就可以实现不同设备的宽度，实现页面元素大小的动态变化。

8. 媒体查询中的 引入资源（理解）

1. 当样式比较繁多的时候，我们可以针对不同的媒体使用不同stylesheets(样式表)。
2. 原理，就是直接在link中判断设备尺寸，然后引用不同的CSS文件。
3. **语法规范：**

```
<link rel="stylesheet" media="mediatype and | not | only (media feature) "
href="mystylesheet.css">

<!— 媒体查询引入资源方式（引用资源是针对不同的屏幕尺寸，调用不同的CSS样式，按照从小到大的方式进行书写） —>
<link rel="stylesheet" media="screen and(min-width:320px)" href="style320.css">
<link rel="stylesheet" media="screen and(min-width:640px)" href="style640.css">
```

08. flex布局

1. 传统布局和flex弹性布局

1. 传统布局

- 兼容性好
- 布局繁琐
- 局限性，不能在移动端很好的布局

2. flex弹性布局

- 操作方便，布局极为简单，移动端应用很广泛。

- PC端浏览器支持情况比较差（不过现在好了很多2022年，也算是主流布局）。
- IE11或更低版本，不支持或仅支持部分。

2. flex布局原理

1. flex是flexible box的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性，任何一个容器都可以指定为flex布局。
2. 当我们为父盒子设为flex布局以后，子元素的float，clear和vertical-align属性失效。
3. 伸缩布局 = 弹性布局 = 伸缩盒布局 = 弹性盒布局 = flex布局。
4. 使用flex布局的元素，称为flex容器（flex container）简称"容器"。它的所有子元素自动成为容器的成员，称为flex项目（flex item）简称"项目"。
5. **重点：子容器可以横向排列也可以纵向排列。**
6. **总结flex布局原理：就是通过给父盒子添加flex属性，来控制子盒子的位置和排列方式。**

3. flex布局父项常见属性

- flex-direction：设置主轴的方向
- justify-content：设置主轴上的子元素排列方式
- flex-wrap：设置子元素是否换行
- align-content：设置侧轴上子元素的排列方式（多行）
- align-items：设置侧轴上的子元素排列方式（单行）
- flex-flow：复合属性，相当于同时设置了flex-direction和flex-wrap

4.【父项属性】flex-direction

1. flex-direction：设置主轴的方向
2. justify-content：设置主轴上的子元素排列方式
3. flex-wrap：设置子元素是否换行
4. align-content：设置侧轴上子元素的排列方式（多行）
5. align-items：设置侧轴上的子元素排列方式（单行）
6. flex-flow：复合属性，相当于同时设置了flex-direction和flex-wrap

flex-direction 设置主轴的方向：

- flex-direction属性决定主轴的方向（即项目的排列方向）。
- 注意：主轴和侧轴是会变化的，就看flex-direction设谁为主轴，剩下的就是侧轴，而子元素是跟着主轴来排列的。

属性值	说明
row（常用）	默认值从左到右
row-reverse	从右到左
column（常用）	从上到下
column-reverse	从下到上

5.【父项属性】justify-content

justify-content 设置主轴上的子元素排列方式：

> justify-content属性定义了项目在主轴上的对齐方式。
>
> **注意：注意使用这个属性之前一定要确定好主轴是哪个。**

属性值	说明
flex-start	默认值从头部开始，如果主轴是x轴，则从左到右
flex-end	从尾部开始排列
center	在主轴居中对齐（如果主轴是x轴则水平居中）
space-around	平分剩余空间
space-between	先两边贴边，再平分剩余空间（重要）

6.【父项属性】flex-wrap

flex-wrap 设置子元素是否换行：

> 默认情况下，项目都排在一条线上（又称"轴线"）上。flex-wrap属性定义，flex布局中默认是不换行的。

属性值	说明
nowrap	默认值，不换行
wrap	**换行**

7.【父项属性】align-items

align-items 设置侧轴上的子元素排列方式（单行）：

> 该属性是控制子项在侧轴（默认是y轴）上的排列方式，在子项为单项的时候使用。

属性值	说明
flex-start	从上到下
flex-end	从下到上
center	挤在一起居中（垂直居中）
stretch	拉伸（默认值）

8. align-content和align-items区别

1. align-items适用于单行情况，只有上对齐，下对齐，居中和拉伸。
2. align-contents适用于换行（多行）的情况（单行的情况下无效），可以设置上对齐，下对齐，居中，拉伸以及平均分配剩余空间等属性值。
3. 总结就是单行使用**align-items**，多行使用**align-contents**。

9.【父项属性】flex-flow

1. flex-flow属性是flex-direction和flex-wrap属性的符合属性。
2. 语法：**flex-flow:row wrap;**

10. flex布局子项常见属性

1. flex子项占的份数（可以是数字也可以是百分比）。
2. align-self控制子项自己在侧轴的排列方式。
3. order属性定义子项的排列顺序（前后顺序）。

11.【子项属性】flex

1. flex属性定义子项目分配剩余空间，用flex来表示占多少份数。
2. 可以是份数（数字值），也可以是百分比。

12.【子项属性】align-self

1. align-self属性允许单个项目有与其它项目不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。
2. 语法：

```
span:nth-child(2){
  /* 设置自己在侧轴上的排列方式 */
  align-self:flex-end;
}
```

