深入理解 flex-grow、flex-shrink、flex-basis

蘑菇街前端团队 LV3
2019年12月09日 02:36 · 阅读 14981

1. Flex布局

> Flex 是 Flexible Box 的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性 flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为 0 1 auto

任何一个容器都可以用 Flex 进行布局(如果不会 flex 布局的可见阮老师的 Flex 布局教程)，而且 Flex 是发生在父容器和子容器之间的布局关系的，那么父容器与子容器的关系是怎么样子的，又是怎么计算子容器所占用的空间的呢，怎么进行弹性布局的呢?

欲要解决上面的问题，首先得了解 flex-grow 和 flex-shrink 是怎么计算的？flex-basis 和 width 又有什么关系和区别?

接下来，我们先提出两个概念：***剩余空间和溢出空间***，具体是什么意思我们后面慢慢解释。

2. flex-grow

flex-grow属性在MDN上的定义是：

> 定义弹性盒子项（flex-item）的拉伸因子，默认值0”

传统的布局是子容器在父容器中从左到右进行布局，应用 flex 进行布局，那么父容器一定设置 `display: flex`，子容器要“占有”并且“瓜分”父容器的空间，如何占有、瓜分的策略就是弹性布局的策略。这里就要解释到“剩余空间”的概念：

> 子容器在父容器的“主轴”上还有多少空间可以“瓜分”，这个可以被“瓜分”的空间就叫做剩余空间。

文字总是很抽象，举个例子就能理解剩余空间了，现在有如下的代码：

HTML 代码：

复制代码

```
<div class="container">
    <div class="item a">
      <p>A</p>
      <p> width:100</p>
    </div>
    <div class="item b">
      <p>B</p>
      <p> width:150</p>
    </div>
    <div class="item c">
      <p>C</p>
      <p> width:100</p>
```

```
    </div>
</div>
```

