- 第一节：【十分钟学会编程的本质】BV1AF411s78P
  - 编程=数据+函数
- 第二节：【有手就会的第一行代码】BV12L4y1775u
  - 函数=接收参数+处理+输出
- 第三节：【十分钟学会写网页】BV1hY411j7UM
  - 盒子div
  - 块p = display:block
  - style
    - margin 外边距
    - border 边框
    - padding 内边距
    - height\width
- 第四节：【上线！年轻人的第一个网站】BV17S4y1P7qH
  - HTML
    - <> </>：标签
    - 标签名后空格：属性
    - 文档内容
      - <!DOCTYPE html>
      - <html lang="en(zh-CN)">
      - head
        - ```undefined
          <meta charset="UTF-8">
          <meta http-equiv="X-UA-Compatible" content="IE=edge">
          <meta content="[!--pagekey--]" name="keywords" />
          <title>[!--pagetitle--]</title>
          ```
      - body
        - p
          ```

        - b 加粗
        -
    - div
    - img
        - src=
        - alt=
        - width=
        - height=
    - a
        - hret=
        - target="_blank"

- 补充1：【原来写一个网页这么简单】BV14S4y1K774
- 补充2：【我就不信教不会你HTML】BV12S4y1m7GR
    - 绝对路径：显示完整的描述文件路径
      例如：C:\Users-\Desktop\乱放\截图壁纸\123.jpg

      相对路径：相对文件所在的路径
      相同目录：./ 文件名字
      例如：./123.jpg
      上级目录：../ 文件名字
      例如：../123.jpg
      下级目录：./目录名字/文件名字
      例如：./目录名字/123.jpg
    - 基础标签

      <!DOCTYPE>　　定义文档类型

      <html>　定义一个HTML文档

      <title> 为文档定义一个标题

      <body>　定义文档主体

      <h1> to <h6>　　定义HTML标题

<p>　　定义一个段落

　　定义简单的折行（换行）

<hr>　　定义水平线

<!--...-->　　定义一个注释

- 格式
  - <acronym>　　（HTML5不再支持）定义支取首字母缩写
  - <abbr>　定义一个缩写
  - <address>　　定义文档作者或拥有者信息
  - <b>　　定义粗体文本
  - <bdi>　允许您设置一段文本，使其脱离其父元素的文本方向设置
  - <bdo>　定义文本的方向
  - <big>　（HTML5不再支持）定义大号文本
  - <blockquote>　　定义块引用
  - 　　（HTML5不再支持。HTML 4.01已废弃）定义居中文本
  - <cite>　定义引用（citation）
  - <code>　定义计算机代码文本
  - <del>　定义被删除文本
  - <dfn>　定义定义项目
  - 　　定义强调文本
  - <font>　（HTML5不再支持。HTML 4.01已废弃）定义文本的字体、尺寸和颜色
  - <i>　　定义斜体文本
  - <ins>　　定义被插入文本
  - <mark>　定义带有记号的文本
  - <merer>　定义度量衡。仅用于已知最大和最小值的度量
  - <pre>　定义预格式文本
  - <progress>　　定义运行中的任务进度（进程）

- <q>　　定义短的引用（熟悉的不能再熟悉了，段落标签）
- <rp>　　定义不支持 ruby 元素的浏览器所显示的内容
- <rt>　　定义字符（中文注音或字符）的解释或发音
- <ruby>　定义 ruby 注释（中文注音或字符）
- <s>　　定义加删除线的文本
- <samp>　定义计算机代码样本
- <small> 定义小号文本
- <strike>　　（HTML5不再支持。 HTML 4.01已废弃）定义加删除线文本
- 　　定义语气更为强烈的强调文本
- <sub>　定义下标文本
- <sup>　定义上标文本
- <time>　定义一个日期/时间
- <tt>　　（HTML5不再支持）定义打印机文本
- <u>转义</u>下标线文本
- <var>　　定义文本的变量部分
- <wbr>　规定在文本中的何处适合添加换行符

- 表单
  - <form>　定义一个人HTML表单，用于用户输入
  - <input> 定义一个输入控件（输入框，也是常用的标签！）
  - <textarea>　　定义多行的文本输入控件
  - <buttom>　　定义按钮（常用！）
  - <select>　　定义选择列表（下拉列表）
  - <optgroup>　　定义选择列表中相关选项的组合
  - <option>　　定义选择列表中的选项
  - <label> 定义 input 元素的标注
  - <fieldset>　　定义围绕表单中元素的边框

        - <legend>　　定义 fieldset 元素的标题
        - <datalis>　　规定了 input 元素可能的选项列表
        - <keygen>　　规定用于表单的密钥对生成器字段
        - <output>　　定义一个计算的结果
    - 框架
        - <frame>　(HTML5不再支持) 定义框架的窗口或框架
        - <frameset>　　(HTML5不再支持) 定义框架集
        - <noframes>　　(HTML5不再支持) 定义针对不支持框架的用户的替代内容
        - <iframe>　　定义内联框架
    - 图像
        - <img>　定义图像（常用！）
        - <map>　定义图像映射
        - <area>　定义图像地图内部的区域
        - <canvas>　　通过脚本（通常是 JavaScript）来绘制图形（比如图标和其他图像）
        - <figcaption>　　定义一个 caption for a <figure> element
        - <figure>　　figure 标签用于对元素进行组合
    - Audio/Video
        - <audio> 定义声音，比如音乐或其他音频流
        - <source>　　定义media元素（<video>和<audio>）的媒体资源。 media
        - <track> 为媒体（<video> 和 <audio>）元素定义外部文本轨道
        - <vidio> 定义一个音频或者视频（常用！）
    - 链接
        - <a>　　定义一个链接（常用！）
        - <link>　定义文档与外部资源的关系（常用！）
        - <main>　定义文档的主体部分

    - <nav>　定义导航链接（常用！）
- 列表
    - 　　定义一个无序列表（常用！）
    - 　　定义一个有序列表（常用！）
    - 　　定义一个列表项（常用！）
    - <dir>　　（HTML5不再支持。HTML 4.01已作废）定义目录列表
    - <dl>　　定义一个定义列表
    - <dt>　　定义一个定义定义列表中的项目（常用！）
    - <dd>　　定义定义列表中项目的描述
    - <menu>　定义菜单列表
    - <command>　　定义用户可能调用的命令（比如单选按钮、复选框或按钮）
- 表格
    - <table> 定义一个表格（常用！）
    - <caption>　　定义表格标题
    - <th>　　定义表格中的表头单元格（常用！）
    - <tr>　　定义表格中的行（常用！）
    - <td>　　定义表格中的单元（常用！）
    - <thead> 定义表格中的表头内容（常用！）
    - <tbody> 定义表格中的主体内容（常用！）
    - <tfoot> 定义表格中的表注内容（脚注）（常用！）
    - <col>　　定义表格中一个或多个列的属性值
    - <colgroup>　　定义表格中供格式化的列组
- 样式/节
    - <style> 定义文档的样式信息（太熟悉了，常用！）
    - 　　定义文档中的节（此处的div的上面的div稍有不同）
    - 　定义文档中的节（常用！）

