一峰大牛 flex.css

Flex 更高效的布局方案。

传统的布局方案，基于盒装模型， display + position + float 属性。对于特殊布局来说不方便， 例如，垂直居中。

Flex布局，简便完整，响应式地实现各种网页布局。

Flex： flexible box。 弹性布局，用来为盒装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。

.box{

display: flex;

}

行内元素也可以使用flex布局：

.box{

display: inline-flex;

}

兼容性适配： -webkit前缀

.box{

display: -webkit-flex; /\* Safari \*/

display: flex;

}

Attention 设置为flex布局后，子元素的float，clear和vertical-align属性将失效。

**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，成为flex item。



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。

项目默认沿着主轴排列，单个项目占据的主轴空间叫做main size，占据的交叉轴空间叫做cross size。

**容器的属性：**

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

**flex-direction 属性：**

flex-direction决定主轴的方向，即项目的排列方向。

.box {

flex-direction: row | row-reverse | column | column-reverse;

}



四个可选值：

* row（默认值）：主轴为水平方向，起点在左端。
* row-reverse：主轴为水平方向，起点在右端。
* column：主轴为垂直方向，起点在上沿。
* column-reverse：主轴为垂直方向，起点在下沿。

**flex-wrap属性：**

flex-wrap属性定义，如果在一条轴线放不下所有元素的情况下如何换行。

.box{

flex-wrap: nowrap | wrap | wrap-reverse;

}

nowrap （默认），不换行



wrap：换行，第一行在上方。



wrap-reverse: 换行，第一行在下方。



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

.box {

flex-flow: <flex-direction> || <flex-wrap>;

}

justify-content属性

justify-content 属性定义了项目在主轴上的对齐方式。

.box {

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

}



如上图所示，一共有五种可能取值：

* flex-start（默认值）：左对齐
* flex-end：右对齐
* center： 居中
* space-between：两端对齐，项目之间的间隔都相等。
* space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

**align-items属性：**

定义项目在交叉轴上如何对齐。

.box {

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

}



如上图所示有5种可能取值，具体的效果与交叉轴的方向有关。在交叉轴从上到下的情况下：

* flex-start：交叉轴的起点对齐。
* flex-end：交叉轴的终点对齐。
* center：交叉轴的中点对齐。
* baseline: 项目的第一行文字的基线对齐。
* 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属性：**

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，该属性无效。

.box {

align-content: flex-start | flex-end | center | space-between | space-around | stretch;

}

* flex-start：与交叉轴的起点对齐。
* flex-end：与交叉轴的终点对齐。
* center：与交叉轴的中点对齐。
* space-between：与交叉轴两端对齐，轴线之间的间隔平均分布。
* space-around：每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。
* stretch（默认值）：轴线占满整个交叉轴。

**项目的属性：**

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

以上六条设置在项目上：

**order属性:**

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，允许负值。

.item {

order: <integer>;

}



**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（如果存在的话）。如果一个item的flex-grow属性为2，其他为1，则前者占据的剩余空间将比其他项多一倍。



**flex-shrink属性：**

定义了item的缩小比例，默认为1，即如果空间不足，item自动缩小。

.item {

flex-shrink: <number>; /\* default 1 \*/

}

如果所有item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item为0，其他项目都为1，则空间不足时，前者不缩小。

负值无效。

数值越大缩小倍数越大。

**flex-basis属性：**

flex-basis属性定义了在分配多余空间之前，item占据的主轴空间（main size）。浏览器根据这个属性来计算主轴是否有多余空间。默认值为auto，即item的本来大小。

.item {

flex-basis: <length> | auto; /\* default auto \*/

}

当它设置为和height或者width属性一样的值，项目将占据固定空间。

**flex属性：**

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

**0 不放大** **1 空间不足时等比例缩小** auto **item的本来大小**

.item {

flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]

}

两个快捷值

auto (1 1 auto) 同等放大 同等缩小 本身大小

none (0 0 auto) 不放大 不缩小 本身大小

建议优先使用这个属性而不是写三个分离的属性，因为浏览器会根据这个属性进行推算。

**align-self属性：**

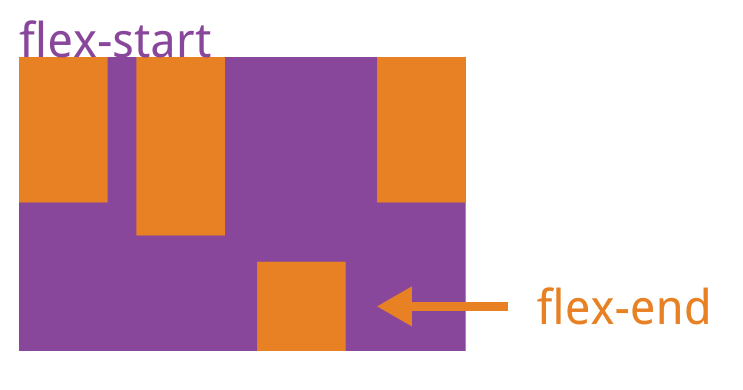
align-self 属性允许单个item 有与其他item不一样的对齐方式，可覆盖align-items属性。

默认值auto，表示继承父元素的align-items属性，如果没有父元素，等同于stretch。

.item {

align-self: auto | flex-start | flex-end | center | baseline | stretch;

}



该属性可能取六个值，除了auto，其他的和align-items属性完全一致。