AIGC

第7章 核心布局：盒模型

盒模型是网页布局的核心基础，它定义了网页中的每个元素在页面中所占据的空间结构。本章将对CSS3盒模型及其布局进行介绍，包括盒模型外边距设置、内边距设置、尺寸设置、内容溢出、传统盒模型布局以及现代盒模型布局。

7.1 盒模型的基础

盒模型是CSS中的一个核心概念，是实现网页布局的基础。本节将对盒模型进行介绍。

7.1.1 认识盒模型

CSS盒模型将每个HTML元素抽象为一个由内容区、内边距、边框与外边距嵌套构成的矩形结构，如图7-1所示。它不仅定义了这4部分如何共同决定元素的最终尺寸与空间占用，还规范了元素外边距与相邻元素的交互规则，是实现网页精准布局的底层基础。

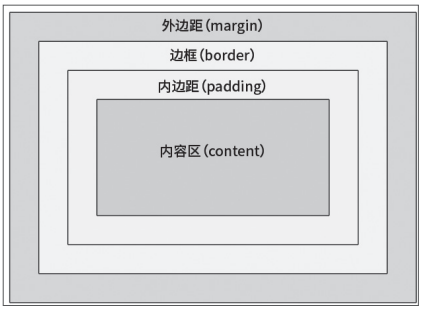

图 7-1

盒模型各结构的介绍如下。

- **内容区**。元素实际内容（文本、图片等）区域。
- **内边距**。内容与边框之间的空白区域。
- **边框**。包裹内边距的线条。
- **外边距**。边框外的空白区域，用于分隔元素与其他元素。

7.1.2 盒子的基本类型

根据display属性可以将盒子分为块级盒、行内盒、行内块盒、弹性盒、网格盒等不同的类型。CSS3常用的盒子类型如表7-1所示。

表7-1

类型	描述	典型场景
none	元素完全隐藏，不占用布局空间	条件性隐藏元素（如弹窗关闭）
block	块级元素，独占一行，宽高 / 内外边距可自由设置	页面分区（头部、内容区）
inline	行内元素，与其他行内元素同行，无法设置宽高	文本内的小元素（标签、链接）
inline-block	行内块元素，同行显示且可自由设置宽高 / 内外边距	图标、导航项、表单控件等需要同行但需控制尺寸的元素
list-item	列表元素，默认带项目符号，独占一行	列表项（ ）
flex	弹性盒容器，子元素按一维布局（现代布局核心）	导航栏、自适应容器
grid	网格盒容器，子元素按二维行列布局（现代布局核心），包含 grid（块级）和 inline-grid（行内）	商品列表、表单网格、卡片网格、仪表板、复杂布局系统等
inherit	继承父元素的 display 属性值	统一子元素布局规则

7.1.3 外边距设置

外边距用于定义元素边框与周围元素/容器的空白距离，使用margin属性设置，属性值单位可以为长度单位或百分比，取值可以为正值或负值。同时，外边距还有专门设置某一方向上外边距的属性，如表7-2所示。

表7-2

属性	定义
margin	简写属性。在一个声明中设置所有的外边距属性
margin-top	设置元素的上外边距
margin-right	设置元素的右外边距
margin-bottom	设置元素的下外边距
margin-left	设置元素的左外边距

margin属性的示例用法如下：

```
margin:10px 15px 5px 20px;
```

这段代码表示上外边距为10px，右外边距为15px，下外边距为5px，左外边距为20px。使用单方向属性精准设置，其示例用法如下：

```
margin-top: 10px;
margin-right: 15px;
margin-bottom: 5px;
margin-left: 20px;
```

当margin取值有重复时，可以进行简写，示例用法如下：

```
margin: 5px; /* 4个方向的外边距均为5px */
margin: 5px 15px; /* 上下外边距为5px，左右外边距为15px */
margin: 10px 20px 15px; /* 上外边距为10px、右外边距为20px、下外边距为15px、左外边距为
20px */
```

7.1.4 外边距合并

外边距合并（Margin Collapsing）是CSS盒模型的核心特性之一，它主要是指在垂直方向上，相邻的两个或多个块级元素的外边距在满足特定条件时会合并为一个外边距，合并后外边距的大小等于各发生合并外边距中的最大值。外边距合并主要发生在满足以下条件的垂直方向。

- 处于同一块级格式化上下文中的块级元素。
- 仅作用于垂直外边距（margin-top/margin-bottom）。
- 元素之间没有边框、内边距、行内内容或BFC创建属性等隔断。