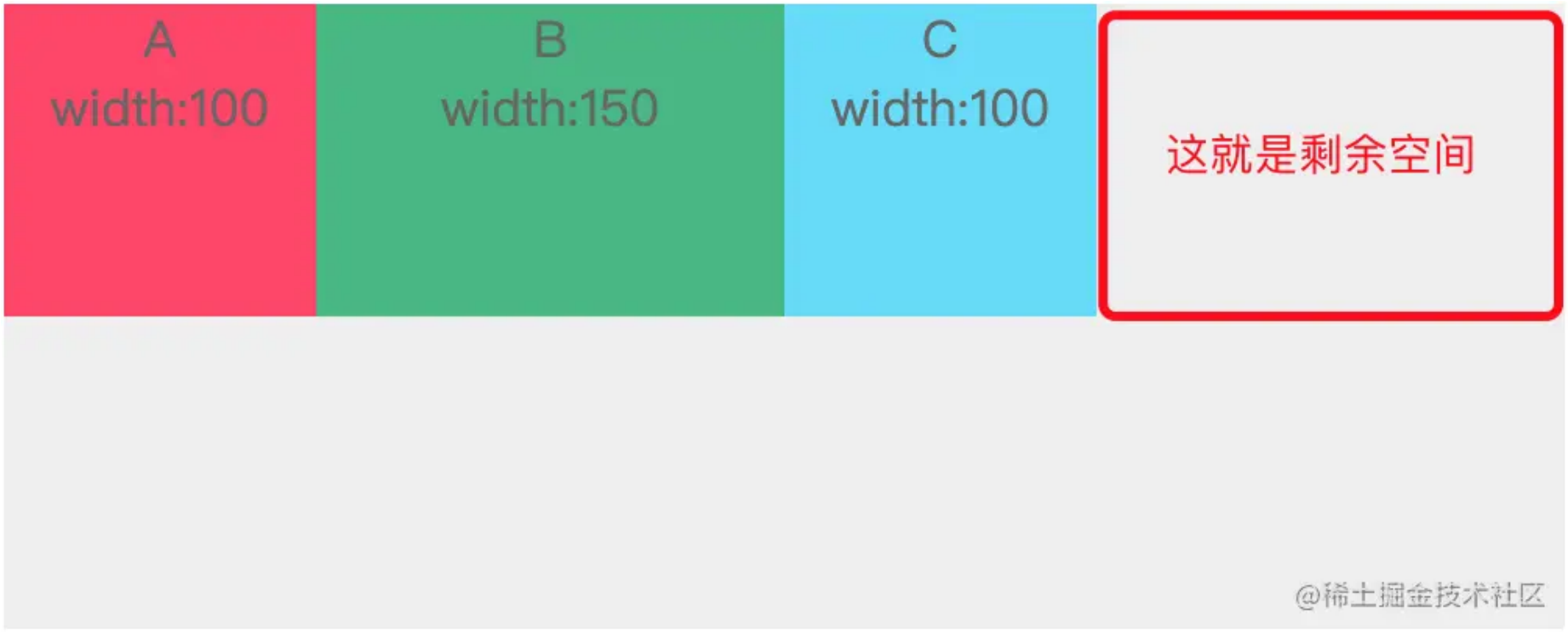
CSS代码：

复制代码

```
.container {
    margin:10px;
    display: flex;
    width: 500px;
    height: 200px;
    background-color: #eee;
    color: #666;
    text-align: center;
}
.item {
    height: 100px;
}
.item p {
  margin: 0;
}
.a{
    width: 100px;
    background-color:#ff4466;
}
.b{
    width: 150px;
    background-color:#42b983;
}
```

```
    background-color:#61dafb;
}
```

展示的效果如下(最后那个框是截图的时候的标注，不是展示出来的效果)：



一图胜千言，看到这个图应该就明白什么是剩余空间了。

父容器的主轴还有这么多剩余空间，子容器有什么办法将这些剩余空间瓜分来实现弹性的效果呢?

`flex-grow` **例子**，将上面的例子改成如下代码：

HTML 代码(代码只增加了 flex-grow 的说明，没有其他结构的变动)：

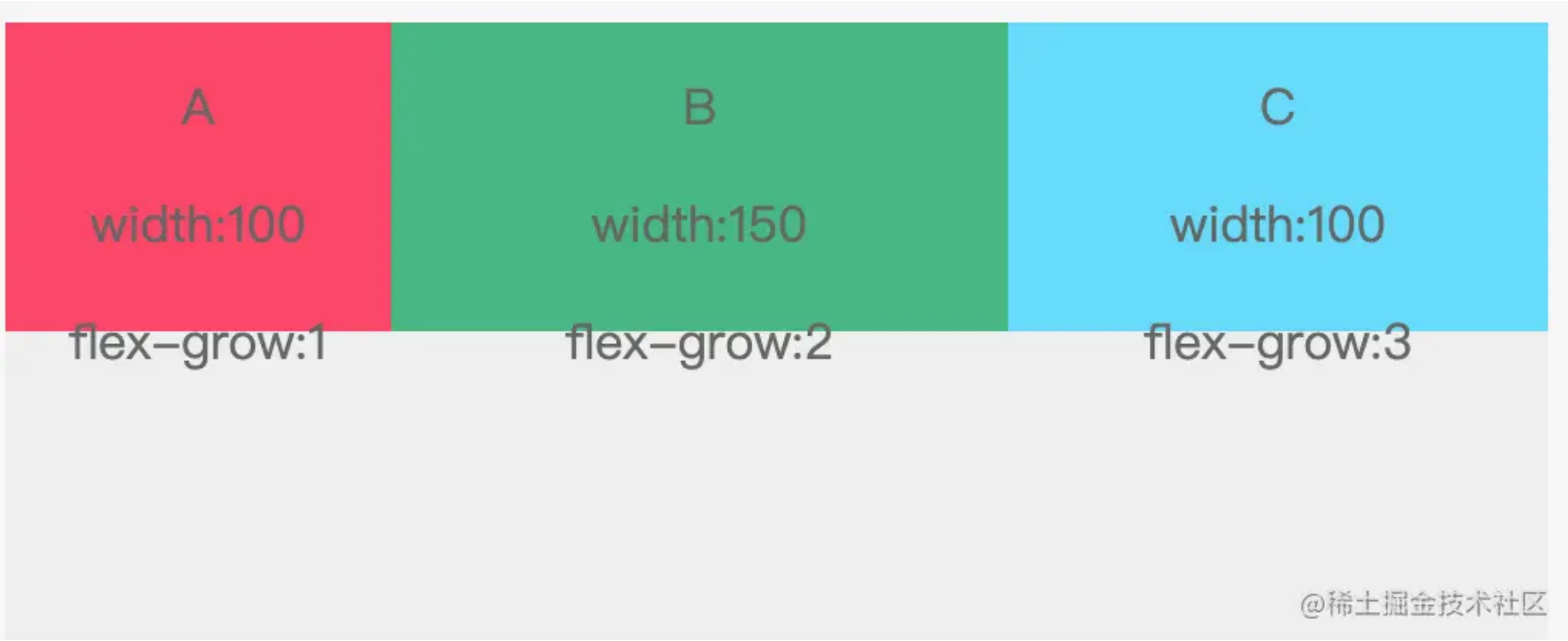
```
复制代码
<div class="container">
    <div class="item a">
      <p>A</p>
      <p> width:100</p>
      <p>flex-grow:1</p>
    </div>
    <div class="item b">
      <p>B</p>
      <p> width:150</p>
      <p>flex-grow:2</p>
    </div>
    <div class="item c">
      <p>C</p>
      <p> width:100</p>
      <p>flex-grow:3</p>
    </div>
</div>
```

