移动端项目开发笔记

1. 基本概念

屏幕尺寸：手机对角线的长度（厘米）

屏幕分辨率：横纵方向上物理像素的个数

屏幕密度：每英寸上物理像素的个数

视口尺寸：横纵方向上 css 像素的个数（css 像素）

2. 4 个像素 3 个视口 2 个操作 1 个比例

I . 4 个像素

物理像素

也称为设备像素，屏幕的最小物理单位。需要注意的是一个像素并不一定是一个小正方形区块，也没有标准的宽高，只是用于显示丰富色彩的一个“点”而已。可以参考公园里的景观变色彩灯，一个彩灯(物理像素)由红、蓝、绿小灯组成，三盏小灯不同的亮度混合出各种色彩。比如当我们打开 Iphone12 的官网，产品参数里“2340 x 1080 像素分辨率”指的就是设备像素，表示屏幕水平有 1080 个点，垂直有 2340 个点。

Q1 一个物理像素占据的实际屏幕尺寸在不同设备上是否一样？

不一样 设备出厂时，该款设备所包含的物理像素的点数和一个物理像素所占据的实际屏幕尺寸是不会变的

CSS 像素

浏览器使用的单位，用来精确度量网页上的内容，比如 div { width: 100px; }。在一般情况下（页面缩放比为 1），1 个 CSS 像素 等于 1 个设备独立像素。比如，假设把屏幕独立像素分辨率设置为 1440x900，给页面元素设置为宽度 720px，则视觉上元素的宽度是屏幕宽度的一半。这也解释了为什么当我们把独立像素分辨率调高后网页的文字感觉变小了。当页面缩放比不为 1 时，CSS 像素和设备独立像素不再对应。比如当页面放大 200%，则 1 个 CSS 像素等于 4 个设备独立像素。

Q2 一个 css 像素到底占据多少个物理像素和谁有关？

屏幕特性 用户的缩放行为

不考虑用户缩放

没有 viewport：

这块屏幕横向上占据了多少个物理像素（横向分辨率）

这块屏幕横向上占据了多少个 css 像素（视觉视口的横向尺寸）

有 viewport

像素比（一个方向上所占据的物理像素个数/一个方向上所占据的 css 像素的个数

考虑用户缩放

在屏幕的特性的基础上

放大：css 像素占据更多的物理像素

缩小：css 像素占据更少的物理像素

设备独立像素

也称作逻辑像素。比如我们偶尔会说“电脑屏幕在 2560x1600 分辨率下字太小了，我们把它调为 1440x900”，这里的“分辨率”（非严谨说法）指的就是设备独立像素。可以通过 window.screen.width/ window.screen.height 查看。另外，平时我们所说的 iphone12mini 的逻辑像素是 375 x 812，iPhone12 Pro Max 的

的逻辑像素是 400 x 880。一个设备独立像素里可能包含 1 个或者多个物理像素点，包含的越多则屏幕看起来越清晰。

位图像素

图片的最小单位

位图像素与物理像素一比一时，图片才可以完美清晰的展示

II.3 个视口

布局视口（layout viewport）

手机上，为了容纳为桌面浏览器设计的网站，默认的布局视口的宽度远大于屏幕的宽度

布局视口的出现，极大程度上帮助了桌面网站到移动设备上的转移。

可以通过 doucument.documentElemen.clientwidth 来获取

在 pc 端，单独一个 width 为 20%的元素最终拿到的值要根据初始包含块的 width 来决定，因为我们横向布局都是安初始包含块开始填的，在移动端一样，不过这时我们应该叫他为布局视口

视觉视口（visual viewport）

视觉视口与设备屏幕一样宽，并且它的 css 像素的数量会随用户的缩放而改变

Visual viewport 的宽度可以通过 window.innerwidth 来获取（但在 Android 2，Oprea mini 和 UC8 中无法正确获取）

理想视口

我们分析知道，布局视口的默认宽度并不是一个理想的宽度，对于我们移动设备来说，最理想的情况是用户刚进入页面时不再需要缩放。这就是为什么苹果和其他厢房苹果的浏览器厂商会引进理想视口

