学习目标

1. 了解各种常见布局
2. 常见布局实战
3. 响应式布局实战和原理解剖
4. 张云鹏（鹏叔）3个小时 8:00-11:00

什么是布局

简单来说就是HTML页面的整体结构或骨架，类似于传统的报纸或杂志中的排版；布局不是某个技术内容，而是一种设计思想；CSS知识体系的重中之重，早期以table为主(简单)，后来以技巧性布局为主(难)，现在有flexbox/grid(偏简单)，响应式布局是必备知识。

几种常见布局

table布局

父级容器—display: table

子级容器—display:table-cell

例子

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      .box {
        width: 600px;
        height: 100px;
        display: table;
      }
      .left,
      .right {
        display: table-cell;
      }
      .left {
        background: yellowgreen;
      }
      .right {
        background: skyblue;
      }
    </style>
  </head>
```

```
  <body>
    <div class="box">
      <div class="left"></div>
      <div class="right"></div>
    </div>
  </body>
</html>
```

float布局

特点:

- 元素"浮动"
- 脱离文档流
- 但不脱离文本流

对自身的影响: 形成"块"(BFC)、位置尽量靠上、位置尽量靠左(右)，无法满足会靠下

> ### 何为BFC
>
> BFC简单理解就是，一块元素区域被赋予了某些样式特性，就会形成一个独立的区域，在此区域内的元素进行margin，padding等改变内部元素样式的操作，不会影响BFC区域外的其他元素。

对兄弟的影响: 上面贴非float元素、旁边贴float元素、不影响其它块级元素位置、影响其它块级元素内部文本

对父级的影响: 从布局上"消失"、高度塌陷(可以通过overflow:hidden | clearfix)

float 布局实战

两列布局解决方案(一边固定，另外一边动态变化)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>两列布局解决方案1 - float+margin</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #left {
        width: 400px; /* 定宽 */
        height: 300px;
        background-color: #c9394a;
```

```
        float: left;
      }

      #right {
        height: 300px;
        background-color: #cccccc;
        margin-left: 400px;
      }
    </style>
  </head>

  <body>
    <div id="left"></div>
    <!-- 自适应容器 -->
    <div id="right"></div>
  </body>
</html>
```

三列布局解决方案

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>三列布局解决方案-float+margin</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #left {
        width: 200px; /* 定宽 */
        height: 300px;
        background-color: #c9394a;
        float: left;
      }

      #center {
        width: 200px; /* 定宽 */
        height: 300px;
        background-color: green;
        float: left;
      }

      #right {
```

```
        height: 400px;
        background-color: #cccccc;
        margin-left: 400px;
      }
    </style>
  </head>

  <body>
    <div id="parent">
      <div id="left"></div>
      <div id="center"></div>
      <div id="right"></div>
    </div>
  </body>
</html>
```

清除浮动的例子

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>清除浮动</title>
    <style>
      .clearfix::after {
        content: '';
        display: block;
        clear: both;
      }
      .float {
        float: left;
        width: 200px;
        height: 200px;
      }
    </style>
  </head>
  <body>
    <div class="clearfix">
      <div class="float" style="background-color: pink;"></div>
      <div class="float" style="background-color: silver;"></div>
    </div>
    <div style="width: 500px; height: 300px; background-color: orange;"></div>
  </body>
</html>
```

inline-block 布局

inline-block 元素会占据一行而且可以调整宽高很适合将这些元素排列在一行，而且使用 inline-block 元素排列没有清除浮动这样的问题。但是，使用 inline-block 布局两个元素之间会有一个空白间隙，下面一起来看一下。

因为现在使用的是 inline-block 元素，为了方便理解，可以将 inline-block 元素看成是两个文字，文字与文字之间不可能是连在一起的，肯定是有间隙的。

既然知道了是文字的问题，那我们就将父元素 container 的字体大小设置为 0，可是这个时候会发现 left 和 right 这两个单词也没有了，这是因为 left 和 right 元素继承了父级元素的字体大小，这时候我们只需要分别设置 left 和 right 元素的字体大小即可。

