1.设备像素、设备独立像素、CSS像素、分辨率、PPI、devicePixelRatio 的区别

- 分辨率
  - 1920px * 1080px ,指屏幕横向可以显示1920个物理像素点，纵向可以显示1080个物理像素点。
- 像素
  - 我们看到所显示在屏幕上的图像,实际上都是由非常多的像素点所组成。各个像素点通过发出不同颜色的光来呈现屏幕的色彩。
- 设备像素（物理像素）
  - 也称为物理像素，显示器的最小物理单位，没有标准的宽高，只是用于显示丰富色彩的一个“点”而已
  - 随着设备生产出来就已经被确定了。例如iPhone5的分辨率是`640x1136px`,代表屏幕由640行,1136列像素点组成。
- 设备独立像素（逻辑像素）（DIP/DIPS）
  - 也称设备无关像素 Device Independent Pixels
  - 可以通过 window.screen.width/ window.screen.height 查看
  - chrome 模拟器iphonex 375 * 812 指的就是设备独立像素
- PPI（pix per inch）
  - 每英寸的物理每英寸的物理像素数。以尺寸为5.8英寸（屏幕对角线长度）、分辨率为`1125x2436`的iphonex为例， `ppi = Math.sqrt(1125*1125 + 2436*2436) / 5.8` ，值为463ppi。
- CSS 像素
  - 在电脑上，1个css像素等于1个设备独立像素（逻辑像素），页面缩放比为1。
  - 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。
- devicePixelRatio
  - window.devicePixelRatio指的是设备物理像素和设备独立像素（device-independent pixels, dips）的比例。
  - window.devicePixelRatio = 物理像素 / 设备独立像素（dips）` 。经计算， iphone x的devicePixelRatio 是3。
- 尺寸的区别
  - 获取屏幕尺寸（设备独立像素值）

    ```
    screen.width
    screen.height
    ```

  - 获取窗口尺寸（css像素）
    - 包含滚动条

    ```
    window.innerWidth
    window.innerHeight
    ```

    - 不包含滚动条

```
document.documentElement.clientWidth
document.documentElement.clientHeight
```

- 总结

```
// iphonex 的设备独立像素值
screen.width  // 375
screen.height // 812

// 物理像素与设备独立像素比值
window.devicePixelRatio  // 3

// 物理像素
375 * 3 = 1125
812 * 3 = 2436
```

2. rem是什么?

- rem（font size of the root element）是指相对于根元素的字体大小的单位。简单的说它就是一个相对单位。看到rem大家一定会想起em单位，em（font size of the element）是指相对于父元素的字体大小的单位。
- 它们之间其实很相似，只不过rem计算的规则是依赖根元素，em计算的规则是依赖父元素。

3. 怎么使用rem?

- rem是通过根元素进行适配的，网页中的根元素指的是html，我们通过设置html的字体大小就可以控制rem的大小。
- 不同的移动设备，屏幕分辨率不同，根元素的大小也需要跟随着变化，一般是使用felxible.js, JS动态计算根元素font-size

```
// flexible.js
(function(doc, win) {
    var docEl = doc.documentElement,
        resizeEvt = 'orientationchange' in window ? 'orientationchange' :
'resize',
        recalc = function() {
            var clientWidth = docEl.clientWidth;
            if (!clientWidth) return;
            docEl.style.fontSize = 20 * (clientWidth / 375) + 'px';
        };
    if (!doc.addEventListener) return;
    win.addEventListener(resizeEvt, recalc, false);
    doc.addEventListener('DOMContentLoaded', recalc, false);
})(document, window);
```

- 这里把屏幕分成了375份再乘以20，则font-size 会随着屏幕的变化而变化，且始终是20的倍数

```
Vue.prototype.px2rem = (px) => {
    return px / 20 + 'rem';
}

/* 自动计算rem */
@function px2rem($px) {
    $baseFontSize: 20px;
    @return $px / $baseFontSize * 1rem;
}
```

- 通过此函数可以把px快速转换为rem
- 如果不想使用使用自动转换的，在确定了设计稿的尺寸后，可以取巧利于计算
    - 设计稿 375
    - docEl.style.fontSize = 100 * (clientWidth / 375) + 'px'
    - 设计稿上的100px则可以直接换算为100，即1rem