打断以上任一条件，都将限制外边距合并。

外边距合并具有以下常见场景。

- **兄弟元素合并**。两个垂直相邻的块级兄弟元素，上元素的下外边距与下元素的上外边距会合并。
- **父子元素合并**。当父元素没有边框、内边距或内容分隔时，父元素的上外边距会与第一个子元素的上外边距合并；下外边距会与最后一个子元素的下外边距合并。
- **空元素自身合并**。一个没有边框、内边距、高度和内容的空块级元素，其自身的上、下外边距会合并。

提示 只有普通文档流中块框的垂直外边距会发生外边距合并，行内框、浮动框或绝对定位之间的外边距不会合并。

7.1.5 内边距设置

内边距是盒模型中内容区域与边框之间的空白区域，使用padding属性设置。其属性值可以是长度单位、百分比或全局值，其中百分比值是相对于包含块的宽度计算的。与外边距不同，内边距的取值不可以为负值，负值会被浏览器视为无效声明。padding属性的示例用法如下：

```
p{padding: 10px;}
```

该代码表示所有p元素的各边都有10px的内边距。还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

```
p{padding: 10px 0.25em 2ex 20%;}
```

这段代码等同于以下代码：

```
p {
padding-top: 10px;
padding-right: 0.25em;
padding-bottom: 2ex;
padding-left: 20%;
}
```

7.1.6 盒模型尺寸

CSS3盒模型的尺寸是页面布局的核心基础，其基础尺寸属性如表7-3所示。

表7-3

属性	作用
width/height	设置盒模型基础尺寸
min-width/max-width	限制最小 / 最大宽度，优先级高于 width
min-height/max-height	限制最小 / 最大高度

盒模型的尺寸由box-sizing属性决定计算规则，涵盖内容区、内边距、边框、外边距4个维度的尺寸控制，该属性具有两个属性值。

1. content-box（标准盒模型）

该属性值为默认值，width和height属性仅代表内容区尺寸，内边距和边框会额外扩大元素的总渲染尺寸。其计算公式如下：

```
元素总宽度（渲染尺寸）= width（内容区） + padding-left + padding-right + border-
left + border-right
元素总高度（渲染尺寸）= height（内容区） + padding-top + padding-bottom + border-
top + border-bottom
页面占用总空间 = 元素总宽度/高度 + margin 对应方向值
```

该属性的示例用法如下：

```
.box {
```

```
    box-sizing: content-box;
    width: 400px;     /* 仅内容区宽度 */
    padding: 10px;    /* 左右各10px，上下各10px */
    border: 2px solid #ccc; /* 左右各2px，上下各2px */
    margin: 5px;      /* 仅影响间距，不计入元素自身尺寸 */
}
```

最终渲染宽度=400px+10px × 2+2px × 2=424px，高度同理。

2. border-box（边框盒模型）

该属性值为现代布局首选，width和height属性包含内容区+内边距+边框，设置的宽高即为元素最终的渲染尺寸，无须手动计算。其计算公式如下：

```
内容区宽度 = width - padding-left - padding-right - border-left - border-right
内容区高度 = height - padding-top - padding-bottom - border-top - border-bottom
页面占用总空间 = 设置的 width/height + margin 对应方向值
```

该属性的示例用法如下：

```
.box {
    box-sizing: border-box;
    width: 400px;     /* 包含内容区+padding+border */
    padding: 10px;
    border: 2px solid #ccc;
}
```

最终渲染宽度为400px（内容区宽度=400px-10px × 2-2px × 2=376px），高度同理。

7.2 盒模型的属性设置

本节将对盒模型常用的属性进行介绍。

7.2.1 内容溢出

内容溢出是指元素内容超出其指定的宽/高、内边距和边框范围的情况，开发者可以使用overflow属性指定如何显示溢出内容。overflow属性可以控制元素水平和垂直方向的溢出行为，其属性值如表7-4所示。

表7-4

属性值	说明
visible	默认值，溢出内容直接显示在元素盒模型外（不裁剪）
hidden	裁剪溢出内容，溢出部分完全隐藏
scroll	裁剪溢出内容，但显示滚动条（水平 + 垂直）以便查看，滚动条占用元素内空间
auto	自动：内容溢出时显示对应方向的滚动条，未溢出时不显示（最常用）
clip	裁剪溢出内容，且不允许滚动（类似 hidden，但禁止编程滚动）
inherit	继承父元素的 overflow 属性值

