

布局

一、几个概念

1. GFC(GridLayout Formatting Contexts)

- 网格布局格式化上下文
- GFC触发条件--**display: grid**
- 布局规则:

通过在网格容器 (grid container) 上定义网格定义行 (grid definition rows) 和网格定义列 (grid definition columns) 属性各在网格项目 (grid item) 上定义网格行 (grid row) 和网格列 (grid columns) 为每一个网格项目 (grid item) 定义位置和空间

2. FFC(Flex Formatting Contexts)

- 自适应格式化上下文
- FFC触发条件--**display: flex 或 inline-flex**
- 布局规则:
 - 设置为 **flex** 的容器被渲染为一个块级元素
 - 设置为 **inline-flex** 的容器则渲染为一个行内元素
 - 弹性容器中的每一个子元素都是一个弹性项目。弹性项目可以是任意数量的。弹性容器外和弹性项目内的一切元素都不受影响。简单地说, Flexbox 定义了弹性容器内弹性项目该如何布局

3. justify-content

- justify-content 用于设置或检索弹性盒子元素在 主轴 (横轴) 方向上的对齐方式。

参数说明如下:

flex-start

弹性项目向行头紧挨着填充。这个是默认值。第一个弹性项的main-start外边距边线被放置在该行的main-start边线, 而后续弹性项依次平齐摆放。

flex-end

弹性项目向行尾紧挨着填充。第一个弹性项的main-end外边距边线被放置在该行的main-end边线, 而后续弹性项依次平齐摆放。

- center

弹性项目居中紧挨着填充。(如果剩余的自由空间是负的, 则弹性项目将在两个方向上同时溢出)。

space-between

弹性项目平均分布在该行上。如果剩余空间为负或者只有一个弹性项, 则该值等同于flex-start。否则, 第1个弹性项的外边距和行的main-start边线对齐, 而最后1个弹性项的外边距和行的main-end边线对齐, 然后剩余的弹性项分布在该行上, 相邻项目的间隔相等。

space-around

弹性项目平均分布在该行上, 两边留有一半的间隔空间。如果剩余空间为负或者只有一个弹性项, 则该值等同于center。否则, 弹性项目沿该行分布, 且彼此间隔相等 (比如是20px), 同时首尾两边和弹性容器之间留有一半的间隔 ($1/2 * 20px = 10px$)。



◦

4. flex属性

flex 属性是 flex-grow、flex-shrink 和 flex-basis 属性的简写属性。

- `flex-grow` 一个数字，规定项目将相对于其他灵活的项目进行扩展的量。
- `flex-shrink` 一个数字，规定项目将相对于其他灵活的项目进行收缩的量。
- `flex-basis` 项目的长度。合法值："auto"、"inherit" 或一个后跟 "%"、"px"、"em" 或任何其他长度单位的数字。

5. column-count属性

指定某个元素应分为的列数。

- 语法:
`column-count: number| auto;`
- number 列的最佳数目将其中的元素的内容无法流出
- auto 列数将取决于其他属性，例如："column-width"

6. break-inside 属性

规定在指定元素内部是否应发生分页（page-break）、分列（column-break）或分区（region-break）。

值	描述
auto	默认。在元素内自动进行分页、分列、分区。
avoid	避免在元素内出现页、列、区域中断。
avoid-column	避免在元素内分列。
avoid-page	避免在元素内分页。
avoid-region	避免在元素内分区。

二、实际应用

1. flex布局实现水平垂直居中

核心点在于使用了 FFC/GFC 使 `margin: auto` 在垂直方向上居中元素。

原理：

在 flex 格式化上下文中（父容器设置display: flex), 设置了 **margin: auto** 的元素, 在通过 **justify-content** 和 **align-self** 进行对齐之前, 任何 正处于空闲的空间 都会分配到该方向的自动

margin 中去。

```
<style>
  .g-container {
    width: 100vw;
    height: 100vh;
    /* 关键点一 设置布局为flex 触发FFC */
    display: flex;
  }

  .g-box {
    width: 40vmin;
    height: 40vmin;
    background: #000;
    /* 关键点二 设置外边距为自动 */
    margin: auto;
  }
</style>
</head>

<body>
  <div class="g-container">
    <div class="g-box"></div>
  </div>
</body>
```

2. Flex实现双飞翼布局

- 由来: 经过淘宝UED工程师"玉伯"[改造圣杯布局而来](#)
- 双飞翼布局是指 左中右三列布局
- 渲染顺序: 中间盒子优先渲染, 两边盒子宽度固定不变

双飞翼布局的实现

- left、center、right三种都设置左浮动
- - 设置center宽度为100%
 - 设置负边距, left设置负边距为100%, right设置负边距为自身宽度
 - 设置content的margin值为左右两个侧栏留出空间, margin值大小为left和right宽度



3. flex实现圣杯布局

- 对比双飞翼:

圣杯布局与双飞翼布局的不同之处，圣杯布局的左中右三列容器没有多余子容器存在，通过控制父元素的padding空出左右两列的宽度。

4. 瀑布流布局

- 特点: 等宽不等高 根据图片原比例缩放直至宽度达到固定的要求
- 实现
 1. CSS column 实现瀑布流
 - column 实现瀑布流主要依赖两个属性。
 - 一个是 column-count 属性，是分为多少列。
 - 一个是 column-gap 属性，是设置列与列之间的距离。