弹性盒

Flexbox是一种用于按行或列排列项目的一维布局方法。项目*弯曲*（扩展）以填充额外的空间或收缩以适应更小的空间。本文解释了所有基础知识。

先决条件：	HTML 基础知识（学习 HTML 简介），以及 CSS 工作原理的概念（学习 CSS 简介。）
客观的：	学习如何使用 Flexbox 布局系统来创建网页布局。

为什么选择 Flexbox?

长期以来，可用于创建 CSS 布局的唯一可靠的跨浏览器兼容工具是floats和positioning等功能。这些有效，但在某些方面它们也有限制和令人沮丧。

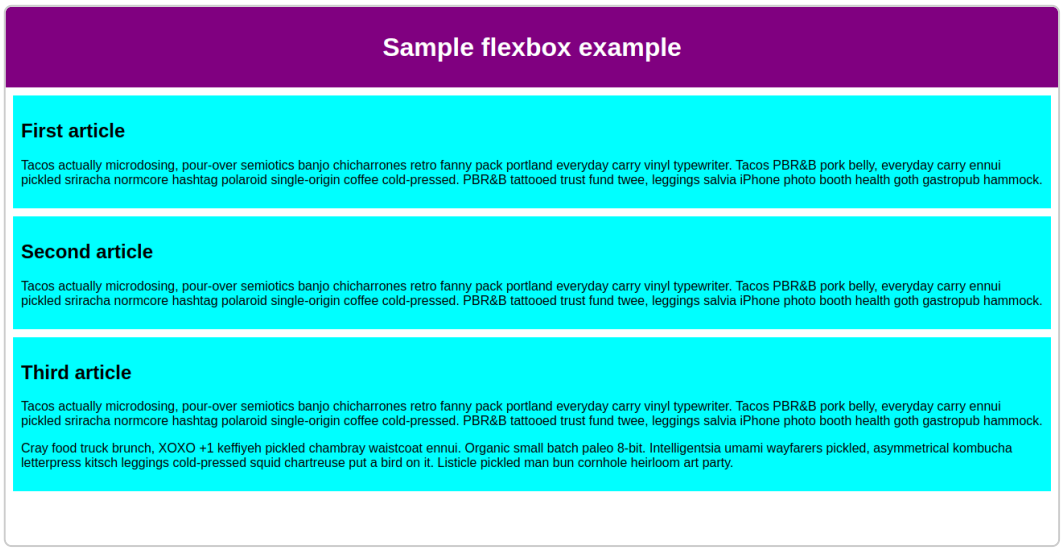
使用此类工具以任何方便、灵活的方式很难或不可能实现以下简单的布局设计：

- 将内容块在其父项内垂直居中。
- 使容器的所有子容器占用相同数量的可用宽度/高度，而不管可用的宽度/高度有多少。
- 使多列布局中的所有列采用相同的高度，即使它们包含不同数量的内容。

正如您将在后续部分中看到的，flexbox 使许多布局任务变得更加容易。让我们开始吧！

引入一个简单的例子

在本文中，您将通过一系列练习来帮助您了解 flexbox 的工作原理。要开始，您应该从我们的 GitHub 存储库中制作第一个启动文件的本地副本 — flexbox0.html 。 在现代浏览器（如 Firefox 或 Chrome）中加载它，并在代码编辑器中查看代码。你也可以在这里看到它 。



