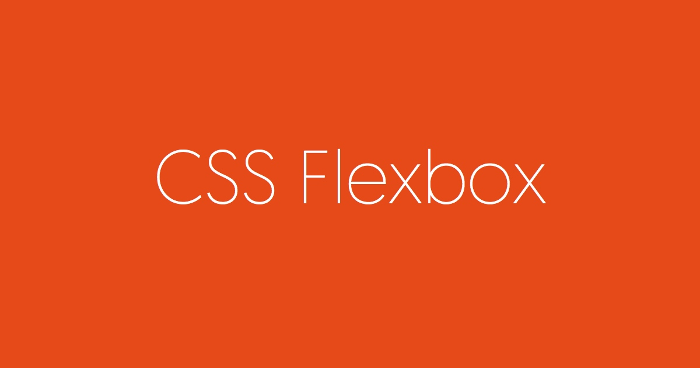
布局的传统解决方案，基于[盒状模型](https://developer.mozilla.org/en-US/docs/Web/CSS/box_model)，依赖 [display](https://developer.mozilla.org/en-US/docs/Web/CSS/display)属性 + [position](https://developer.mozilla.org/en-US/docs/Web/CSS/position)属性 + [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float)属性。它对于那些特殊布局非常不方便，比如，[垂直居中](https://css-tricks.com/centering-css-complete-guide/)就不容易实现。



2009年，W3C提出了一种新的方案—-Flex布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。



Flex布局将成为未来布局的首选方案。

**一、Flex布局是什么？**

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。

.box{

display: flex;

}

行内元素也可以使用Flex布局。

.box{

display: inline-flex;

}

Webkit内核的浏览器，必须加上-webkit前缀。

.box{

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

display: flex;

}

注意，设为Flex布局以后，子元素的float、clear和vertical-align属性将失效。

**二、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。

项目默认沿主轴排列。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。

**三、容器的属性**

以下6个属性设置在容器上。

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

**3.1 flex-direction属性**

flex-direction属性决定主轴的方向（即项目的排列方向）。

.box {

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

}

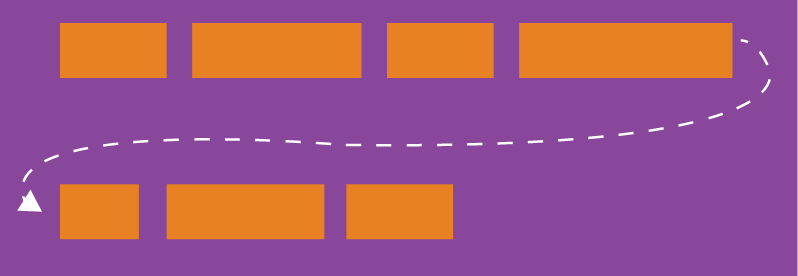


它可能有4个值。

* 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。

### 3.2 flex-wrap属性

默认情况下，项目都排在一条线（又称”轴线”）上。flex-wrap属性定义，如果一条轴线排不下，如何换行。



.box{

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

}



（2）wrap：换行，第一行在上方。



（3）wrap-reverse：换行，第一行在下方。



**3.3 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。

.box {

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

}

**3.4 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;

}



它可能取5个值，具体对齐方式与轴的方向有关。下面假设主轴为从左到右。

* 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。

### 3.5 align-items属性

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;

}



它可能取5个值。具体的对齐方式与交叉轴的方向有关，下面假设交叉轴从上到下。

* 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。

### 3.6 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，该属性不起作用。

.box {

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

}



该属性可能取6个值。

* 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。

### 四、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

### 4.1 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。

* .item {
* order: <integer>;
* }