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      .container {
        width: 800px;
        height: 200px;
        font-size: 0;
      }
      .left {
        font-size: 14px;
        background-color: red;
        display: inline-block;
        width: 200px;
        height: 200px;
      }
      .right {
        font-size: 14px;
        background-color: blue;
        display: inline-block;
        width: 500px;
        height: 200px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="left">
        left
      </div>
      <div class="right">
```

```
        right
      </div>
    </div>
  </body>
</html>
```

flexbox布局

布局的传统解决方案，基于盒状模型，依赖 display 属性 + position属性 + float属性。它对于那些特殊布局非常不方便，比如，垂直居中就不容易实现；2009年，W3C 提出了一种新的方案----Flex 布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。
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；任何一个容器都可以指定为Flex布局，行内元素也可以使用Flex布局; webkit内核的浏览器，必须加上-webkit-前缀。

- 弹性盒子
- 盒子本来就是并列的
- 指定宽度即可

水平竖直居中 flex 实现

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>水平竖直居中布局解决方案-flex+justify-content</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #parent {
        width: 100%;
        height: 400px;
        background: #ccc;
        display: flex;
        align-items: center;
        justify-content: center;
      }

      #child {
        width: 300px;
        height: 200px;
        background: #c9394a;
      }
    </style>
```

```
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child"></div>
    </div>
  </body>
</html>
```

两列布局布局flex的实现

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>flex-两列布局解决方案</title>
    <style>
      .container {
        width: 100%;
        height: 200px;
        display: flex;
      }
      .left {
        background: red;
        width: 200px;
        height: 100%;
      }
      .right {
        flex: 1;
        background: blue;
        height: 100%;
        padding-left: 10px;
      }
    </style>
  </head>

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

```
    </div>
  </body>
</html>
```

Grid布局

网格布局（Grid）是最强大的 CSS 布局方案。
它将网页划分成一个个网格，可以任意组合不同的网格，做出各种各样的布局。以前，只能通过复杂的CSS框架达到的效果，现在浏览器内置了;Grid 布局与 Flex 布局有一定的相似性，都可以指定容器内部多个项目的位置。但是，它们也存在重大区别。Flex 布局是轴线布局，只能指定"项目"针对轴线的位置，可以看作是一维布局。Grid 布局则是将容器划分成"行"和"列"，产生单元格，然后指定"项目所在"的单元格，可以看作是二维布局。Grid 布局远比Flex布局强大。

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      .wrapper {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
        grid-gap: 10px;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div>One</div>
      <div>Two</div>
      <div>Three</div>
      <div>Four</div>
      <div>Five</div>
    </div>
  </body>
</html>
```

columns布局

CSS属性 `columns` 用来设置元素的列宽和列数。

例子