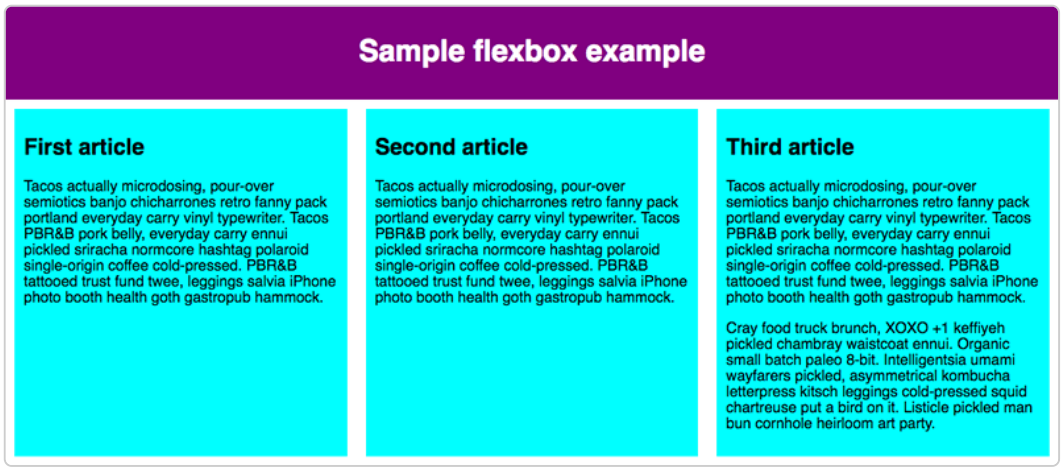
您会看到我们有一个 <header> 元素，其中包含一个顶级标题和一个 <section> 包含三个 <article> s 的元素。我们将使用这些来创建一个相当标准的三列布局。

指定将哪些元素布置为灵活的盒子

首先，我们需要选择要将哪些元素布置为灵活的盒子。为此，我们在 display 要影响的元素的父元素上设置一个特殊值。在这种情况下，我们想要布置 <article> 元素，因此我们将其设置在 <section> ：

```
section {
  display: flex;
}
```

这会导致 <section> 元素成为**弹性容器**，其子元素成为**弹性项目**。结果应该是这样的：

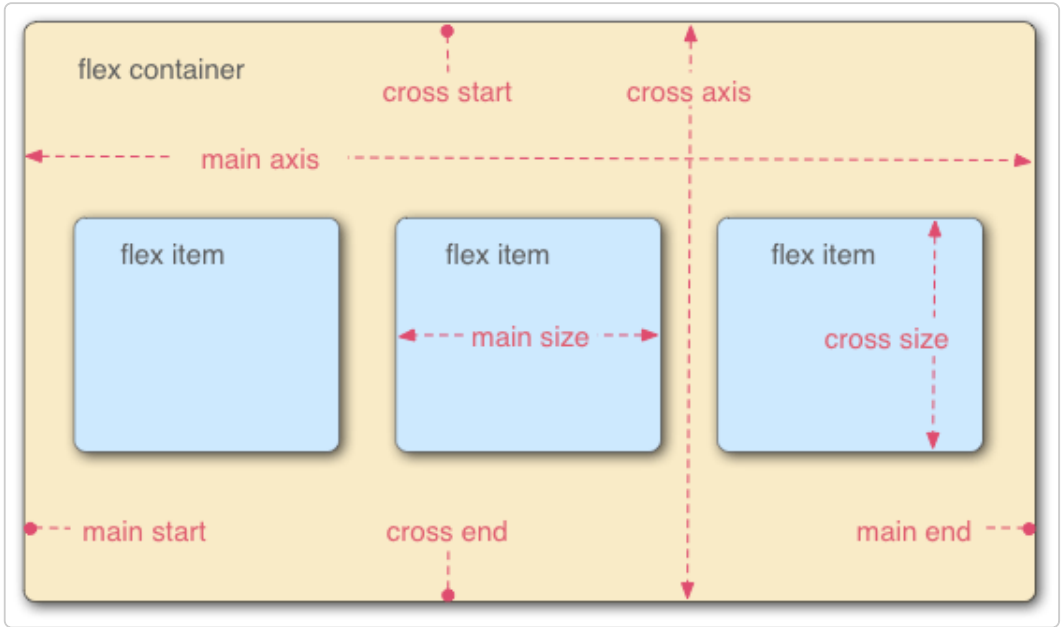


所以，这个单一的声明给了我们所需要的一切。难以置信，对吧？我们的多列布局具有相同大小的列，并且列的高度都相同。这是因为赋予弹性项目（弹性容器的子项）的默认值是为了解决诸如此类的常见问题而设置的。

