

24 | CSS排版：从毕升开始，我们就开始用正常流了

winter 2019-03-14



00:00

14:52

讲述：winter 大小：13.62M

你好，我是 winter。今天我们来聊聊 CSS 的正常流。

我想，在 CSS 中，大家最讨厌的大概就是排版部分了。因为早年的 CSS 设计上不能够很好地支持软件排版需求，导致大家需要使用很多黑科技，让很多新人望而却步。

现在 CSS 提供了很多种排版方式，我们有很多选项可以选择自己适合的那一种，然而，正常流却是我们绕不开的一种排版。

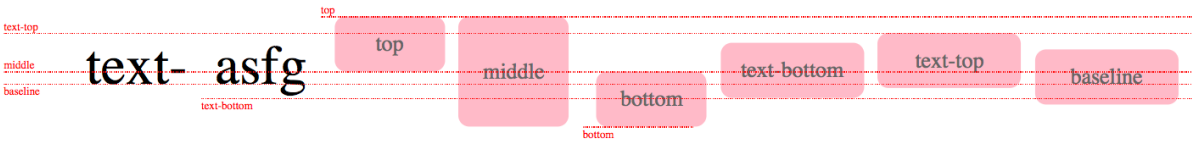
我们能够在网上看到关于正常流的各种资料，比如：块级格式化上下文、margin 折叠等等.....这一系列的概念光是听起来就令人非常头痛。

所以我相信很多同学一定会奇怪：正常流到底正常在哪里。事实上，我认为正常流本身是简单和符合直觉的东西。

我们之所以会觉得它奇怪，是因为如果我们从严苛的 CSS 标准角度去理解正常流，规定排版的算

复杂，但是实际上，基线、文字顶 / 底、行顶 / 底都是我们正常书写文字时需要用到的概念，只是我们平时不一定会总结它们。

下图展示了在不同的 vertical-align 设置时，盒与文字是如何混合排版的。为了方便你理解，我们用代码给大家标注了基线、文字顶 / 底、行顶 / 底等概念



(点击大图查看)

除此之外，margin 折叠是很多人非常不理解的一种设计，但是实际上我们可以把 margin 理解为“一个元素规定了自身周围至少需要的空间”，这样，我们就非常容易理解为什么 margin 需要折叠了。

正常流的原理

我们前面描述了正常流的行为，接下来我们要切换一下模式，用比较严谨的姿势来理解一下正常流。

在 CSS 标准中，规定了如何排布每一个文字或者盒的算法，这个算法依赖一个排版的“当前状态”，CSS 把这个当前状态称为“格式化上下文 (formatting context) ”。

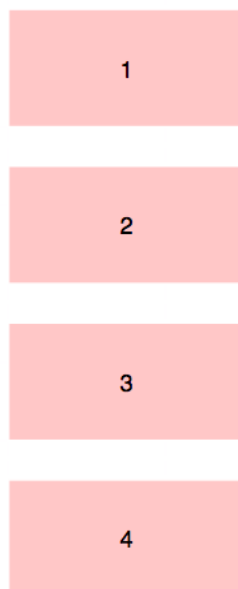
我们可以认为排版过程是这样的：

- 格式化上下文 + 盒 / 文字 = 位置
- formatting context + boxes/charater = positions

我们需要排版的盒，是分为块级盒和行内级盒的，所以排版需要分别为它们规定了块级格式化上下文和行内级格式化上下文。

与正常流一样，如果我们单纯地看格式化上下文，规则其实是非常简单的。

块级格式化上下文顺次排列元素：



行内级格式化上下文顺次排列元素：



注意，块级和行内级元素的排版，受文字书写方向的影响，这里我们讲上下左右只是为了方便你直观理解。

当我们要把正常流中的一个盒或者文字排版，需要分成三种情况处理。

当遇到块级盒：排入块级格式化上下文。

当遇到行内级盒或者文字：首先尝试排入行内级格式化上下文，如果排不下，那么创建一个行盒，先将行盒排版（行盒是块级，所以到第一种情况），行盒会创建一个行内级格式化上下文。

遇到 float 盒：把盒的顶部跟当前行内级上下文上边缘对齐，然后根据 float 的方向把盒的对应边缘对到块级格式化上下文的边缘，之后重排当前行盒。

我们以上讲的都是一个块级格式化上下文中的排版规则，实际上，页面中的布局没有那么简单，一些元素会在其内部创建新的块级格式化上下文，这些元素有：

1. 浮动元素；
2. 绝对定位元素；

2. 绝对定位元素，

3. 非块级但仍能包含块级元素的容器（如 inline-blocks, table-cells, table-captions）；

4. 块级的能包含块级元素的容器，且属性 overflow 不为 visible。

这里的最后一条比较绕，实际上，我个人喜欢用另一种思路去理解它：

自身为块级，且 overflow 为 visible 的块级元素容器，它的块级格式化上下文和外部的块级格式化上下文发生了融合，也就是说，如果不考虑盒模型相关的属性，这样的元素从排版的角度就好像根本不存在。