```
<!DOCTYPE html>
```

```html
<html lang="zh_CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      .content-box {
        columns: 3 auto;
      }
    </style>
  </head>
  <body>
    <p class="content-box">
      JavaScript 框架 WHY VUE.JS?起步 GITHUB 易用 已经会了
      HTML、CSS、JavaScript?即刻阅读指南开始构建应用! 灵活
      不断繁荣的生态系统,可以在一个库和一套完整框架之间自如伸.JavaScript 框架
      WHY VUE.JS?起步 GITHUB 易用 已经会了
      HTML、CSS、JavaScript?即刻阅读指南开始构建应用! 灵活
      不断繁荣的生态系统,可以在一个库和一套完整框架之间自如伸.JavaScript 框架
      WHY VUE.JS?起步 GITHUB 易用 已经会了
      HTML、CSS、JavaScript?即刻阅读指南开始构建应用! 灵活
      不断繁荣的生态系统,可以在一个库和一套完整框架之间自如伸.JavaScript 框架
      WHY VUE.JS?起步 GITHUB 易用 已经会了
      HTML、CSS、JavaScript?即刻阅读指南开始构建应用! 灵活
      不断繁荣的生态系统,可以在一个库和一套完整框架之间自如伸.JavaScript 框架
      WHY VUE.JS?起步 GITHUB 易用 已经会了
      HTML、CSS、JavaScript?即刻阅读指南开始构建应用! 灵活
      不断繁荣的生态系统,可以在一个库和一套完整框架之间自如伸.
    </p>
  </body>
</html>
```

常见布局实战

水平居中

- 文本 .parent{text-align:center}
- 单个块级元素 .son{width:1000px(定宽)，margin:0 auto}
- 多个块级元素 .parent{text-align:center} .son{display:inline-block}
- 使用绝对定位: 父相子绝，top、right、bottom、left的值是相对于父元素尺寸的，然后margin或者transform是相对于自身尺寸的，组合使用达到水平居中的目的;
- 任意个元素(flex): #parent{display: flex; justify-content: center; }

文本居中

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>水平居中布局解决方案-text-align</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }
      #child {
        width: 200px;
        height: 200px;
        background: #c9394a;
        text-align: center;
      }
    </style>
  </head>

  <body>
    <!-- 定义子级元素 -->
    <div id="child">居中布局</div>
  </body>
</html>
```

单个块级元素居中

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>单个块级元素居中-margin-auto</title>
    <style>
      #parent {
        width: 100%;
        height: 200px;
        background: #ccc;
      }

      #child {
        width: 200px;
```

```
        height: 200px;
        margin: 0 auto;
        background: #c9394a;
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child">居中布局</div>
    </div>
  </body>
</html>
```

多个块级元素居中

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>多个块级元素居中-text-algin+inline-block</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #parent {
        width: 100%;
        height: 200px;
        background: #ccc;
        text-align: center;
      }

      .child {
        width: 300px;
        height: 200px;
        background: #c9394a;
        display: inline-block;
      }
    </style>
  </head>

  <body>
```

```
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div class="child"></div>
      <div class="child"></div>
      <div class="child"></div>
    </div>
  </body>
</html>
```

使用绝对定位（已知父子宽高）

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>水平居中布局解决方案:绝对定位 transform</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #parent {
        width: 800px;
        height: 200px;
        background: #ccc;
        position: relative;
      }

      #child {
        position: absolute;
        left: 300px;
        height: 200px;
        width: 200px;
        background: #c9394a;
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child">123</div>
    </div>
```

```
  </body>
</html>
```

使用transform

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>水平居中布局解决方案:绝对定位 transform</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #parent {
        width: 800px;
        height: 200px;
        background: #ccc;
        position: relative;
      }

      #child {
        position: absolute;
        left: 400px;
        height: 200px;
        background: #c9394a;
        transform: translateX(-50%);
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child">123</div>
    </div>
  </body>
</html>
```

水平竖直居中布局解决方案-flex+justify-content

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>水平竖直居中布局解决方案-flex+justify-content</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #parent {
        width: 100%;
        height: 400px;
        background: #ccc;
        display: flex;
        align-items: center;
        justify-content: center;
      }

      #child {
        width: 300px;
        height: 200px;
        background: #c9394a;
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child"></div>
    </div>
  </body>
</html>
```

竖直居中方法

- 单行文本 .parent{height:150px;line-height:150px;} 高度等于行高的值；

```
<!DOCTYPE html>
<html lang="en">
```

```
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>高度等于行高</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      .parent {
        height: 150px;
        line-height: 150px;
        background-color: red;
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div class="parent">
      123
    </div>
  </body>
</html>
```

- 多行文本 .parent{height:150px;line-height:30px;} 行高等于height/行数；

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>多行文本/行内元素/行内块级元素</title>
    <style>
      .parent {
        height: 150px;
        width: 30px;
        line-height: 25px;
        background-color: red;
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
```

```
    <div class="parent">
      123 123 123 123 123 123
    </div>
  </body>
</html>
```