为了清楚起见，让我们重申这里发生的事情。 `display` 我们赋予 to值的元素 `flex` 在与页面其余部分交互方面表现得像一个块级元素，但它的子元素被布置为 flex 项目。下一节将更详细地解释这意味着什么。另请注意，如果您希望将元素的子元素布置为弹性项目，但让该元素表现得像内联元素，则可以使用 `display` 值。 `inline-flex`

弹性模型

当元素作为弹性项目布局时，它们沿着两个轴布局：



- 主轴是在 flex 项目布局方向上运行的轴（例如，作为页面上的一行，或页面下方的一列）。该轴的起点和终点称为**main start**和**main结束**。
- 交叉**轴**是垂直于弹性项目布局方向的轴。该轴的起点和终点称为交叉**起点**和**交叉终点**。
- 在其上设置的父元素 `display: flex` （`<section>` 在我们的示例中）称为**flex container**。
- 在 flex 容器中布置为灵活框的项目称为**flex 项目**（`<article>` 我们示例中的元素）。

在阅读后续部分时请牢记此术语。如果您对所使用的任何术语感到困惑，可以随时参考它。

列还是行?

Flexbox 提供了一个名为的属性 `flex-direction`，它指定主轴运行的方向（flexbox 子元素的布局方向）。默认情况下，它设置为 `row`，这会导致它们按照浏览器默认语言的工作方向排成一行（对于英文浏览器，从左到右）。

尝试将以下声明添加到您的 `<section>` 规则中：

```
flex-direction: column;
```

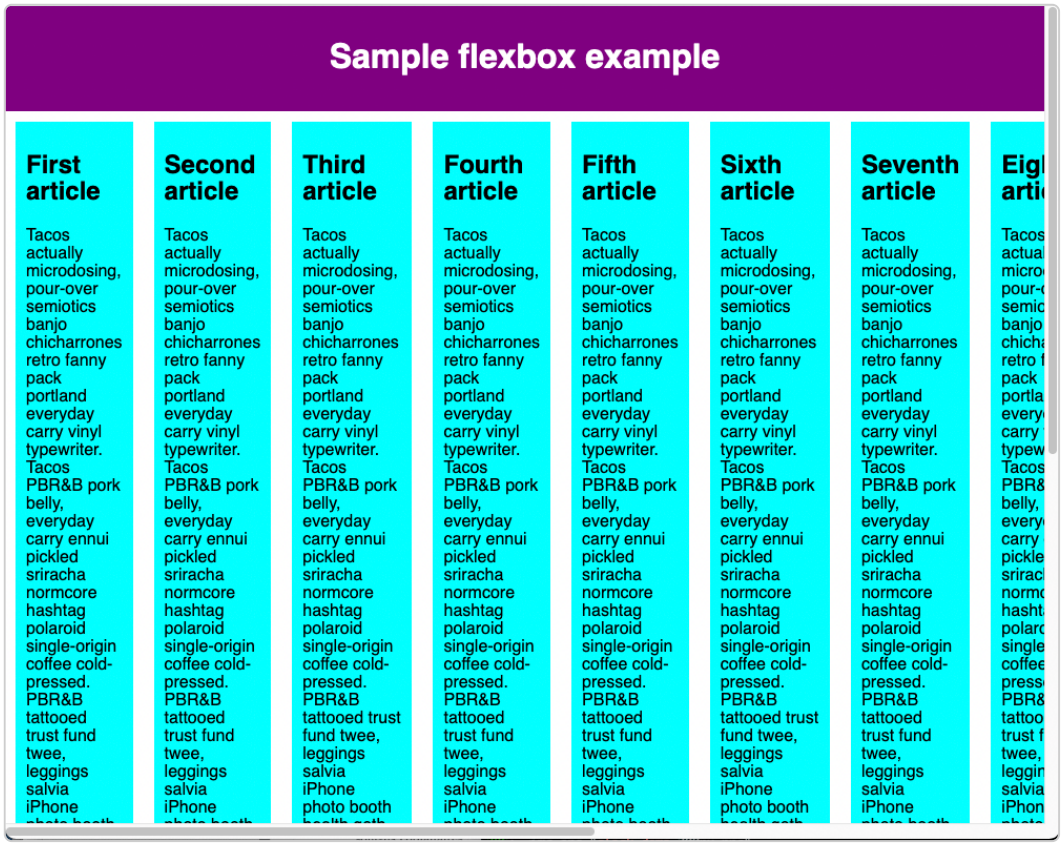
您会看到这会将项目放回到列布局中，就像我们添加任何 CSS 之前一样。在您继续之前，请从您的示例中删除此声明。

