第九章 响应式设计与弹性盒子

9.1 ViewPort

9.1.1 什么是viewport?

视口（viewport）是用户在网页上的可见区域。

视口随设备而异，在移动电话上会比在计算机屏幕上更小。

在平板电脑和手机之前，网页仅设计为用于计算机屏幕，并且网页拥有静态设计和固定大小是很常见的。

然后，当我们开始使用平板电脑和手机上网时，固定大小的网页太大了，无法适应视口。为了解决这个问题，这些设备上的浏览器会按比例缩小整个网页以适合屏幕大小。

这并不是完美的！勉强是一种快速的修正。

9.1.2 设置Viewport

HTML5 引入了一种方法，使 Web 设计者可以通过 标签来控制视口。

应该在所有网页中包含以下 `<meta>` 视口元素：

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

它为浏览器提供了关于如何控制页面尺寸和缩放比例的指令。

- width：控制viewport的大小，可以指定一个值。如 600，或者特殊的值，如device-width 为设备的宽度（单位为缩放为 100% 时的 CSS 的像素）。
- height：和width相对应，指定高度
- initial-scale：初始缩放比例，也即是页面第一次load的时候缩放比例
- maximum-scale:允许用户缩放到的最大值
- minximum-scale：允许用户缩放到的最小比例
- user-scaleble：用户是否可以手动缩放

用户习惯在台式机和移动设备上垂直滚动网站，而不是水平滚动！

因此，如果迫使用户水平滚动或缩小以查看整个网页，则会导致不佳的用户体验。

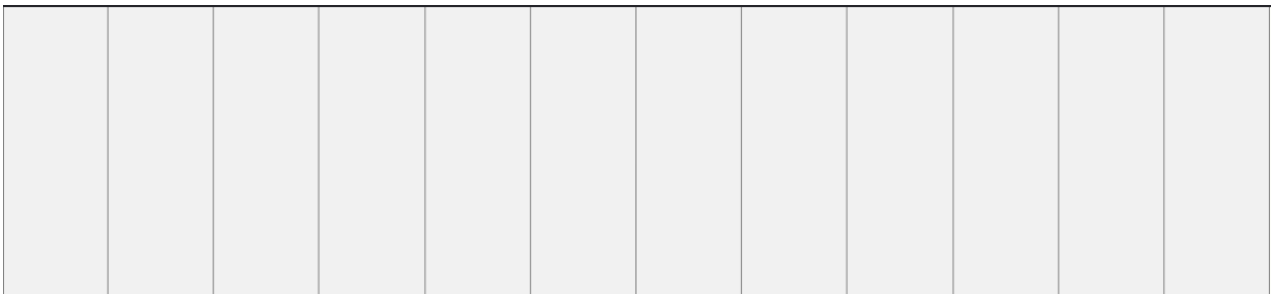
还需要遵循的一些附加规则：

1. 请勿使用较大的固定宽度元素 - 例如，如果图像的宽度大于视口的宽度，则可能导致视口水平滚动。务必调整此内容以适合视口的宽度。
2. 不要让内容依赖于特定的视口宽度来呈现好的效果 - 由于以 CSS 像素计的屏幕尺寸和宽度在设备之间变化很大，因此内容不应依赖于特定的视口宽度来呈现良好的效果。
3. 使用 CSS 媒体查询为小屏幕和大屏幕应用不同的样式 - 为页面元素设置较大的 CSS 绝对宽度将导致该元素对于较小设备上的视口太宽。而是应该考虑使用相对宽度值，例如 width: 100%。另外，要小心使用较大的绝对定位值，这可能会导致元素滑落到小型设备的视口之外。

