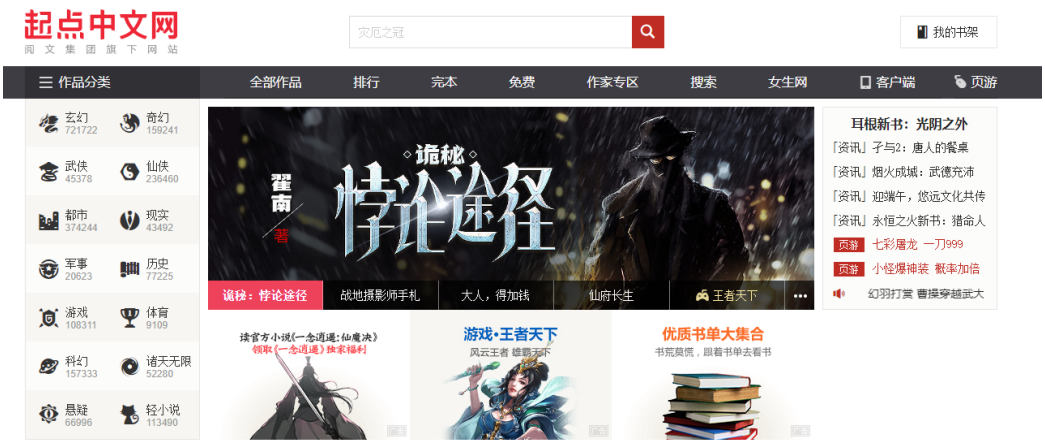
【前端面试】你有几种方式实现页面的三栏...

前言

常见的网页布局有很多种，比如单列布局、等高布局、三栏布局等等。其中三栏布局在很多网站都有用到，这也是一种比较经典的网页布局方式，当然，想要实现网页三栏布局的方式有非常之多，我们今天就主要学习一些哪些方式可以实现三栏布局，这样小伙伴们遇到三栏布局需求的时候就可以根据需求自行选择合适的布局方式了。

1.何为三栏布局

我们可以先来看几个有三栏布局的网站页面：



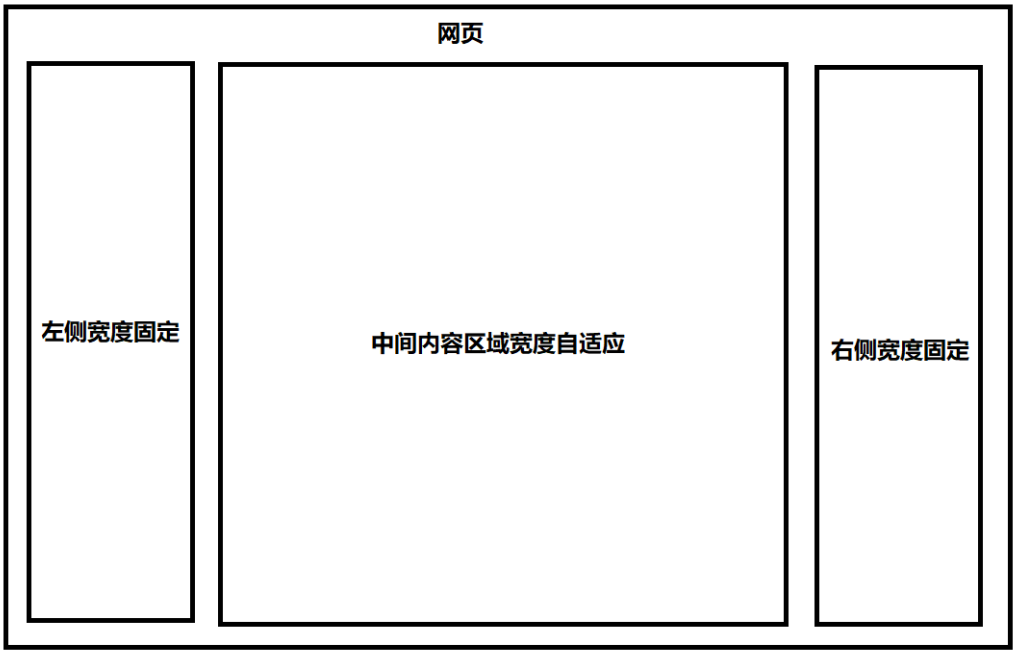
上面两张图都有三栏布局的身影，大家仔细看这两张图，会发现它们的布局的形式差不多类似的。