> **注意：** `row-reverse` 您还可以使用和值以相反方向布置弹性项目 `column-reverse`。也可以尝试这些值！

包装

当你的布局中有一个固定的宽度或高度时出现的一个问题是最终你的 flexbox 孩子会溢出他们的容器，破坏布局。查看我们的flexbox-

wrap0.html 示例并尝试实时查看它 （如果您想跟随此示例，请立即获取此文件的本地副本）：



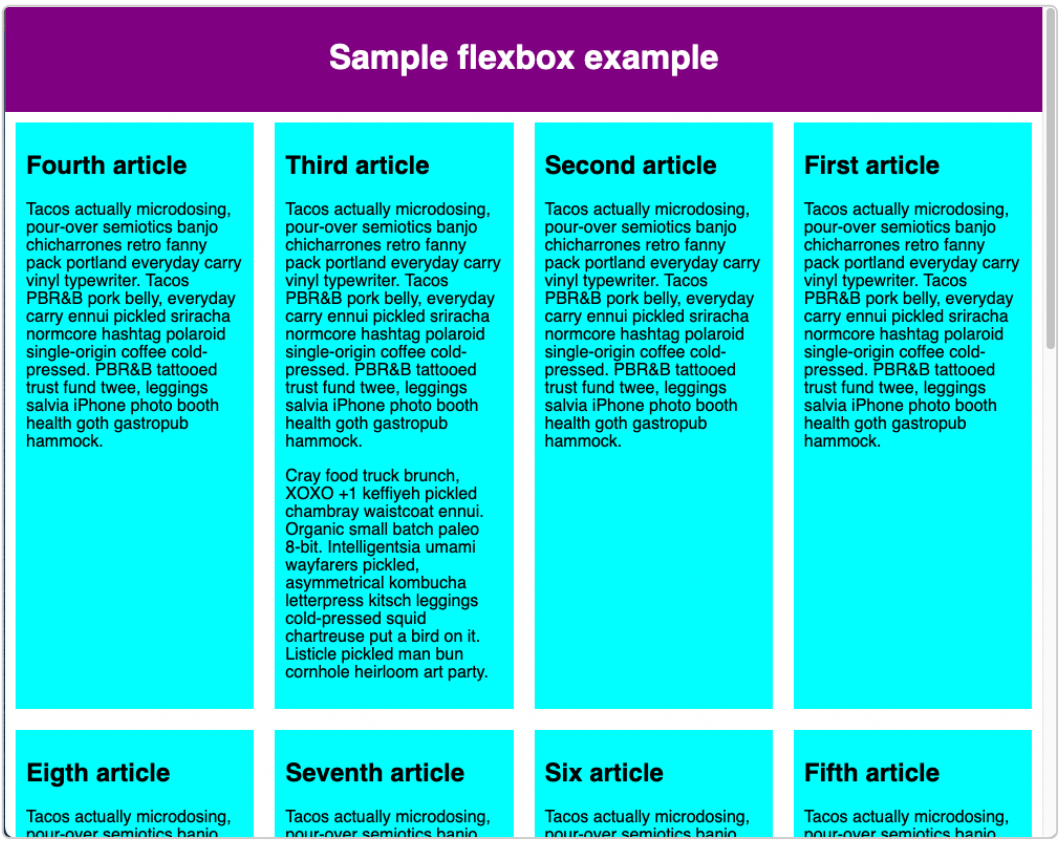
在这里我们看到孩子们确实打破了他们的容器。解决此问题的一种方法是将以下声明添加到您的 <section> 规则中：

```
flex-wrap: wrap;
```