好了，到这里我们已经讲完了正常流的排版详细规则，但是理解规则仅仅是基础，我们还需要掌握一些技巧。

正常流的使用技巧

现在，我们就一起来动手用实际的例子来研究一下。我们今天来看看等分布局和自适应宽，从这两种经典布局问题入手，一起来探索一下正常流的使用技巧。

等分布局问题

横向等分布局是一个很常见的需求，按照一般的思路，我们可以使用百分比宽度来解决，我们参考以下代码：

 复制代码

```
1 <div class="outer">
2     <div class="inner"></div>
3     <div class="inner"></div>
4     <div class="inner"></div>
5 </div>
6 .inner {
7     width:33.33%;
8     height:300px;
9     display:inline-block;
10    outline:solid 1px blue;
11 }
12
```

在这段 HTML 代码中，我们放了三个 div，用 CSS 给它们指定了百分比宽度，并且指定为 inline-block。

但是这段代码执行之后，效果跟我们预期不同，我们可以发现，每个 div 并非紧挨，中间有空白，这是因为我们为了代码格式加入的换行和空格被 HTML 当作空格文本，跟 inline 盒混排了的缘故。

解决方案是修改 HTML 代码，去掉空格和换行：

```
1 <div class="outer"><div class="inner"></div><div class="inner"></div><div class="inner"
2
```

但是这样做影响了源代码的可读性，一个变通的方案是，改变 outer 中的字号为 0。

```
1 .inner {
2     width:33.33%;
3     height:300px;
4     display:inline-block;
5     outline:solid 1px blue;
6     font-size:30px;
7 }
8 .outer {
9     font-size:0;
10 }
11
```

在某些浏览器中，因为像素计算精度问题，还是会出现换行，我们给 outer 添加一个特定宽度：

```
1 .inner {
2     width:33.33%;
3     height:300px;
4     display:inline-block;
5     outline:solid 1px blue;
6 }
7 .outer {
8     width:101px
9 }
10
```

这个代码在某些旧版本浏览器中会出现换行。为了保险起见，我们给最后一个 div 加上一个负的右 margin：

```
1 .outer {
2     width:101px
3 }
4
5 .inner {
6     width:33.33%;
7     height:300px;
8     display:inline-block;
9     outline:solid 1px blue;
10 }
11
12 .inner:last-child {
13     margin-right: -5px;
14 }
```

```
13     margin-right:-5px;
14 }
15
```


这样就可以解决旧版本浏览器的问题了。

除了使用 inline-block，float 也可以实现类似的效果，但是 float 元素只能做顶对齐，不如 inline-block 灵活。

自适应宽

我们再来说说自适应宽。在 IE6 统治浏览器市场的旧时代，自适应宽（一个元素固定宽度，另一个元素填满父容器剩余宽度）是个经典的布局问题，我们现在就看一下如何使用正常流来解决：


我们首先来看一下问题：

 复制代码

```
1 <div class="outer">
2     <div class="fixed"></div>
3     <div class="auto"></div>
4 </div>
5 .fixed {
6     width:200px;
7 }
8 .fixed, .auto {
9     height:300px;
10    outline:solid 1px blue;
11 }
12
```

这里 fixed 这个 div 宽度已经被指定好，我们需要添加 css 代码尝试让 auto 填满剩余宽度。


使用正常流解决这个问题的思路是，利用负 margin：

 复制代码

```
1 .fixed {
2     display:inline-block;
3     vertical-align:top;
4 }
5 .auto {
6     margin-left:-200px;
7     width:100%;
8     display:inline-block;
9     vertical-align:top;
10 }
11
```

但是，这样做会导致 auto 中的内容位置不对，所以我们还需要使用 padding 把内容挤出来，最

终完整代码如下：

 复制代码

```
1 .fixed {
2     display:inline-block;
3     vertical-align:top;
4 }
5 .auto {
6     margin-left:-200px;
7     padding-left:200px;
8     box-sizing:border-box;
9     width:100%;
10    display:inline-block;
11    vertical-align:top;
12 }
13
```

这样就给 auto 添加了 padding-left 和 box-sizing 两个属性。

总的来说，正常流布局主要是使用 inline-block 来作为内容的容器，利用块级格式化上下文的纵向排布和行内级格式化上下文的横向排布来完成布局的，我们需要根据需求的横向和纵向排布要求，来选择元素的 display 属性。

结语

这次的文章中，我们一起学习了正常流，我们可以用一句话来描述正常流的排版行为，那就是：依次排列，排不下了换行。这也是理解它最简单最源头的方式。

我们将正常流的知识分成了三个部分。

正常流的行为部分，我们从一些感性认知出发，帮助你从思路和源头上理解正常流的行为。

正常流的原理部分，我用更严格的描述方式，给你讲解了 CSS 标准中规定的正常流排版逻辑。

最后的正常流应用部分，我以两个经典布局问题等分布局和自适应宽为例，为你讲解了正常流实际使用的一些技巧。

最后，留给你一个思考题：用 JavaScript 写一个仅包含 inline-block 的正常流布局算法。你写好的话，可以留言给我，我们一起讨论。



重学前端

每天 10 分钟，重构你的前端知识体系

winter 程劭非
前手机淘宝前端负责人



新版升级：点击「👤 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得转载



由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Ctrl + Enter 发表

0/2000字

提交留言

精选留言(2)



Scorpio

我大flex天下第一!!! 😊



2019-03-14



William

1. 等宽布局，不用外层font-size:0的方法的话，应该是.inner:not(last-child) {
margin-right: -5px;
}吧，前面元素均添加一个负外边距抵消掉空格大小。

2. 因为也是用inline-block, 所以自适应宽需要加上

```
.outer {
```

```
    font-size: 0;
```

```
}
```



2019-03-14