--

[CSS中的单位[尺寸] 1](#_Toc7340)

[[ em rem ] + [ vh vw vmin vmax ] 1](#_Toc450)

[应用1 1](#_Toc24801)

[应用 2](#_Toc2256)

[应用2 4](#_Toc12240)

<http://www.cnblogs.com/lyzg/p/4877277.html>

[从网易与淘宝的font-size思考前端设计稿与工作流](http://www.cnblogs.com/lyzg/p/4877277.html)

<https://smohan.net/blog/mggrai>

从网易与淘宝的FONT-SIZE思考前端设计稿与工作流

<http://hcysun.me/2015/10/19/%E4%B8%80%E7%AF%87%E7%9C%9F%E6%AD%A3%E6%95%99%E4%BC%9A%E4%BD%A0%E5%BC%80%E5%8F%91%E7%A7%BB%E5%8A%A8%E7%AB%AF%E9%A1%B5%E9%9D%A2%E7%9A%84%E6%96%87%E7%AB%A0-%E4%BA%8C/>

<http://hcysun.me/2015/10/16/%E4%B8%80%E7%AF%87%E7%9C%9F%E6%AD%A3%E6%95%99%E4%BC%9A%E4%BD%A0%E5%BC%80%E5%8F%91%E7%A7%BB%E5%8A%A8%E7%AB%AF%E9%A1%B5%E9%9D%A2%E7%9A%84%E6%96%87%E7%AB%A0(%E4%B8%80)/>

[2015-10-19一篇真正教会你开发移动端页面的文章(二)](http://hcysun.me/2015/10/19/%E4%B8%80%E7%AF%87%E7%9C%9F%E6%AD%A3%E6%95%99%E4%BC%9A%E4%BD%A0%E5%BC%80%E5%8F%91%E7%A7%BB%E5%8A%A8%E7%AB%AF%E9%A1%B5%E9%9D%A2%E7%9A%84%E6%96%87%E7%AB%A0-%E4%BA%8C/)

[2015-10-16一篇真正教会你开发移动端页面的文章(一)](http://hcysun.me/2015/10/16/%E4%B8%80%E7%AF%87%E7%9C%9F%E6%AD%A3%E6%95%99%E4%BC%9A%E4%BD%A0%E5%BC%80%E5%8F%91%E7%A7%BB%E5%8A%A8%E7%AB%AF%E9%A1%B5%E9%9D%A2%E7%9A%84%E6%96%87%E7%AB%A0(%E4%B8%80)/)

CSS中的单位[尺寸]

|  |  |  |
| --- | --- | --- |
| 各种单位 | % | 百分比 |
| in | 英寸 |
| cm | 厘米 |
| mm | 毫米 |
| em | 1em 等于当前的字体尺寸。  2em 等于当前字体尺寸的两倍。  例如，如果某元素以 12pt 显示，那么 2em 是24pt。  在 CSS 中，em 是非常有用的单位，因为它可以自动适应用户所使用的字体。 |
| ex | 一个 ex 是一个字体的 x-height。 (x-height 通常是字体尺寸的一半) |
| pt | 磅 (1 pt 等于 1/72 英寸) |
| pc | 12 点活字 (1 pc 等于 12 点) |
| px | 像素 (计算机屏幕上的一个点) |