概念：

> 三栏布局顾名思义有左中右三栏，其中左边列和右边列的宽度都是固定的，中间列的宽度是自适应的。

我们理解三栏布局重点抓住几个关键词即可：**左右宽度固定，中间区域自适应**。

比如我们用示意图就可以很好的表示出来，如下图所示：



2.实现方式

我们既然知道三栏布局是怎么回事了，那么我们就可以尝试着去用不同的方式去实现它了，毕竟这是在项目中很常见的一种布局方式。

2.1 浮动布局

这是一种比较经典的方式，因为它采用了浮动的方式来实现三栏布局，这种方式很好理解，很多初学者都是使用这种方式来实现三栏布局。

实现原理：

- 左栏设置左浮动
- 右栏设置右浮动
- 中间栏设计合适的margin实现自适应

代码如下：

```
<head>
  <style>
```

```
    .left {
      float: left;
      height: 100vh;
      width: 100px;
      background-color: rgb(194, 123, 123);
    }
    .right {
      width: 200px;
      height: 100vh;
      background-color: rgb(69, 69, 92);
      float: right;
    }
    .main {
      margin-left: 120px;
      margin-right: 220px;
      height: 100vh;
      background-color: rgb(95, 214, 95);
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="left"></div>
    <div class="right"></div>
    <div class="main"></div>
  </div>
</body>
```

实现效果：

这种方式主要利用了浮动导致元素脱离文档流的特点，让左右两栏各浮动在两边。

利用浮动布局的方式虽然很简单，但是它也有不少缺点，比如下列几点。

缺点：

- 浮动布局会脱离文档流，容易造成高度塌陷
- 需要手动清楚浮动
- 中间内容区域无法最先加载，因为按照html顺序来看，中间区域是排在最后的。

2.2 浮动布局升级版

上面我们说简单的浮动布局会造成元素脱离文档流，容易造成父元素高度塌陷，这个时候我们需要手动清除浮动。我们可以让元素形成BFC，不了解BFC的建议先去学一学，形成BFC之后我们就可以不用给内容区域设置margin了。

实现原理：

- 左栏设置左浮动
- 右栏设置右浮动
- 中间栏形成BFC

代码如下：

```
<head>
  <style>
    .left {
      float: left;
      height: 100vh;
      width: 100px;
      background-color: rgb(194, 123, 123);
    }
    .right {
      width: 200px;
      height: 100vh;
      background-color: rgb(69, 69, 92);
      float: right;
    }
    .main {
      overflow: hidden;
      height: 100vh;
      background-color: rgb(95, 214, 95);
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="left"></div>
    <div class="right"></div>
    <div class="main"></div>
  </div>
</body>
```

实现效果：

上端代码我们只改变了main区域的样式，让它形成BFC，这样它就不会与浮动元素发生重叠了，不过它的缺点仍然是存在的。

缺点：

- 浮动元素不设置高度会造成高度塌陷
- 内容区域仍然不是最先渲染的

2.3 双飞翼布局

前面两种方式我们都说了渲染顺序的问题，如果页面的内容一旦多起来，就会影响用户的体验，双飞翼布局方式就可以解决该问题。

实现原理：

- 左、中、右三栏都设置为左浮动
- 中间栏宽度设置为100%
- 通过设置左右两栏的margin-left负边距，让三栏保证在同一行

代码如下：

```
<head>
  <style>
    .content {
      float: left;
      width: 100%;
      height: 100vh;
      background-color: rgb(197, 145, 145);
    }
    .left {
      float: left;
      margin-left: -100%;
      height: 100vh;
      width: 100px;
      background-color: rgb(114, 52, 52);
    }
```

```
    .right {
      float: left;
      margin-left: -200px;
      width: 200px;
      height: 100vh;
      background-color: rgb(69, 69, 92);
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="content">
      <div class="center"></div>
    </div>
    <div class="left"></div>
    <div class="right"></div>
  </div>
</body>
```