该属性的示例用法如下：

```
.box {
```

```
    width: 300px;
    height: 150px;
    border: 2px solid #000;
    padding: 10px;
    overflow: auto; /* 自动，溢出时显示滚动条 */
}
```

除了overflow属性外，开发者还可以使用overflow-x和overflow-y属性分别控制水平和垂直方向的溢出，其属性值与overflow一致。示例用法如下：

```
.box {
    width: 300px;
    height: 150px;
    overflow-x: hidden; /* 隐藏水平溢出 */
    overflow-y: scroll;   /* 垂直方向始终显示滚动条 */
}
```

7.2.2 自由缩放

resize属性允许用户通过拖曳的方式来修改元素的尺寸，但仅对溢出容器或明确设置尺寸的元素生效。resize的属性值如表7-5所示。

表7-5

属性值	作用
none	默认值，禁止用户缩放元素（所有元素默认此值）
both	允许用户同时水平 + 垂直方向拖曳缩放
horizontal	仅允许水平方向缩放（调整宽度）
vertical	仅允许垂直方向缩放（调整高度）
block	按块级方向缩放（取决于书写模式，如中文竖排时为水平，横排时为垂直）
inline	按行内方向缩放（与 block 相反）

该属性的示例用法如下：

```
.resize-box {
    width: 300px;
    height: 160px;
    padding: 20px;
    border: 1px solid #000;
    overflow: auto; /* 必须设置，否则resize失效 */
    resize: both; /* 允许双向缩放 */
    box-sizing: border-box; /* 避免padding/border影响尺寸计算 */
}
```

7.2.3 外轮廓

外轮廓（outline）在页面中呈现的效果和边框（border）呈现的效果相似，但它不占用盒模型空间，也不影响元素布局。outline属性的语法与border类似，可以拆分为outline-width、outline-style、outline-color三种属性。除此之外，CSS3中还新增了outline-offset属性，以设置偏移。该属性的示例用法如下：

```
.box {
```

```
    outline: 6px double #000;
    outline-offset: 10px;
}
```

该代码等价为：

```
.box {
    outline-width: 6px;
    outline-style: double;
    outline-color: #000;
    outline-offset: 10px;
}
```


扫码看视频

动手练 AIGC助力矩形缩放

下面练习使用盒模型制作一个可缩放的矩形，并使用AIGC工具优化，涉及的知识点包括padding、resize等属性的设计，以及豆包的应用等。

（1）新建一个记事本文档，并输入代码：

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>调整矩形缩放</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            background-color: #f5f7fa;
        }

        .container {
            width: 100%;
            max-width: 800px;
            padding: 20px;
        }

        .title {
            text-align: center;
            margin-bottom: 30px;
            color: #333;
            font-weight: 300;
            font-size: 1.8rem;
        }

        .resizable-rectangle {
            width: 300px;
            height: 200px;
            background-color: #4a6fa5;
            border: 2px solid #2e4a7d;
            border-radius: 8px;
            margin: 0 auto;
```

```
            resize: both;
            overflow: auto;
            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
        }