只有是专门为移动设备开发的网站，他才有理想视口这一说。而且只有当你在页面中加入 viewport 的 meta 标签，理想视口才会生效。

理想视口可以通过 screen.width 来获取（兼容性极大）

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

这一行代码告诉我们，布局视口的宽度应该与理想视口的宽度一致

III. 2 个操作

用户

只影响布局视口

系统（initial-scale）

影响布局视口和视觉视口

放大

放大一个 css 像素的面积，视觉视口的尺寸变小，一个 css 像素包含的物理像素的个数变多

缩小

缩小一个 css 像素的面积,视觉视口的尺寸变大，一个 css 像素包含的物理像素的个数变少

VI.一个比例

像素比

官方定义：物理像素/设备独立像素

一个方向上所占据的物理像素个数/一个方向上所占据的 css 像素的个数

3.等比问题

没有 viewport：

等比，页面展示太小，用户体验不好

有 viewport：

不等比，每一个 css 像素在不同的设备占据的实际屏幕尺寸一样

每一个 css 像素在不同设备占据的物理像素的个数不一样（像素比）。

一个物理像素占据的实际屏幕尺寸在不同设备上是不一样的

PS：

总的来说一定要适配！！！！

完美视口解决大部分问题 记得移动端开发加此行代码：

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

（使用完美视口解决元素过大超过视觉视口的问题后会出现不出现滚动条问题）

（当 width 和 initial-scale 冲突时谁大听谁）

三种适配方法：

1.rem 适配（音悦台项目应用的适配方式）

步骤

第一步　创建 style 标签

第二三步　将根标签的 font-size 置为布局视口的宽/16

第四步　将 style 标签添加到 head 中

原理

改变一个元素在不同设备上的 css 像素的个数

优缺点

优点：可以使用完美视口

缺点：px 到 rem 的转化特别麻烦（可以借助 less）

2.viewport 适配

步骤

将所有设备的布局视口的宽置为设计图的宽度

第一步　定义设计图的宽度

第二步　确定系统缩放比例

第三步　选中 viewport 标签，改变其 content 指

原理

改变不同设备上一个 css 像素根物理像素的比例

优缺点

优点：所量即所得

缺点：破坏了完美视口

3.百分比适配（vw 适配）
　百分比参照谁
优点
1.适配原理简单
2.不需要 JS 即可适配
3.设计稿标注的 px 换算到 CSS 的 vw 计算复杂
4.方案灵活技能实现整体缩放又能实现局部不缩放
缺点
1.计算繁琐

4.流体（弹性布局 flex）+固定（不是一种适配方法）

一物理像素的实现
 rem+系统缩放
1.主体适配采用 rem 适配 并放大 rem 的基值（dpr 倍）
2.再通过系统缩放 缩回 dpr 倍 initial-scale=1/dpr
 响应式+变换伸缩

移动端事件基础
1.queryselectorAll：静态列表　queryselector：静态列表的第一个(前者选中列表中的全部元素，后者只获取列表中的第一个元素)
2.touchstart（手指按到屏幕上）　touchmove（手指在屏幕上滑动）　touchend（手指离开屏幕）

属性名	作用
type	事件类型
target	事件源
preventDefault（returnValue）	阻止默认行为
stopPropagation(cancleBubble)	停止事件的传播
touches[0].clientX	触碰位置的 x 值
changedTouches	当前的值和离开的值

3.阻止事件的默认行为，阻止事件的冒泡
4.怎么全局阻止默认事件

```
<script type="text/javascript">
    window.onload=function(){
        document.addEventListener("touchstart",function(ev){
            ev=ev||event;
            ev.preventDefault();
        })
 </script>
```

5.事件点透(关闭浮层用 click，不用 touchStart)

```
<script type="text/javascript">
        window.onload=function(){
                1.pc 端的事件可以在移动端触发
                2.PC 端事件有 300 毫秒延迟
                3.移动端事件不会有延迟
            var item = document.querySelector(".item");
            var a = document.querySelector("a");
            item.addEventListener("touchstart",function(){
                this.style.display="none";
            })
            a.addEventListener("click",function(){
                console.log("click from a");
            })
        }
    </script>
```

6.event 3 类手指列表

---changedTouches:触发当前事件的手指列表

---targetTouches:触发当前事件时元素上的手指列表

//touches:触发当前事件时屏幕上的手指列表

7.常见问题

-webkit-appearance：none

-webkit-tap-highlight-color

font boosting

布局

布局包裹器

滑屏元素（动态生成）

---queryselector 的坑

---有时绘制跟不上 js 引擎的渲染

解决方法：定时器

基本的滑屏

拿到元素和手指一开始（点击到布局包裹器上时）的位置

拿到元素实时的位置，再去计算手指实时的偏移量，将偏移量实时的同步给滑屏元素

基本滑屏的两个版本

定位板（两种抽象：1.图片的下标 2.ul 的位置）

OffsetLeft：累加的过程

2d 变换版（单独的图层）

---变量（业务逻辑变得复杂）

---定义了 css 函数

节点的属性来管理变换类型与他所对应的值

2 个参数
读取
3 个参数
设置（单位）
循环节点的属性

无缝 自动滑屏

无缝：复制一组图片，当点击第一组第一张瞬间跳到第二组的第一张
当点击第二组最后一张时瞬间跳到第一组的最后一张

自动滑屏：循环定时器
函数包裹器(重启定时器)
清定时器（循环定时器的逻辑没有必要同一时刻开启多次；暂停逻辑）

自动滑屏和无缝的冲突
使用同一个 index 变量就可以

项目笔记

1.头部布局+效果
---怎么使用 less 来弥补 rem 适配缺点
定义了一个变量@rem（1rem 包含多少位图像素）
---表单
表单高亮 outline：none
表单内阴影 border：none
---1 物理像素的现实
less 混合版
---移动端骨架搭建
meta 标签
挑选一个适配方案（百分比 rem 适配 viewport 适配）
布局形式（流体+固体 275px）
全面禁止事件默认行为的 js（每次冒泡的时候，记住阻止事件的默认行为）