### [ em rem ] + [ vh vw vmin vmax ]

|  |  |
| --- | --- |
| em | 14px |
| rem | r是“root”的缩写，意思就是1rem等于根元素的字体大小；大部分情况下，根元素就是<html>元素了。  html { font-size: 14px;}div { font-size: 1.2rem;}  这样在上面的那三个嵌套的div娃们的字体大小都是 1.2\*14px = 16.8px 了。 |
|  | <http://caibaojian.com/rem-vs-em.html>  [rem与em的使用和区别详解](http://caibaojian.com/rem-vs-em.html) |
| vh vw  视口的百分比 | 响应式web设计离不开百分比。但是，CSS百分比并不是所有的问题的最佳解决方案。CSS的宽度是相对于包含它的最近的父元素的宽度的。但是如果你就想用视口（viewpoint）的宽度或者高度，而不是父元素的，那该肿么办？ 这就是 vh 和 vw 单位为我们提供的。  1vh 等于1/100的视口高度。栗子：浏览器高度900px, 1 vh = 900px/100 = 9 px。同理，如果视口宽度未750， 1vw = 750px/100 = 7.5 px。  可以想象到的，他们有很多很多的用途。比如，我们用很简单的方法只用一行CSS代码就实现同屏幕等高的框。  .slide { height: 100vh;}  假设你要来一个和屏幕同宽的标题，你只要设置这个标题的font-size的单位为vm，那标题的字体大小就会自动根据浏览器的宽度进行缩放，以达到字体和viewport大小同步的效果. |
| vmin vmax | vh和 vm 依据于视口的高度和宽度，相对的，vmin 和 vmax则关于视口高度和宽度两者的最小或者最大值。比如，浏览器的宽度设置为1100px，高度设置为700px， 1vmin = 1px， 1vmax = 11px。如果宽度设置为800px,高度设置为1080px, 1vmin就等于8px, 1vmax则未10.8px。 那么问题来了，我们应该在什么场景下使用这两个单位呢？ 假设有一个元素，你需要让它始终在屏幕上可见。只要对其高度和宽度使用vmin单位，并赋予其低于100的值就可以做到了。再来个栗子，可以这样定义一个至少有两个边触摸到屏幕的方形：  .box { height: 100vmin; width: 100vmin;}  IMG_256  如果你要让这个方形框框始终铺满整个视口的可见区域（四边始终触摸到屏幕的四边）  .box { height: 100vmax; width: 100vmax;}  IMG_257  结合使用这些单位可以为我们提供一个新颖有意思的方式来灵活地利用我们视口的大小。 |
| 应用1 | [认识视口单位vm、vh在网页中的排版应用](http://caibaojian.com/viewport-based-typography.html)  原文链接：<http://caibaojian.com/viewport-based-typography.html> |
|  | 视口单位是什么？  在 [CSS](http://caibaojian.com/css3/" \o "CSS) 规范中，有4种类型的可用视口单位:  vw --- 1vw 等于视口宽度的 1%  vh --- 1vh 等于视口高度的 1%  vmin --- vw 和 vh 中的较小值  vmax --- vw 和 vh 中的较大值  视口，即浏览器屏幕大小，1vw 等于浏览器宽度的 1%，100vw 即整个浏览器的宽度。  IMG_256  视口的单位大小会根据视口大小的改变自动计算，视口大小的改变常发生于页面加载、页面缩放或者屏幕方向的改变(横纵切换)。正因为如此，创建一个大小总为视口四分之一大小的容器是非常容易滴：  .component {  width: 50vw;  height: 50vh;  background: rgba(255, 0, 0, 0.25)  } |
|  | 将视口单位用于网页排版  将视口单位用于网页排版的唯一理由就是视口的单位大小会根据客户端浏览器的视口大小自动计算。也就是说，我们不必明确地通过媒体查询来声明字体大小。举个demo来简要说明一下。  代码如下，将断点设置为 800px，字体大小从 16px 变为 20px：[·](http://caibaojian.com/viewport-based-typography.html)  // Note: CSS are all written in SCSS  html {  font-size: 16px;  @media (min-width: 800px) {  font-size: 20px;  }  }  对于上述代码，当视口大小是 800px 时，字体会从 16px "突变" 到 20px。  在[响应式](http://caibaojian.com/356.html" \o "响应式布局)排版中，这是经常采用的方式。  有时，你会碰到在两个断点之间添加额外的媒体查询来确保页面排版适应所有设备：  html {  font-size: 16px;  @media (min-width: 600px) {  font-size: 18px;  }  @media (min-width: 800px) {  font-size: 20px;  }  }  尽管这样做能达到效果，但需要更多特定的媒体查询规则和字体大小。通常，会选择 3~4 中字体大小。  但是，如何不同媒体查询或字体大小的设置来达到同样的效果呢？  当然是有滴，这就是视口单位的用处了。你可以用视口单位来表示字体大小：  html { font-size: 3vw; } |
|  | 一段简短的代码就实现了适配。但也有明显的缺点，就是视口的单位大小是根据设备屏幕的视口大小计算的，对于小屏幕设备(如宽度 320px 的手机)，字体太小，难以阅读；对于大屏幕设备(如宽度 1440px 的笔记本)，字体会变的非常大，同样也会难以阅读。  所以，现在面临的一个有意思的挑战是---怎么解决不同设备的视口宽度对视口单位计算的影响？  一种简单地方式是设置 font-size 的最小值，然后通过 calc() 属性来动态计算小屏幕设备上的字体大小值：  html { font-size: calc(18px + 0.25vw) }  参考 Mike Riethmuller 的 [Precise control over responsive typography](http://madebymike.com.au/writing/precise-control-responsive-typography)  但是，不是所有的浏览器都支持 calc() 的这种计算方式(px+vw)。解决方式也很简单，结合使用百分比和vw 用于 calc 计算能获得更好地浏览器支持：  html { font-size: calc(112.5% + 0.5vw) } |