        - <header>　　定义一个文档头部部分
        - <footer>　　定义一个文档底部
        - <section>　　定义了文档的某个区域
        - <article>　　定义一个文章内容
        - <aside> 定义其所处内容之外的内容
        - <details>　　定义了用户可见的或者隐藏的需求的补充细节
        - <summary>　　定义一个可见的标题。当用户点击标题时会显示出详细信息
    - 元信息：（需了解！）
        - <head>　定义关于文档的信息
        - <meta>　定义关于HTML文档的元信息
        - <base>　定义页面中所有链接的默认地址或默认目标
        - <basefont>　　（HTML5不再支持。HTML 4.01已废除）定义页面中文本的默认字体、颜色或尺寸
    - 程序
        - <script>　　定义客户端脚本（重要！）
        - <noscript>　　定义针对不支持客户端脚本的用户的代替内容
        - <basefont>　　（HTML5不再支持。HTML4.01已废除）定义嵌入的 applet
        - <embed> 定义了一个容器，用来嵌入外部应用或者互动程序（插件）
        - <object>　　定义嵌入的对象
        - <param> 定义对象的参数
- 补充3：【SS：让你的网页也能女装】BV1su411i72x
- 补充4：【CSS 布局及学习小游戏】BV1X3411H76q
    - 一个css规则由选择器和声明两部分组成，选择器可以是标签名，id和class名；声明由属性和属性值组成。
    - 深度解析 CSS Flexbox 布局 - 2020年最新版[1]
    - 基础认知
        - 语法规则

    - id和class选择器
        - 内联，直接在标签里面写style
        - 内部，head标签内写style里面
            - 标签名{}：对某类标签的属性进行美化
            - #id名{}：使对应id="id名"的标签的css属性进行更改（id属性在一个页面中是唯一的,不可重复,一般用于配合js做动画效果）
            - .class名{}：对应的 class="类名" ,类名相同则生效属性
            - *{ }通配符选择器：选中所有标签,较少用到
        - 外部，单独写css文件
    - 字体和文本样式
        - font-size 字体大小
        - font-weight 字体粗细
        - (以上单位一般为像素大小 px)
        - font-style 字体样式 单个字体
        - font-family字体类型 可以支持多种字体,从左往右查找渲染
        - 常用 ： Microsoft YaHei sans-serif
        - font属性连写 字体类型
    - 文本
        - 文本缩进 text-indent /单位: px || em相当于font-size的大小
        - 文本水平对齐 text-align left /center / right
        - 文本修饰属性 text-decoration underline下划线 || line-through || none 无装饰线 || overline 上划线
        - 行高 line-height 单位px // font-size的字体倍数
        - 颜色 color || background-color
        - 外边距 ： margin
- 补充5：【编程环境配置】BV1k34y1Y7xk
    - vscode常用插件和快捷键

- 插件:
  - Chinese 汉化

    prettier 代码美化插件

    wisen-translator 翻译

    live preview 本地打开网页

    Live server : 实时预览html

    VSCode icons 插件

    Git History Diff : 后面git教程中出现的插件

    HTMLHint : 检查html文件出错的地方
  - / HTMl必备

    Chinese (Simplified) Language Pack for Visual Studio Code （中文)

    open in browser （打开浏览器，快捷键“Alt+B”)

    vscode-icons （编辑器的文件图标 => 好看)

    Auto Rename Tag （自动匹配HMTL标签)

    Bracket Pair Colorizer （彩色的括号)

    Highlight Matching Tag （高亮对应HTML标签 & 表示对应括号，高效)

    stylelint （css/sass/less语法检查)

    Path Intellisense （智能路径提示)
  - // Vue必备

    Vetur （Vue必备，提示的嘛，方便)

    Live Server （代码保存后，浏览器自动更新)

    // React必备

    Prettier （格式化插件，比Beautify好)

- // Egg框架必备
  Egg.js dev tools (NodeJs中 EggJs框架，方便)
- // other
  carbon-now-sh (截获代码为PNG，Ctrl+Shift+P => Carbon)
  background (界面右下角有个小人)

- 设置
  Auto Save :自动保存，选择afterDelay/On focus change
  Format On Save，保存文件后会自动格式化你的代码，简称自动美化器
  Default Formatter默认的格式化插件，选择prettier,默认格式化插件
  encoding自动猜测语言
  ascii突出显示非ascii字符
  linked Editing 编辑标签前后两侧
- node.js，它是运行java的后端框架
  重启完之后点击新建终端，node -v。可以看出nodejs是否安装好。
- 快捷键:
  1.control 加 D 会选择下一个相同的字符。就是当你选择一个单词之后，按一下 control 加 D 它会同时选择下一个，然后一直按会一直选。
  2.按住 alter 再按一下鼠标左键，就会添加一个光标，也可以添加多个光标。同时添加之后我们输入文字，它会同时改变多个地方，这个功能也是非常方便。
  3. control 加 S 保存， control 加 Z 是撤销
  4.用 Alt 加 shift 加上下键，就可以快速地把一行代码复制到上面，或者是复制到下面。
  5.查找 control 加 F 当你一个文件中的内容比较多的时候，我们想找一个字符找不到，就会用这个 control 加 F 查找。

- 补充6: 【CSS Flex 网站布局】BV1di4y1U75N
  - display:flax 弹性布局 -盒子模型
    - 语法: 设置标签的display属性 并设置为 flex
    - 然后就可以通过flex的属性进行调整，依赖属性有 position +float
    - 容器默认有两个轴 ： 水平的主轴 (main axis) 和垂直的交叉轴 (cross axis)

- flex-direction属性
  - 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
  - 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
    - 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
- justify-content属性
  - justify-content属性定义了项目在主轴上的对齐方式
    1. flex-start（默认值）：左对齐
    2. flex-end：右对齐
    3. center： 居中
    4. space-between：两端对齐，项目之间的间隔都相等。
    5. 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
  1. align-items属性定义项目在交叉轴上如何对齐。
  2. flex-start：交叉轴的起点对齐。
  3. flex-end：交叉轴的终点对齐。
  4. center：交叉轴的中点对齐。
  5. baseline：项目的第一行文字的基线对齐。
  6. 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