- 图片元素：.parent{height:150px;line-height:150px;font-size:0;} .son{vertical-align:middle}

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      .parent {
        height: 150px;
        line-height: 150px;
        font-size: 0;
      }
      .son {
        vertical-align: middle;
        height: 50px;
        width: 50px;
      }
    </style>
  </head>
  <body>
    <div class="parent">
      <img class="son" src="./布局.jpg" alt="" />
    </div>
  </body>
</html>
```

- 单个块级元素:
  - 使用table-cell实现：.parent{display:table-cell;vertical-align:middle}
  - 使用position实现：子绝父相，top、right、bottom、left的值是相对于父元素尺寸的，然后margin或者transform是相对于自身尺寸的，组合使用达到垂直居中的目的；
  - 利用flex实现 .parent{display:flex; align-items: center;}
- 任意个元素：.parent{display:flex; align-items: center;} 或者 .parent{display:flex;flex-direction: column;justify-content: center;}

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
```

```
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>居中</title>
    <style>
      .parent {
        height: 150px;
        display: flex;
        align-items: center;
        background-color: red;
      }
      .son {
        height: 50px;
        width: 50px;
        background-color: #fff;
      }
    </style>
  </head>
  <body>
    <div class="parent">
      <div class="son"></div>
    </div>
  </body>
</html>
```

居中布局(水平+垂直)

- table + margin实现水平方向居中，table-cell + vertical-align实现垂直方向居中

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>居中布局解决方案-table+margin-auto</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #parent {
        width: 1000px;
        height: 600px;
        background: #ccc;
        /* td */
        display: table-cell;
```

```
        /* 垂直对齐 - 居中 */
        vertical-align: middle;
      }

      #child {
        width: 200px;
        height: 200px;
        background: red;
        margin: 0 auto;
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child"></div>
    </div>
  </body>
</html>
```

- postion + transform

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>居中布局解决方案-position定位</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }

      #parent {
        /* 开启定位 */
        position: relative;
        width: 1000px;
        height: 600px;
        background: #ccc;
      }

      #child {
        position: absolute;
        top: 50%;
```

```
        left: 50%;

        /* left: 0;
        right: 0;
        top: 0;
        bottom: 0; */
        margin: auto;
        width: 200px;
        height: 200px;
        background: #c9394a;

        /* transform:translate: ; */
        transform: translate(-50%, -50%);
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child"></div>
    </div>
  </body>
</html>
```

- flex + align-items + justify-content

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>居中布局解决方案-flex+justify-content+align-items</title>
    <style>
      * {
        margin: 0;
        padding: 0;
      }
      /* flex会控制子元素的排列方式 - 水平排列*/
      #parent {
        width: 1000px;
        height: 600px;
        background: #ccc;
        display: flex;
        /* 水平排列 */
        justify-content: center;
```

```
        /* 垂直排列 */
        align-items: center;
      }

      #child {
        width: 200px;
        height: 200px;
        background: red;
      }
    </style>
  </head>

  <body>
    <!-- 定义父级元素 -->
    <div id="parent">
      <!-- 定义子级元素 -->
      <div id="child"></div>
    </div>
  </body>
</html>
```

圣杯布局

圣杯布局是来源于该布局效果类似圣杯而得名。简单来说，就是指三行三列布局；
圣杯布局核心：主要是实现中间主体部分中的左右定宽+中间自适应的布局效果；

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>圣杯布局</title>
    <style>
      * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
      }
      #header,
      #footer {
        background: gray;
        text-align: center;
        height: 60px;
        line-height: 60px;
      }
      .column {
        position: relative;
```

```
        height: 200px;
        float: left;
      }
      #left {
        width: 200px;
        margin-left: -100%;
        background-color: blue;
        left: -200px;
      }
      #right {
        width: 200px;
        margin-left: -200px;
        background-color: yellow;
        right: -200px;
      }
      #center {
        width: 100%;
        background-color: red;
      }
      #container {
        padding: 0 200px;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <body>
      <div id="header">#header</div>
      <div id="container">
        <div id="center" class="column">#center</div>
        <div id="left" class="column">#left</div>
        <div id="right" class="column">#right</div>
      </div>
      <div id="footer">#footer</div>
    </body>
  </body>
</html>
```

