

写给自己看的display: flex布局教程

这篇文章发布于 2018年10月28日，星期日，22:01，归类于 [CSS相关](#)。阅读 11962 次, 今日 70 次 [26 条评论](#)

by zhangxinxu from <https://www.zhangxinxu.com/wordpress/?p=8063>

本文可全文转载，但需得到原作者书面许可，同时保留原作者和出处，摘要引流则随意。

//zxx: 本教程所有布局效果（含交互）为实时渲染，若布局异常，可以[点击这里](#)访问原文。

一、前言&索引

给div这类块状元素设置 `display:flex` 或者给span这类内联元素设置 `display:inline-flex`，flex布局即创建！其中，直接设置 `display:flex` 或者 `display:inline-flex` 的元素称为flex容器，里面的子元素称为flex子项。

//zxx: flex和inline-flex区别在于，inline-flex容器为inline特性，因此可以和图片文字一行显示；flex容器保持块状特性，宽度默认100%，不和内联元素一行显示。

而Flex布局相关属性正好分为两拨，一拨作用在flex容器上，还有一拨作用在flex子项上。具体参见下表，点击可快速索引。

作用在flex容器上
作用在flex子项上
<ul style="list-style-type: none">• <code>flex-direction</code>• <code>flex-wrap</code>• <code>flex-flow</code>• <code>justify-content</code>• <code>align-items</code>• <code>align-content</code>
<ul style="list-style-type: none">• <code>order</code>• <code>flex-grow</code>• <code>flex-shrink</code>• <code>flex-basis</code>• <code>flex</code>• <code>align-self</code>

无论作用在flex容器上，还是作用在flex子项，都是控制的flex子项的呈现，只是前者控制的是整体，后者控制的是个体。

其他说明：

- 本教程所有案例HTML结构为：

```
container(flex容器)
  div(flex子项) > img
  div(flex子项) > img
  div(flex子项) > img
```

同时，为了便于区分，flex容器区域使用虚框标示，flex子项增加了白蓝径向渐变背景色，图片上显示了原始序号。

- Flex布局中还有主轴和交叉轴的概念，为避免过多概念干扰，本教程省略相关措辞，而是使用水平方向和垂直方向代替^①。

//zxx: ① writing-mode属性可以改变文档流方向，此时主轴是垂直方向，但实际开发很少遇到这样场景，因此，初学的时候，直接使用水平方向和垂直方向理解不会有任何问题，反而易于理解。

二、作用在flex容器上的CSS属性

1. flex-direction

`flex-direction` 用来控制子项整体布局方向，是从左往右还是从右往左，是从上往下还是从下往上。和CSS的`direction`属性相比就是多了个 `flex`。

语法如下：

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

其中：

row

默认值，显示为行。方向为当前文档水平流方向，默认情况下是从左往右。如果当前水平文档流方向是 `rtl`（如设置 `direction: rtl`），则从右往左。

row-reverse

显示为行。但方向和 `row` 属性值是反的。

column

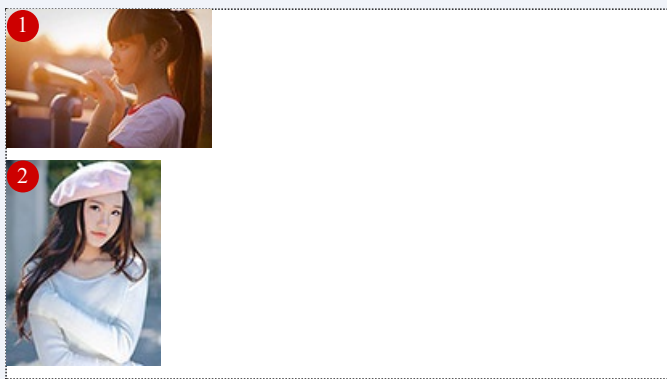
显示为列。

column-reverse

显示为列。但方向和 `column` 属性值是反的。

眼见为实，点击下面对应单选项，可以看到实时的布局效果：

☒ row ☐ row-reverse ☐ column ☐ column-reverse



2. flex-wrap

`flex-wrap` 用来控制子项整体单行显示还是换行显示，如果换行，则下面一行是否反方向显示。这个属性比较好记忆，在CSS世界中，只要看到单词wrap一定是与换行显示相关的，`word-wrap` 属性或者 `white-space: nowrap` 或者 `pre-wrap` 之类。

语法如下：

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

其中：

nowrap

默认值，表示单行显示，不换行。于是很容易出现宽度溢出的场景，其渲染表现比较复杂，需要对CSS3宽度有一定了解，可以阅读“[理解CSS3 max/min-content及fit-content等width值](#)”这篇文章。具体表现如下（以水平布局举例）：

- flex子项最小内容宽度 `min-content` 之和大于flex容器宽度，则内容溢出，表现和 `white-space: nowrap` 类似。
- 如果flex子项最小内容宽度 `min-content` 之和小于flex容器宽度，则：
 - flex子项默认的 `fit-content` 宽度之和大于flex容器宽度，则flex子项宽度收缩，正好填满flex容器，内容不溢出。
 - flex子项默认的 `fit-content` 宽度之和小于flex容器宽度，则flex子项以 `fit-content` 宽度正常显示，内容不溢出。

