# 专题七: 移动端响应式布局技巧



大纲目录:

- 设备像素、设备独立像素、CSS像素、PPI、devicePixelRatio
- layout viewport 与 visual viewport
- viewport缩放适配
- 媒体查询@media
- vw弹性适配
- 动态rem适配
- 弹性flex适配

## 设备像素、设备独立像素、CSS像素、PPI、devicePixelRatio

设备像素（物理像素 / 像素分辨率）

- 显示器的最小物理单位（对于一个显示器来说是固定的）
- 以手机屏幕为例，iphonex像素分辨率为1125x2436，是指屏幕横向能显示1125个物理像素点，纵向能显示2436个物理像素点。
- 通常说的4K显示屏指的是 4096x2160

设备独立像素（dips）

- 比如我们偶尔会说“电脑屏幕在 2560x1600分辨率下不适合玩游戏，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。
- 可在控制台通过 window.screen.width/ window.screen.height 查看。
- 另外，平时我们所说的 iphoneX的逻辑分辨率375 x 812指的就是设备独立像素。chrome检查元素模拟调试手机设备时显示如375x667和 320x480都是设备独立像素。
- 一个设备独立像素可能包含多个物理像素，包含的越多，显示越清晰

CSS像素

- 在页面不缩放的情况下，1px的CSS像素 === 1设备独立像素
- 页面放大200%时，页面的设备独立像素依旧不变，放大的是CSS像素。但是此时CSS像素与设备独立像素的关系变化了，1px === 4独立像素（宽x2，高x2）

PPI

- 指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。（屏幕对角线上的像素点 / 对角线的英寸数）

devicePixelRatio

- 像素比window.devicePixelRatio
- devicePixelRatio指的是物理像素和设备独立像素的比，即1独立像素由多少物理像素渲染。
- dpr（device pixel ratio）：设备像素比，设备像素/设备独立像素，代表设备独立像素到设备像素的转换关系，在JS中可以通过 window.devicePixelRatio 获取；
- window.devicePixelRatio = 物理像素 / 设备独立像素（dips）。
- 经计算， iphonex的 devicePixelRatio 是3。

高清屏部分图片失真：

- 一些像素比较低的图片，在普通显示屏上可以显示，但在高清屏上会出现模糊的现象
- 原因是：假如有一张图片，设置宽高为100px，在不同屏幕上，呈现的都是100px设备独立像素的图片，但对于高清屏来说，100px独立像素所需的物理像素比普通屏多得多
- 1px独立像素所含的物理像素越多，屏幕越高清； 假如普通屏100px独立像素需要1W个像素点，高清屏得3W个，但是图片本身包含的像素点可能远远达不到3W，这时候，图片就会拉伸自己的像素点，所以看起来就显得模糊了。
- 解决办法是：高清屏上图片的宽高设小一点，这样所需的物理像素就不用那么多了，屏幕显示图片所需的物理像素越接近图片，图片越高清

矢量图永不失真

- 因为矢量图形不是一个个像素点显色的，而是通过给定的坐标数据进行绘制的，所以不会失真。

## layout viewport 与 visual viewport

https://www.html.cn/archives/5975
layout viewport（布局视口）
visual viewport（视觉视口）和物理像素
ideal viewport（理想视口）和 dip （设备逻辑像素）

## viewport缩放适配

屏幕的尺寸：window.screen.width  // 指设备独立像素值

浏览器窗口尺寸： window.innerWidth 、window.innerHeight // 指的是CSS像素
注：innerWidth innerHeight不包括滚动条的宽度高度， 精确计量用
document.documentElement.clientWidth和document.documentElement.clientHeight

| 属性名 | 取值 | 描述 |
| --- | --- | --- |
| width | 正整数 或?device-width | 定义视口的宽度，单位为像素 |
| height | 正整数 或?device-height | 定义视口的高度，单位为像素，一般不用 |
| initial-scale | [0.0-10.0] | 定义初始缩放值 |
| minimum-scale | [0.0-10.0] | 定义缩小最小比例，它必须小于或等于maximum-scale设置 |
| maximum-scale | [0.0-10.0] | 定义放大最大比例，它必须大于或等于minimum-scale设置 |
| user-scalable | yes/no | 定义是否允许用户手动缩放页面，默认值yes |