此外，将以下声明添加到您的 <article> 规则中：

```
flex: 200px;
```

现在试试这个。您会看到包含以下内容的布局看起来好多了：



我们现在有多行。每行都有尽可能多的 flexbox 子元素。任何溢出都向下移到下一行。文章上的声明 `flex: 200px` 意味着每篇文章的宽度至少为 200 像素。稍后我们将更详细地讨论此属性。您可能还会注意到，最后一行的最后几个孩子都变宽了，因此整行仍然被填满。

但是我们可以在这里做更多的事情。首先，尝试将您的 `flex-direction` 属性值更改为 `row-reverse` . 现在您会看到您仍然拥有多行布局，但它从浏览器窗口的对角开始并反向流动。

弹性流简写

在这一点上值得注意的是存在一个速记 `flex-direction` 和 `flex-wrap` : `flex-flow` 。因此，例如，您可以替换

```
flex-direction: row;
flex-wrap: wrap;
```

和

```
flex-flow: row wrap;
```

弹性项目的灵活大小

现在让我们回到第一个例子，看看我们如何控制弹性项目与其他弹性项目相比所占空间的比例。启动flexbox0.html的本地副本，或将 flexbox1.html 的副本作为新起点（实时查看 ）。

首先，将以下规则添加到 CSS 的底部：

```
article {
  flex: 1;
}
```

这是一个无单位的比例值，它决定了与其他弹性项目相比，每个弹性项目将沿主轴占用多少可用空间。在本例中，我们为每个 <article> 元素赋予了相同的值（值为 1），这意味着在设置 padding 和 margin 等属性后，它们将占用等量的剩余空间。这个值在弹性项目之间按比例共享：给每个弹性项目一个 400000 的值会产生完全相同的效果。

现在在前一个规则下面添加以下规则：

```
article:nth-of-type(3) {
  flex: 2;
}
```

现在，当您刷新时，您会看到第三个 <article> 占用的可用宽度是其他两个的两倍。现在总共有四个比例单位可用（因为 1 + 1 + 2 = 4）。前两个弹性项目各有一个单元，因此它们各占可用空间的 1/4。第三个有两个单元，所以它占用可用空间的 2/4（或二分之一）。

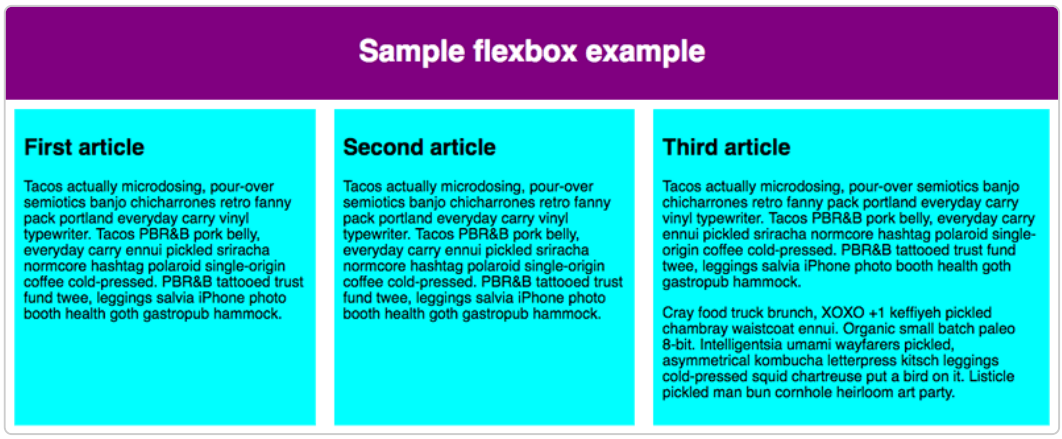
您还可以在 flex 值内指定最小尺寸值。尝试像这样更新您现有的文章规则：

```
article {
  flex: 1 200px;
```

```
}

article:nth-of-type(3) {
  flex: 2 200px;
}
```