在下面案例中，示意的图片默认有设置 `max-width:100%`，flex子项div没有设置宽度，因此，flex子项最小宽度是无限小，表现为图片宽度收缩显示。如果我们取消 `max-width:100%` 样式，则此时flex子项最小宽度就是图片宽度，就可以看到图片溢出到了flex容器之外。

wrap

宽度不足换行显示。


wrap-reverse

宽度不足换行显示，但是是从下往上开始，也就是原本换行在下面的子项现在跑到上面。

眼见为实，点击下面对应单复选框，可以看到实时的布局效果：

☒ `img{max-width:100%;}`

☒ nowrap ☐ wrap ☐ wrap-reverse



3. flex-flow

`flex-flow` 属性是 `flex-direction` 和 `flex-wrap` 的缩写，表示flex布局的flow流动特性，语法如下：

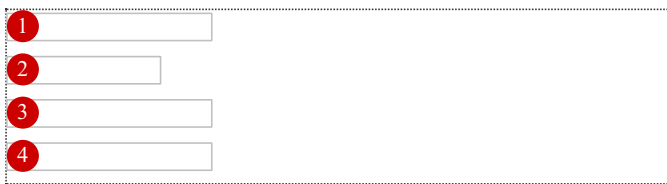
```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

当多属性同时使用的时候，使用空格分隔。

举个例子，容器元素如下设置：