移动前端中常说的 viewport （视口）就是浏览器显示页面内容的屏幕区域;

- width: 控制 viewport 的大小，可以指定的一个值，如 600，或者特殊的值，如 device-width 为设备的宽度（单位为缩放为 100% 时的 CSS 的像素）。
- height: 和width相对应，指定高度。
- initial-scale：初始缩放比例，也即是当页面第一次 load 的时候缩放比例。
- maximum-scale：允许用户缩放到的最大比例。
- minimum-scale：允许用户缩放到的最小比例。
- user-scalable：用户是否可以手动缩放

```
// 快捷键生成：meta:vp tab
<meta id="viewport" name="viewport" content="width=device-width; initial-scale=1.0;
maximum-scale=1.0; user-scalable=no;">
```

## 媒体查询@media

语法: @media 媒体类型 逻辑操作符（媒体属性）｛样式代码｝

逻辑操作符

- and： 操作符用来把多个媒体属性组合起来，合并到同一条媒体查询中。只有当每个属性都为真时，这条查询的结果才为真； @media all and （min-width:700px）and （orientation: lanscape）{...}
- not：操作符用来对一条媒体查询的结果进行取反;
  @media not all and （monochrome){...} <=> @media not (all and (monochrome)){...}
- only：操作符表示仅在媒体查询匹配成功时应用指定样式。可以通过它让选中的样式在老式浏览器中不被应用； media = "only screen and（max-width：1000px)" {...}

媒体属性

- width | min-width | max-width
- height | min-height | max-height
- device-width | min-device-width | max-device-width
- device-height | min-device-height | max-device-height
- aspect-ratio | min-aspect-ratio | max-aspect-ratio
- device-aspect-ratio | min-device-aspect-ratio | max-device-aspect-ratio
- color | min-color | max-color
- color-index | min-color-index | max-color-index
- monochrome | min-monochrome | max-monochrome
- resolution | min-resolution | max-resolution
- scan | grid

横竖屏

- @media (orientation: portrait) { 竖屏 }
- @media (orientation: landscape) { 横屏 }

## vw弹性适配

- 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。

## 动态rem适配

- 相对长度单位。相对于根元素(即html元素)font-size计算值的倍数;
- 根元素html默认的font-size为16px;
- 为了方便计算，我们一般给父元素的font-size设置为100px;
- 移动端适配 rem & vw 计算工具 http://www.jq22.com/demo/jqueryremvw201812251323/

```
// 针对750的设计稿
<script type="text/javascript">
function refreshRem() {
  var desW = 750,
  winW = document.documentElement.clientWidth,
  ratio = winW / desW;
  document.documentElement.style.fontSize = ratio * 100 + 'px';
}
refreshRem();
```

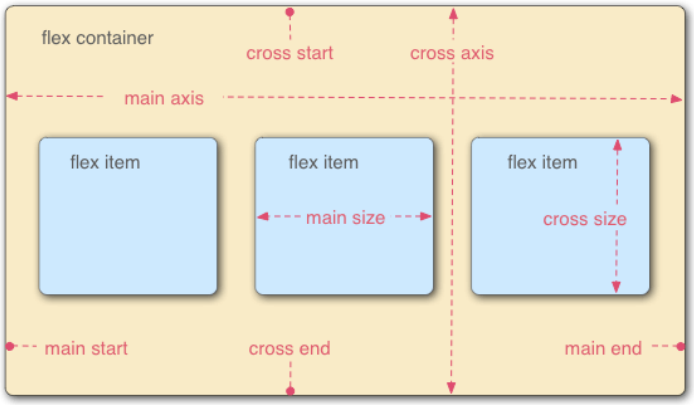
```
10 window.addEventListener('resize', refreshRem);
11 </script>
```

## 弹性flex适配

> http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html
> 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。flex 布局有两个值：
> - display:flex; 代表的是块级
> - display:inline-flex; 代表的是行内块

**flex布局的基本概念**

采用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成员，称为Flex项目（flex item），简称”项目”。