        .instructions {
            text-align: center;
            margin-top: 30px;
            color: #666;
            font-size: 0.9rem;
            line-height: 1.5;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1 class="title">调整矩形缩放</h1>
        <div class="resizable-rectangle"></div>
        <div class="instructions">
            <p>拖动矩形右下角调整大小</p>
        </div>
    </div>
</body>
</html>
```

（2）保存文档后，修改后缀名称为.html，双击打开，预览效果如图7-2所示。

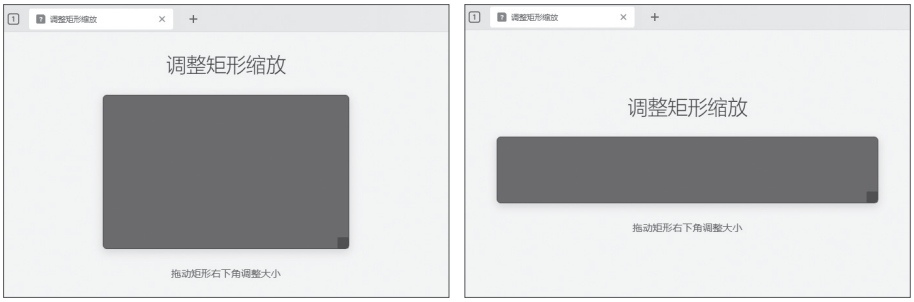

图 7-2

（3）复制代码内容至豆包对话框中，并输入提示词：

基于HTML5和CSS3优化代码，并在下方添加实时显示尺寸。

（4）单击“发送”按钮等待生成，生成内容具有随机性，详见示例文档“rectangle-measure.html”。其预览效果如图7-3所示。

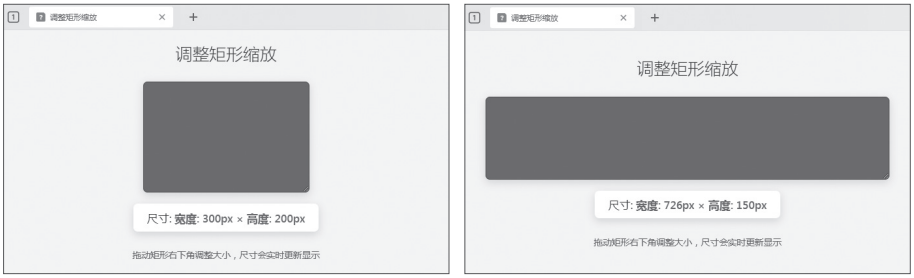

图 7-3

扫码看彩图

至此，我们完成了矩形缩放效果的制作。

7.3 传统盒模型布局

传统CSS布局主要基于元素的display属性类型，通过标准文档流、浮动和定位三种机制进行排列，并使用box-sizing设置尺寸计算方式，解决溢出问题。

7.3.1 标准流布局

标准流布局（也称为文档流/常规流）是HTML元素在没有任何CSS布局干预（如float/position/flex等）时的默认排列方式。它遵循HTML文档的自然流动顺序，基于元素的盒类型形成分层布局结构：块级元素自上而下垂直排列，内联/行内块元素在块级元素的内容区内从左到右水平排列，整体构成页面的基础布局流。

标准流布局常用的元素类型如表7-6所示。

表7-6

元素类型	排列方向	核心特征
块级元素（div、p、h1、footer等）	垂直流（自上而下）	独占一行，宽度默认撑满父容器，高度由内容决定
内联元素（span、a、em、label等）	水平流（从左到右）	同行排列，行宽不足时换行，宽高由内容决定
行内块元素（img、input、button等）	水平流为主	同行排列（内联特性），可手动设置宽高（块级特性）

7.3.2 浮动布局

浮动布局主要通过float属性实现，该属性用于定义元素在哪个方向浮动，其属性值包括以下4种。

- **left**：定义向左浮动。
- **right**：定义向右浮动。
- **none**：默认值，表示元素不浮动，并会显示其在页面中出现的位置。
- **inherit**：继承父元素的float属性值。

当一个元素被设置为浮动元素后，元素本身的属性也会发生一些改变。

- **空间的改变**。块级元素浮动后，宽度从默认撑满父容器收缩为包裹内容的宽度；内联和行内块元素浮动后宽度无收缩变化。若显式设置宽和高，则按指定尺寸显示；若未设置则宽度适配内容、高度由内容和盒模型决定；无尺寸无内容时宽高为0。
- **位置的改变**。浮动元素向左或向右移动，直到外边框边缘触碰到父元素的内边距边缘，或其他浮动元素的外边框边缘。当前行空间不足时，元素下移至足够空间的行，再继续水平浮动。
- **布局环绕**。浮动元素部分脱离标准流，后续非浮动块级元素的盒模型区域仍占据其原始垂直空间，但内容区域会上移并环绕浮动元素排列。浮动元素之间不会相互环绕，仅影响后续非浮动内容。
- **块级化特性**。浮动元素被隐式转为块级元素（display: block），支持手动设置宽或高，且不会像普通块级元素那样占满整行、强制换行，多个浮动元素可在同一行水平排列。
- **上下位置限制**。浮动元素的初始垂直位置由其在HTML中的原始行决定，无法通过float属性直接调整；但可通过margin-top或bottom改变垂直位置，或调整HTML结构、使用定位布局。

float属性的用法示例如下：