```
.container {  
  display: flex;  
  flex-flow: row-reverse wrap-reverse;  
}
```

实时效果如下：



可以看到水平排序从右往左（`row-reverse` 属性值的作用），以及换行的那一行在上面（`wrap-reverse` 属性值的作用）。

4. justify-content

`justify-content` 属性决定了水平方向子项的对齐和分布方式。CSS `text-align` 有个属性值为 `justify`，可实现两端对齐，所以，当我们想要控制flex元素的水平对齐方式的时候，记住 `justify` 这个单词，`justify-content` 属性也就记住了。

`justify-content` 可以看成是 `text-align` 的远房亲戚，不过前者控制flex元素的水平对齐外加分布，后者控制内联元素的水平对齐。

语法如下：

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

其中：

`flex-start`

默认值。逻辑CSS属性值，与文档流方向相关。默认表现为左对齐。

`flex-end`

逻辑CSS属性值，与文档流方向相关。默认表现为右对齐。

`center`

表现为居中对齐。

`space-between`

表现为两端对齐。between是中间的意思，意思是多余的空白间距只在元素中间区域分配。使用抽象图形示意如下：



space-around

around是环绕的意思，意思是每个flex子项两侧都环绕互不干扰的等宽的空白间距，最终视觉上边缘两侧的空白只有中间空白宽度一半。使用抽象图形示意如下：



space-evenly

evenly是匀称、平等的意思。也就是视觉上，每个flex子项两侧空白间距完全相等。使用抽象图形示意如下：



眼见为实，点击下面对应单复选框，可以看到实时的布局效果：



5. align-items

`align-items` 中的 `items` 指的就是flex子项们，因此 `align-items` 指的就是flex子项们相对于flex容器在垂直方向上的对齐方式，大家是一起顶部对齐呢，底部对齐呢，还是拉伸对齐呢，类似这样。

语法如下：

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

其中：

stretch

