**9.9**

flex布局

flex布局可以简便、完整、响应式地实现各种页面布局

在实现一些特殊布局如垂直居中等时flex布局的CSS代码更加简洁，而且一般性能也相对不错

实现flex布局，需要把容器指定为弹性容器，这样的容器的元素即为弹性元素

弹性容器与弹性元素是基于相对关系的概念，所以说弹性容器可以同时为弹性元素，只是参照的不同

设置flex布局，可以将display属性设置为flex，或是inline-flex，前一种更加广泛

flex布局主轴（水平方向）和辅轴（垂直方向）

一般元素按主轴方向

弹性容器属性

**flex-direction**属性

决定主轴的方向，亦即弹性元素方向

* row 水平方向 自左向右 **默认值**
* row-reverse 水平方向 自右向左
* column 垂直方向 自上向下
* column-reverse 垂直方向 自下而上

所以默认为上或左方向，reverse使之反向

**flex-wrap**属性

确定主轴上元素换行方式

* nowrap 不换行 **默认值**
* wrap 换行
* wrap-reverse 换行，且首行在末尾

**flex-flow** 属性

flex-direction和flex-wrap属性的简写

先flex-direction后flex-wrap

**justify-content**属性

设置如何处理弹性容器剩余空白空间

* flex-start 元素自起边排列，即元素集中于元素左侧，元素间不分配空白空间 **默认值**
* flex-end 元素自终边排列，除排列起始方向外类似flex-start
* center 元素居中，同样元素间不分配空白空间
* space-around 元素间与元素&容器边沿间距分配空白空间，且元素间是元素&容器边沿2倍，同类型间距相对
* space-between 仅元素间分配空白空间，间距相等，元素&容器边沿不分配空白空间
* space-evenly 元素间与元素&容器边沿间距分配空白空间，元素间与元素&容器边沿相等

**align-items**属性

设置元素在辅轴上的对齐方式

* stretch 拉伸一行元素使各元素高度相同 **默认值**
* flex-start 沿着辅轴起点对齐
* flex-end 沿着辅轴终点对齐
* center 垂直居中对齐
* baseline 沿第一行文字基线对齐

**align-content** 属性

* flex-start 元素自起点排列，即元素集中于元素上侧，元素间不分配空白空间 **默认值**
* flex-end 元素自终点排列，除排列起始方向外类似flex-start
* center 元素居中，同样元素间不分配空白空间
* space-around 元素间分配空白空间，且元素间是元素&容器边沿2倍，同类型间距相对
* space-between 仅元素间分配空白空间，间距相等
* stretch 拉伸同一行元素使各元素高度相同

弹性元素属性

**flex-grow**属性

指定弹性元素伸展系数

* + 0 不可 默认值
  + 非0 按比例分配

**flex-shrink**属性

指定弹性元素收缩系数

* 1 默认值
* 非0值 按收缩系数收缩
* 0 不收缩

**flex-basis**属性

定义元素在主轴上基础长度

auto自动取决于元素本身width或者height

具体数值则按该数值

**flex**属性

flex-grow、flex-shrink、flex-basis 的简写

顺序要求是flex-grow flex-shrink flex-basis

* + - * 数值 设置方法与上述相同
      * initial 和 flex: 0 1 auto 相同
      * auto 和 flex: 1 1 auto 相同
      * none 和 flex: 0 0 auto 相同