实现效果：

双飞翼布局将三个元素都浮动了起来，这样我们就可以把中间内容区域排到前面有限渲染，通过设置左右两栏的左负边距，让它们按照规则排列在左右两边，但是双飞翼布局还是有一点点的额缺点的。

缺点：

- 增加了DOM节点，浏览器渲染是生成渲染树的时间会增长，也就是渲染会增长时间

2.4 圣杯布局

圣杯布局其实和双飞翼布局非常的类似，都是利用浮动和负边距来实现三栏布局。但是我们的双飞翼布局有一个缺点，就是DOM节点稍显复杂。我们这里的圣杯布局不经可以做到DOM节点简

单，而且还能做到内容区域优先加载。

实现原理：

- 父级元素分别设置左右外边距，给左右两栏留出空间
- 左中右三栏分别设置左浮动
- 中间栏设置宽度100%
- 左边栏设置左负边距-100%，右边距通过负边距和定位来实现在最右边的效果

代码如下：

```
<head>
  <style>
    /* 留出左右两栏位置 */
    .container {
      margin-left: 100px;
      margin-right: 200px;
    }
    /* 中间栏 */
    .center {
      float: left;
      width: 100%;
      height: 100vh;
      background-color: rgb(197, 145, 145);
    }
    .left {
      float: left;
      margin-left: -100%;
      height: 100vh;
      width: 100px;
      background-color: rgb(114, 52, 52);
      position: relative;
      left: -100px;
    }
    .right {
      float: left;
      margin-left: -200px;
      width: 200px;
      height: 100vh;
      background-color: rgb(69, 69, 92);
      position: relative;
      right: -200px;
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="center"></div>
    <div class="left"></div>
```

```
    <div class="right"></div>
  </div>
</body>
```

实现效果：

上段代码中我们DOM元素布局将中间栏内容区域放在了最前面，然后我们让左右两栏使用定位的方式定位在页面左右两端。试想一下，如果我们不使用定位或者container不给左右两栏留出空间，那么我们这种DOM节点的排列是不会满足要求的，大家可以下来试一试。

缺点：

虽然圣杯布局解决了内容优先渲染以及DOM节点复杂的问题，但是它还是有缺点的，比如它的CSS样式代码比我们前面几种方式要复杂一点，没有那么好理解。

2.5 Flex 布局

如果说你现在还不会使用flex布局，那么只能说明你的基础知识不牢固。flex布局可以说是能够解决我们前端页面布局中的90%的问题，还不熟悉的小伙伴可以去阮一峰老师的博客去学一学。flex布局当然也能够很方便快捷的实现我们的三栏布局。

实现原理：

- 利用flex布局的特有属性进行布局

代码如下：

```
<head>
  <style>
    .container {
      display: flex;
      height: 100vh;
    }
```

```
    .left {
      width: 100px;
      background-color: rgb(194, 123, 123);
    }
    .right {
      width: 200px;
      background-color: rgb(69, 69, 92);
    }
    .main {
      flex: 1;
      background-color: rgb(95, 214, 95);
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="left"></div>
    <div class="main"></div>
    <div class="right"></div>
  </div>
</body>
```