默认值。flex子项拉伸。在演示中我们可以看到白蓝径向渐变背景区域是上下贯穿flex容器的，就是因为flex子项的高度拉伸到容器高度导致。如果flex子项设置了高度，则按照设置的高度值渲染，而非拉伸。

flex-start

逻辑CSS属性值，与文档流方向相关。默认表现为容器顶部对齐。

flex-end

逻辑CSS属性值，与文档流方向相关。默认表现为容器底部对齐。

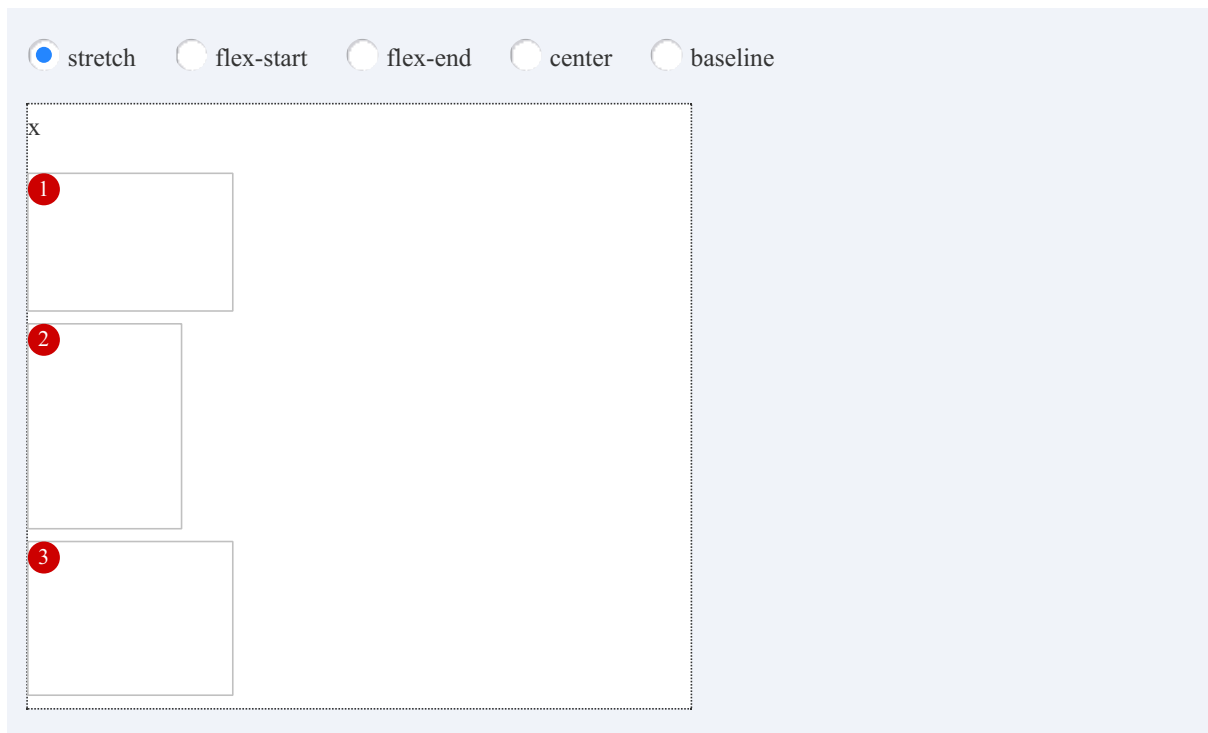
center

表现为垂直居中对齐。

baseline

表现为所有flex子项都相对于flex容器的基线（字母x的下边缘）对齐。

眼见为实，点击下面对应单选框，可以看到实时的布局效果：



6. align-content

`align-content` 可以看成和 `justify-content` 是相似且对立的属性，`justify-content` 指明水平方向flex子项的对齐和分布方式，而 `align-content` 则是指明垂直方向每一行flex元素的对齐和分布方式。如果所有flex子项只有一行，则 `align-content` 属性是没有任何效果的。

语法如下：

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

其中：

stretch

默认值。每一行flex子元素都等比例拉伸。例如，如果共两行flex子元素，则每一行拉伸高度是50%。

flex-start

逻辑CSS属性值，与文档流方向相关。默认表现为顶部堆砌。

flex-end

逻辑CSS属性值，与文档流方向相关。默认表现为底部堆放。

center

表现为整体垂直居中对齐。

space-between

表现为上下两行两端对齐。剩下每一行元素等分剩余空间。

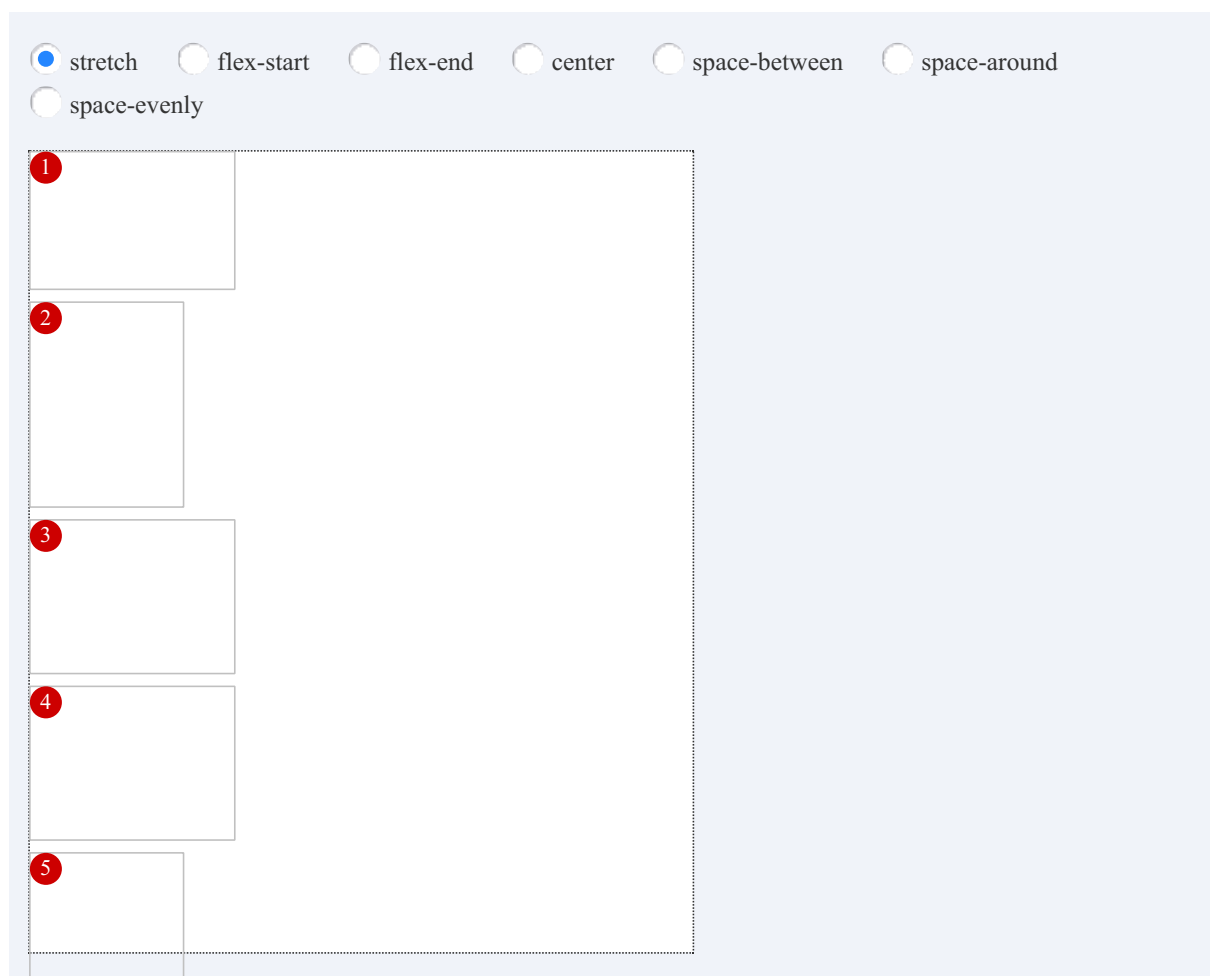
space-around

每一行元素上下都享有独立不重叠的空白空间。

space-evenly

每一行元素都完全上下等分。

眼见为实，我们给flex容器设置高度500像素，然后点击下面对应单选框，可以看到实时的布局效果：



三、作用在flex子项上的CSS属性

6 1. order

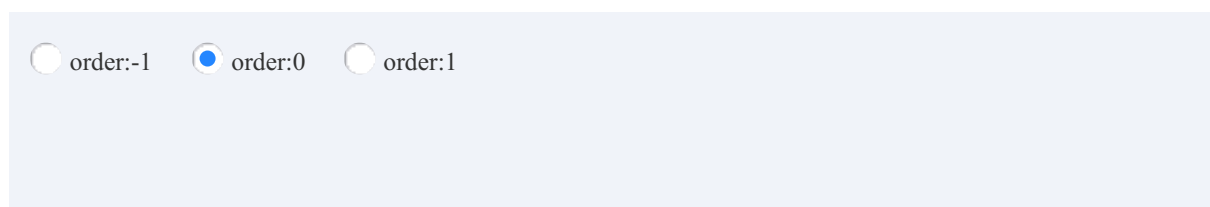
我们可以通过设置 `order` 改变某一个flex子项的排序位置。

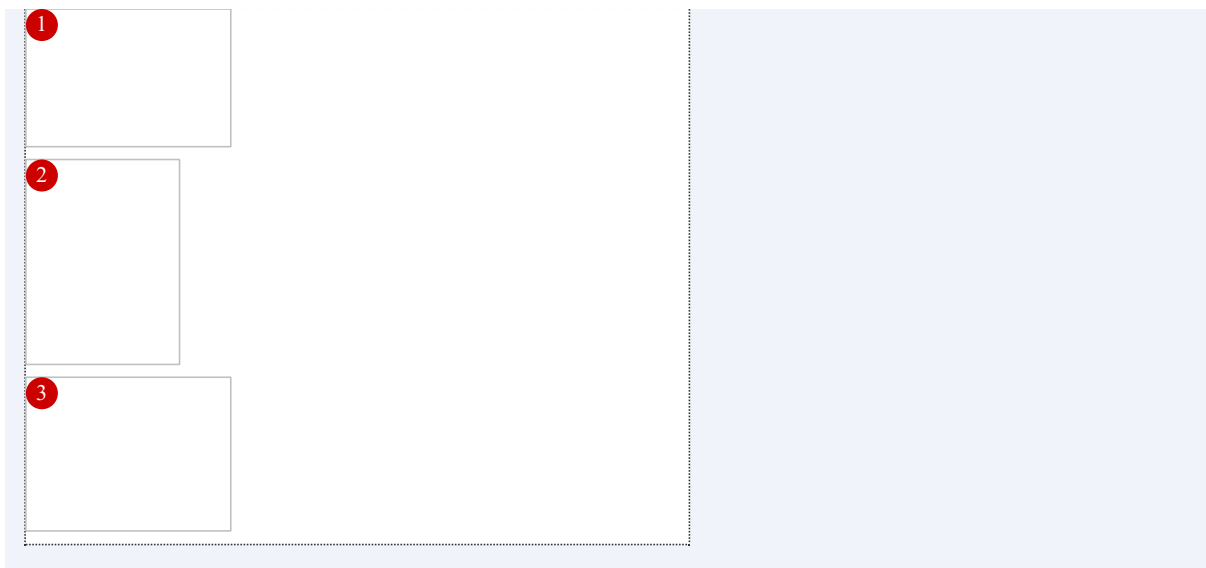
7 用法：