容器默认存在两根轴：水平的主轴（main axis）和垂直的交叉轴（cross axis）。主轴的开始位置（与边框的交叉点）叫做main start，结束位置叫做main end；交叉轴的开始位置叫做cross start，结束位置叫做cross end。

**容器的属性**

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

flex-direction

设置主轴的方向

- row：主轴的方向是水平，从从左到右
- column：主轴的方向是垂直的，从上到下
- row-reverse：主轴的方向是水平，从右到左

- column-reverse：主轴的方向是垂直的，从下到上

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Document</title>
<style>
    *{
        margin:0;padding:0;
    }
    ul,ol{
        list-style: none;
    }
    ul{
        display:flex;
        /* flex-direction: row; */
        /* flex-direction: column; */
        /* flex-direction: row-reverse; */
        flex-direction: column-reverse;
    }
    ul>li{
        width:100px;
        height:100px;
        background:green;
        margin-left:10px;
    }
</style>
</head>
<body>
<ul>
<li>1</li>
<li>2</li>
```

```
<li>3</li>
<li>4</li>
<li>5</li>
<li>6</li>
</ul>
</body>
</html>
```

## flex-wrap

- wrap: 换行
- nowrap: 不换行（默认）
- wrap-reverse：换行，不过第一行在最下面


## flex-flow

flex-flow属性是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap。

flex-flow:row wrap;写了这个之后，下面的案例效果跟上一个一样

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.
0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>flex-flow</title>
<style>
    *{
        margin:0;padding:0;
    }
```

```
    /*
      flex-flow 是flex-direction 和flex-wrap 的简写
     */
    ul,ol{
        list-style: none;
    }
    ul{
        display:flex;
        flex-flow:row wrap
    }
    ul>li{
        width:400px;
        height:400px;
        background:green;
        margin-left:10px;
        margin-bottom:10px;
    }
</style>
</head>
<body>
<ul>
<li>1</li>
<li>2</li>
<li>3</li>
<li>4</li>
<li>5</li>
<li>6</li>
<li>7</li>
</ul>
</body>
</html>
```

justify-content

属性定义了项目在主轴上的对齐方式。

justify-content: flex-start | flex-end | center | space-between | space-around| space-evenly;





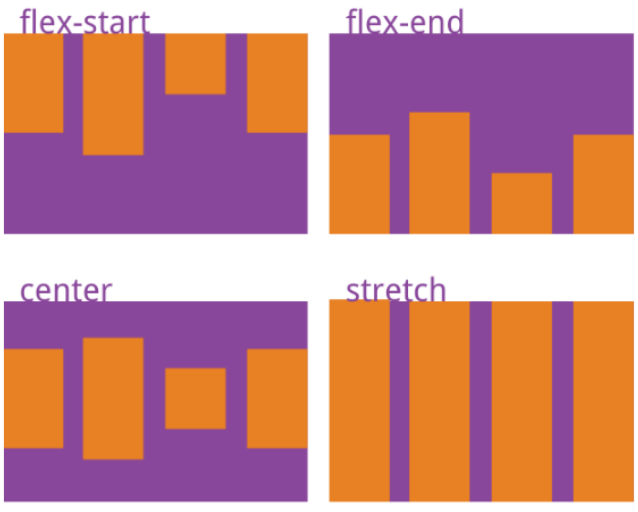

space-evenly

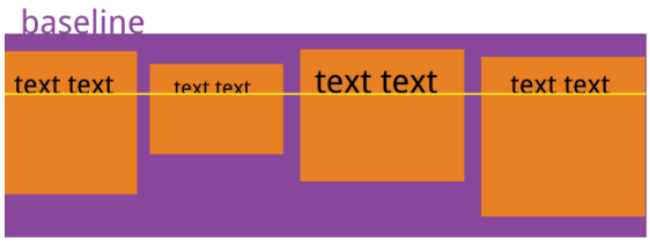
## align-items

align-items属性定义项目在交叉轴上如何对齐。

align-items: flex-start | flex-end | center | baseline | stretch;

stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度。

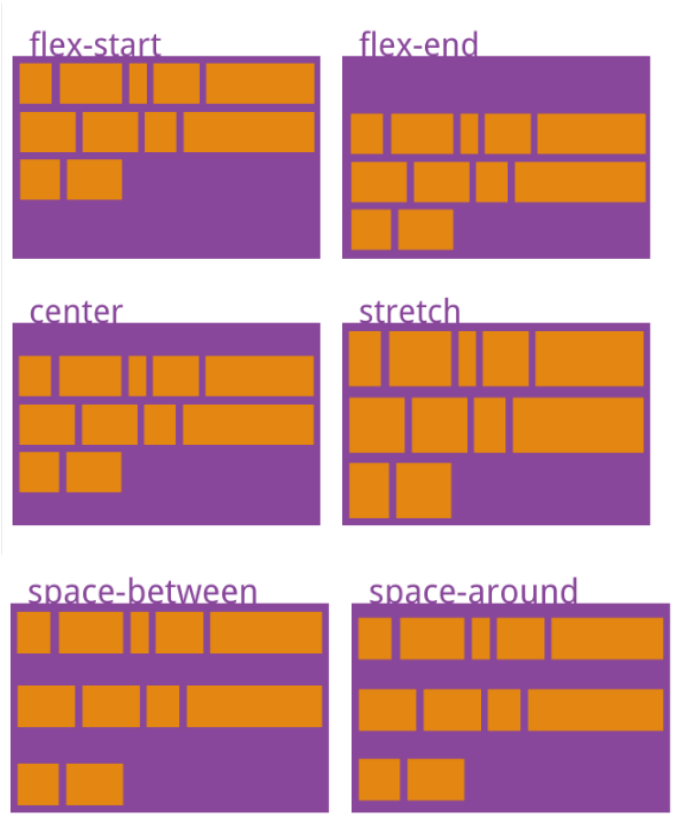



## align-content

align-content属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。（也就是说得有项目换行）

```
.box {
    align-content: flex-start | flex-end | center | space-between | space-around | stretch;
}
```




**项目的属性**

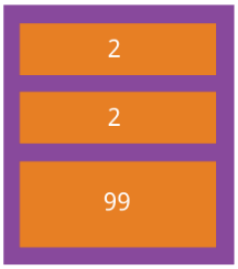
以下6个属性设置在项目上。

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

order

属性定义项目的排列顺序。数值越小，排列越靠前，默认为0。



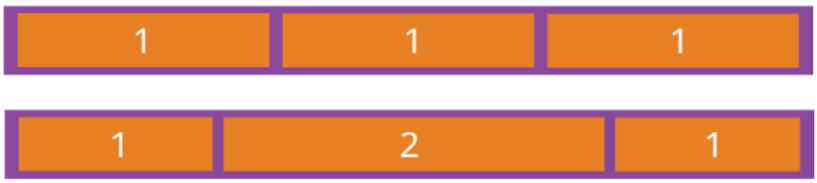



### flex-grow

flex-grow属性定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大。

如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。

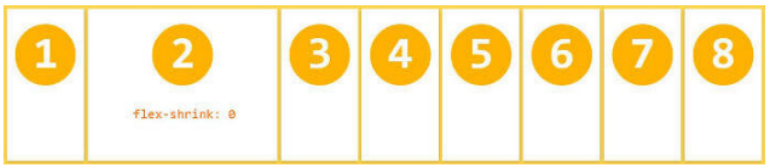
```
.item {
    flex-grow: <number>; /* default 0 */
}
```




### flex-shrink

flex-shrink属性定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小。

```
.item {
flex-shrink: <number>; /* default 1 */
}
```




如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小。如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小。

负值对该属性无效。

### flex-basis

flex-basis属性定义了在分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。它可以设为跟width或height属性一样的值（比如350px），则项目将占据固定空间。

```
.item {
flex-basis: <length> | auto; /* default auto */
}
```

flex

flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。

flex：0 1 auto；默认值

flex：none； 代表的意思等同于 flex：0 0 auto；

flex：auto；代表的意思是flex：1 1 auto；

flex：number；当flex取值为一个非负数字，则该数字为flex-grow的值，flex-shrink的值为1，flex-basis的值为0%；

```
.item {
    flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

align-self

align-self属性允许单个项目有与其他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。

```
.item {
align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```