|  | 下一个需要克服的挑战就是用视口单位来设置排版元素(h1-h6)的字体大小。  用视口单位设置其它排版元素的字体大小  首先，我创建一个 <h1> 元素，将其字体大小设置为body 的两倍：  html { font-size: calc(112.5% + 0.25vw) }  h1 { font-size: calc((112.5% + 0.25vw) \* 2); }  IMG_256  我试图将 html 元素的字体大小乘以2，但并不可行，对于 <h1>，字体大小是基于百分比计算的。在字体大小继承了 <html> 的大小之后，又重新计算了 <h1> 的字体大小。  现在假设视口宽度是 800px，默认的 font-size 是 16px：  对于 html 元素，112.5% 意味着 font-size 是 18px(112.5/100 \* 16px)  0.25vw == 2px(800px \* 0.25/100)  所以，html 的 font-size 的最终值是 20px(18px + 2px)  按照同样的方法来计算 h1 元素的 font-size，但需要特别注意的是此时百分比(112.5%)的相对计算量值：  对于 h1 元素，112.5% 意味着 font-size 是 22.5px(112.5/100 \* 20px)  0.25vw == 2px(800px \* 0.25/100)  所以，h1 的 font-size 的最终值是 49px((22.5px + 2px) \* 2)  这与最初想把 h1 元素的 font-size 设置成 Body 的两倍大小的想法相违背。但我们知道了造成差异的原因是由于 h1 继承了 html 的 font-size，有两种方式来解决这个问题。  第一种方式就简单滴将 112.5% 改为 100%：  h1 { font-size: calc((100% + 0.25vw) \* 2) }  第二种方式是确保 font-size 不被跨元素继承：  h1 { font-size: calc((100% + 0.25vw) \* 2) }  p { font-size: calc((100% + 0.25vw)) }  这两种方式看起来有点 hack，看起来不爽，于是又继续尝试其它方法。最终，最干净的方式是使用 [Rem & Em](http://zellwk.com/blog/rem-vs-em/):  html { font-size: calc(112.5% + 0.25vw) }  h1 { font-size: 2em; }  IMG_257  既然讲到了字体大小的计算，那接下来的问题是："视口单位的垂直和标准化计算是怎么样的？"  视口单位的垂直和标准化计算  这个相对比较容易回答。不知是否注意到，视口单位常仅被用于 html 元素？其它元素仍用 [rem](http://caibaojian.com/t/rem" \o "rem) 和 em 作为计算的单位。  这就意味着，你仍然能使用 rem 和 em 用于视口单位的垂直和标准化计算，这和我之前在 [Everything I Know about Responsive Typography](http://zellwk.com/blog/responsive-typography) 一文中讨论的一样。  结束这篇文章之前，最后一个需要谈到的问题是：要怎么样去计算 vw 的值，才能在视口宽度是 800px 时，排版的字体大小为 20px？很多人问到了这个问题，因而，将这个问题简化成一个词就是---精确。换句话说，如何才能字体大小更加精确？  精确  结果是，Mike 已经替我解决了这个问题，我只需要再简单解释下计算方式。  假设你要处理下面两种情况：  视口宽度是 600px 时，font-size 是 18px  视口宽度是 1000px 时，font-size 是 20px  首先，我们必须将较小的 font-size 值转为百分比。  第一部计算是：calc(18/16 \* 100%) (或者 calc(112.5%))。  接下来，计算出 vw 值。\*\*这部分的计算略繁琐:  计算 font-size 的最大差值 v1(22-18=4)  用 v1 除以视口宽度的最大差值 v2(1000-600)  将上述结果再乘以 100vw - smaller-viewport-width(100vw - 600)  最终，结果如下：  html {  font-size: calc(112.5% + 4 / 400 \* (100vw - 600px) )  }  开始接触可能会比较复杂，但是熟悉之后，你可以把它简化成 Sass 混入([simple sass mixin](http://www.sassmeister.com/gist/7f22e44ace49b5124eec))。  原文参照  [Viewport Unit Based Typography](http://zellwk.com/blog/viewport-based-typography/)  [rem和em的使用](https://css-tricks.com/rem-global-em-local/)  原文：[基于视口单位的网页排版](https://github.com/dwqs/blog/issues/5" \t "http://caibaojian.com/_blank) |
| 应用 | [纯CSS3使用vw和vh视口单位实现自适应](http://caibaojian.com/vw-vh.html)  http://caibaojian.com/vw-vh.html |
|  | 认识视口单位（ Viewport units )  首先，我们要了解什么是视口。  在业界，极为推崇的一种理论是 Peter-Paul Koch (江湖人称“PPK大神”)提出的关于视口的[解释](http://weizhifeng.net/viewports.html" \t "_blank)——在桌面端，视口指的是在桌面端，指的是浏览器的可视区域；而在移动端较为复杂，它涉及到三个视口：分别是 Layout Viewport（布局视口）、 Visual Viewport（视觉视口）、Ideal Viewport。  而视口单位中的“视口”，在桌面端，毫无疑问指的就是浏览器的可视区域；但是在移动端，它指的则是三个 Viewport 中的 Layout Viewport 。  IMG_256  视口单位中的“视口”  根据[CSS3规范](https://drafts.csswg.org/css-values-3/" \l "viewport-relative-lengths" \t "_blank)，视口单位主要包括以下4个：  vw : 1vw 等于视口宽度的1%  vh : 1vh 等于视口高度的1%  vmin : 选取 vw 和 vh 中最小的那个  vmax : 选取 vw 和 vh 中最大的那个  视口单位区别于%单位，视口单位是依赖于视口的尺寸，根据视口尺寸的百分比来定义的；而%单位则是依赖于元素的祖先元素。  IMG_257  用视口单位度量，视口宽度为100vw，高度为100vh（左侧为竖屏情况，右侧为横屏情况）  例如，在桌面端浏览器视口尺寸为650px，那么 1vw = 650 \* 1% = 6.5px（这是理论推算的出，如果浏览器不支持0.5px，那么实际渲染结果可能是7px）。  兼容性  其兼容性如下图所示，可以知道：在移动端 [ios](http://caibaojian.com/t/ios" \o "ios) 8 以上以及 [Android](http://caibaojian/t/android" \o "android) 4.4 以上获得支持，并且在微信 x5 内核中也得到完美的全面支持。  