```
order: <integer>; /* 整数值，默认值是 0 */
```

所有flex子项的默认 `order` 属性值是0，因此，如果我们想要某一个flex子项在最前面显示，可以设置比0小的整数，如 `-1` 就可以了。

眼见为实，下面flex容器有3个子元素，现在，我们给第2个子元素设置 `order` 属性值，看看其排列位置有何变化。点击下面的单选框，可以看到实时的交互效果：





2. flex-grow

`flex-grow` 属性中的grow是扩展的意思，扩展的就是flex子项所占据的宽度，扩展所侵占的空间就是除去元素外的剩余的空白间隙。

具体的扩展比较复杂。在展开之前，我们先看下语法。

语法：

```
flex-grow: <number>; /* 数值, 可以是小数, 默认值是 0 */
```

`flex-grow` 不支持负值，默认值是0，表示不占用剩余的空白间隙扩展自己的宽度。如果 `flex-grow` 大于0，则flex容器剩余空间的分配就会发生，具体规则如下：

- 所有剩余空间总量是1。
- 如果只有一个flex子项设置了 `flex-grow` 属性值：
 - 如果 `flex-grow` 值小于1，则扩展的空间就总剩余空间和这个比例的计算值。
 - 如果 `flex-grow` 值大于1，则独享所有剩余空间。

具体可参见下面“grow案例1”。

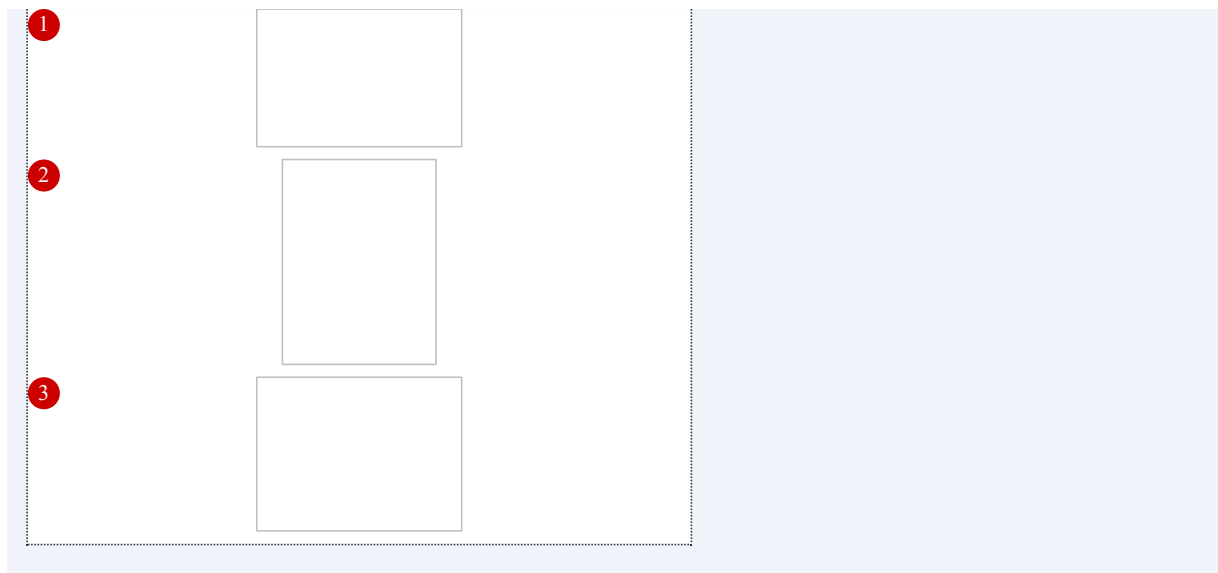
- 如果有多个flex设置了 `flex-grow` 属性值：
 - 如果 `flex-grow` 值总和小于1，则每个子项扩展的空间就总剩余空间和当前元素设置的 `flex-grow` 比例的计算值。
 - 如果 `flex-grow` 值总和大于1，则所有剩余空间被利用，分配比例就是 `flex-grow` 属性值的比例。例如所有的flex子项都设置 `flex-grow:1`，则表示剩余空白间隙大家等分，如果设置的 `flex-grow` 比例是1:2:1，则中间的flex子项占据一半的空白间隙，剩下的前后两个元素等分。

具体可参见下面“grow案例2”。

grow案例1：

flex容器有3个子元素，现在，我们仅给第2个子元素设置 `flex-grow` 属性值，看看其占据尺寸有何变化。点击下面的单选框，可以看到实时的交互效果：

☒ flex-grow:0 ☐ flex-grow:0.5 ☐ flex-grow:1 ☐ flex-grow:2



此实例演示中，仅一个flex子项设置了 `flex-grow` 属性值，当我们选择 `0.5` 的时候，值小于1，剩余空间用不完，因此，扩展的宽度是总剩余宽度是0.5，也就是一半；当我们选择 `1` 的时候，正好所有空间都使用；当我们选择 `2` 的时候，效果一样，因为没有其他参与分配的子项，因此渲染表现和 `1` 一样。

grow案例2:

flex容器有3个子元素，默认所有子项都设置了 `flex-grow:0.25`，现在我们点击下面的单选框，改变第2个子元素的 `flex-grow` 属性值，看看其占据尺寸有何变化：



此实例演示中，因为3个子项都是0.25，因此默认还剩余25%的剩余空间；如果我们选择 `flex-grow:0`，则加起来的 `flex-grow` 是 `0.5`，因此剩余50%空间；如果我们选择 `flex-grow:0.5`，则加起来的 `flex-grow` 是 `1`，因此没有剩余空间，同时空间占用比例为1:2:1，最终效果符合此预期；如果我们选择 `flex-grow:1`，则加起来的 `flex-grow` 大于 `1`，剩余空间按比例分配，为1:4:1，最终效果也确实如此。

以上就是 `flex-grow` 属性的作用规则。

3. flex-shrink

shrink是“收缩”的意思， `flex-shrink` 主要处理当flex容器空间不足时候，单个元素的收缩比例。

语法如下：

```
flex-shrink: <number>; /* 数值, 默认值是 1 */
```

`flex-shrink` 不支持负值，默认值是1，也就是默认所有的flex子项都会收缩。如果设置为0，则表示不收缩，保持原始的 `fit-content` 宽度。

`flex-shrink` 的内核跟 `flex-grow` 很神似，`flex-grow` 是空间足够时候如何利用空间，`flex-shrink` 则是空间不足时候如何收缩腾出空间。总有点CP的味道。

两者的规则也是类似。已知flex子项不换行，且容器空间不足，不足的空间就是“完全收缩的尺寸”：

- 如果只有一个flex子项设置了 `flex-shrink` :
 - `flex-shrink` 值小于1，则收缩的尺寸不完全，会有一部分内容溢出flex容器。
 - `flex-shrink` 值大于等于1，则收缩完全，正好填满flex容器。
- 如果多个flex子项设置了 `flex-shrink` :
 - `flex-shrink` 值的总和小于1，则收缩的尺寸不完全，每个元素收缩尺寸占“完全收缩的尺寸”的比例就是设置的 `flex-shrink` 的值。
 - `flex-shrink` 值的总和大于1，则收缩完全，每个元素收缩尺寸的比例和 `flex-shrink` 值的比例一样。下面案例演示的就是此场景。

眼见为实，flex容器有4个子元素，现在，我们给第2个子元素设置不同的 `flex-shrink` 属性值，看看其占据尺寸有何变化。点击下面的单选框，可以看到实时的交互效果：

☐ flex-shrink:0 ☐ flex-shrink:0.5 ☒ flex-shrink:1 ☐ flex-shrink:2



此实例演示中，因为4个子项都是1，和远大于1，因此，完全收缩，不会有内容溢出。如果我们选择 `flex-shrink:0`，则第2个flex子项不收缩，剩下3个flex子项等比例收缩；如果我们选择 `flex-shrink:1`，则4个子项1:1:1:1收缩；如果我们选择 `flex-shrink:2`，则完全收缩尺寸比例分配为1:2:1:1，第2个flex子项收缩的宽度最大，是其他元素的2倍。

以上就是 `flex-shrink` 属性的作用规则。

4. flex-basis

`flex-basis` 定义了再分配剩余空间之前元素的默认大小。相当于对浏览器提前告知：浏览器兄，我要占据这么大的空间，提前帮我预留好。

语法如下：