实现效果：

可以看到CSS的代码量一下就减少了很多倍，可见flex布局时如此的优秀。有些小伙伴可能会问：上段代码的DOM节点还是没有让内容优先渲染。其实flex布局是可以将中间栏的DOM节点放在前面的，比如下面的代码：

```
<head>
  <style>
    .container {
      display: flex;
      height: 100vh;
    }
    .main {
```

```
      flex-grow: 1;
      background-color: red;
    }
    .left {
      order: -1;
      flex: 0 1 200px;
      margin-right: 20px;
      background-color: blue;
    }
    .right {
      flex: 0 1 100px;
      margin-left: 20px;
      background-color: green;
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="main"></div>
    <div class="left"></div>
    <div class="right"></div>
  </div>
</body>
```

上段代码既将中间栏内容节点放在了最前面，而且也实现了三栏布局，只不过这就得大家下去好好学习flex布局了。

2.6 table布局

table布局在现阶段来说已经不是那么流行了，毕竟我们有了如此优秀得flex布局，但是我们这里还是提一下，使用table布局也能实现三栏布局。

实现原理：

- 利用flex布局的特有属性进行布局

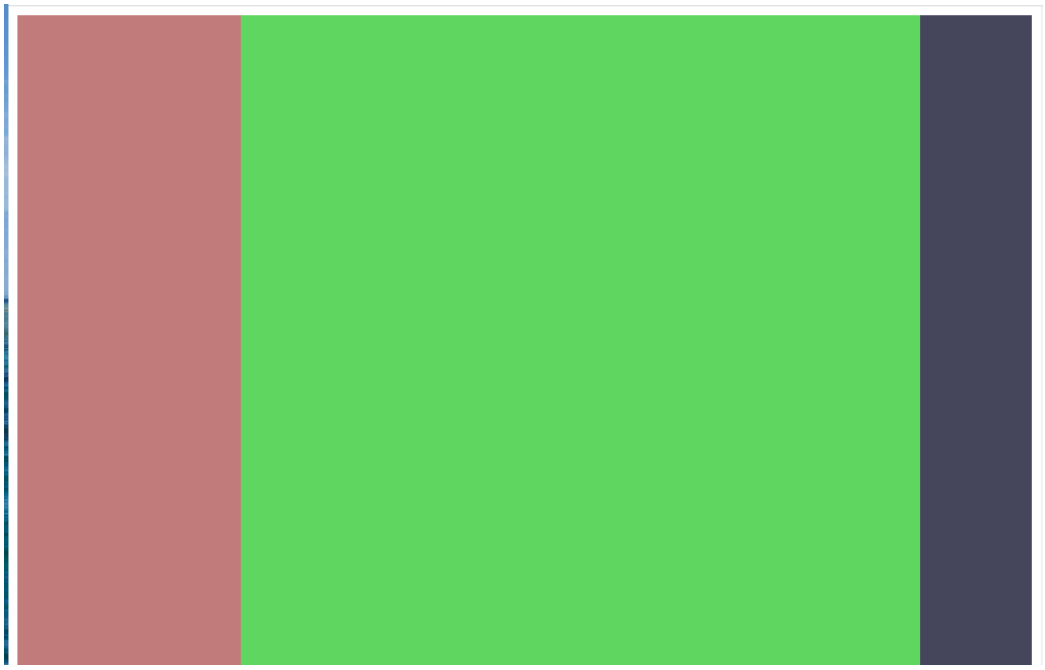
代码如下：

```
<head>
  <style>
    .container {
      display: table;
      width: 100%;
    }
    .left,
    .main,
    .right {
      display: table-cell;
    }
    .left {
```

```
      width: 200px;
      height: 100vh;
      background-color: rgb(194, 123, 123);
    }
    .main {
      background-color: rgb(95, 214, 95);
    }
    .right {
      width: 100px;
      height: 100vh;
      background-color: rgb(69, 69, 92);
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="left"></div>
    <div class="main"></div>
    <div class="right"></div>
  </div>
</body>
```