这基本上是说，“每个 flex 项目将首先获得 200px 的可用空间。之后，剩余的可用空间将根据比例单位共享。” 尝试刷新，您会发现空间共享方式有所不同。



flexbox 的真正价值可以在它的灵活性/响应能力中看出。如果您调整浏览器窗口的大小或添加另一个 `<article>` 元素，布局将继续正常工作。

flex：速记与普通

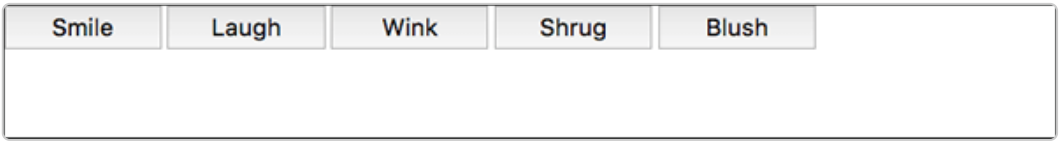
`flex` 是一个速记属性，最多可以指定三个不同的值：

- 我们上面讨论的无单位比例值。这可以使用 longhand 属性单独指定 `flex-grow` 。
- 第二个无单位比例值， `flex-shrink` 当弹性项目溢出其容器时开始发挥作用。该值指定项目将收缩多少以防止溢出。这是一个非常高级的 flexbox 特性，我们不会在本文中进一步介绍它。
- 我们上面讨论的最小尺寸值。这可以使用普通值单独指定 `flex-basis` 。

我们建议不要使用普通的 flex 属性，除非你真的必须这样做（例如，覆盖以前设置的东西）。它们会导致编写大量额外的代码，并且可能会有些混乱。

水平和垂直对齐

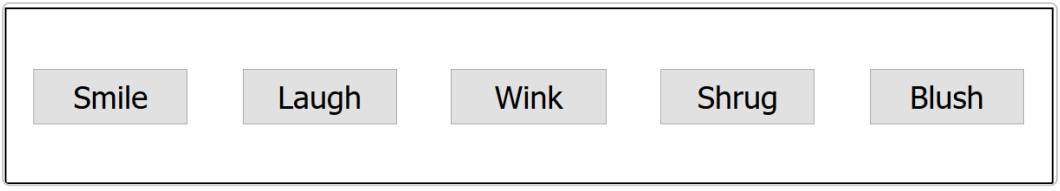
您还可以使用 flexbox 功能沿主轴或交叉轴对齐弹性项目。让我们通过查看一个新示例来探索这一点：flex-align0.html （也可以实时查看 ）。我们要把它变成一个整洁、灵活的按钮/工具栏。现在，您会看到一个水平菜单栏，左上角塞满了一些按钮。



首先，获取此示例的本地副本。

现在，将以下内容添加到示例 CSS 的底部：

```
div {
  display: flex;
  align-items: center;
  justify-content: space-around;
}
```



刷新页面，您会看到按钮现在在水平和垂直方向上都很好地居中。我们通过两个新属性完成了此操作。

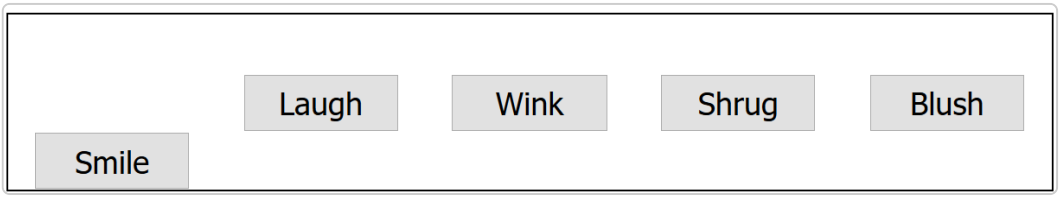
`align-items` 控制弹性项目在横轴上的位置。

- 默认情况下，该值为 `stretch` ，它会在交叉轴方向拉伸所有弹性项目以填充父级。如果父项在横轴方向上没有固定的高度，那么所有的弹性项目都会变得和最高的弹性项目一样高。这就是我们的第一个示例默认情况下具有等高列的方式。
- 我们在上面的代码中使用的值 `center` 使项目保持其固有尺寸，但沿交叉轴居中。这就是我们当前示例的按钮垂直居中的原因。