1. 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。
2. flex-start：与交叉轴的起点对齐。
3. flex-end：与交叉轴的终点对齐。
4. center：与交叉轴的中点对齐。
5. space-between：与交叉轴两端对齐，轴线之间的间隔平均分布。
6. 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。
7. stretch（默认值）：轴线占满整个交叉轴。

- 项目的属性

1. order　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。
2. flex-grow　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。
3. flex-shrink　flex-shrink属性定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小。
4. flex-basis　flex-basis属性定义了在分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。
5. flex　flex属性是flex-grow，flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。
6. align-self　align-self属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。

- 补充7：【发布正式网站的注意事项】BV1iY4y1a7X1
- 答疑1：【常见问题】BV1LS4y127Vz
- 作业展示：【学习是高级的快乐】BV1hS4y1c7q9

---

1. 深度解析 CSS Flexbox 布局 - 2020年最新版

在 CSS flexbox 布局出现以前，如果要控制 HTML 元素的布局，要用到很多种奇葩的方式。在水平方向上得用`float`控制左右对齐，稍一不注意，就会有浮动的元素飞来飞去~。在垂直方向上就更是百家争鸣了：要么手动计算高度然后算出中心点，要么用`line-height`和`height`的结合，要么用十之八九不生效的`vertical-align`属性等等等等。自从 flex-box 出现以后，一切似乎就豁然开朗了，水平垂直各种花式对齐，空间分配由你做主。当然，要用好它，用对它也不是一件容易的事，今天就给你说说 flex-box 布局，看完之后你也能熟练的运用它!

2 分钟视频入门版：2 分钟掌握 CSS Grid 布局

真实 HTML 代码（非图片）示例版：请点此访问（示例都是真实的 HTML 代码，可以使用 chrome 开发者工具查看属性。

开启 Flexbox 布局

假设有下边这么一个 html 结构：

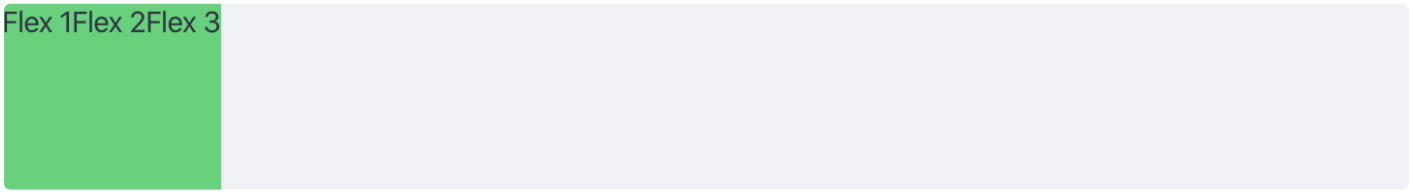
```
<div class="flex">
  <div class="flex1">Flex 1</div>
  <div class="flex2">Flex 2</div>
  <div class="flex3">Flex 3</div>
</div>
```

一个`div`容器包含了三个`div`子元素，按照默认的布局方式进行排列。因为`div`是块级元素，每个`div`占了整个一行的空间：

Flex 1
Flex 2
Flex 3

如果要开启容器的 flex 布局，只需要在 css 里边给`.flex`设置`display: flex`属性，同时为了演示效果，我给它加上了 100px 的高度：

```
display: flex;
height: 100px;
```



可以看到里边的三个元素自动变成了一行，因为 flex 默认是按行进行排列的。Flexbox 布局是一维布局方式，要么按行排列，要么按列排列。

对齐方式

Flex 布局有一个隐式的坐标空间，水平方向有一条主轴(main-axis)，垂直方向上有一条交叉轴(cross-axis)：

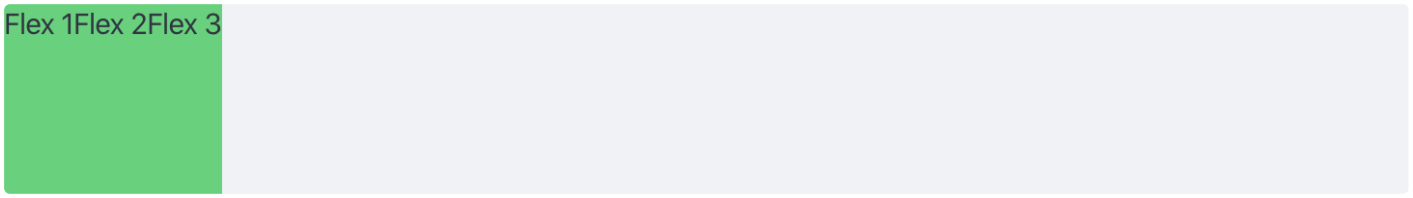
justify-content

控制主轴（即水平方向）对齐方式使用`justify-content`属性，它有下边几种对齐方式：

flex-start

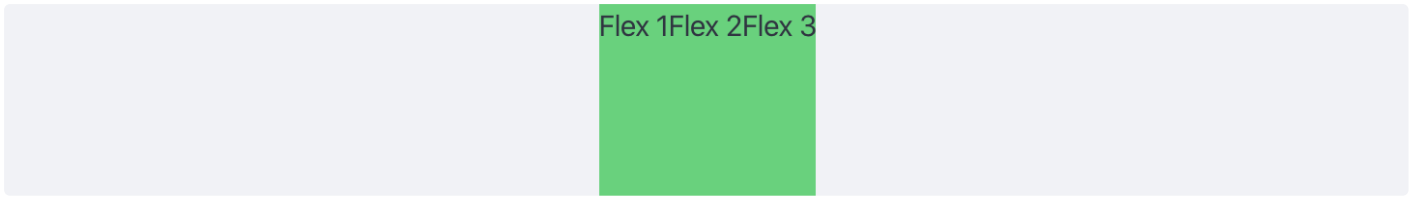
`flex-start` 是默认值，如果是从左到右的文字阅读习惯(LTR)，就是靠左对齐。因为默认的对齐方式，所以跟上边的例子没有什么区别：

```
justify-content: flex-start;
```