IMG_258  截图来自[Can I Use](http://caniuse.com/" \l "search=vm" \t "_blank)  IMG_259  截图来自[X5内核－Can I Use](http://res.imtt.qq.com/tbs/incoming20160419/home.html" \t "_blank)  利用视口单位适配页面  对于移动端开发来说，最为重要的一点是如何适配页面，实现多终端的兼容，不同的适配方式各有千秋，也各有缺点。  就主流的响应式布局、弹性布局来说，通过 Media Queries 实现的布局需要配置多个响应断点，而且带来的体验也对用户十分的不友好：布局在响应断点范围内的分辨率下维持不变，而在响应断点切换的瞬间，布局带来断层式的切换变化，如同卡带的唱机般“咔咔咔”地一下又一下。  而通过采用rem单位的动态计算的弹性布局，则是需要在头部内嵌一段脚本来进行监听分辨率的变化来动态改变根元素字体大小，使得 CSS 与 JS 耦合了在一起。  有没有办法能够解决这样的问题呢？  答案是肯定的，通过利用视口单位实现适配的页面，是既能解决响应式断层问题，又能解决脚本依赖的问题的。 |
|  | 做法一：仅使用vw作为CSS单位  在仅使用 vw 单位作为唯一应用的一种 CSS 单位的这种做法下，我们遵守：  1.对于设计稿的尺寸转换为vw单位，我们使用Sass函数编译  //iPhone 6尺寸作为设计稿基准  $vm\_base: 375;  @function vw($px) {  @return ($px / 375) \* 100vw;  }  2.无论是文本还是布局高宽、间距等都使用 vw 作为 CSS 单位  .mod\_nav {  background-color: #fff;  &\_list {  display: flex;  padding: vm(15) vm(10) vm(10); // 内间距  &\_item {  flex: 1;  text-align: center;  font-size: vm(10); // 字体大小  &\_logo {  display: block;  margin: 0 auto;  width: vm(40); // 宽度  height: vm(40); // 高度  img {  display: block;  margin: 0 auto;  max-width: 100%;  }  }  &\_name {  margin-top: vm(2);  }  }  }  }  3.1物理像素线（也就是普通屏幕下 1px ，高清屏幕下 0.5px 的情况）采用 transform 属性 scale 实现。  .mod\_grid {  position: relative;  &::after {  // 实现1物理像素的下边框线  content: '';  position: absolute;  z-index: 1;  pointer-events: none;  background-color: #ddd;  height: 1px;  left: 0;  right: 0;  top: 0;  @media only screen and (-webkit-min-device-pixel-ratio: 2) {  -webkit-transform: scaleY(0.5);  -webkit-transform-origin: 50% 0%;  }  }  ...  }  4.对于需要保持高宽比的图，应改用 padding-top 实现  .mod\_banner {  position: relative;  padding-top: percentage(100/700); // 使用padding-top  height: 0;  overflow: hidden;  img {  width: 100%;  height: auto;  position: absolute;  left: 0;  top: 0;  }  }  由此，我们能够实现一个常见布局的页面效果如下：  IMG_256  体验地址[点击此处](https://jdc.jd.com/demo/ting/vw_layout.html) |
|  | 做法二：搭配vw和rem，布局更优化  这样的页面虽然看起来适配得很好，但是你会发现由于它是利用视口单位实现的布局，依赖于视口大小而自动缩放，无论视口过大还是过小，它也随着视口过大或者过小，失去了最大最小宽度的限制。[·](http://caibaojian.com/vw-vh.html)  当然，你可以不在乎这样微小的不友好用户体验，但我们还是尝试下追求修复这样的小瑕疵吧。  于是，联想到不如结合rem单位来实现布局？rem 弹性布局的核心在于动态改变根元素大小，那么我们可以通过：  给根元素大小设置随着视口变化而变化的 vw 单位，这样就可以实现动态改变其大小。  限制根元素字体大小的最大最小值，配合 body 加上最大宽度和最小宽度  这样我们就能够实现对布局宽度的最大最小限制。因此，根据以上条件，我们可以得出代码实现如下：  //code from http://caibaojian.com/vw-vh.html  // rem 单位换算：定为 75px 只是方便运算，750px-75px、640-64px、1080px-108px，如此类推$vm\_fontsize: 75;  // iPhone 6尺寸的根元素大小基准值  @function rem($px) {  @return ($px / $vm\_fontsize ) \* 1rem;  }  // 根元素大小使用 vw 单位  $vm\_design: 750;  html {  font-size: ($vm\_fontsize / ($vm\_design / 2)) \* 100vw;  // 同时，通过Media Queries 限制根元素最大最小值  @media screen and (max-width: 320px) {  font-size: 64px;  }  @media screen and (min-width: 540px) {  font-size: 108px;  }  }  // body 也增加最大最小宽度限制，避免默认100%宽度的 block 元素跟随 body 而过大过小  body {  max-width: 540px;  min-width: 320px;  }  这里就不再给出截图，但你可以[点击此处在线地址](https://jdc.jd.com/demo/ting/vw_rem_layout.html)进行体验。 |