- 您还可以使用 `flex-start` 和之类的值 `flex-end` ，它们将分别在交叉轴的起点和终点对齐所有项目。 `align-items` 有关详细信息，请参阅。

您可以 `align-items` 通过将属性应用于单个弹性项目来覆盖 `align-self` 它们的行为。例如，尝试将以下内容添加到您的 CSS：

```
button:first-child {
  align-self: flex-end;
}
```



看看这有什么影响，完成后再次将其删除。

`justify-content` 控制弹性项目在主轴上的位置。

- 默认值为 `flex-start` ，这使得所有项目都位于主轴的起点。
- 你可以用来 `flex-end` 让他们坐在最后。
- `center` 也是 的一个值 `justify-content` 。它将使弹性项目位于主轴的中心。
- 我们在上面使用的值 `space-around` 非常有用——它沿主轴均匀分布所有项目，并在两端留有一点空间。
- 还有另一个值， `space-between` 它与 非常相似， `space-around` 只是它在两端不留任何空间。

该 `justify-items` 属性在 flexbox 布局中被忽略。

我们鼓励您在继续之前尝试使用这些值，看看它们是如何工作的。

订购弹性项目

Flexbox 还有一个功能，可以在不影响源代码顺序的情况下更改弹性项目的布局顺序。这又是传统布局方式无法做到的事情。

代码很简单。尝试将以下 CSS 添加到您的按钮栏示例代码中：

```
button:first-child {
  order: 1;
}
```

刷新，你会看到“微笑”按钮已经移动到主轴的末端。让我们更详细地谈谈它是如何工作的：

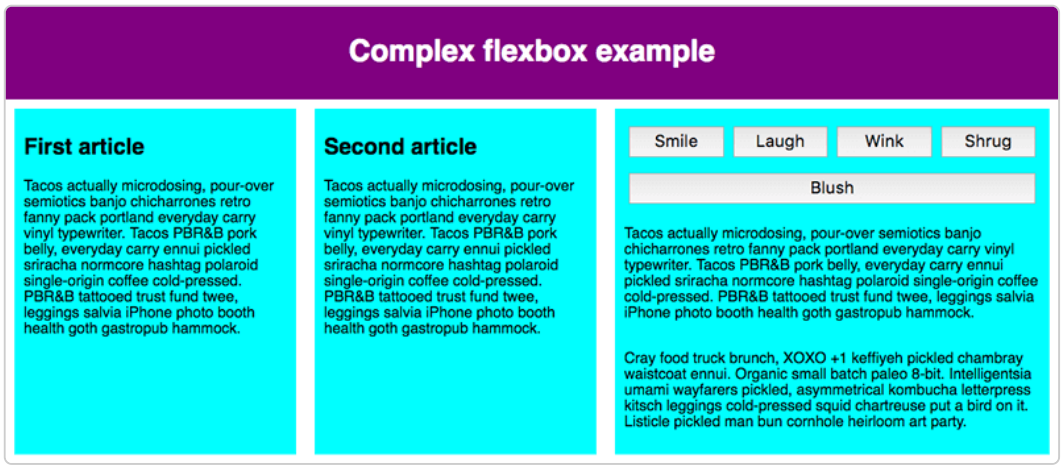
- 默认情况下，所有弹性项目 `order` 的值为 0。
- 具有较高指定订单值的弹性项目将在显示顺序中晚于具有较低订单值的项目出现。
- 具有相同订单价值的弹性项目将出现在它们的来源订单中。因此，如果您有四个项目，其顺序值分别设置为 2、1、1 和 0，则它们的显示顺序将为第 4、第 2、第 3，然后第 1。
- 第 3 个项目出现在第 2 个之后，因为它具有相同的顺序值并且在源顺序中位于它之后。