```
flex-basis: <length> | auto; /* 默认值是 auto */
```

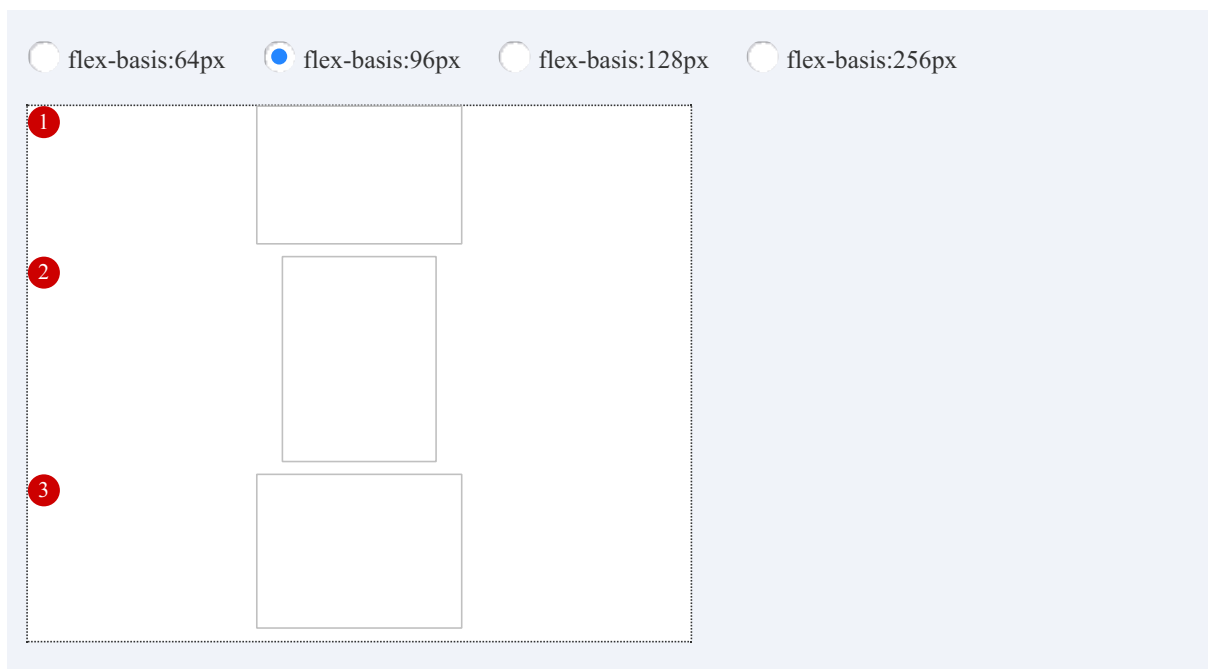
默认值是 `auto`，就是自动。有设置 `width` 则占据空间就是 `width`，没有设置就按内容宽度来。

如果同时设置 `width` 和 `flex-basis`，就渲染表现来看，会忽略 `width`。`flex` 顾名思义就是弹性的意思，因此，实际上不建议对 `flex` 子项使用 `width` 属性，因为不够弹性。

当剩余空间不足的时候，`flex` 子项的实际宽度并通常不是设置的 `flex-basis` 尺寸，因为 `flex` 布局剩余空间不足的时候默认会收缩。

实例一则：

`flex` 容器有3个子元素，现在，我们给第2个子元素设置不同的 `flex-basis` 属性值，看看其占据尺寸有何变化。点击下面的单选框，可以看到实时的交互效果：



选择最后一个 `flex-basis:256px` 会发现 `flex` 子项的宽度并不是 `256px`，这是因为此时剩余空间不足，3个子项1:1:1收缩的缘故。

5. flex

`flex` 属性是 `flex-grow`，`flex-shrink` 和 `flex-basis` 的缩写。

语法：

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

其中第2和第3个参数（`flex-shrink` 和 `flex-basis`）是可选的。默认值为 `0 1 auto`。

案例演示：

第2个 `flex` 子项设置 `flex:none` 或者 `flex:auto`，我们看看实时布局效果会有怎样的变化：

☐ 隐藏部分图片使空间剩余（为了测flex-grow）

☒ flex默认值 ☐ flex:none ☐ flex:auto



此时第2个flex子项的 `flex-grow`，`flex-shrink` 和 `flex-basis` 属性值分别是 `0`，`1`，和 `auto`

。

经过上面一番测试，我们可以得到如下结论：

- `flex` 默认值等同于 `flex:0 1 auto`；
- `flex:none` 等同于 `flex:0 0 auto`；
- `flex:auto` 等同于 `flex:1 1 auto`；

6. align-self

`align-self` 指控制单独某一个flex子项的垂直对齐方式，写在flex容器上的这个 `align-items` 属性，后面是items，有个s，表示子项们，是全体；这里是self，单独一个个体。其他区别不大，语法几乎一样：

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

唯一区别就是 `align-self` 多了个 `auto`（默认值），表示继承自flex容器的 `align-items` 属性值。其他属性值含义一模一样，如下案例示意：

首先我们设置flex容器 `baseline` 对齐，然后点击下面的单选框，给第2个flex子项设置不同 `align-self` 属性值，观察其表现：