双飞翼布局

双飞翼布局最早是淘宝团队提出，是针对圣杯布局的优化解决方案。
主要是优化了圣杯布局中开启定位的问题。

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
  <meta http-equiv="X-UA-Compatible" content="ie=edge" />
  <title>双飞翼布局</title>
  <style>
    * {
      margin: 0;
      padding: 0;
    }

    #parent {
      height: 500px;
    }

    #left,
    #center,
    #right {
      float: left;
    }

    #left,
    #right {
      width: 300px;
    }
    #left {
      background-color: #c9394a;
      margin-left: -100%;
    }

    #center {
      width: 100%;
      background-color: green;
    }

    #right {
      background-color: #cccccc;
      margin-left: -300px;
    }
    #inner {
      height: 300px;
      background-color: hotpink;
      margin-left: 300px;
      margin-right: 300px;
    }
  </style>
</head>

<body>
  <!-- 在圣杯基础的结构的基础上,中间容器里面嵌套了一个子容器,解决了圣杯用定位进行再次移动的问题
双飞翼布局,没有定位进行位置的移动,简化了圣杯的代码(定位)
-->
```

```
    <div id="parent">
      <div id="center">
        <div id="inner">inner</div>
      </div>
      <div id="left">left</div>
      <div id="right">right</div>
    </div>
  </body>
</html>
```

全屏布局

全屏布局就是指HTML页面铺满整个浏览器窗口，并且没有滚动条。而且还可以跟着浏览器的大小变化而变化；

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>全屏布局解决方案-position+fixed</title>
    <style>
      /* position:absolute; => 绝对实现 */
      * {
        margin: 0;
        padding: 0;
      }

      header {
        height: 100px;
        background-color: gray;
        /* 固定在头部 - 定位元素的宽度是元素本身内容的宽度 */
        position: fixed;
        top: 0;
        /* 宽度撑开 - width:100% */
        left: 0;
        right: 0;
      }

      .content {
        background-color: blue;
        /* 高度撑开 */
        position: fixed;
        top: 100px;
        bottom: 100px;
        /* 宽度撑开 */
        left: 0;
        right: 0;
```

```
        /* 隐藏 - auto   overflow-x  overflow-y     */
      }

      .content .left {
        width: 300px;
        height: 100%;
        background-color: red;
        /* 定宽 300 */
        position: fixed;
        top: 100px;
        bottom: 100px;
        left: 0;
      }

      .content .right {
        background-color: yellow;

        /* 自适应  */
        position: fixed;
        /* 宽度撑开 */
        left: 300px;
        right: 0;
        /* 高度撑开 */
        top: 100px;
        bottom: 100px;
      }

      footer {
        height: 100px;
        background-color: lightslategray;
        /* 固定底部 */
        position: fixed;
        bottom: 0;
        left: 0;
        right: 0;
      }
    </style>
  </head>

  <body>
    <header></header>
    <div class="content">
      <div class="left"></div>
      <div class="right"></div>
    </div>
    <footer></footer>
  </body>
</html>
```

响应式布局实战和原理解剖

早年设计Web时，页面是以适配特定的屏幕大小为考量创建的。如果用户正在使用比设计者考虑到的更小或者更大的屏幕，那么结果从多余的滚动条，到过长的行和没有被合理利用的空间，不一而足。随着人们使用的屏幕尺寸的种类越来越多，出现了响应式网页设计的概念（*responsive web design，RWD*），RWD指的是允许Web页面适应不同屏幕宽度因素等，进行布局和外观的调整的一系列实践。