您可以设置负顺序值以使项目比值为 0 的项目更早出现。例如，您可以使用以下规则使“腮红”按钮出现在主轴的开头：

```
button:last-child {
  order: -1;
}
```

嵌套弹性盒子

使用 flexbox 可以创建一些非常复杂的布局。将一个弹性项目也设置为一个弹性容器是完全可以的，这样它的子项也可以像弹性盒子一样布局。看看 complex-flexbox.html （也可以实时查看 ）。



这个的 HTML 相当简单。我们有一个 <section> 包含三个 s 的元素 <article> 。第三个 <article> 包含三个 ， 第一个 包含五个 <button> ：

```
section – article
          article
          article – div – button
                    div   button
                    div   button
                          button
                          button
```

让我们看看我们用于布局的代码。

首先，我们将 <section> 要布置的 的子项设置为灵活的盒子。

```
section {
  display: flex;
}
```

接下来，我们在 s 本身上设置一些弹性值 <article> 。特别注意这里的第二条规则：我们将第三条规则设置 <article> 为让它的子项也像弹性项目一样布局，但这次我们将它们像列一样布局。

```
article {
  flex: 1 200px;
}
```

```
article:nth-of-type(3) {
  flex: 3 200px;
  display: flex;
  flex-flow: column;
}
```

接下来，我们选择第一个 。我们首先使用 flex: 1 100px; 有效地给它一个 100px 的最小高度，然后我们设置它的子元素（ <button> 元素）也像 flex 项目一样布局。在这里，我们将它们放置在一个环绕行中，并将它们对齐在可用空间的中心，就像我们之前看到的单个按钮示例所做的那样。

```
article:nth-of-type(3) div:first-child {
  flex: 1 100px;
  display: flex;
  flex-flow: row wrap;
```


最后，我们在按钮上设置一些大小。这次给它一个 1 auto 的 flex 值。这有一个非常有趣的效果，如果您尝试调整浏览器窗口宽度，就会看到这种效果。这些按钮将占用尽可能多的空间。尽可能多的人可以放在一条线上；除此之外，他们将换行。

```
button {
  flex: 1 auto;
  margin: 5px;
  font-size: 18px;
  line-height: 1.5;
}
```

跨浏览器兼容性

大多数新浏览器都支持 Flexbox：Firefox、Chrome、Opera、Microsoft Edge 和 IE 11，更新版本的 Android/iOS 等。但是，您应该知道，仍然有不支持的旧浏览器在使用Flexbox（或支持，但支持它的一个非常旧的、过时的版本。）

当您只是在学习和试验时，这并不重要；但是，如果您考虑在真实网站中使用 flexbox，则需要进行测试并确保您的用户体验在尽可能多的浏览器中仍然可以接受。

Flexbox 比某些 CSS 特性要复杂一些。例如，如果浏览器缺少 CSS 投影，则该站点可能仍然可用。然而，不支持 flexbox 功能可能会完全破坏布局，使其无法使用。

我们在跨浏览器测试模块中讨论克服跨浏览器支持问题的策略。

测试你的技能！

您已读完本文，但您还记得最重要的信息吗？在继续之前，您可以找到一些进一步的测试来验证您是否保留了这些信息——请参阅测试您的技能：Flexbox。

概括

我们的 Flexbox 基础知识之旅到此结束。我们希望您玩得开心，并在继续学习的过程中玩得开心。接下来，我们将了解 CSS 布局的另一个重要方面：CSS 网格。

也可以看看

- CSS-Tricks Guide to Flexbox — 一篇以视觉上吸引人的方式解释关于 Flexbox 的一切的文章
- Flexbox Froggy — 一款学习和更好地理解 Flexbox 基础知识的教育游戏

此页面最后修改于 2023 年 2 月 23 日由MDN 贡献者提供。