# 35 | CSS Flex排版：为什么垂直居中这么难？ | 极客时间

winter 2019-04-13

你好，我是 winter。今天我们来谈谈 Flex 排版。

我们在前面多次讲过，正常流排版的设计来源于数百年来出版行业的排版经验，而 HTML 诞生之初，也确实是作为一种“超文本”存在的。

但是，自上世纪 90 年代以来，Web 标准和各种 Web 应用蓬勃发展，网页的功能逐渐从“文本信息”向着“软件功能”过渡，这个思路的变化导致了：CSS 的正常流逐渐不满足人民群众的需求了。

这是因为文字排版的思路是“改变文字和盒的相对位置，把它放进特定的版面中”，而软件界面的思路则是“改变盒的大小，使得它们的结构保持固定”。

因此，在早年的 CSS 中，“使盒按照外部尺寸变化”的能力非常弱。在我入行前端的时间（大约 2006 年），CSS 三大经典问题：垂直居中问题，两列等高问题，自适应宽问题。这是在其它 UI 系统中最为基本的问题，而到了 CSS 中，却变成了困扰工程师的三座大山。

机智的前端开发者们，曾经创造了各种黑科技来解决问题，包括著名的 table 布局、负 margin、float 与 clear 等等。在这种情况下，Flex 布局被随着 CSS3 一起提出（最初叫 box 布局），可以说是解决了大问题。

React Native 则更为大胆地使用了纯粹的 Flex 排版，不再支持正常流，最终也很好地支持了大量的应用界面布局，这一点也证明了 Flex 排版的潜力。

今天，我们就从设计、原理和应用三个方面来学习一下 Flex 布局，我们先从设计开始。

## Flex 的设计

Flex 在英文中是可伸缩的意思，一些翻译会把它译作弹性，我觉得有点不太准确，但是确实中文中没有更好的词。

Flex 排版的核心是 display:flex 和 flex 属性，它们配合使用。具有 display:flex 的元素我们称为 flex 容器，它的子元素或者盒被称作 flex 项。

flex 项如果有 flex 属性，会根据 flex 方向代替宽 / 高属性，形成“填补剩余尺寸”的特性，这是一种典型的“根据外部容器决定内部尺寸”的思路，也是我们最常用的 Windows 和 Apple 窗口系统的设计思路。

## Flex 的原理

说完了设计，我们再来看看原理，Flex 的实现并不复杂，我曾经写过一个基本实现提交给 spritejs 项目，代码可以参考这里。

下面我们就来讲解一下，如何实现一个 Flex 布局。

首先，Flex 布局支持横向和纵向，这样我们就需要做一个抽象，我们把 Flex 延伸的方向称为“主轴”，把跟它垂直的方向称为“交叉轴”。这样，flex 项中的 width 和 height 就会称为交叉轴尺寸或者主轴尺寸。

而 Flex 又支持反向排布，这样，我们又需要抽象出交叉轴起点、交叉轴终点、主轴起点、主轴终点，它们可能是 top、left、bottom、right。

Flex 布局中有一种特殊的情况，那就是 flex 容器没有被指定主轴尺寸，这个时候，实际上 Flex 属性完全没有用了，所有 Flex 尺寸都可以被当做 0 来处理，Flex 容器的主轴尺寸等于其它所有 flex 项主轴尺寸之和。

接下来我们开始做 Flex 排版。

**第一步是把 flex 项分行，有 Flex 属性的 flex 项可以暂且认为主轴尺寸为 0，所以，它可以一定放进当前行。**

接下来我们把 flex 项逐个放入行，不允许换行的话，我们就“无脑地”把 flex 项放进同一行。允许换行的话，我们就先设定主轴剩余空间为 Flex 容器主轴尺寸，每放入一个就把主轴剩余空间减掉它的主轴尺寸，直到某个 flex 项放不进去为止，换下一行，重复前面动作。

分行过程中，我们会顺便对每一行计算两个属性：交叉轴尺寸和主轴剩余空间，交叉轴尺寸是本行所有交叉轴尺寸的最大值，而主轴剩余空间前面已经说过。

**第二步我们来计算每个 flex 项主轴尺寸和位置。**

如果 Flex 容器是不允许换行的，并且最后主轴尺寸超出了 Flex 容器，就要做等比缩放。

如果 Flex 容器有多行，那么根据我们前面的分行算法，必然有主轴剩余空间，这时候，我们要找出本行所有的带 Flex 属性的 flex 项，把剩余空间按 Flex 比例分给他们即可。

做好之后，我们就可以根据主轴排布方向，确定每个 flex 项的主轴位置坐标了。

如果本行完全没有带 flex 属性的 flex 项，justify-content 机制就要生效了，它的几个不同的值会影响剩余空白如何分配，作为实现者，我们只要在计算 Flex 项坐标的时候，加上一个数值即可。

例如，如果是 flex-start 就要加到第一个 flex 项身上，如果是 center 就给第一个 flex 项加一半的尺寸，如果是 space-between，就要给除了第一个以外的每个 flex 项加上“flex 项数减一分之一”。

**第三步我们来计算 flex 项的交叉轴尺寸和位置。**

交叉轴的计算首先是根据 align-content 计算每一行的位置，这部分跟 justify-content 非常类似。

再根据 alignItems 和 flex 项的 alignSelf 来确定每个元素在行内的位置。

计算完主轴和交叉轴，每个 flex 项的坐标、尺寸就都确定了，这样我们就完成了整个的 flex 布局。

# Flex 的应用

接下来我们来尝试用 flex 排版来解决一下当年的 CSS 三大经典问题（简直易如反掌）。

垂直居中：