**order**属性

定义弹性元素排列顺序

数值越小，排列越前

**align-self**属性

允许单个弹性元素不同的对齐方式

默认继承被包含的弹性容器align-items属性

取值类似于弹性容器align-items属性

flex：0 1 auto

按比例1收缩，不伸展，未设置基础长度

效果同initial

若空间不足，这个弹性元素可收缩；但空间多余时，该弹性元素不可扩展

flex布局其他一些杂七杂八事情

常见布局类型

* 静态布局（Static Layout）

不管浏览器具体尺寸是多少，网页布局始终按照最初写代码时候的布局来显示

尺寸使用px作为单位

* 流式布局（Liquid Layout）

页面元素的宽度按照屏幕分辨率进行适配的调整，整体布局不变

主要的划分区域的尺寸使用百分数%

* 自适应布局（Adaptive Layout）

分别为不同的屏幕分辨率定义布局，使用@media媒体查询给不同尺寸和介质的设备切换不同的样式

* 响应式布局（Responsive Layout）

创建多个流式布局，分别对应一个屏幕分辨率范围。可以把响应式布局看作是流式布局和自适应布局设计理念的融合

综合响应式、流动等上述技术通过CSS给单一网页不同设备返回不同样式

* 弹性布局（rem/em布局）

正是上面的flex布局

尺寸使用em/rem作为单位

flex布局很大一部分程度上更加灵活，因为它让元素尺寸可以一定程度上按浏览器或页面尺寸来调整，适应性更广

定位目的是把元素依据需要而实现摆到对应的位置的效果

定位通过position属性控制和实现

无 static 默认值

绝对定位 absolute 相对第一个非static定位元素定位

相对定位 relative 相对正常位置定位

固定定位 fixed 固定位置不随页面翻动而移动，相对浏览器窗口定位

粘性定位 sticky 基于用户滚动的位置定位，之前像absolute，之后像relative

可以用于实现一些特殊效果，比如学校网站，上下滚动时页面图片和左右导航条明显地不会随页面上下移动而移动；包括导航条，但可能这是JavaScript实现的应该

**9·11**

substr start开始位置，length截取字符串长度【可不用】 允许负数

substring start开始位置，stop结束位置后一位【可不用】 禁止负数

slice start开始位置，end结束位置 允许负数

这几个都是字符串方法，用于实现裁剪字符串

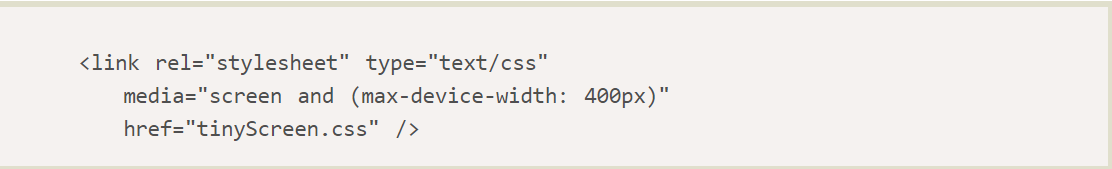
主要是调用的参数的区别，实际上一些场合下可以互相替代

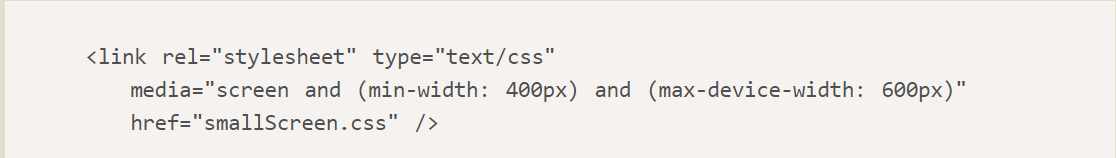
若允许负数，就是指自后而前

响应式布局

目的正是为了同时有效地保证多个终端的兼容

* <meta name="viewport" content="width=device-width, initial-scale=1" />代码实现自适应网页设计，即令网页宽度默认等于屏幕宽度（width=device-width），原始缩放比例（initial-scale=1）为1.0，即网页初始大小占屏幕面积的100%
* 不使用绝对定位
* 不直接指定宽度为px，可%或auto
* 元素大小和字体大小亦使用相对定位
* 使用流动布局即float属性
* 自动探测网页大小，并加载对应css文件

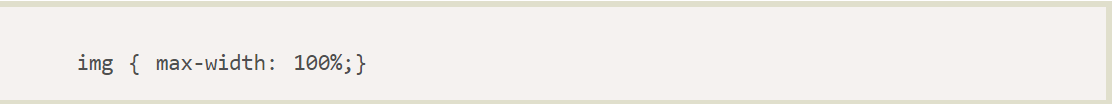


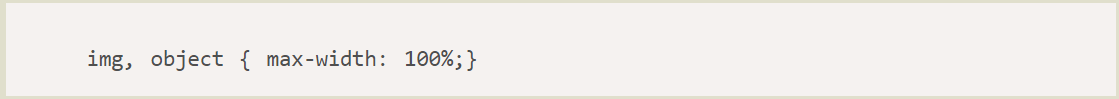


* css文件中按网页大小的调整



* 图片按网页大小的调整





参考csdn一篇文章