```
<!DOCTYPE html>
<html>
<head>
    <style>
        img {
            float: left;
            margin: 10px 15px 10px 5px;
            width: 240px;
        }
        p {
            text-align: justify;
            line-height: 1.6;
        }
    </style>
</head>
<body>
    <img src="01.png" alt="等待加载">
    <p>环滁皆山也。其西南诸峰，林壑尤美，望之蔚然而深秀者，琅琊也。山行六七里，渐闻水声潺潺，而泻出于两峰之间者，酿泉也。峰回路转，有亭翼然临于泉上者，醉翁亭也。作亭者谁？山之僧智仙也。名之者谁？太守自谓也。太守与客来饮于此，饮少辄醉，而年又最高，故自号曰醉翁也。醉翁之意不在酒，在乎山水之间也。山水之乐，得之心而寓之酒也。</p>
    <p>若夫日出而林霏开，云归而岩穴暝，晦明变化者，山间之朝暮也。野芳发而幽香，佳木秀而繁阴，风霜高洁，水落而石出者，山间之四时也。朝而往，暮而归，四时之景不同，而乐亦无穷也。</p>
    <p>至于负者歌于途，行者休于树，前者呼，后者应，伛偻提携，往来而不绝者，滁人游也。临溪而渔，溪深而鱼肥，酿泉为酒，泉香而酒洌，山肴野蔌，杂然而前陈者，太守宴也。宴酣之乐，非丝非竹，射者中，弈者胜，觥筹交错，起坐而喧哗者，众宾欢也。苍颜白发，颓然乎其间者，太守醉也。</p>
    <p>已而夕阳在山，人影散乱，太守归而宾客从也。树林阴翳，鸣声上下，游人去而禽鸟乐也。然而禽鸟知山林之乐，而不知人之乐；人知从太守游而乐，而不知太守之乐其乐也。醉能同其乐，醒能述以文者，太守也。太守谓谁？庐陵欧阳修也。</p>
</body>
</html>
```

上述代码的运行效果如图7-4所示。

图 7-4

7.3.3 定位布局

position属性用于控制网页元素的精确定位方式，该属性具有5种属性值，如表7-7所示。

表7-7

属性值	定位类型	作用与特性
static	静态定位（默认值）	元素按标准文档流正常排列，完全不受 top、right、bottom、left、z-index 影响（z-index 仅对非 static 定位元素生效）。position: static 即无特殊定位，是所有元素的默认定位方式

（续表）

属性值	定位类型	作用与特性
relative	相对定位	元素仍在文档流中，原位置的物理空间完全保留（后续元素不会填补）。可设置 top/right/bottom/left 进行相对于自身原位置的偏移
absolute	绝对定位	元素完全脱离文档流，原位置空间被释放（后续元素会填补）。定位基准为最近的已定位祖先元素（即 position 为 relative/absolute/fixed/sticky 的祖先），且参照该祖先的内边距边缘（padding edge）而非内容区边缘；若无已定位祖先，则相对于根元素（<html>）的视口区域定位。可通过 z-index 控制层叠顺序
fixed	固定定位	元素完全脱离标准文档流，默认相对于浏览器视口（viewport）定位，滚动页面时位置固定不变
sticky	黏性定位	混合定位：在阈值范围内表现为 relative（仍在流中），超过阈值后表现为 fixed（固定定位），但不会脱离父容器的范围。必须指定 top/right/bottom/left 之一作为滚动阈值，且父容器不能有 overflow: hidden/auto/scroll

position属性的用法示例如下：

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>定位布局示例</title>
    <style>
        body {
            height: 200vh; /* 让页面可滚动 */
            margin: 0;
            padding: 20px;
        }

        /* 固定定位 */
        .fixed-box {
            position: fixed;
            bottom: 20px;
            left: 20px;
            background: lightgreen;
            padding: 15px;
        }
    </style>
</head>
<body>
    <div class="fixed-box">固定定位 - 始终在视口左下角</div>
</body>
</html>
```

上述代码的运行效果如图7-5所示。

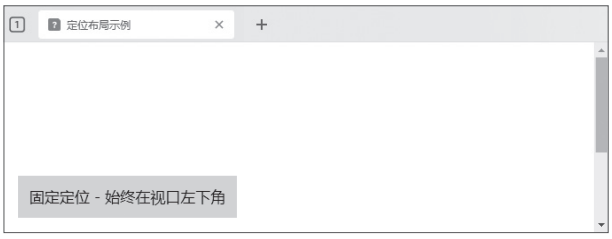

图 7-5

7.4 现代盒模型布局

现代CSS布局以弹性盒（Flexbox）布局和网格（Grid）布局为核心，通过引入全新的布局模型、尺寸单位和对齐系统，从根本上解决了传统布局的复杂性和局限性。

7.4.1 弹性盒布局

CSS3弹性盒（Flexible Box或Flexbox）是一种当页面需要适应不同的屏幕大小以及设备类型时确保元素拥有恰当的行为的布局方式。引入弹性盒布局模型的目的是提供一种更加有效的方式来对一个容器中的子元素进行排列、对齐和分配空白空间。需要注意的是，设为Flex box布局以后，子元素的float、clear和vertical-align属性将失效。