CSS 代码（给每个子容器增加了 flex-grow）：

```
复制代码
.container {
    margin:10px;
    display: flex;
```

```
    background-color: #eee;
    color: #666;
    text-align: center;
}
.item {
    height: 100px;
    p {
      margin: 0;
    }
}
.a{
    width: 100px;
    flex-grow:1;
    background-color:#ff4466;
}
.b{
    width: 150px;
    flex-grow:2;
    background-color:#42b983;
}
.c{
    width: 100px;
    flex-grow:3;
    background-color:#61dafb;
}
```

结果如下：



最初，我们发现，子容器的宽度总和只有 350px，父容器宽度为 500px，那么剩余空间就出现了，为 150px。当设置了 `flex-grow` 之后，A，B，C三个子容器会根据自身的 `flex-grow` 去“瓜分”剩余空间。

在这里我们总结为 flex-grow 属性决定了子容器要占用父容器多少剩余空间。

计算方式如下：

- 剩余空间：x
- 假设有三个flex item元素，flex-grow 的值分别为a, b, c

以 A 为例子进行说明： A 占比剩余空间：1/(1+2+3) = 1/6，那么 A “瓜分”到的 `150*1/6=25`，实际宽度为 `100+25=125`。

考虑是否可以把 flex-grow 设置的值小于 1，而且 flex-grow 的和也小于 1 呢？只要把上面公式的分母（flex-grow 的和）设置为 1 就好啦！

3. flex-shrink

说完 flex-grow，我们知道了子容器设置了 flex-grow 有可能会被拉伸。那么什么情况下子容器被压缩呢？考虑一种情况：如果子容器宽度超过父容器宽度，即使是设置了 flex-grow，但是由于没有剩余空间，就分配不到剩余空间了。这时候有两个办法：换行和压缩。由于 flex 默认不换行，那么压缩的话，怎么压缩呢，压缩多少？此时就需要用到 flex-shrink 属性了。

flex-shrink属性在MDN上的定义是：

> 指定了 flex 元素的收缩规则，默认值是 1

此时，剩余空间的概念就转化成了**“溢出空间”**

计算方式：

- 三个flex item元素的width: w1, w2, w3
- 三个flex item元素的flex-shrink：a, b, c
- 计算总压缩权重： sum = a * w1 + b * w2 + c * w3
- 计算每个元素压缩率： S1 = a * w1 / sum，S2 =b * w2 / sum，S3 =c * w3 / sum
- 计算每个元素宽度：width - 压缩率 * 溢出空间