```
.container {  
  display: flex;  
  align-items: baseline;  
  height: 240px;  
}
```

☒ auto ☐ flex-start ☐ flex-end ☐ center ☐ baseline ☐ stretch



四、其他Flex知识点

- 在Flex布局中，flex子元素的设置 `float`，`clear` 以及 `vertical-align` 属性都是没有用的。
- Flexbox布局最适合应用程序的组件和小规模布局（一维布局），而Grid布局则适用于更大规模的布局（二维布局），关Grid布局请参见“[写给自己看的display: grid布局教程](#)”一文。
- 已经8102年了，Flex老语法不用在管了，舒爽弃之，然后私有前缀也不用再加了，看到就烦。

本教程优点在于交互效果可以实时体验，更直观。如果是转载文章，必定没有效果，访问原文即可。

说实话，自己之前Flex布局用得很少，本文内容自己也是边学边写，文中若有表述不准确的地方欢迎指正。

感谢阅读！

参考文章：[A Complete Guide to Flexbox](#)

《CSS世界》签名版独家发售，包邮，可指定寄语，[点击显示购买码](#)

（本篇完） // 想要打赏？[点击这里](#)。有话要说？[点击这里](#)。



« [CSS margin-inline和margin-block区别是什么？](#)

[写给自己看的display: grid布局教程](#) »

猜你喜欢

- [写给自己看的display: grid布局教程](#)
- [粉丝群第1期CSS小测点评与答疑](#)
- [基于CSS3 column多栏布局实现水平滑页翻页交互](#)
- [CSS box-flex属性，然后弹性盒子模型简介](#)
- [关于文字内容溢出用点点点\(...\)省略号表示](#)
- [聊聊CSS世界中的margin-box](#)
- [深入CSS ::first-letter伪元素及其实例等](#)
- [CSS3 Media Queries的些野史外传](#)
- [我熟知的三种三栏网页宽度自适应布局方法](#)
- [css行高line-height的一些深入理解及应用](#)
- [CSS3 transition实现超酷图片墙动画效果](#)

分享到：

1

标签：[align-content](#), [align-items](#), [align-self](#), [box-flex](#), [css3](#), [flex](#), [flex-basis](#), [flex-direction](#), [flex-flow](#), [flex-grow](#), [flex-shrink](#), [flex-wrap](#), [justify-content](#), [order](#), [布局](#)

发表评论（目前26条评论）

<input type="text"/>	名称 (必须)
<input type="text"/>	邮件地址(不会被公开) (必须)
<input type="text"/>	网站
<div></div>	
<div>提交评论</div>	

1. 无下限的王五说道:

2019年01月1日 17:17



对3. flex-shrink的规则有几点不同的看法

- 1.不可能只给一个flex子项设置flex-shrink，其他flex子项会有flex-shrink的默认值，为1；可以把其他flex子项的flex-shrink设为0，再设置另一个flex子项的flex-shrink值。
- 2.当其他flex子项flex-shrink: 0;另一个flex子项的flex-shrink值大于等于1，不一定收缩完全，要考虑flex子项的宽度（尺寸），当溢出容器的空间大于flex子项的宽度时，flex子项会完全收缩，仍然会有空间溢出。
- 3.如果多个flex子项设置了flex-shrink，且flex-shrink值的总和大于1，也不一定完全收缩，当其中一个flex子项收缩完全时，有可能仍然会有空间溢出
- 4.每个元素收缩尺寸占“完全收缩的尺寸”的比例不是flex-shrink值，比例与flex子项自身的宽度（尺寸）有关，可以参考谢然大大的文章[回复](#)

2. zhyt说道:

2018年12月26日 17:49



学习了，非常感谢！

[回复](#)

3. hhhh说道:

2018年12月11日 17:29



很好，学习了

[回复](#)

4. 很好说道:

2018年11月16日 16:44



很好

[回复](#)

5. ranjing说道:

2018年11月15日 11:12



建议作者使用不常用属性的时候，备注一下兼容性问题，毕竟很多人可能看到了就拿去开发使用，结果发生手机上不支持，或者某些版本的谷歌都不支持。比如这里的justify-content : space-evenly，在除了Firefox Mobile (Gecko) 52.0 (52.0) 版本以上支持，其余浏览器，手机浏览器 都是No support

[回复](#)

6.

夜晚硬邦邦说道:
2018年11月14日 17:24
发现了个Flex布局的在线游戏<https://flexboxfroggy.com/>
回复

猫咪君-VRIie说道:
2018年11月30日 10:16
居然还有这种游戏!!!
回复

Ittplus说道:
2018年12月20日 16:37
赞!
回复

7.

外评网说道:
2018年11月5日 13:25
每当遇到pc项目，还是得老老实实用float、 position
回复

8.

阿桂说道:
2018年11月2日 23:02
完全看不懂
回复

9.

ub说道:
2018年11月2日 10:34
'justify-content'属性决定了水平方向子项的对齐和分布方式。`应该是主轴方向，flex-direction 属性决定主轴方向（默认row: 水平方向，column: 垂直方向）。