Bootstrap 实战

bootstrap简介
一个CSS框架，twitter出品，提供基础样式、常用组件和JS插件;

- 简单灵活可用于架构流行的用户界面和交互接口的html、css、javascript工具集。
- 基于html5、css3的bootstrap，具有大量的诱人特性：友好的学习曲线，卓越的兼容性，响应式设计，12列格网，样式向导文档。
- 自定义JQuery插件，完整的类库，基于Less等；

Bootstrap优缺点

- 优点:CSS代码结构合理 现成的样式可以直接用
- 缺点：定制较为繁琐 体积大

快速开始

入门模版

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <!-- 必须的 meta 标签 -->
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <!-- Bootstrap 的 CSS 文件 -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
    />

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, 你好</h1>

    <!-- JavaScript 文件是可选的。 -->

    <!-- 包含 JS 插件 的 Bootstrap 集成包 -->
```

```
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js">
</script>
  </body>
</html>
```

例子

Bootstrap 布局例子

容器

网格

网格实例

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/css/bootstrap.min.css"
    />
    <title>Bootstrap 表单例子</title>
  </head>
  <body>
    <div class="container-fluid">
      <h2 class="col-sm-6">注册</h2>
      <div class="col-sm-6">
        <label class="col-sm-4 col-form-label">姓名</label>
        <div>
          <input name="name" type="text" class="form-control" />
        </div>
      </div>
      <div class="col-sm-6">
        <label class="col-sm-4 col-form-label">密码</label>
        <div>
          <input name="password" type="password" class="form-control" />
        </div>
      </div>
      <div class="col-sm-6">
        <label class="col-sm-4 col-form-label">电话</label>
```

```
        <div>
          <input name="cellphone" type="text" class="form-control" />
        </div>
      </div>
      <div class="col-sm-6">
        <label class="col-sm-4 col-form-label">地址</label>
        <div>
          <input name="address" type="text" class="form-control" />
        </div>
      </div>
    </div>
  </body>
</html>
```

网格

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <!-- <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.0/dist/css/bootstrap.min.css" /> -->
    <link rel="stylesheet" href="./my-bootstrap.css" />
    <title>Bootstrap responsive</title>
    <style>
      .content > div {
        height: 100px;
        line-height: 100px;
        text-align: center;
        color: white;
        background: gray;
        margin-bottom: 10px;
      }
    </style>
  </head>

  <body>
    <!--
        https://v5.bootcss.com/docs/layout/grid/
        网格 12等分
    -->
    <div class="container">
      <div class="row">
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
```

```
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
        <div class="content col- col-sm-6 col-md-4 col-lg-3">
          <div>内容</div>
        </div>
      </div>
    </div>
  </body>
</html>
```

使用自定义bootstrap

使用@media媒体查询可以针对不同的媒体类型定义不同的样式，特别是响应式页面，可以针对不同屏幕的大小，编写多套样式，从而达到自适应的效果 .

```
.content > div {
  float: left;
  height: 100px;
  line-height: 100px;
  text-align: center;
  color: #333;
  background: #cccccc;
  margin-bottom: 10px;
  border: 10px solid #fff;
  box-sizing: border-box;
}
.container {
  padding: 0 100px;
}

@media screen and (max-width: 576px) {
  /* col- */
```

```
  .content > div {
    width: 100%;
  }
}
@media screen and (min-width: 576px) and (max-width: 768px) {
  /* col-ms-6 */
  .content > div {
    width: 50%;
  }
}
@media screen and (min-width: 768px) and (max-width: 992px) {
  /* col-md-4 */
  .content > div {
    width: 33.33%;
    background-color: yellow !important;
  }
}
@media screen and (min-width: 992px) and (max-width: 1200px) {
  /* col-lg-4 */
  .content > div {
    width: 25%;
  }
}
@media screen and (min-width: 1200px) {
  /* col-xl */
  .content > div {
    width: 25%;
  }
}
```