复制代码

```
<div class="container">
   <div class="item a">
     <p>A</p>
     <p> width:300</p>
     <p>flex-shrink: 1</p>
   </div>
   <div class="item b">
     <p>B</p>
     <p> width:150</p>
     <p>flex-shrink: 2</p>
   </div>
   <div class="item c">
     <p>C</p>
     <p> width:200</p>
     <p>flex-shrink: 3</p>
   </div>
</div>
```

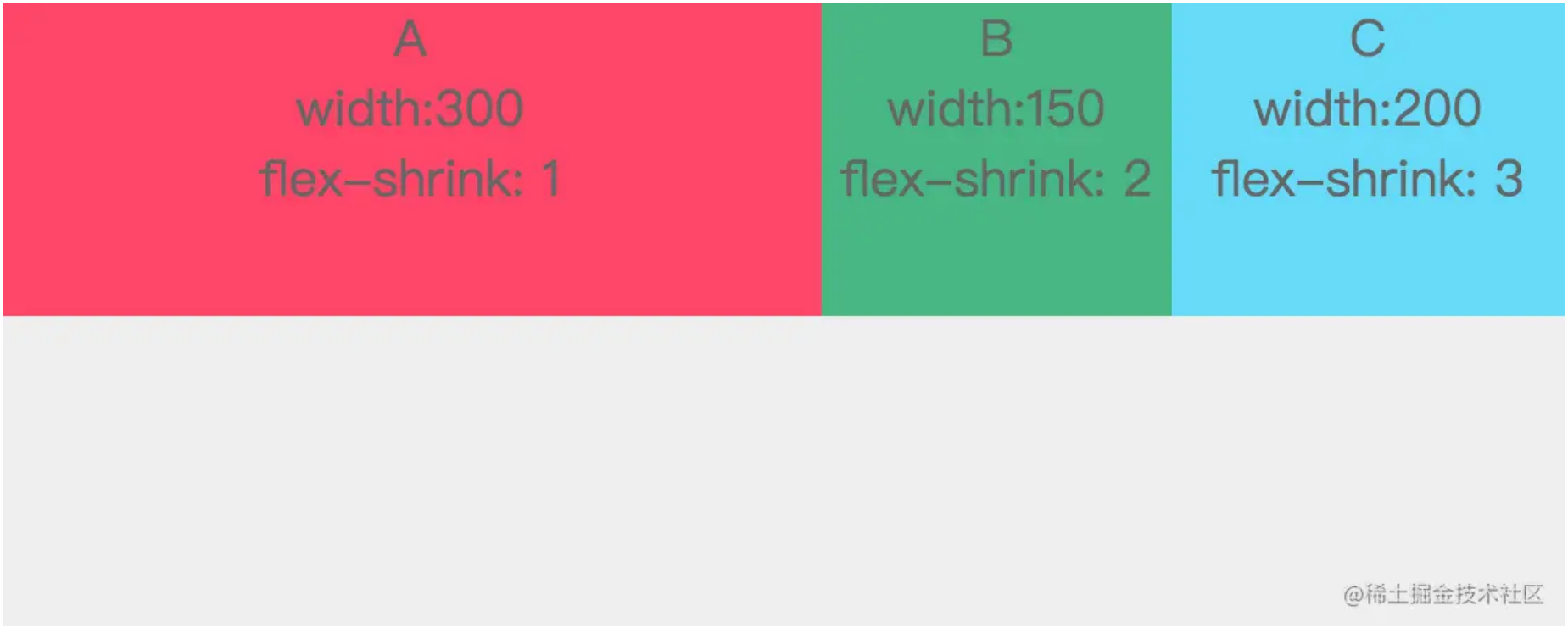
复制代码

```
.container {
        margin:10px;
        display: flex;
        width: 500px;
        height: 200px;
        background-color: #eee;
        color: #666;
        text-align:center;
}
.item {
```

```
.item p {
                margin: 0;
}
.a{
        width: 300px;
        flex-grow: 1;
        flex-shrink: 1;
        background-color:#ff4466;
}
.b{
        width: 150px;
        flex-shrink: 2;
        background-color:#42b983;
}
.c{
        width: 200px;
        flex-shrink: 3;
        background-color:#61dafb;
}
```