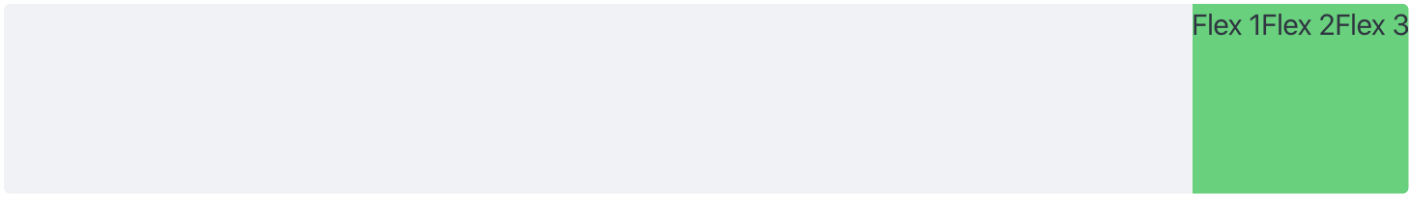
center

居中对齐，此时整个 flex 容器被居中到了页面中间：



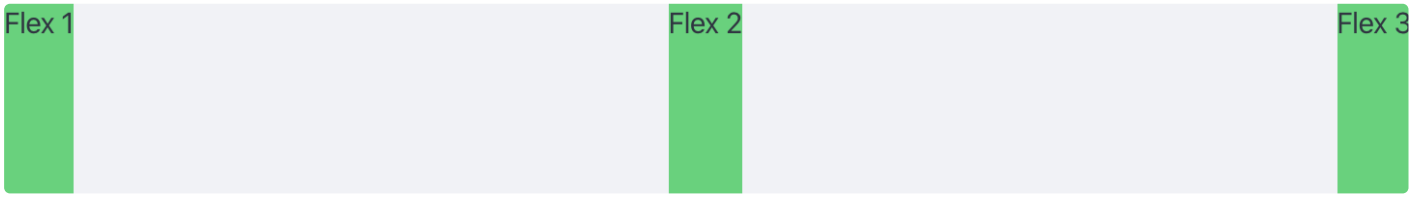
flex-end

靠右对齐：



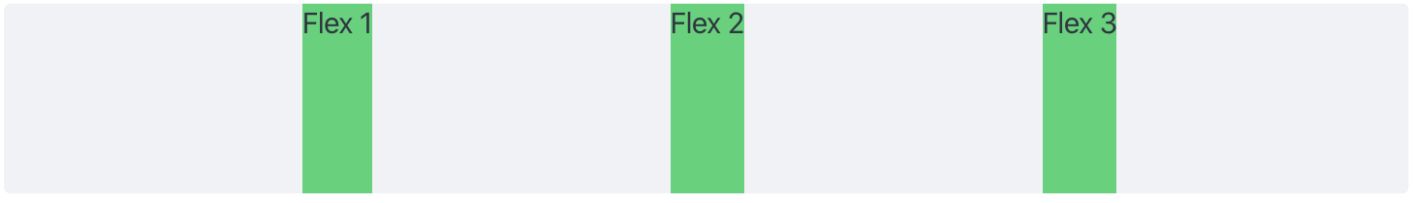
space-between

两端对齐，这种对齐方式是第一个和最后一个元素贴边，中间的元素平分剩余的空间：



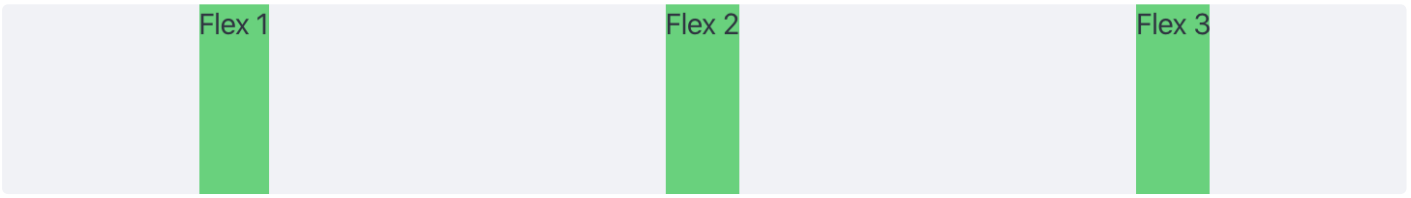
space-evenly

分散对齐，所有的元素都平分空间：



space-around

跟`space-evenly`类似，但是左右两边的留白为平分空间的 1/2.

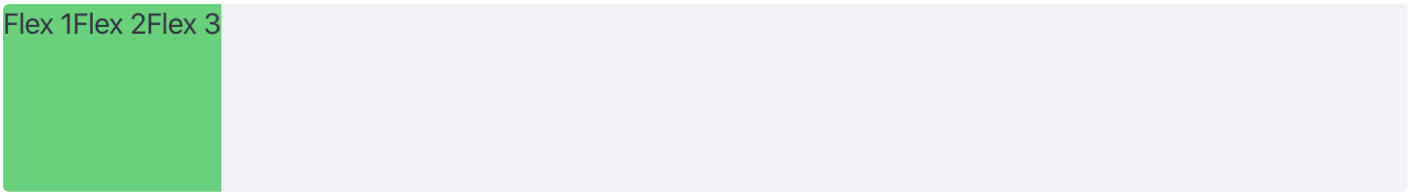


align-items

控制交叉轴方向（即垂直方向）上的对齐方式使用`align-items`属性，有下边几种对齐方式：

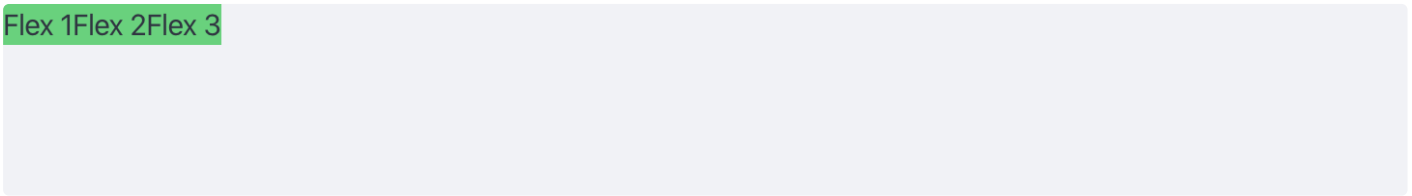
stretch

`stretch` 是 `align-items` 的默认值，它会自动把子元素拉伸成容器的高度，所以之前的例子里子元素在垂直方向上都占满了容器，只要改变容器的`align-items`的值，它就会变成内容的高度。`stretch` 对齐效果如下：