示例：<https://codepen.io/ubbcou/full/yRmMPN/>
回复

10.

许海琼说道:
2018年11月2日 09:50
一直看你的文章，受教很多，开启我的自学之路啊，
回复

11.

zx说道:
2018年11月1日 11:32
前段在线资源里的jquery怎么没有了 我经常用的 鑫哥放出来吧
回复

张鑫旭说道:
2018年11月1日 20:41
好
回复



12.

zc说道:

2018年11月1日 09:23

display:flex和display:inline-flex有什么区别吗? 我平时用都是统一设置display:flex, 很少用到display:inline-flex。

回复

jojo说道:

2018年11月8日 10:08

就跟block和inline-block一样的区别

回复
13.

Misaya_Q说道:

2018年10月31日 16:11

期待大佬写一篇关于Grid布局的文章

回复
14.

DeathGhost说道:

2018年10月30日 23:11

????????????哈哈, 写的比我详细多了, 我前几天也写了scroll-behavior与flex????????

回复
15.

icenzhao说道:

2018年10月30日 10:16

最近刚好在团队内部分享flex入门, 看看大佬的文章, 再看看自己准备的ppt, 那个汗啊 哈哈 看完文章感觉有学习了一遍

回复
16.

Beme说道:

2018年10月29日 21:04

现在一直用 flex 布局, 居然被人鄙视说不会 margin + float 布局。

我能怎么办?

回复
17.

nicholasurey说道:

2018年10月29日 17:07

真棒, 以前一直对flex敬而远之

回复
18.

meepo说道:

2018年10月29日 16:25

现在的项目需要兼容IE9, 用不了flex。

唉。

回复
19.

fe_bean说道:

2018年10月29日 13:04

头排沙发


回复

20.

网站建设说道:
2018年10月29日 11:29

很齐全


[回复](#)


21.

紫气东来的紫东说道:
2018年10月29日 10:16

受教了，有些点平常都没注意到 哈哈


[回复](#)



1说道:
2018年12月25日 18:02

44

[回复](#)


- #### 最新文章
- » [常见的CSS图形绘制合集](#)
 - » [粉丝群第1期CSS小测点评与答疑](#)
 - » [分享三个纯CSS实现26个英文字母的案例](#)
 - » [小tips: 纯CSS实现打字动画效果](#)
 - » [CSS/CSS3 box-decoration-break属性简介](#)
 - » [CSS :placeholder-shown伪类实现Material Design占位符交互效果](#)
 - » [从天猫某活动视频不必要的3次请求说起](#)
 - » [CSS vector-effect与SVG stroke描边缩放](#)
 - » [CSS ::backdrop伪元素是干嘛用的?](#)
 - » [周知: CSS -webkit-伪元素选择器不再导致整行无效](#)
- #### 今日热门
- » [常见的CSS图形绘制合集](#) ⁽¹⁷⁹⁾
 - » [粉丝群第1期CSS小测点评与答疑](#) ⁽¹¹³⁾
 - » [未来必热: SVG Sprite技术介绍](#) ⁽¹¹¹⁾
 - » [HTML5终极备忘大全 \(图片版+文字版\)](#) ⁽⁸⁵⁾
 - » [让所有浏览器支持HTML5 video视频标签](#) ⁽⁸³⁾
 - » [Selectivizr-让IE6~8支持CSS3伪类和属性选择器](#) ⁽⁸⁰⁾
 - » [CSS3下的147个颜色名称及对应颜色值](#) ⁽⁷⁸⁾
 - » [小tips: 纯CSS实现打字动画效果](#) ⁽⁷³⁾
 - » [写给自己看的display: flex布局教程](#) ⁽⁷⁰⁾
 - » [分享三个纯CSS实现26个英文字母的案例](#) ⁽⁷⁰⁾
- #### 今年热议
- » [《CSS世界》女主角诚寻靠谱一起奋斗之人](#) ⁽⁷⁶⁾
 - » [不借助Echarts等图形框架原生JS快速实现折线图效果](#) ⁽⁶⁴⁾
 - » [看, for..in和for..of在那里吵架!](#) ⁽⁶⁰⁾
 - » [是时候好好安利下LuLu UI框架了!](#) ⁽⁴⁷⁾
 - » [原来浏览器原生支持JS Base64编码解码](#) ⁽³⁵⁾
 - » [妙法攻略: 渐变虚框及边框滚动动画的纯CSS实现](#) ⁽³³⁾
 - » [炫酷H5中序列图片视频化播放的高性能实现](#) ⁽³¹⁾
 - » [CSS scroll-behavior和JS scrollIntoView让页面滚动平滑](#) ⁽³⁰⁾
 - » [windows系统下批量删除OS X系统.DS_Store文件](#) ⁽²⁶⁾
 - » [写给自己看的display: flex布局教程](#) ⁽²⁶⁾
- #### 猜你喜欢
- [写给自己看的display: grid布局教程](#)
 - [粉丝群第1期CSS小测点评与答疑](#)

- 基于CSS3 column多栏布局实现水平滑页翻页交互
- CSS box-flex属性，然后弹性盒子模型简介
- 关于文字内容溢出用点点点(...)省略号表示
- 聊聊CSS世界中的margin-box
- 深入CSS ::first-letter伪元素及其实例等
- CSS3 Media Queries的些野史外传
- 我熟知的三种三栏网页宽度自适应布局方法
- css行高line-height的一些深入理解及应用
- CSS3 transition实现超酷图片墙动画效果