在弹性盒布局中，可以通过对父级元素进行一系列的设置从而起到约束子级元素排列布局的目的。可以对父级元素设置的属性如表7-8所示。

表7-8

属性	作用	属性值
flex-direction	设置主轴方向，控制子项排列方向	● row（默认值）：水平排列，从左到右。 ● row-reverse：水平排列，从右到左。 ● column：垂直排列，从上到下。 ● column-reverse：垂直排列，从下到上
justify-content	控制子项在主轴上的对齐方式	● flex-start（默认值）：主轴起点对齐，剩余空间在末尾。 ● flex-end：主轴终点对齐，剩余空间在开头。 ● center：主轴居中对齐，剩余空间在两侧。 ● space-between：两端对齐，剩余空间仅在项目之间均匀分配（首尾项目贴容器边缘）。 ● space-around：环绕对齐，剩余空间环绕每个项目。 ● space-evenly：均匀对齐，剩余空间均匀分布
align-items	控制子项在交叉轴上的对齐方式	● stretch（默认值）：拉伸填满交叉轴高度。 ● flex-start：交叉轴起点对齐。 ● flex-end：交叉轴终点对齐。 ● center：交叉轴居中对齐。 ● baseline：基线对齐
flex-wrap	控制子项是否换行	● nowrap（默认值）：不换行，所有项目挤在一行。 ● wrap：正常换行，从上到下 / 从左到右。 ● wrap-reverse：反向换行，从下到上 / 从右到左
align-content	控制多行 flex 容器中行与行在交叉轴的对齐方式	● stretch（默认值）：拉伸行填满交叉轴。 ● flex-start：交叉轴起点对齐。 ● flex-end：交叉轴终点对齐。 ● center：交叉轴居中对齐。 ● space-between：两端对齐。 ● space-around：环绕对齐。 ● space-evenly：均匀对齐
flex-flow	flex-direction 和 flex-wrap 的简写	取值格式：[flex-direction] [flex-wrap]，无固定顺序。 默认值：row nowrap（与单独设置默认值一致）。 示例如下。 flex-flow: column wrap;（垂直排列 + 换行）。 flex-flow: row-reverse wrap-reverse;（水平反向 + 反向换行）

除了设置父级容器外，也可以对子级元素进行设置，对子级元素设置的属性如表7-9所示。

表7-9

属性	作用	属性值
flex	控制项目的尺寸与空间分配（flex-grow+flex-shrink+flex-basis 的简写属性，优先级高于单独设置的 width/height）	默认值为 flex: 0 1 auto（不放大、可缩小、基准尺寸自适应内容）。 常用取值（开发中优先用简写）如下。 ● flex: 1 等价于 flex: 110%（等分剩余空间，最常用）。 ● flex: 0 0 200px 等价于不放大、不缩小、基准尺寸为 200px（固定尺寸）。 ● flex: none 等价于 flex: 0 0 auto（完全固定尺寸，不放大不缩小）
order	控制项目的视觉排列顺序（仅改变显示顺序，不改变 HTML 源码顺序，不影响可访问性）	一般为整数（正 / 负 /0，默认值为 0），数值越小越靠前，也可以选择 initial（重置为默认值 0）或 inherit（继承父元素的 order 值）
align-self	覆盖父容器的 align-items，单独控制当前项目在交叉轴的对齐方式（仅作用于当前项目，不影响其他项目）	与 align-items 属性完全一致

7.4.2 网格布局

网格（Grid）布局是一种二维布局系统，它允许开发者同时精确控制行和列的排列。它专为复杂栅格和页面布局设计，提供强大的单元格尺寸控制、跨列/跨行合并能力，以及灵活的内容对齐和间隙管理。

为父元素设置display:grid/inline-grid，可以使其成为网格容器，同时，直接子元素自动变为网格项目，父级元素的核心属性如表7-10所示。

表7-10

属性	作用
grid-template-columns	定义列数与列宽
grid-template-rows	定义行数与行高
gap（row-gap/column-gap）	定义行列间距（不占用元素盒模型空间）
justify-items	控制项目在单元格水平方向对齐，取值包括 start/end/center/stretch（默认值为拉伸）
align-items	控制项目在单元格垂直方向对齐，取值包括 start/end/center/stretch（默认值为拉伸）