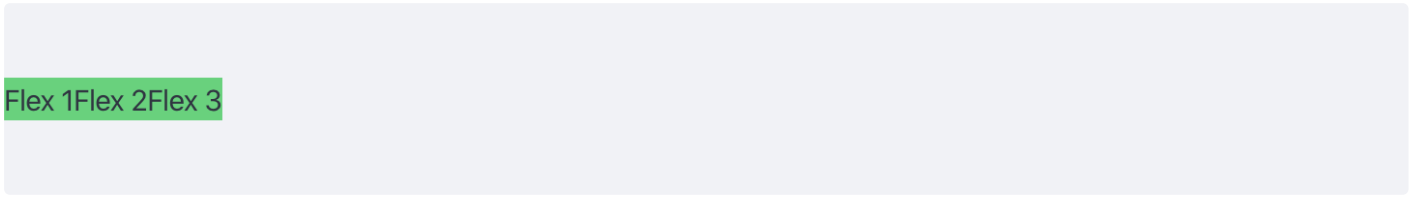
flex-start

靠上对齐，在交叉轴开始的最上方，可以看到子元素不再占满容器宽度：



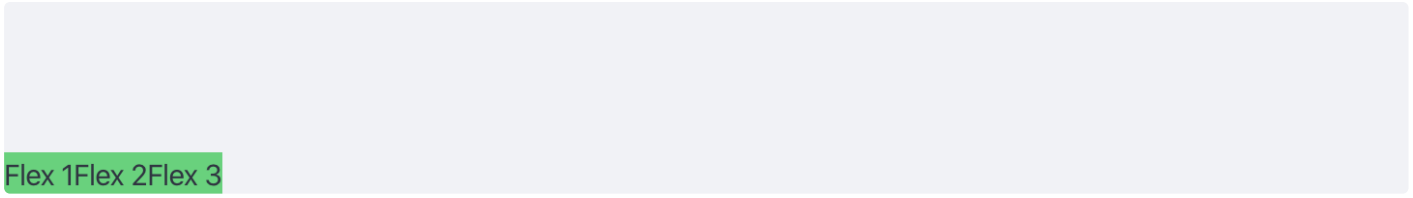
center

居中对齐：



flex-end

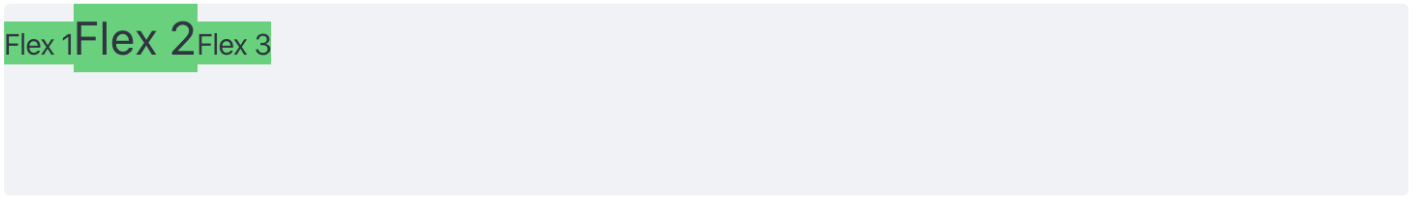
靠下对齐：



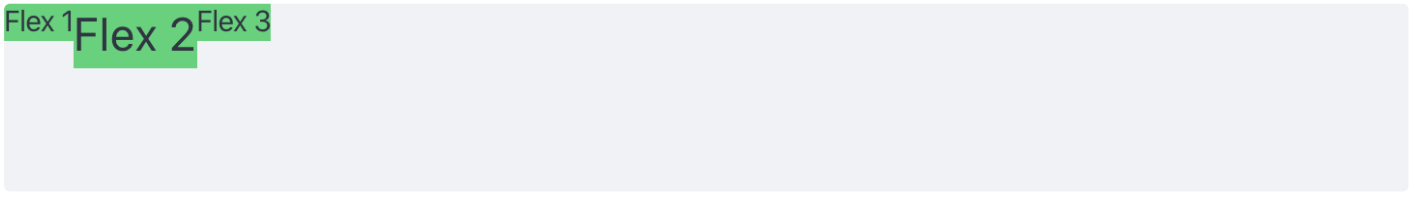
baseline

基线对齐，如果子元素文字尺寸和行高不同，则子元素会按照文字的基线进行对齐：

```
.flex2 {
  font-size: 24px;
}
```



如果是 `flex-start` 对齐方式：



align-content

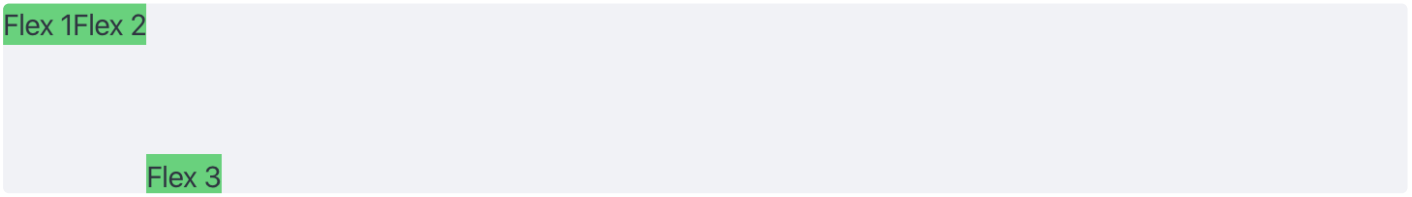
本小节在下边讲到折行时再介绍

子元素覆盖对齐方式

子元素可以通过设置 `align-self` 来控制自己在交叉轴上的对齐方式，例如把 `.flex3` 子元素在垂直方向上靠下对齐：

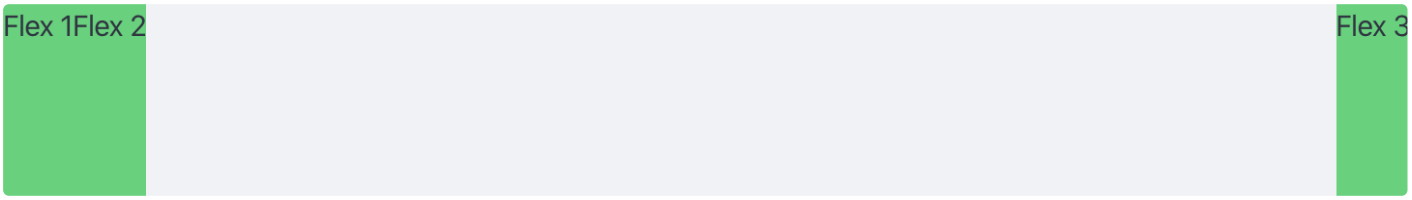
```
.flex {
  display: flex;
  align-items: flex-start;
}

.flex3 {
  align-self: flex-end;
}
```