|  | 小结  相对于做法一，个人比较推崇做法二，有以下两点原因：  第一，做法二相对来说用户视觉体验更好，增加了最大最小宽度的限制；  第二，更重要是，如果选择主流的rem弹性布局方式作为项目开发的适配页面方法，那么做法二更适合于后期项目从 rem 单位过渡到 vw 单位。只需要通过改变根元素大小的计算方式，你就可以不需要其他任何的处理，就无缝过渡到另一种CSS单位，更何况vw单位的使用必然会成为一种更好适配方式，目前它只是碍于兼容性的支持而得不到广泛的应用。  后语  这是笔者在偶然中阅读到[[翻译]使用VH和VW实现真正的流体排版](http://www.cnblogs.com/wengxuesong/archive/2016/05/16/5497653.html" \t "_blank)这一篇文章得到的感悟与成果，也满心欢喜地期待这篇文章同样能够带给读者一些启发，并提出一些的vw单位使用秘笈来交流交流～:）  原文：凹凸实验室 https://aotu.io/notes/2017/04/28/2017-4-28-CSS-viewport-units/  推荐文章  [Bootstrap CSS3媒体查询断点](http://caibaojian.com/bootstrap-media.html)  [CSS预处理器框架与CSS后后处理器框架](http://caibaojian.com/css-post-processor.html)  [CSS自适应导航菜单](http://caibaojian.com/css-responsive-menu.html)  [站外视频/iframe/web内嵌内容响应式代码](http://caibaojian.com/responsive-embed.html)  [CSS多行文字超出隐藏加省略号](http://caibaojian.com/multiline-text-with-ellipsis.html) |
| 应用2 | [使用VH和VW实现真正的流体排版](http://caibaojian.com/fluid-typography.html)  <http://caibaojian.com/fluid-typography.html> |
|  | 原文链接：<http://caibaojian.com/fluid-typography.html>  前言  不像[响应式](http://caibaojian.com/356.html" \o "响应式布局)布局，通过media query，设置几个变化点来适配，流体排版通过调整大小，适配所有设备宽度。这个方法可以使我们开发的网页，在几乎所有屏幕尺寸上都可以使用。但出于一些原因，它的使用率还远远没有[响应式](http://caibaojian.com/t/%e5%93%8d%e5%ba%94%e5%bc%8f" \o "View all posts in 响应式" \t "_blank)技术高。  在印刷的历史上，排版是根深蒂固的。关于“流体”的概念，在传统思想里并不存在。这是因为，在印刷上，尺寸大小都是有固定的，不用考虑在页面上使用。我认为流体排版技术可以和网页很好的匹配。这是在不同媒介上的一种解决方法。  并不意味着我们要推翻之前的所有关于排版的认识，只需要去学习如何用不同方式，去运用掌握新的技术。只要注意细节，就可以制作出适配所有屏幕尺寸的完美的页面。  流体排版上手  视口（viewport）单位，使流体排版在页面上应用变为可能。视口单位是根据浏览器的视口尺寸的百分比来定义的。[·](http://caibaojian.com/fluid-typography.html)  举个例子，1视口宽度（vw）等于视口宽度的1%，它不同于百分比的地方是，它的宽度是依赖于视口的宽度的，而百分比是元素的祖先元素来决定的。  视口单位，不同于其它单位，它不依赖于基础字体的大小。这种差别很重要，使它变得有意义而且独特。  可以使用的4种视口单位：  vw:视口宽度(viewport width)  vh:视口高度(viewport height)  vmin:视口宽度或高度，选择小的那个  vmax:视口宽度或高度，选择大的那个  使用流体排版最简单的方法，是把html元素的font-size，设置为一个流体单位：  html{ font-size: 2vw; }  这个例子中，我们把根元素的字体大小设置为2vw。这里我们已经修改了根元素字体大小。因为直接或间接使用的em或[rem](http://caibaojian.com/t/rem" \o "rem)单位，都是依赖于"root rem"的，所以也变成流体的了。例如  h1{  font-size:2em; }  h1的font-size:2em，如果它依赖的是根节点的字体大小，那么 font-size:2\*2vw=4vw。  只使用视口相关的单位也有一些不足。  无法精确地控制放大比率。  没有最大或最小字体大小。  和使用font-size相比，用户更容易使用像素来声明大小。  控制视口单位来设置最小最大font-size  html设置为font-size:2vw的方法，看起来既是流体的，调用起来也很方便，但有些场景没法覆盖。视口单位不是万能的，它也需要使用一些其它的方法，来解决无法覆盖的场景。由于视口单位都是依赖于视口的，在非常小的屏幕上，会得到很小的字体大小，已致无法查看。（chrome的最小字体是12px，好像可以解决了。）  理想情况下，我们可以通过设置最小字体来避免这种情况，但[CSS](http://caibaojian.com/css3/" \o "CSS)中没有min-font-size这个属性。通过一些横向思维，我们可以得到实现这种效果的方法。  首先，我们可以使用calc()表达式。  html{ font-size: calc(1em + 1vw); }  这样就算我们在一个0宽度的视口中，font-size的大小也会存在，并为1em。在大的屏幕上，1vw也会在最小字体1em的基础上增加。但是这种解决方法也不是最理想的。通常我们只是想在一些小的屏幕上去设置最小字体。我们可以使用media query来解决：  @media screen and (min-width: 50em) {  html {  font-size: 2vw;  }  }  上面代码设置，是仅在当视口宽度大于50em的时候，使用流体。虽然这样能很好的工作，但也表明会在固定值和流体值之间跳动。为了解决这个问题，我们可以计算出流体值和固定值之间的对应关系。  如果默认字体大小是16像素，并且2vw是2%的视口宽度，那么可以知道在800像素的视口时两个值是相等的。（16/(2/100)=800）  因为我们想在media query中，使用em单位来匹配。现在来换算一下像素到em。800像素除以16：800/16=50。我们也可以使用这个公式来算1/(2/100)=50。和上面例子里使用的变化条件一样，2vw对应的是50em。最终，我们得到了一个不会在固定和流体之间，产生跳动变化的值。  同样的可以用公式去算出最大字体。如果我们想使用的最大字体是24像素，那么对应的视口宽度就是24/(2/100)=1200px。换算成字体，就是1.5/(2/100)=75em。如果视口宽度大于75em时，我们就把字体大小设置成固定值。  @media screen and (min-width: 75em) {  html {  font-size: 1.5em;  }  }  这些计算都不是太难，下面做了一个表格，显示了各视口宽度，流体大小对应的字体大小。  IMG_256  通过这个表格，你可以获得一些控制视口单位变化。使用单独的视口单位，可以只看对应的一列中的字体大小。  控制缩放比率  在不使用media query的情况下，我们不可能解决，在400像素的屏幕上显示16像素的字体，在800像素显示24像素的问题。  还有，你可能会说，可以使用设置最大最小字体的方式，来解决这个问题。但这里不是像那样来处理的。  我们要怎么去解决这个问题呢？答案是使用[calc](https://www.smashingmagazine.com/2015/12/getting-started-css-calc-techniques/" \t "_blank)()。