13.【子项属性】order

1. order属性定义项目的排列顺序。
2. 数值越小，排列越靠前，默认值为0。
3. 注意：和z-index不一样。z-index是数值越大越靠上。

09. 响应式布局

1. 响应式开发原理

> 响应式开发原理就是**使用媒体查询**针对不同宽度的设备进行布局和样式设置，从而适配不同设备的目的。

设备划分	尺寸区间
超小屏幕（手机）	< 768px
小屏设备（平板）	⩾768px ~ ⩽992px
中等屏幕（桌面显示器）	⩾992px ~ <1200px
宽屏设备（大桌面显示器）	⩾1200px

2. 响应式布局容器

1. 响应式需要一个父级做为布局容器，来配合子级元素来实现变化效果。
2. 原理就是在不同屏幕下，通过媒体查询来改变这个布局容器的大小，再改变里面子元素的排列方式和大小，从而实现不同屏幕下，看到不同的页面布局和样式变化。

3. 响应式尺寸的划分

1. 超小屏幕（手机，小于768px）：设置宽度为100%。
2. 小屏幕（平板，大于等于768px）：设置宽度为750px。
3. 中等屏幕（桌面显示器，大于等于992px）：宽度设置为970px。

4. 大屏幕（大桌面显示器，大于等于1200px）：宽度设置为1170px。

4. bootstrap

> 响应式布局可以使用bootstrap。

1. Bootstrap来自Twitter（推特），是目前最受欢迎的前端框架。
2. Bootstrap是基于HTML，CSS，Javascript的，它简介灵活，使得web开发更加快捷。
3. 中文官网：https://www.bootcss.com/
4. 官网：https://getbootstrap.com/

10. 移动端布局vw和vh

1. 市场比较常见布局方案

1. 需要不断修改html文字大小
2. 需要媒体查询@media
3. 需要flexible.js

2. 现在的布局趋势

1. 省去各种判断和修改
2. vw和vh布局方式就会省去媒体查询等。

3. vw和vh

1. vw和vh是一个相对单位（类似em和rem相对单位），是相对单位总是相对视口来说的。**1vw就是当前视口的1/100。**

- vw是：viewport width 视口宽度单位
- vh是：viewport height 视口高度单位

2. 相对视口的尺寸计算结果

- 1vw = 1/100 视口宽度
- 1vh = 1/100 视口高度
- 例如：当前屏幕视口是375像素，则1vw就是3.75像素。如果当前屏幕视口为414，则1vw就是4.14像素。

4. vw和rem区别

1. **rem: root em**

- 动态设置html的font-size。
- font-size = 视口的宽度 / 10（例如：375 / 10 = 37.5是根元素html字体的大小）。
- 元素rem = 元素 px / 37.5。

2. **vw: viewport width**

- 动态设置html的font-size。
- font-size = 视口的宽度 / 100（例如：375 / 100 = 3.75）。
- 那么1vw就等于3.75px（1vw是视口宽度的一百分之一）。

5. vw和百分比的区别

1. vw/vh是相对于当前视口来说的。
2. 百分比%相对于父元素来说的。

6. vw/vh使用

> 如何还原设计稿?

1. 前提：我们设计稿按照iPhone678来设计的，有个盒子50像素*50像素，如何使用vw呢?
2. 分析：

- 设计稿参照iPhone678，所以视口宽度尺寸是375像素（**像素大厨切换到2x模式**）。
- 那么 1vw 是多少像素? 375px / 100 = 3.75px。
- 我们元素的目标是多少像素? 50px * 50px。
- 那么50*50是多少个vw? 50 / 3.75 = 13.3333vw。

开发中使用vw，需要像素大厨有那么改动?

- 把模式改为2x模式

7. 开发中使用vw，如何还原设计稿?

1. 确定设计稿视口的宽度。比如：375（375实际上是2x模式，屏幕的尺寸是750px）。
2. vw=直接使用测量数值px / （视口宽度 / 100）。
3. 比如：50 / （375/100）。

8. vscode中安装 px2vw

1. 在vscode中安装**px2vw**，这个和rem转换使用的 **(px to rem&rpx&vw)** 是一样的，可以将px单位自动转换成vw。
2. **px2vw**和 **(px to rem & rpx & vw)** 两者任选一个就行。
3. 安装完成后需要设置（扩展设置），将design width设置成375，重启vscode即可。

11. 移动布局总结

---

1. 移动端开发方式

1. 单独制作移动端页面（主流）
2. 响应式页面兼容移动端（其次）

2. 移动端技术选型

1. 流式布局（也就是百分比布局）
2. flex布局（优先先择）
3. rem布局（优先选择）
4. vw布局
5. 响应式布局

注意：一般情况下选取一种作为主要技术，其它技术作为辅助技术，采取混合式开发。

3. vw和rem的对比

1. rem事实上是作为一个过渡方案，他利用的也是vw的思想。

- 前面不管是我们自己编写的js，还是flexible的源码。
- 都是1rem等于设计稿的1/10（十分之一），在利用1rem计算相对于整个屏幕的尺寸大小。
- 那么我们来思考，1vw不是刚好等于屏幕的1/100吗?
- 而且相对于rem还更加有优势。

2. vw相比rem的优势

- 优势一：不需要去计算html的font-size大小，也不需要给html设置一个font-size。
- 优势二：不会因为设置html的font-size大小，而必须给body再设置一个font-size，防止继承。
- 优势三：因为不会依赖font-siez尺寸，所以不用担心某些原因html的font-size尺寸被篡改（rem依赖html的font-size），页面尺寸混乱。
- 优势四：vw相比于rem更加语义化，1vw刚好是1/100的viewport的大小。
- 优势五：可以具备rem之前所有的优点。

3. vw我们面临一个问题，将尺寸换算成vw的单位即可（px to vw插件）。
4. 目前相比于rem，更加推荐大家使用vw（但是理解rem也是非常重要的）。