子容器宽度总和为650，溢出空间为150 总压缩：300 * 1 + 150 * 2 + 200 * 3 = 1200 A的压缩率：300*1 / 1200 = 0.25 A的压缩值：150 * 0.25 = 37.5 A的实际宽度：300 - 37.5 = 262.5

结果如下：



同样，如果出现flex-shrink总和小于1？那么计算溢出空间（收缩总和）的结果有所变化。比如：shrink设置为0.1, 0.2, 0.3， 原溢出空间为200，实际溢出空间：200 * （0.1 + 0.2 + 0.3）/ 1 = 120。

注意：如果子容器没有超出父容器，设置 flex-shrink 无效

4. flex-basis

> MDN定义：指定了 flex 元素在主轴方向上的初始大小

一旦 flex item 放进 flex 容器，并不能保证能够按照 flex-basis 设置的大小展示。浏览器会根据 flex-basis 计算主轴是否有剩余空间。既然是跟宽度相关，那么 max-width，min-width，width 和 box 的大小优先级是怎么样的。

举例说明：

复制代码

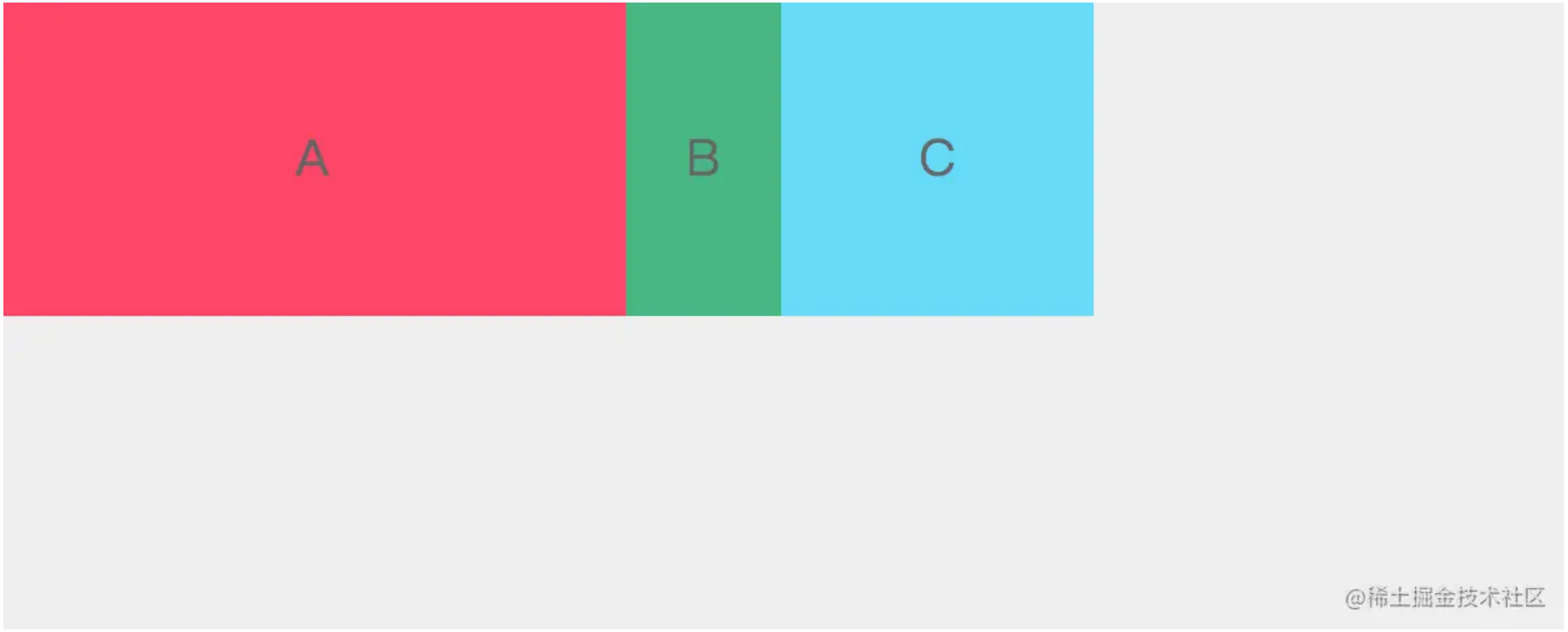
```
<div class="container">
    <div class="item a">A</div>
    <div class="item b">B</div>
    <div class="item c">C</div>
</div>
```