实现效果：



tab布局得缺点也是显而易见得。

缺点：

table布局无法设置栏间距。

2.7 定位布局

既然我们需要内容区域优先加载，还要保证样式简洁，DOM节点简单，那么绝对定位无疑是一种很好得选择，你想要把元素放在哪儿就放在哪儿，和DOM节点得排序无关。

实现原理：

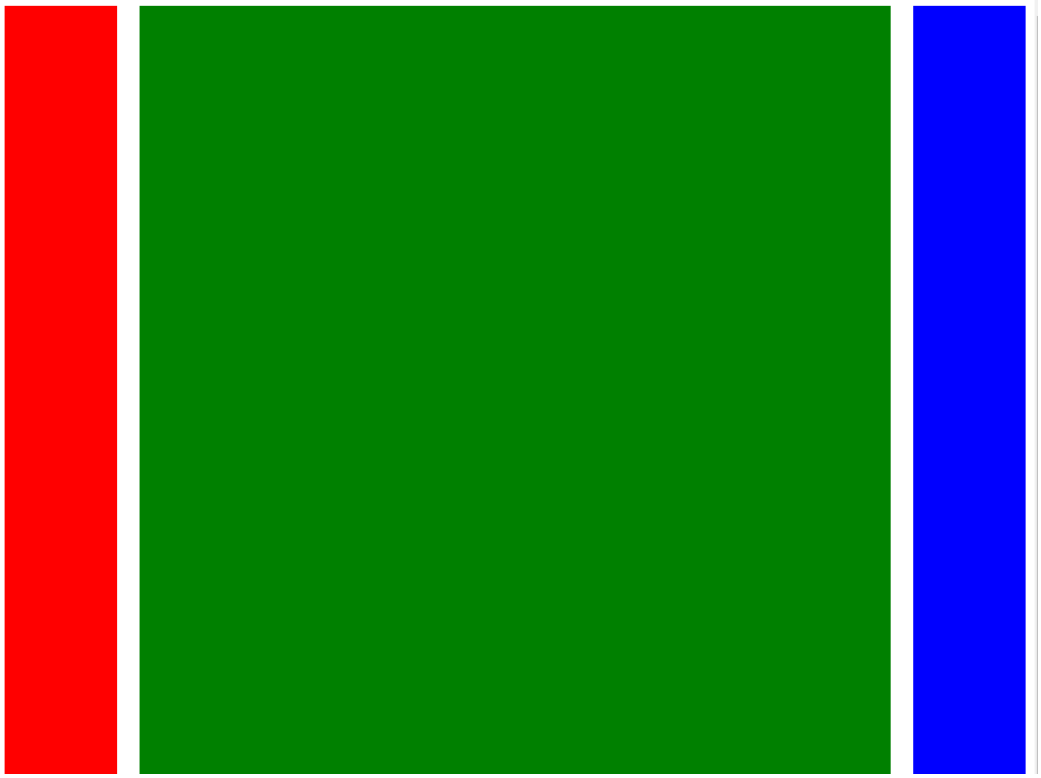
- 左边栏定位到左边
- 右边栏定位到右边
- 中间栏宽度自适应

代码如下：

```
<head>
  <style>
    .container {
      position: relative;
    }
    .main {
      height: 100vh;
      margin: 0 120px;
      background-color: green;
    }
    .left {
      position: absolute;
      width: 100px;
      height: 100vh;
      left: 0;
      top: 0;
      background-color: red;
    }
    .right {
      position: absolute;
      width: 100px;
      height: 100vh;
      background-color: blue;
      right: 0;
      top: 0;
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="main"></div>
    <div class="left"></div>
    <div class="right"></div>
  </div>
</body>
```

实现效果：



可以看到定位的方式非常简单，所需要的CSS知识也不是很多，只需要了解定位即可。

总结

我们每天都在逛各种各样的网页，你是否注意过它们的布局方式呢？今天我们只是讲了三栏布局这种常见的布局方式，实现它的方式就有这么多种，那么你可以下来再仔细思考一下其它布局模式有哪些实现方式。