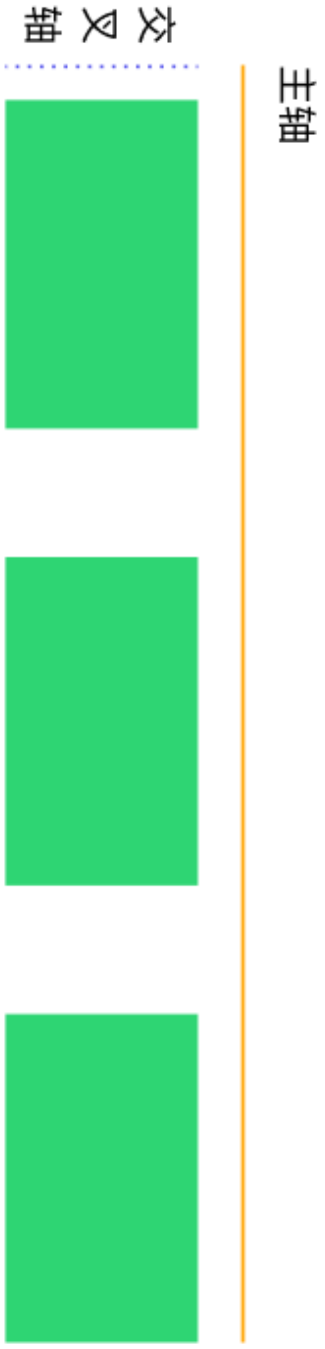
在水平方向上控制子元素对齐并没有 `justify-self` 属性，而是使用 `margin` 属性，通过把左或右边距设置为 `auto` 来控制水平对齐，比如把 `flex3` 放到最右边：

```
.flex3 {
  margin-left: auto;
}
```



排列方式

flex 支持按行排布，也支持按列排布。按列排布时，主轴和交叉轴换了方向，但是 align-items 和 justify-content 控制的轴线不变，即 `align-items` 还是控制交叉轴，`justify-content` 控制主轴：



所以说，在水平方向上对齐变成了使用`align-items`，垂直方向则用`justify-content`。

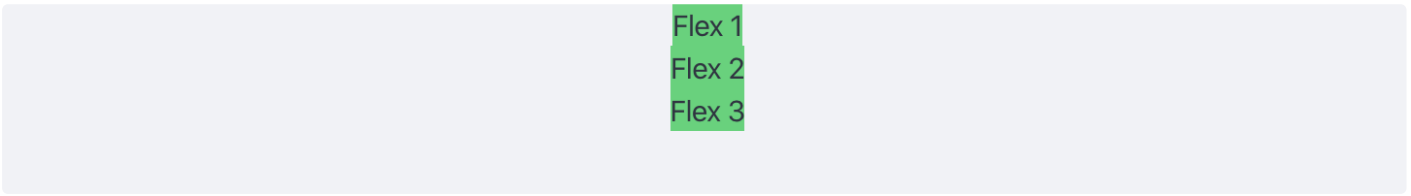
要使 flex 按列排布，只需要设置：

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

来看几个例子：

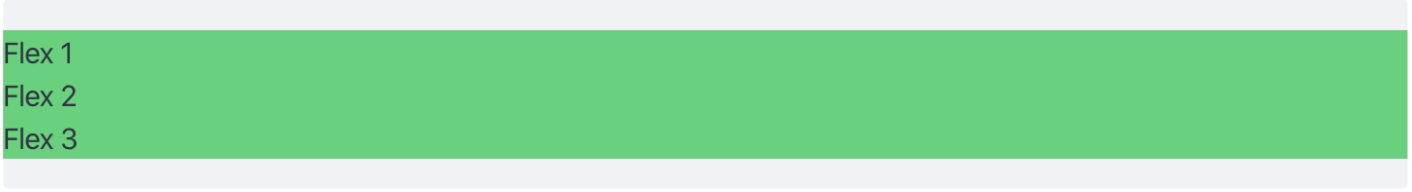
水平居中对齐

```
.flex {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```



垂直居中对齐

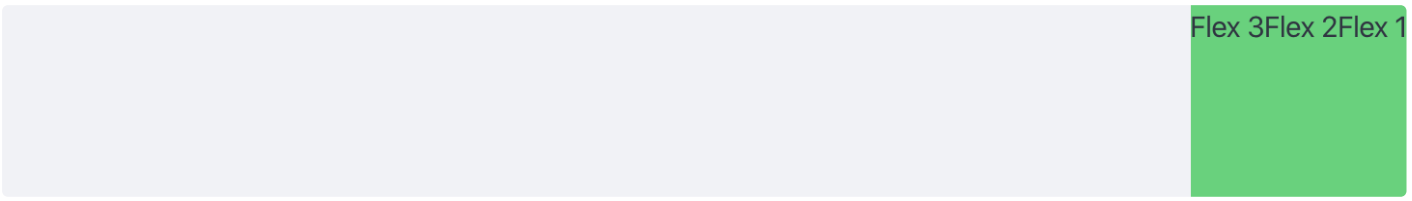
```
.flex {
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```



另外 flex 布局也可以支持反向按行和列布局，相当于按容器中心线进行 180 度翻转：

row-reverse

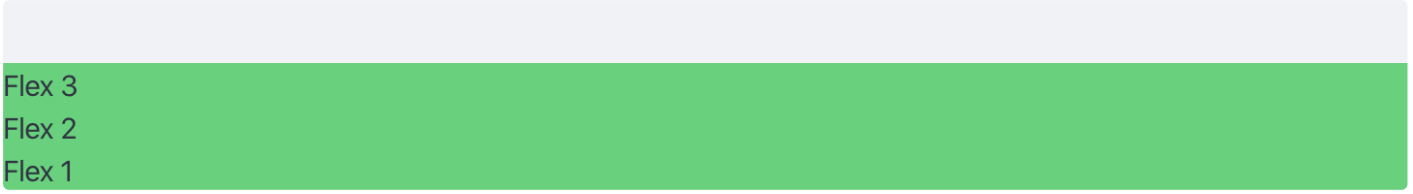
```
.flex {
  display: flex;
  flex-direction: row-reverse;
}
```



`column-reverse`

列模式下会垂直翻转：

```
.flex {
  display: flex;
  flex-direction: column-reverse;
}
```



空间占比

子元素可以通过设置`flex`属性来调整空间的占比，例如让`flex2`在水平方向上占据其他子元素的 2 倍大小，可以设置：

```
.flex1,
.flex3 {
  flex: 1;
}
.flex2 {
  flex: 2;
}
```

Flex-basis

在介绍 flex-basis 之前，先讲一个概念 `main size`，即主轴方向的尺寸，那么，在行排布模式下，也就是水平方向的尺寸，其实就是子元素的宽度，而在列模式下，它是子元素的高度，相对应的也有 `cross size`，即行模式下是子元素的高度，列模式下是宽度。 而 `flex-basis` 是用来设置 `main size` 的，它的优先级会高于 `width`。它的默认值是 `auto`，即在行模式下，如果子元素设置了宽度，它就取自这个宽度值，没有设置的话，就是内容的宽度。使用 `flex-basis`，可以同时管理行模式下的宽度和列模式下的高度。