复制代码

```
.container {
    margin:10px;
    display: flex;
    width: 500px;
    height: 200px;
    background-color: #eee;
    text-align: center;
    line-height: 100px;
    color: #666;
}
.item {
    width: 100px;
    height: 100px;
}
.a{
    flex-basis: 200px;
```

```
.b{
    max-width: 50px;
    flex-basis: 150px;
    background-color:#42b983;
}
.c{
    background-color:#61dafb;
}
```

结果如下：

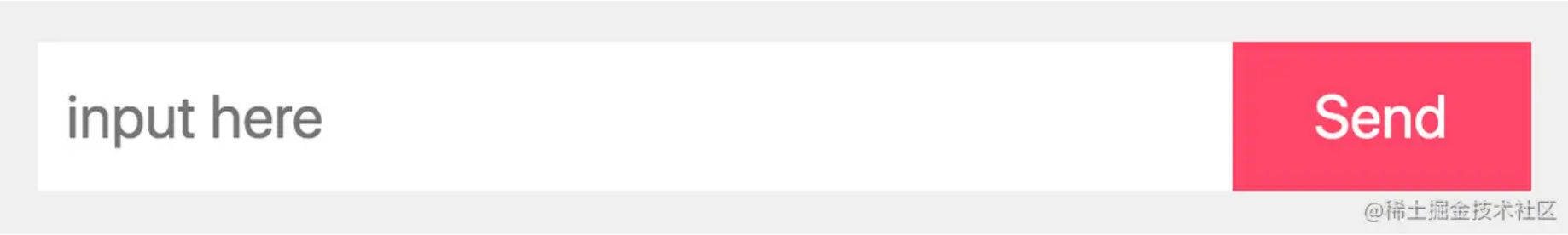


> max-width/min-width > flex-basis > width > box

5. 应用场景

1. 一种很常见的布局：当内容区域高度不够的时候，footer仍然需要固定在底部。这时候，我们可以给main使用flex-grow: 1，使它自动填满剩余空间。

2 . 在我们开发一种常见的表单组件的时候，使用flex布局，可以使输入框占满剩余空间。



而大部分场景下我们不希望元素被压缩，所以flex-shrink通常设置为0。

6. 总结

最后，我们需要注意的是：

- flex items 总和超出 flex 容器，会根据 flex-shrink 的设置进行压缩
- 如果有剩余空间，如果设置 flex-grow，子容器的实际宽度跟 flex-grow 的设置相关。如果没有设置flex-grow，则按照 flex-basis 展示实际宽度

参考文献：

- developer.mozilla.org/zh-CN/docs/...
- developer.mozilla.org/zh-CN/docs/...
- developer.mozilla.org/zh-CN/docs/...

- gedd.ski/post/the-di…
- zhuanlan.zhihu.com/p/24372279

分类： 前端　　标签： CSS

评论

输入评论（Enter换行，Ctrl + Enter发送）

热门评论

ß 小 1年前

如何设置felx-grow 3个item和的值为1呢？ 在父容器中有这样的属性设置吗？

1 1

az22c 前端小工 2年前

max-width/min-width > flex-basis > width > box。最后的box是啥?