使用calc和视口单位配合，我们可以用高级的流体排版，可以精确地控制一定范围内的视口宽度使用特定大小的像素值。这里需要建立一个基本的数字函数：  IMG_257  这个函数可以能过一个范围内的一个值，可以计算出另一个对应范围的值。如果你取1-100的中的50，用它获取1-200的值 ，你将得到100。(其实我得到的是（1+(200-1)\*(50-1)/(100-1)=99.4949495))。这两个值都是各自范围的中间值。  这是一个求映射值的工具函数，在[JS](http://caibaojian.com/javascript/" \o "js)中处理数据时经常使用。当我认识到这只是一个纯粹的数学函数，我只需要一个变化的量，我使用calc()来运行这个函数。这里的变化关键值是视口大小：视口大小就是一个变量。100vw是一个变化的值，它是根据视口的尺寸变化的。有了最终的设想，花了一段时间来获得了能工作的函数表达式。calc()函数以其独特的方式来处理不同的单位类型。如果你对这方面感兴趣，强烈推荐《 [W3C对单位类型和值规范](https://www.w3.org/TR/css3-values/" \t "_blank)》，不为别的，就为了你在同事之间沟通时，可以吹牛B。  计算看上去很复杂，其实很简单。我们选择最小和最大字体值，并且可以根据屏幕的尺寸来获取精确的字体大小。我们可以使用任何单位，包括em,rem,px。这里拿像素来举例，是因为可以解释起来更方便并更好理解，但正常工作中我是使用rem单位的。看你自己的喜好啦，但有一点，方程式里所有的值必须是单位统一的，而且你也要像上面的例子中一样，去除一些单位。  如果你不想输入上面的代码，有一些工具可以帮助你实现。比如：[SASS](http://www.sassmeister.com/gist/7f22e44ace49b5124eec" \t "_blank),[LESS](http://codepen.io/MadeByMike/pen/RWJyML" \t "_blank),[PostCSS](https://www.npmjs.com/package/postcss-responsive-type" \t "_blank)插件。这样你就可以轻松的工作了。  保持理想的阅读长度  在《The elements of typographic style》,Robert Bringhurst 提出一个合理的阅读长度 大约是45到75个字符的长度  Anything from 45 to 75 characters is widely regarded as a satisfactory length of line for a single-column page set in a serifed text face in a text size. The 66-character line (counting both letters and spaces) is widely regarded as ideal.  同样的规则可以直接运用到流体排版上，很多情况下，文件缩放实现一致的阅读长度是可能的。  在[响应式](http://caibaojian.com/t/%e5%93%8d%e5%ba%94%e5%bc%8f" \o "View all posts in 响应式" \t "_blank)方法里，我们在media query里设置不同的字体大小，调整容器的宽度，来保持正确的阅读长度。然而，使用流体排版，调整media query下的值方法已经行不通了。 只是设置容器的大小，使它的比率和字体相同。我们可以使用像上文提到的计算font-size的方法，使用 calc（）来计算width属性的值。这可以保持阅读长度-而且样式表也更容易阅读和维护了。  在非常小的屏幕维护理想的阅读长度是不可能的。这种情况下，我们选择把容器的宽度设置为智能移动设备的宽度。  实现模块化缩放  模块化缩放是指一组成比例的数值。看图可以更直观地看到效果：  每个标题都是下面标题的1.2倍大小。  不用的屏幕尺寸显示不同的绽放效果会更好。  IMG_258IMG_259IMG_259  在小屏幕的时候，标题大小显示得更统一。在大屏幕上有足够的空间可以显示更大的差别。我们可以使用排版技术来使在不同的模板缩放变化时更平滑。简单地在不同的屏幕上选择不的缩放比率，然后计算出每个的标题级别，不同的最小和最大的字体大小。  计算模块化的缩放，使用基础字体大小乘以一个期望的比率来得到一个大的值。得到小值是用除法。使用相同的方法去计算出每个缩放对应的值。  大值：  1em × 1.125em = 1.125em  1.125em × 1.125em = 1.266em  1.266em × 1.125em = 1.424em  1.424em × 1.125em = 1.602em  小值：  1em ÷ 1.125em = 0.889em  0.889em ÷ 1.125em = 0.790em  0.790em ÷ 1.125em = 0.702em  0.702em ÷ 1.125em = 0.624em  我们使用1.125做为我们最小缩放比率，1.250做为我们最大缩放比率。下面我们把这个缩放比率应用到每个级别的标题级别上。  最小缩放比：1.125  最大缩放比：1.250  IMG_260  让我们在[codepen上演示](http://codepen.io/MadeByMike/pen/VvwqvW" \t "_blank)一下流体模块缩放标题。你可以从Tim Brown的《[排版的意义](http://alistapart.com/article/more-meaningful-typography" \t "_blank)》是去了解更多的模块绽放信息。  如何去选择一个合适的缩放比率，推荐 Jeremy Church的网站[Type Scale](http://type-scale.com/" \t "_blank)，Tim brown、Scott Kellum的[Modular Scale](http://www.modularscale.com/" \t "_blank)网站  保持垂直比例  纵向排版有必要保持页面元素之间的空白一致。这方面信息可以读Espen Brunborg的《[CSS Baseline: The Good, the Bad and the Ugly](https://www.smashingmagazine.com/2012/12/css-baseline-the-good-the-bad-and-the-ugly/)》。  在我们的层上保持纵向排版，需要设置一个元素之间的垂直空白的比例。找到一个在大屏幕和小屏幕上都适配的垂直排版，是很有挑战性的。我们通常希望找出大屏和小屏的基本度量，或使用不同的比例。  在流体排版中，基线也像字体大小一样是流体的。事实上，你给根元素设置一个流体的值，你可以使用em或rem单位。你也可以使用calc()来计算。  我们设置baseline为1.5rem，然后把body的padding设置为一个单位的baseline。  body {  padding: 1.5rem;  }  可以同样来设置line-height和margin  p {  line-height: 1.5rem;  margin-bottom: 1.5rem;  }  给标题元素设置不同的line-height。