来看一个例子，把之前的子元素改成固定宽度，比如 `200px`：

```
.flex > * {
  flex-basis: 200px;
}
```

这样每个子元素宽度变为了 200px：

如果再添加 `width` 属性，发现并不会生效：

```
.flex > * {
  flex-basis: 200px;
  width: 250px;
}
```

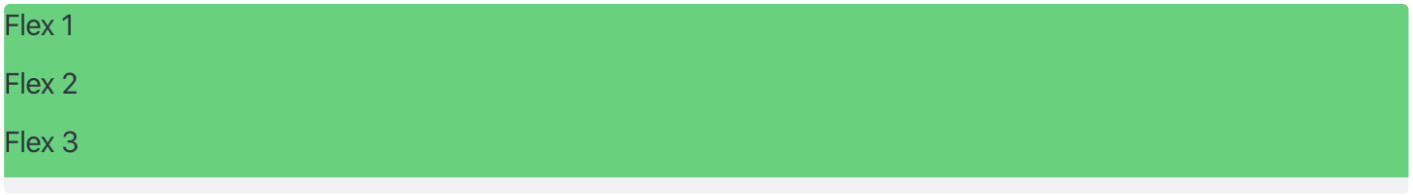
但是，可以通过设置 `min-width` 来强制设置最小宽度：

```
.flex > * {
  flex-basis: 200px;
  min-width: 250px;
}
```

同理的，在列模式下，`flex-basis` 变成了高度，因为容器高度为 `100px`，这里把子元素高度设置成了 `30px` 总计 `90px` 来效果：

```
.flex {
  flex-direction: column;
}

.flex > * {
  flex-basis: 30px;
}
```



同样的，也可以用 `min-height` 来控制最小高度。

缩放

（后续例子都假设是行模式）之前的小节简单说了一下 flex 子元素空间的占比，这里把缩放单独拿出来是为了说明：除了调整 flex 子元素的增长之外，也可以调整收缩，以及 `flex` 属性背后的原理（下一小节）。

flex-grow

先看一下增长，`flex-grow`，这个属性是说 flex 容器在有剩余空间的时候，子元素占据剩余空间的占比。例如，给`.flex2`子元素设置：

```
.flex2 {
  flex-grow: 1;
}
```

其它的元素保持默认的宽度（即内容的宽度，flex-basis 为 auto），那么 `.flex2` 就会自动增长并占据整个剩余空间：

如果把三个元素全部设置成 1，那么所有元素都会自动增长，并各自占据 1/3 的空间：

使用 `flex-grow` 就能够自由的调整元素的空间占比了，非常适合一些浮动的布局。

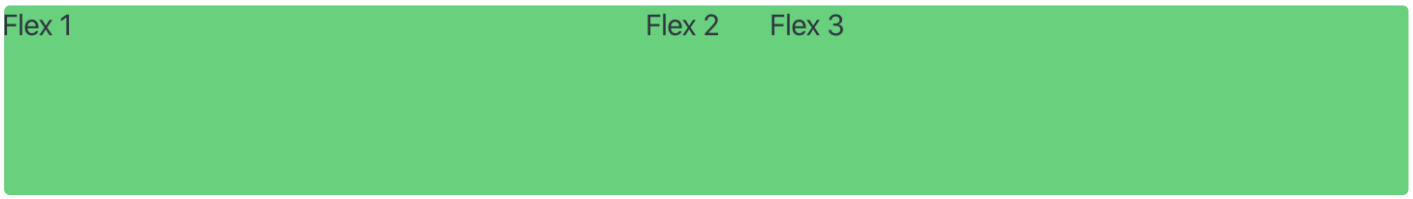
flex-shrink

子元素的收缩是说：当它们的宽度超过 flex 容器之后，该如何进行收缩。通过 `flex-shrink` 来设置一个数值，数值越大，收缩程度也越大，比如`flex-shrink: 2`的元素会比`flex-shrink:1`收缩的值大 2 倍：

```
.flex1,
.flex3 {
  flex-basis: 600px;
  flex-shrink: 1;
}
.flex2 {
  flex-basis: 600px;
```

```
  flex-shrink: 2;
}
```

这里为了方便演示，我把所有的 flex 子元素的 main size（宽度）都设置成了 600px。在我的显示器下，flex 容器的宽度是 728px，三个子元素总和 1800px，显然超出了容器的宽度，那么根据上边定义的收缩规则，`.flex2` 将收缩 2 倍于 `.flex` 和 `.flex3` 收缩的空间。下边的例子中，`.flex1` 和 `.flex3` 的宽度变成了 `332px`，相比于 `600px` 收缩了 `268px`，那么 `.flex2` 就要收缩 `536px (268px * 2)` 的宽度，那么它最后就会剩下 `64px (600px - 536px)` 的宽度：



再说 flex 属性

说完 `flex-grow`、`flex-shrink` 和 `flex-basis` 之后，再来看一下这个 `flex` 属性，它其实是前边三个属性的缩写，默认值是 `0 1 auto`，即不增长，但收缩，收缩比例为 1，flex-basis 为 auto，即取自用户定义的宽度或内容的宽度。

flex 的值可以是下边几种：

- 指定一个数字 - 例如 `flex: 1`，就等同于是 `flex: 1 1 0`，即自动缩放，比例为 1，flex-basis 为 0。
- auto - 等同于 `flex: 1 1 auto`。
- 指定两个数字 - 第一个为 `flex-grow`，第二个，如果是数字则认为是 `flex-shrink`，如果是宽度，则是 `flex-basis`。
- 指定三个值 - 分别为 `flex-grow`，`flex-shrink` 和 `flex-basis`。

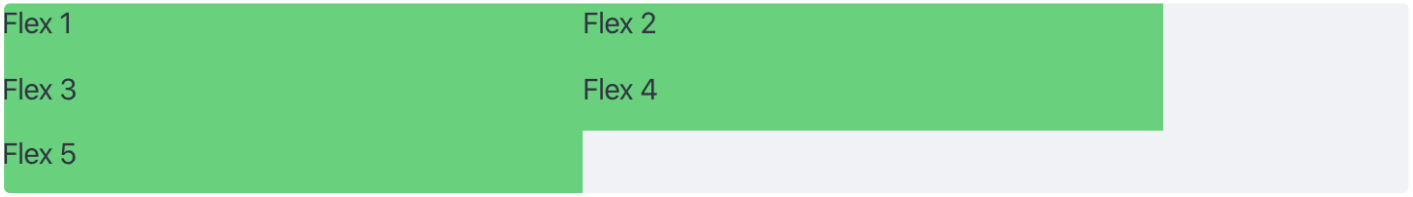
所以说，通过 `flex` 属性可以方便的同时设置 `flex-grow`、`flex-shrink` 和 `flex-basis` 这三个值。