2.导航的布局+橡皮筋效果
---导航 滑屏区域与滑屏元素的布局
滑屏区域宽度必定占满一个视口
滑屏区域宽度百分比
滑屏元素必须被子项撑开
滑屏元素必须浮动（为了能被子项撑开 禁止子项换行）
子项统一 inline-block
---无缝滑屏 滑屏区域与滑屏元素布局
滑屏区域宽度必定占满一个视口
滑屏区域宽度百分比
滑屏元素必须被子项撑开
Width：子项个数*100%
子项：1/子项个数*100%

橡皮筋效果

减少每次 move 的有效距离，最终的移动距离还是一直增大

move：每次手指移动的距离

混合级和继承级的区分

混合

```
.mixin(){
    规则集
}
#test{
  .mixin()
  //规则集
}
#test1{
  .mixin()
  //规则集
}

#test2{
  .mixin()
  //规则集
}
```

继承

```
.extend{
  规则集
}
#test{
 &:：extend（.extend）
}
#test1{
 &:：extend（.extend）
}

#test2{
 &:：extend（.extend）
}
#test，#test1，#test2{
  规则集
}
```

第一版手动橡皮筋效果

disX：touchstart 到 touchend 整个过程中手指滑动的距离

elementX 为基准（每次手指 touchstart 的时候，元素的位置）

pointDis：nowPoint-lastPoint

在 touchmove 的最后一定要记住同步 nowpoint

给 lastPoint

应该以实时位置为基准（damu.css 函数）

手动橡皮筋

减小每次手指移动（一次 touchmove 手指滑动的距离）的有效距离

快速滑屏

拿到手指 touchend 时的 speed

以速度为基准值延长目标位置

tragetX=translate+speed*200;

speed:最后一次 touchmove 过程中的平均速度

pointdis/timedis

速度的正负代表方向

快速滑屏的橡皮筋效果

“贝塞尔”

快速滑屏的橡皮筋效果和手动橡皮筋的冲突

在手动橡皮筋鲜果触发时不开启快速滑屏

2 个 bug

一、不处理 distime；

导致首次点击导航后，滑屏会失效

解决方法：distime=1；

二、速度的残留

导致每次 touchstart 时重置 dispoint

解决方法：dispoint=0；

防抖动

isX

isFirst

判断用户的首次操作

如果是 x 轴滑屏，不管用户以后怎么操作，滑屏逻辑永远都会被触发

如果是 y 轴滑屏，不管用户以后怎么操作，滑屏逻辑永远都不会被触发

怎么防止首次抖动

1.确定了用户首次操作的方向

isFirst 加入逻辑就变为 false，直到下一次 touchstart 置回

如果是 y return

怎么防止后续抖动

在 touchmove 一上来时，就对用户首次操的方向进行判断

如果是 y return

tab 选项卡布局

less 继承（清除浮动 要加 all）

tab 选项卡基本滑屏逻辑

滑屏的必要参数都拿对象来存储

防抖动

防 y 轴的抖动

基于 tab 本身业务的滑屏逻辑

---滑屏区域和滑屏元素是同一个 dom 节点

---手指没有滑过 1/2

触发正常的滑屏逻辑，我们不关心

touchmove 的触发次数

---手指滑过 1/2

我们必须确定 touchmove 中滑过 1/2 的跳转（jump）只触发一次

滑屏元素要跳转到指定的位置

小绿要同步（now++ now--）

tap 几点注意

让一个逻辑只触发一次

1.定时器

在一段时间内触发一次

2.立 flag

在一类操作内触发一次

3.循环定时器 用 dom2 对 transitionend 绑定回调时回调的第一行

清楚循环定时器

解绑事件

4.transition 指定样式的时候，明确出给哪个样式指定

存址存值

在一部分移动端浏览器上，事件的 event 可能会指向同一个应用

---->快照

3d 硬件加速

直接走 gpu，不走软件计算，跨过浏览器这一段

3d 变换：没有重绘重排

竖向滑屏

即点即停：Tween 算法

t:当前是哪一次

b:初始位置

c:初始位置与最终位置之间的差值

d:总次数

s:回弹距离

css 动画（animation）

js 触发 css 动画（过渡 animation）

js 动画（循环定时器）

自定义滚动条

滚动条的高度/滚动条滚动区域的高度=滚动条

滚动区域的高度/可滑动内容的高度

滚动条的偏移量/滚动条可以滚动的最远偏移量=内容的偏移量/内容可以移动的最远偏移量

组件如何去封装外部逻辑

Apk 打包