子级元素的核心属性如表7-11所示。

表7-11

属性	作用
grid-column	控制项目跨列
grid-row	控制项目跨行
justify-self	单独控制当前项目水平对齐（覆盖容器的 justify-items）
align-self	单独控制当前项目垂直对齐（覆盖容器的 align-items）

动手练 AIGC助力导航栏效果制作


扫码看视频

下面练习使用盒模型及弹性盒制作导航栏，并使用AIGC工具进行优化，涉及的知识点包括盒模型的创建、弹性盒布局及豆包的应用等。

（1）新建一个记事本文档，并输入代码：

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>导航栏</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            background-color: #f5f7fa;
            padding: 40px;
            font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
Roboto, "Helvetica Neue", Arial, sans-serif;
        }

        .navbar {
            display: flex;
            justify-content: flex-end;
            align-items: center;
            background-color: #ffffff;
            border-radius: 10px;
            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.08);
            padding: 0 30px;
            height: 70px;
            border: 1px solid #e1e5eb;
        }

        .nav-links {
            display: flex;
            list-style: none;
        }

        .nav-links a {
            color: #2c3e50;
            text-decoration: none;
            padding: 0 20px;
            height: 70px;
            display: flex;
            align-items: center;
            justify-content: center;
            font-weight: 500;
            transition: all 0.3s ease;
            border-bottom: 3px solid transparent;
        }
```

```
        .nav-links a:hover {
            background-color: rgba(52, 152, 219, 0.08);
            border-bottom: 3px solid #3498db;
            color: #3498db;
        }