折行

如果子元素有固定宽度，并且超出了容器的宽度，还不允许收缩的话，那么可以使用`flex-wrap`属性来让元素进行折行排列，使得每行的元素都不超过容器的宽度。这里跟 css grid 布局的主要区别是，它无法控制单独控制行、列的占比，比如跨行、夸列，也不能自由定位元素到特定的位置。下边的示例新增了 2 个元素，一共 5 个，每个元素的 main size 为 300px，然后超出宽度后折行：

```
.flex {
  flex-wrap: wrap;
}

.flex > * {
  flex-shrink: 0;
  flex-basis: 300px;
}
```



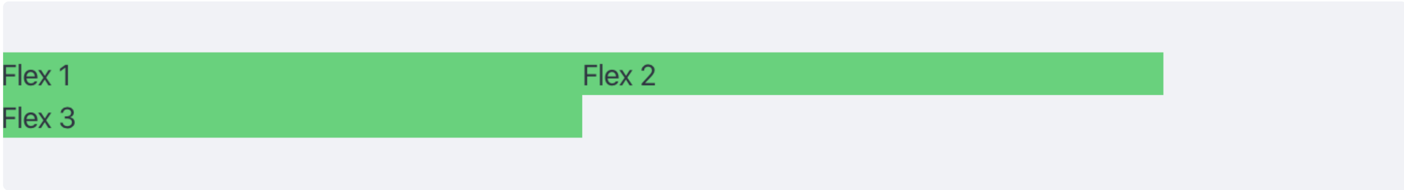
align-content

如果 flex 容器开启了折行，那么两行及以上的内容可以通过`align-content`属性来控制各行之间在交叉轴上的排列规则，它的取值和`justify-content`基本相同，这里演示其中几个，还是使用之前三个元素的 flex 容器，每个容器宽度为 300px，超出后换行：

```
.flex {
  display: flex;
  flex-wrap: wrap;
}
.flex > * {
  flex-basis: 300px;
}
```

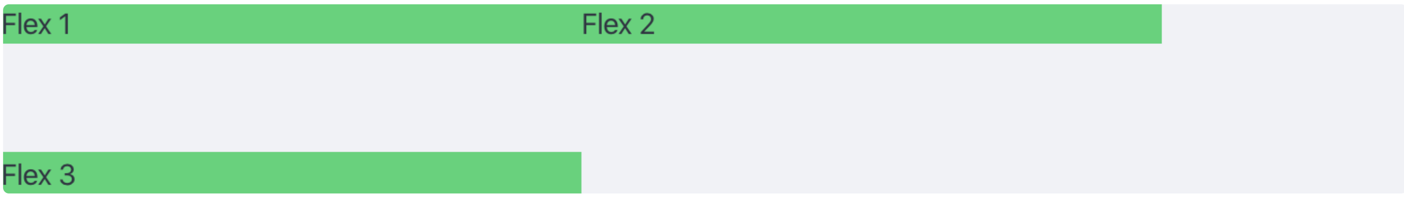
center

垂直居中：



space-between

两端对齐：



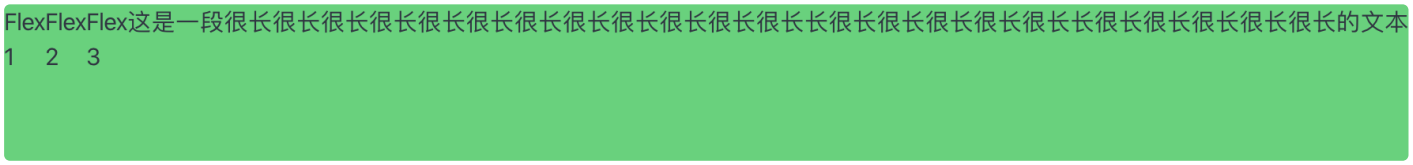
嵌套的 flex 容器的问题

如果 HTML 结构复杂，有嵌套的 flex 容器，很有可能会遇到嵌套的 flex 容器并不能自动收缩的问题，即使设置了 `flex-shrink`。比如有下边一个 html 结构：

```
<div class="flex">
  <div class="flex1">Flex 1</div>
  <div class="flex2">Flex 2</div>
  <div class="flex3">Flex 3</div>
  <div class="flex4">
    <p>
      这是一段很长很长很长很长很长很长很长很长很长很长很长很长长很长很长很长很长很长长很长很长很长很长很长的文本
    </p>
  </div>
</div>
```

这里给之前的 flex 容器添加了一个新的子元素 `.flex4`，这 4 个子元素都设置成 `flex: 1` 来平分空间，但是 `.flex4` 自己本身也是一个 `flex` 布局的容器，里边有一长串文本，我想让它超长之后自动显示省略号，它的 CSS 代码：

```
.flex {
  display: flex;
}
.flex > * {
  flex: 1;
}
.flex4 {
  display: flex;
  flex: 1;
}
.flex4 > p {
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
```



可以看到，最后本应该占 1/4 空间的`.flex4`，因为文本不能换行，直接把 flex 容器撑开了，并且把其他的三个子元素挤成了最小空间，它本应该把文字截短并显示省略号，这是为什么呢？原来，flex 容器的 `min-width` 属性值为 auto，是由浏览器自行计算的，在这里它取了`<p>`元素的宽度，使得宽度成为了一整行 `<p>` 的宽度。那么要解决这个问题，可以把`.flex4` 这个嵌套 flex 容器的 `min-width` 改为`0`，即最小宽度是`0`，那么就可以正常收缩了：

```
.flex4 {
  display: flex;
  flex: 1;
  min-width: 0;
}
```

总结

到这里，整个 flex 布局就介绍完了，还是有不少东西的，但不难。相信通过实例你一定可以掌握它的用法，下边总结一下要点：

- 开启 flex 布局使用`display: flex`属性。
- flex 布局有主轴和交叉轴，分别使用`justify-content`和`align-items`控制对齐方式。
- 支持按行或列进行排列，使用`flex-direction`，另外也支持`row-reverse`和`column-reverse`反向排列。
- 子元素可以通过`flex`简写形式，或者`flex-grow`，`flex-shrink`，`flex-basis`来调整元素的空间占比和缩放。
- 通过`flex-wrap`可以设置 flex 子元素折行显示。
- 嵌套`flex`容器的缩放问题。

你学会了吗？如果有问题，请留下评论吧。