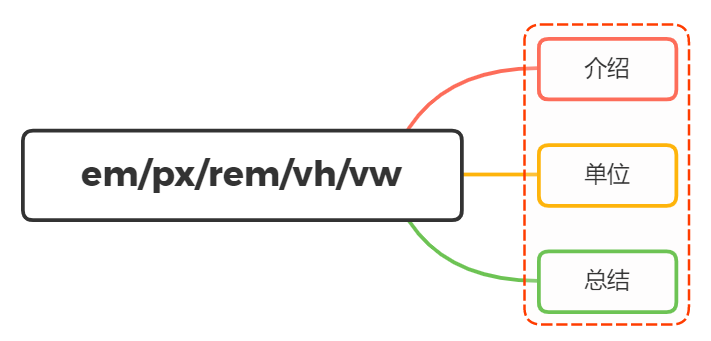
# 面试官：说说em/px/rem/vh/vw区别?



## 一、介绍

传统的项目开发中，我们只会用到px、%、em这几个单位，它可以适用于大部分的项目开发，且拥有比较良好的兼容性

从CSS3开始，浏览器对计量单位的支持又提升到了另外一个境界，新增了rem、vh、vw、vm等一些新的计量单位

利用这些新的单位开发出比较良好的响应式页面，适应多种不同分辨率的终端，包括移动设备等

## 二、单位

在css单位中，可以分为长度单位、绝对单位，如下表所指示

|  |  |
| --- | --- |
| CSS单位 |  |
| 相对长度单位 | em、ex、ch、rem、vw、vh、vmin、vmax、% |
| 绝对长度单位 | cm、mm、in、px、pt、pc |

这里我们主要讲述px、em、rem、vh、vw

### px

px，表示像素，所谓像素就是呈现在我们显示器上的一个个小点，每个像素点都是大小等同的，所以像素为计量单位被分在了绝对长度单位中

有些人会把px认为是相对长度，原因在于在移动端中存在设备像素比，px实际显示的大小是不确定的

这里之所以认为px为绝对单位，在于px的大小和元素的其他属性无关

### em

em是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸（1em = 16px）

为了简化 font-size 的换算，我们需要在css中的 body 选择器中声明font-size= 62.5%，这就使 em 值变为 16px\*62.5% = 10px

这样 12px = 1.2em, 10px = 1em, 也就是说只需要将你的原来的px 数值除以 10，然后换上 em作为单位就行了

特点：

* em 的值并不是固定的
* em 会继承父级元素的字体大小
* em 是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸
* 任意浏览器的默认字体高都是 16px

举个例子

<div class="big">  
 我是14px=1.4rem<div class="small">我是12px=1.2rem</div>  
</div>

样式为

<style>  
 html {font-size: 10px; } /\* 公式16px\*62.5%=10px \*/   
 .big{font-size: 1.4rem}  
 .small{font-size: 1.2rem}  
</style>

这时候.big元素的font-size为14px，而.small元素的font-size为12px

### rem

rem，相对单位，相对的只是HTML根元素font-size的值

同理，如果想要简化font-size的转化，我们可以在根元素html中加入font-size: 62.5%

html {font-size: 62.5%; } /\* 公式16px\*62.5%=10px \*/

这样页面中1rem=10px、1.2rem=12px、1.4rem=14px、1.6rem=16px;使得视觉、使用、书写都得到了极大的帮助

特点：

* rem单位可谓集相对大小和绝对大小的优点于一身
* 和em不同的是rem总是相对于根元素，而不像em一样使用级联的方式来计算尺寸

### vh、vw

vw ，就是根据窗口的宽度，分成100等份，100vw就表示满宽，50vw就表示一半宽。（vw 始终是针对窗口的宽），同理，vh则为窗口的高度

这里的窗口分成几种情况：

* 在桌面端，指的是浏览器的可视区域
* 移动端指的就是布局视口

像vw、vh，比较容易混淆的一个单位是%，不过百分比宽泛的讲是相对于父元素：

* 对于普通定位元素就是我们理解的父元素
* 对于position: absolute;的元素是相对于已定位的父元素
* 对于position: fixed;的元素是相对于 ViewPort（可视窗口）

## 三、总结

**px**：绝对单位，页面按精确像素展示

**em**：相对单位，基准点为父节点字体的大小，如果自身定义了font-size按自身来计算，整个页面内1em不是一个固定的值

**rem**：相对单位，可理解为root em, 相对根节点html的字体大小来计算

**vh、vw**：主要用于页面视口大小布局，在页面布局上更加方便简单