我想让line-height加margin值等于baseline的增量。这里可以使用calc()来计算。  h1 {  font-size: 2rem;  line-height: 2rem;  margin-top: calc((1.5rem - 2rem) + 1.5rem);  }  这个例子里，标题元素的height加上margin值等于3rem，正好是baseline值的两倍。  这里用sass变量创建[一个例子](http://codepen.io/MadeByMike/pen/0dbc9b82769c3e2a0cbfc3192bcaabd3?editors=0100)，方便大家理解。  未来，使用自定义css属性，可以对这项技术进行更多的扩展。我们能在calc()表达式中，使用css变量，计算出对应media query中的baseline基值。    约束条件  当你想在WordPress或Bootstrap，这些已经有预定义的布局中使用，或在已经存在布局的网站上使用，流体排版时。容器有可能不是流体的，或它们不是根据字体大小的比率来改变的。  最近，帮助澳大利亚政府的民众部，实现了流体排版。这是一个很大的网站，而且我必须根据已有的设计进行制作。在这种大流量的网站上实现流体布局，我有两个主要方面的问题。  第一问题，怎样阻止布局被破坏？在这个项目中，我预期导航容器里的内容，会因为字体大小和窗口的改变的比例不同而溢出。但很神奇的是，并没有出现这个问题。  实际情况，正好相反。之前需要单独设置来调整的地方，内容很自然的适配了它的容器。总之，样式表中需要尽量少的media query，大多的组件需要更少的样式声明。  还有一个好处是，我们可以不用单独为平板设备去制作一个单独的版本，因为这已经可以很好地在小屏幕上工作了。  下面是在过去需要去为每种不同屏幕尺寸调整一个版本的导航组件。虽然在小屏幕下表现还是不太完美，但可以不破坏它的整体性和功能性。  IMG_261  第二个问题，因为里面有很多的固定宽度，所以无法保持每行的字符都有理解的阅读长度。我在手机和平板电脑上，对字体大小进行了调整，使其保持在理想的阅读长度--移动设备阅读更重要。  主容器和文本的缩放比是不同的，当浏览器改变大小时，一些文本会重新排列。因为我不想在设备改变方向时，导致阅读处置改变。这是无法避免的情况，我们只能去测试一下主要设备的影响情况。还好，在移动设备上重排的影响并不是太大。  最后，我们对结果满意。毫无疑问，可以做更多关于排版的改进，我们已经提高了网站的可读性。随着流体排版的经验越来越多，在下一个设计迭代中，主要的焦点会集中在排版和可读性上。    在现有网站上，使用流体排版技术的几点提示：  根据你的设计来，确定最适合你的字体的最大最小值及缩放比。  font-size使用em单位。如果你想在一个部分容器上（如导航），使用流体排版，给容器设置一个固定值。在这个容器中的em值，都是依赖于固定值的。  同样的，如果你只是把主体容器设置成流体排版。如果你把一个块容器的字体设置为流体的，那么所有这个容器中的em值都是流体的。  文本重排，如果只是在浏览器改变大小时，或设备改变方向时发生，不用过于在意。  维护一个好的阅读长度是可能的。优先去考虑，手机或平板设备的理想阅读长度。    浏览器支持情况和BUGS  经常听到在Safari和特别是移动版的Safari，在同时使用视口单位和calc()表达式时，会出现明显的bug。但都没有提出什么具体的问题是什么，所以这里做了一些测试。  单独测试了calc()表达式和视口单位，以及同时使用它们来实现流体排版技术。在使用些技术时，没有发现什么问题。  在现代浏览器中，calc()和视口单位都能很好的工作。  在Safari 8以下和ie 11以下浏览器，当窗口中浏览器改变大小时，calc()表达式中的视口单位，不会重新计算。  可以通过media query来加以修正。  /\* Older browsers \*/  html { font-size: 16px; }    /\* Modern browsers only need this one \*/    @media screen and (min-width: 25em){  html { font-size: calc( 16px + (24 - 16) \* (100vw - 400px) / (800 - 400) ); }  }    /\* Safari <8 and IE <11 \*/  @media screen and (min-width: 25em){  html { font-size: calc( 16px + (24 - 16) \* (100vw - 400px) / (800 - 400) ); }  }  @media screen and (min-width: 50em){  html { font-size: calc( 16px + (24 - 16) \* (100vw - 400px) / (800 - 400) ); }  }  设置了Safary 8以下和ie 11以下，通过media query来重新计算字体大小。  老的浏览器则能过js来监听窗口的改变事件来重新计算字体大小。  可以在caniuse和查看支持情况  IMG_262  IMG_263    使用流体排版  如果要使用流体排版，要先想清楚要使用哪种解决方案。  如果整个设计都是流体的，可以考虑使用rem来定义流体，可以声明html字体大小为一个流体单位。可以使用em和rem来定义所有设计部分。  小心选择你的最大最小字体。关于这点，你需要决定是否直接使用视口单位或对更加精确地缩放比率。如果是后者，使用Sass,LESS或PostCSS插件中的函数，可以更简单实现。  确保获得了正确的字体的最大最小值。这是问题的关键。一旦你选定了根元素的字体大小，其它的所有组件都是依赖这个值来计算的。项目中如果后面要调整这个值，那么整个项目都要调整。  不要忘记，在使用流体排版之前定义一个默认的字体大小。默认字体大小，用来在那些不支持流体字体大小的浏览器上使用的，这个值不需要和字体最小值 相同。  最后，考虑你的设计上的限制，如何去解决像标题级别和阅读长度的问题。可以参与文中的处理方法。如果你的标题组件，想和常规的文本，以不同的比率进行缩放，在添加calc()表达式之前，先考虑可读性和样式代码的可维护性。  我希望这可以给你启发，可以让你思考在下个项目中那里可以用到流体排版。  原文：http://www.cnblogs.com/wengxuesong/archive/2016/05/16/5497653.html  推荐文章  [认识视口单位vm、vh在网页中的排版应用](http://caibaojian.com/viewport-based-typography.html)  [CSS3 calc() 会计算的属性](http://caibaojian.com/calc.html)  [纯CSS3使用vw和vh视口单位实现自适应](http://caibaojian.com/vw-vh.html)  [CSS预处理器框架与CSS后后处理器框架](http://caibaojian.com/css-post-processor.html)  [CSS3 calc实现滚动条出现页面不跳动](http://caibaojian.com/css3-calc-vw.html) |