        .nav-links a.active {
            background-color: rgba(52, 152, 219, 0.12);
            border-bottom: 3px solid #3498db;
            color: #3498db;
        }
    </style>
</head>
<body>
    <nav class="navbar">
        <div class="nav-links">
            <a href="#" class="active">首页</a>
            <a href="#">产品</a>
            <a href="#">服务</a>
            <a href="#">关于我们</a>
            <a href="#">联系我们</a>
        </div>
    </nav>
</body>
</html>
```

（2）保存文档后，修改后缀名称为.html，双击打开，预览效果如图7-6所示。

（3）复制代码内容至豆包对话框中，并输入提示词：

基于HTML5和CSS3优化代码，添加动画效果。

（4）单击“发送”按钮等待生成，生成内容具有随机性，详见示例文档“nav-yh.html”。其预览效果如图7-7所示。

图 7-6

图 7-7

至此，完成了导航栏的制作。

7.5 案例实战：AIGC优化网格布局效果


扫码看视频

网格布局是当前主流的布局方式，下面练习制作一个网格布局网页，涉及的知识点包括盒模型、网格布局、豆包的应用等。

（1）新建一个记事本文档，并输入代码：

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>网格布局</title>
```

```
<style>
    /* 全局重置与基础样式 */
    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: 'Source Han Sans CN', 'Noto Sans SC', sans-serif;
    }

    body {
        background-color: #f8f9fa;
        color: #333;
        line-height: 1.6;
    }

    /* 网格容器：核心布局 */
    .grid-container {
        display: grid;
        /* 定义网格列：小屏1列，大屏3列（导航1fr + 主体3fr + 侧边1fr） */
        grid-template-columns: 1fr;
        /* 定义网格行：导航、主体、侧边、页脚 */
        grid-template-rows: auto 1fr auto auto;
        /* 命名网格区域，方便布局 */
        grid-template-areas:
            "header"
            "main"
            "sidebar"
            "footer";
        min-height: 100vh; /* 占满视口高度 */
        gap: 1rem; /* 网格间距 */
        padding: 1rem;
    }

    /* 响应式：大屏布局（768px以上） */
    @media (min-width: 768px) {
        .grid-container {
            grid-template-columns: 1fr 3fr 1fr;
            grid-template-rows: auto 1fr auto;
            grid-template-areas:
                "header header header"
                "main main sidebar"
                "footer footer footer";
        }
    }

    /* 各区域样式 */
    .header {
        grid-area: header;
        background-color: #2c3e50;
        color: white;
        padding: 1rem;
        border-radius: 4px;
        text-align: center;
    }

    /* 主图区域样式 - 核心修改 */
    .main-content {
```

```
        grid-area: main;
        background-color: white;
        padding: 0; /* 去除内边距，让图片填满容器 */
        border-radius: 4px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.05);
        overflow: hidden; /* 裁剪图片超出圆角的部分 */
    }

    /* 响应式主图样式 */
    .hero-img {
        width: 100%; /* 宽度自适应容器 */
        height: auto; /* 高度自动，保持比例 */
        object-fit: cover; /* 图片裁剪填充，保持比例 */
        min-height: 280px; /* 移动端最小高度，保证视觉效果 */
        max-height: 500px; /* 桌面端最大高度，避免过高 */
        display: block; /* 去除图片默认行内间隙 */
    }

    .sidebar {
        grid-area: sidebar;
        background-color: white;
        padding: 1.5rem;
        border-radius: 4px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.05);
    }

    .footer {
        grid-area: footer;
        background-color: #2c3e50;
        color: white;
        padding: 1rem;
        border-radius: 4px;
        text-align: center;
        font-size: 0.9rem;
    }

    /* 极简装饰样式 */
    h1, h2, h3 {
        font-weight: 600;
        margin-bottom: 1rem;
    }

    .nav-links {
        list-style: none;
        display: flex;
        justify-content: center;
        gap: 2rem;
        margin-top: 0.5rem;
    }

    .nav-links a {
        color: white;
        text-decoration: none;
        transition: opacity 0.2s;
    }

    .nav-links a:hover {
```

```
            opacity: 0.8;
        }

        .sidebar-item {
            margin-bottom: 1rem;
            padding-bottom: 1rem;
            border-bottom: 1px solid #eee;
        }

        .sidebar-item:last-child {
            border-bottom: none;
        }

        /* 图片下方说明文字的样式 */
        .img-caption {
            padding: 1rem;
            text-align: center;
            font-size: 0.95rem;
            color: #666;
        }
    </style>
</head>
<body>
    <div class="grid-container">
        <!-- 头部导航 -->
        <header class="header">
            <h1>图像空间</h1>
            <ul class="nav-links">
                <li><a href="#">首页</a></li>
                <li><a href="#">分类</a></li>
                <li><a href="#">加入</a></li>
                <li><a href="#">上传</a></li>
            </ul>
        </header>

        <!-- 主体内容区（主图）- 核心修改 -->
        <main class="main-content">
            <img src="01.png" alt="主视觉图" class="hero-img">
            <!-- 图片说明文字 -->
            <p class="img-caption">图像空间 • 网格布局</p>
        </main>

        <!-- 侧边栏 -->
        <aside class="sidebar">
            <h3>图像信息</h3>
            <div class="sidebar-item">
                <p>下载图像</p>
            </div>
            <div class="sidebar-item">
                <p>在线编辑</p>
            </div>
            <div class="sidebar-item">
                <p>作者：一二三</p>
            </div>
        </aside>

        <!-- 页脚 -->
```

```
            <footer class="footer">
                <p>© 2025 图像空间 • 网格布局</p>
            </footer>
        </div>
    </body>
    </html>
```

（2）保存文档后，修改后缀名称为.html，双击打开，预览效果如图7-8所示。

（3）复制代码内容至豆包对话框中，并输入提示词：

基于HTML5和CSS3，将侧边栏“下载图像”和“在线编辑”变成按钮，单独放置。

（4）单击“发送”按钮等待生成，生成内容具有随机性，详见示例文档“grid- yh.html”。其预览效果如图7-9所示。

扫码看彩图

图 7-8

扫码看彩图

图 7-9

至此，完成了网格布局的应用与优化。

7.6 拓展练习

下面练习使用盒模型制作简易的网页布局，如图7-10所示。

扫码看彩图

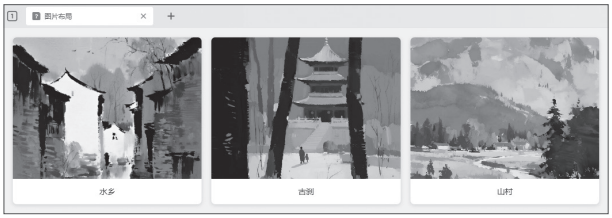

图 7-10

Step

【操作提示】

（1）使用即梦AI生成素材图像，并下载。

（2）使用记事本编写HTML5代码，并修改后缀名称。

AIGC

【思路拓展】

借助AIGC工具，可以丰富网页效果。复制HTML5文档内容并在豆包对话框中输入提示词：基于HTML5和CSS3进行优化，单击图片链接可以打开原图，并添加header和footer区域。豆包会给出相应代码（生成内容具有随机性）。详见示例文档“pic-yh.html”。