1 1

光辉GuangHui Lv2 1年前

max-content

1 回复

全部评论 15

最新 最热

海岸贡献 前端开发工程师 @ Anon... 6月前

好文，讲解形象生动，易于理解

263 15 收藏

阝 忄 1年前

如何设置felx-grow 3个item和的值为1呢？ 在父容器中有这样的属性设置吗?

1 1

莱米 Lv1 1年前

收获很大 👍

点赞 回复

vcxiaohan 1年前

flex: 1为
flex-grow 1
flex-shrink 1
flex-basis 0%

1 回复

az22c 前端小工 2年前

max-width/min-width > flex-basis > width > box。最后的box是啥?

1 1

光辉GuangHui Lv2 1年前

max-content

263 15 收藏

Wei同学 2年前

flex: 1; 能聊聊吗

1 1

feng_cc Lv1 2年前

😄 我也想这么说，后面两个应用场景直接flex:1处理

点赞 回复

鸿蒙OS招聘 Lv1 华为 2年前

大牛，有兴趣来华为一起搞鸿蒙OS吗？目前前端这块有比较多的高端岗位空缺。

点赞 1

1184dac01b041… 2年前

[内容违规]

点赞 回复

TornadoLi 2年前

在提供的例子中，我认为并不能证明优先级关系 flex-basis > width 。class="item a" 本身a的优先级就比item的高

1 回复

263 15 收藏

[内容违规]

点赞 回复

Hitsuki9 接口调用师 2年前

米家龙我男神

点赞 回复

相关推荐

是糖糖啊 | 9月前 | 前端

flex常用属性

558 10 评论

linwanxia | 20天前 | CSS

我用 flex 布局 写了 9 个麻将

1.3w 153 38

蛙人 | 11月前 | CSS

面试官：请说说什么是BFC？大白话讲清楚

3.9w 785 92

263 15 收藏

CSS flex布局踩坑小记：flex-basis属性之0px与0%的差异

466 14 2

答案cp3 | 9月前 | CSS · 前端

你了解flex: 1，flex: auto，flex: 0，flex: none的区别吗?

1046 30 2

ThinkMore28092 | 4年前 | CSS

清除浮动的四种方式及其原理理解

3.9w 407 17

yuxiaoliang | 3年前 | CSS · PostCSS · 浏览器

响应式布局的常用解决方案对比(媒体查询、百分比、rem和vw/vh）

3.1w 714 19

大海我来了 | 1年前 | CSS

1.5 万字 CSS 基础拾遗（核心知识、常见需求）

4.4w 1909 87

手撕红黑树 | 9月前 | 前端 · CSS · HarmonyOS

产品经理：鸿蒙那个开场动画挺帅的 给咱们页面也整一个呗

6.6w 1256 182

为什么要使用flex布局?

3.9w 397 31

一李 | 10月前 | CSS · 前端

flex: 1 flex: auto flex: none flex: 0到底有什么 区别？ 使用场景?

1610 26 1

---

alphardex | 2年前 | CSS

请收下这72个炫酷的CSS技巧

5.1w 1549 80

Running53 | 4月前 | 前端

原来flex布局还能那么细?

2.0w 358 40

IDuxFE | 5月前 | CSS · 前端

深入浅出之 Flex 弹性布局

9180 142 2

---

tiutiu | 1年前 | CSS

明天全国哀悼日，一段css让全站变灰

3.9w 771 100

一文搞懂 flex 属性

1567 10 评论

天明夜尽 | 1年前 | CSS · GitHub

10 个 GitHub 上超火的 CSS 奇技淫巧项目，找到写 CSS 的灵感！

4.2w 1434 38

前端论道 | 2年前 | Vue.js · CSS

【建议收藏】90%的前端都会踩的坑，你中了吗?

5.0w 1637 80

zhangbao90s | 1年前 | CSS

有 width 不就够了吗，为什么还要 flex-basis 呢?

3458 37 4

Jimmy | 1年前 | CSS · 前端

CSS八种让人眼前一亮的HOVER效果

3.9w 1348 95

263 15 收藏