CSS字体大小: em与px、pt、百分比之间的对比

|  |  |
| --- | --- |
| 简介 | CSS样式最混乱的一个方面是应用程序中文本扩展的font-size属性。在CSS中,你可以用四个不同的单位度量来显示在web浏览器中的文本 大小。这四个单位哪一种最适合Web? 这个问题引起了广泛的争论。找到一个确定的答案是困难的, 因为这个问题,本身就是如此难以回答。  接触这些单位  1.  “Ems”(em):“em”是一个可伸缩的单位, 用于web文档媒体展示。一个em等于当前的字体大小,例如,如果文档的字体大小是12 pt,1 em等于12 pt。Ems在本质上是可伸缩的,所以2 em相当于24 pt，.5 em相当于6 pt等。ems由于其可伸缩性和适应移动设备的特性在web文档中正变得越来越受欢迎。  2.  像素(px):像素是固定大小的单元,用于屏幕媒体(即在电脑屏幕上读取)。一个像素等于电脑屏幕上的一个点 (是你屏幕分辨率的最小分割)。许多网页设计师在web文档使用像素单位以生产浏览器渲染的像素完美呈现的网站。像素单元的一个问题是,它没有为视障读者的扩展,以适应移动设备。  3.  点(pt):点通常用于印刷媒体(任何打印在纸上的媒体,等等)。一个点等于一英寸的1/72。点更像像素,他们是固定大小的单位,不能伸缩。  4. 百分比(%):百分比单位更像“em”单位,除了一些根本性的差异。首先,当前的字体大小等于100%(比如12 pt = 100%)。当使用百分比单位,你的文字在移动设备上仍然保持完全的可伸缩性和可访问性。  字体大小单位之间的区别  当你动态地观察他们就很容易理解字体大小单位之间的区别。  一般来说,1 em = 12 pt = 16 px = 100%。  当使用这些font-size, 增加基础本字体大小(使用CSS选择器)从100%到120%，让我们看看会发生什么。  IMG_256  正如你所看到的, 随着基本字体大小增加em和百分比单位变大,但px 和 pt 没有变化。为你的文本设置一个绝对的大小很容易,但它是更容易在你的访客使用可伸缩的文本,可以显示任何设备或机器上。出于这个原因,em和百分比单位是web文档文本的首选。  Em和percent的对比  我们定义点和像素单位不一定是最适合web文档,留下来是的em和百分比单位。在理论上,em和百分比单位是相同的,但在应用程序中,他们实际上有一些细微的差别需要着重考虑。  在上面的示例中,我们使用百分比单位作为我们的基础字体大小(body上标记)。如果你把你的字体大小从百分比变为ems(即body{font- size:1 em;}),你可能不会注意到有什么差别。让我们看看当“1 em”是我们的body字体大小,当客户端改变浏览器的“文字大小”设置(这在一些浏览器中可用,如Internet Explorer)会发生什么。  IMG_257  当客户端的浏览器文本大小设置为“medium”, ems和百分比之间没有区别。然而,当设置改变,差别是相当大的。“Smallest”设置,ems比百分比小得多,而当设置为“Largest”,这时 恰恰相反,ems比百分比显示地更大。有些可能会说,当真正去扩展时em单位在扩展,在实际的应用程序中,em文本尺度变化太大, 在一些客户端机器上最小的文本变得不是很清晰。  结论  理论上,em单位是网上新的和即将到来的字体大小标准,但在实践中,百分比单位似乎给用户提供一个更加一致的和可访问的显示。当客户端设置改变,百分比文本以合理的比例扩展,允许[设计师](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=0&jk=295e2486b7518971&k=%C9%E8%BC%C6%CA%A6&k0=%C9%E8%BC%C6%CA%A6&kdi0=0&luki=2&n=10&p=baidu&q=06011078_cpr&rb=0&rs=1&seller_id=1&sid=718951b786245e29&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1922429&u=http://www.admin10000.com/document/4753.html&urlid=0" \t "http://www.admin10000.com/document/_blank)保持可读性,可访问性和[视觉设计](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=0&jk=295e2486b7518971&k=%CA%D3%BE%F5%C9%E8%BC%C6&k0=%CA%D3%BE%F5%C9%E8%BC%C6&kdi0=0&luki=3&n=10&p=baidu&q=06011078_cpr&rb=0&rs=1&seller_id=1&sid=718951b786245e29&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1922429&u=http://www.admin10000.com/document/4753.html&urlid=0" \t "http://www.admin10000.com/document/_blank)。  赢家:百分比(%)。  附录(2011年1月)  从我写这篇文章已经有几年了,我想总结那段时间所发生的讨论。一般来说,当我开始一个新的[设计](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=0&jk=295e2486b7518971&k=%C9%E8%BC%C6&k0=%C9%E8%BC%C6&kdi0=0&luki=5&n=10&p=baidu&q=06011078_cpr&rb=0&rs=1&seller_id=1&sid=718951b786245e29&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1922429&u=http://www.admin10000.com/document/4753.html&urlid=0" \t "http://www.admin10000.com/document/_blank), 我将在body元素上使用百分比(body { font-size: 62.5%; })，然后使用em单位大小来比较。只要body使用百分比单位设置,您可以选择使用百分比或ems或其他任何CSS规则和选择器并且在你的基础字体大小 上保持使用百分比的好处。在过去的几年中,这确实成为设计的标准。  像素现在被认为是可接受的字体大小单位(用户可以使用浏览器的“缩放”功能读小文本),虽然由于移动设备非常高的密度屏幕(一些Android和 iPhone设备每英寸超过200到 300个像素,让你11 - 12-pixel字体很难看到!)他们也开始引起一些问题。因此,我将继续使用百分比作为web文档中的基础字体大小。一如既往,鼓励和欢迎讨论和想法， 感谢在过去的两年里关于这个问题的所有评论！ |
| px em | px和em的战争  px和em都是长度单位，区别是，px的值是固定的，指定是多少就是多少，计算比较容易。em得值不是固定的，并且em会继承父级元素的字体大小。  任意浏览器的默认字体高都是16px。所以未经调整的浏览器都符合: 1em=16px。那么12px=0.75em, 10px=0.625em。  为了简化计算，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。  <style>  body{Font-size:62.5%}  .div\_px{font-size:10px; border:1px red solid;width:120px; height:20px}  .div\_em{font-size:1em; border:0.1em red solid;width:12em; height:2em}  </style>  <div class="div\_px" >1、这里是div\_px</div><br/>  <div class="div\_em" >2、这里是div\_em</div><br/>  效果一样  em会继承父级元素的字体大小  <style>  body {  Font-size: 62.5%  }  .div\_out {  font-size: 1em;/\* 1em 2em \*/  border: 1px red solid;  width: 10em;  height: 10em  }  .div\_in {  font-size: 1em;/\* 1em 2em \*/  border: 1px blue solid;  width: 10em;  height: 10em  }  </style>  <div class="div\_out">  3、这是div\_out  <div class="div\_in">  4、这是div\_in  </div>  </div>  <br/><br/><br/>  <div class="div\_in">  5、这是在外面的div\_in  </div>  这里有3个div，3和4是套在一起的，5是作对比的。 |
| rem em | Use `rem` for Global Sizing; Use `em` for Local Sizing  <style>  html {  font-size: 16px;  @media screen and (min-width: 900px) {  font-size: 18px;  }  @media screen and (min-width: 1200px) {  font-size: 20px;  }  }  h2 {  font-size: 2em;  }  pre {  font-size: 0.8em;  }  .module {  font-size: 1.1rem;  }  .another-module {  font-size: 1.3rem;  }  </style> |