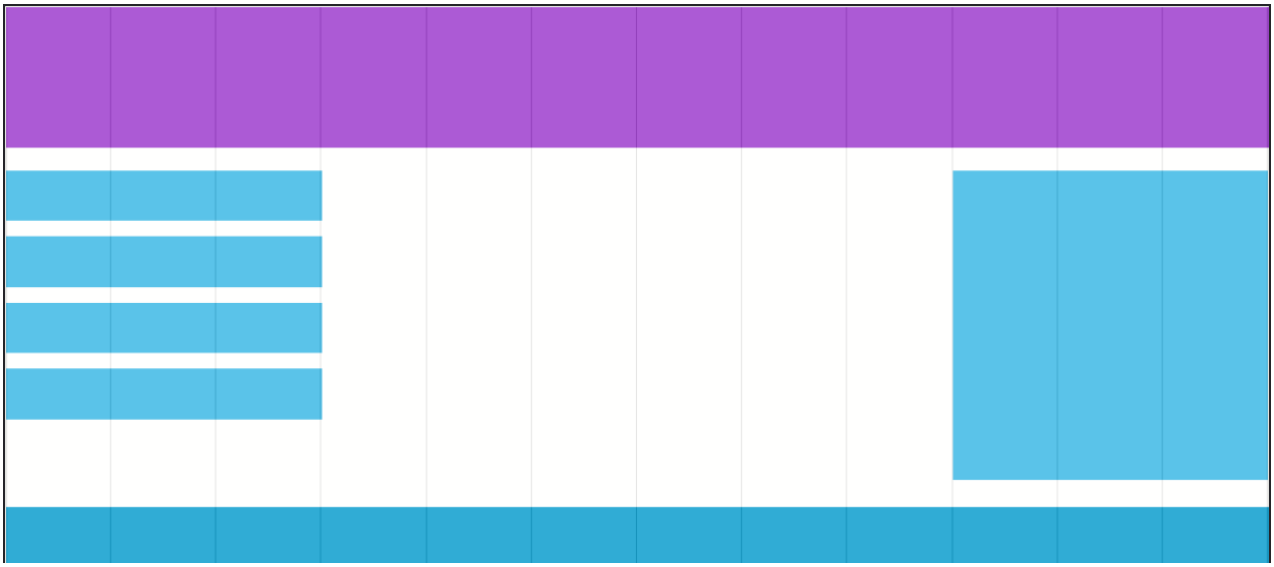
9.2 网格视图

9.2.1 什么是网格视图

许多网页都基于网格视图（grid-view），这意味着页面被分割为几列：



在设计网页时，使用网格视图非常有帮助。这样可以更轻松地在页面上放置元素。



响应式网格视图通常有 12 列，总宽度为 100%，并且在调整浏览器窗口大小时会收缩和伸展。

9.2.2 构建响应式网格视图

接下来我们开始构建响应式网格视图。

首先，确保所有 HTML 元素的 `box-sizing` 属性设置为 `border-box`。这样可以确保元素的总宽度和高度中包括内边距（填充）和边框。

在 CSS 中添加如下代码：

```
* {
  box-sizing: border-box;
}
```

下面的例子展示了一张简单的响应式网页，其中包含两列：

25%	75%

实例：

```
.menu {
  width: 25%;
  float: left;
}
.main {
  width: 75%;
  float: left;
}
```

若网页只包含两列，则上面的例子还不错。

但是，我们希望使用拥有 12 列的响应式网格视图，来更好地控制网页。

首先，我们必须计算一列的百分比：100% / 12 列 = 8.33%。

然后，我们为 12 列中的每一列创建一个类，即 class="col-" 和一个数字，该数字定义此节应跨越的列数：

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

所有这些列应向左浮动，并带有 15px 的内边距：

```
[class*="col-"] {
  float: left;
  padding: 15px;
  border: 1px solid red;
}
```

每行都应被包围在 `<div>` 中。行内的列数总应总计为 12：

HTML

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
```

行内的所有列全部都向左浮动，因此会从页面流中移出，并将放置其他元素，就好像这些列不存在一样。为了防止这种情况，我们会添加清除流的样式：

```
.row::after {
  content: "";
  clear: both;
  display: table;
}
```

我们还想添加一些样式和颜色，使其看起来更美观：

```
html {
  font-family: "Lucida Sans", sans-serif;
}

.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color :#33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
  background-color: #0099cc;

}
```

注意，当我们将浏览器窗口调整为非常小的宽度时，例中的网页看起来并不理想。下面我们将学习如何解决这个问题。

9.3 响应式布局

9.3.1 什么是响应式布局

响应式布局指的是同一页面在不同屏幕尺寸下有不同的布局。传统的开发方式是PC端开发一套，移动端再开发一套，而使用响应式布局只要开发一套就够。

优点：对于不同的视口都可以显示饱满的网页结构，没有横向滚动条。
缺点：制作复杂，需要较重的CSS，对于移动端而言，需要加载很多pc端的样式和图片等资源，影响加载速度，耗费流量。

响应式设计与自适应设计的区别：

> 响应式开发一套界面，通过检测视口分辨率，针对不同客户端在客户端做代码处理，来展现不同的布局和内容；自适应需要开发多套界面，通过检测视口分辨率，来判断当前访问的设备是pc端、平板、手机，从而请求服务层，返回不同的页面。

9.3.2 媒体查询

响应式布局可以在不同的视口下，呈现不同的效果，实现的原理就是通过媒体查询`@media`完成。

媒体查询的使用，有以下几种：

- 可以在引入样式时，限定视口大小
- 直接在样式中，书写不同视口下的样式

在CSS中引入基本语法：

```
/*
  mediatype: 媒体类型
  and|not|only: 逻辑运算符
  media feature: 媒体特性
*/
@media mediatype and|not|only (media feature) {
    ...
}
```

在`style`标签上引入基本语法：

```
<style media="mediatype and|not|only (media feature)">
    ...
</style>
```

使用`@import`时基本语法：

```
@import url(...) mediatype and|not|only (media feature);
```

使用外链式引入CSS时基本语法：

```
<link rel="stylesheet" media="mediatype and|not|only (media feature)" href="...">
```

1.媒体类型

媒体类型描述了设备的类别，是可选的，默认是all类型。具体的媒体类型如下：

类型	描述
all	在所有设备上都加载
print	在打印预览模式下，在屏幕上查看的分页材料和文档
screen	在电脑屏幕，平板电脑，智能手机等其中加载
speech	在屏幕阅读器等发声设备中加载

2.逻辑运算符

逻辑运算符可以被用于组成一个复杂的媒体查询，还可以通过用逗号分隔多个媒体查询，将它们合并为一个规则。具体的逻辑运算符如下：

逻辑运算符	说明
and	用于将多个媒体特征组合到一个媒体查询中，还可以用于将媒体功能与媒体类型连接起来，相当于“且”的意思。
not	用于媒体查询取反值，表示排除某个媒体类型，相当于“非”的意思
only	指定某个特定的媒体类型

3.媒体特性

媒体特性描述了`user agent`、输出设备，或是浏览环境的具体特征。媒体特性表达式是可选的，它负责测试这些特性或特征是否存在、值为多少。每条媒体特性表达式都必须用括号括起来。常用的特性如下：

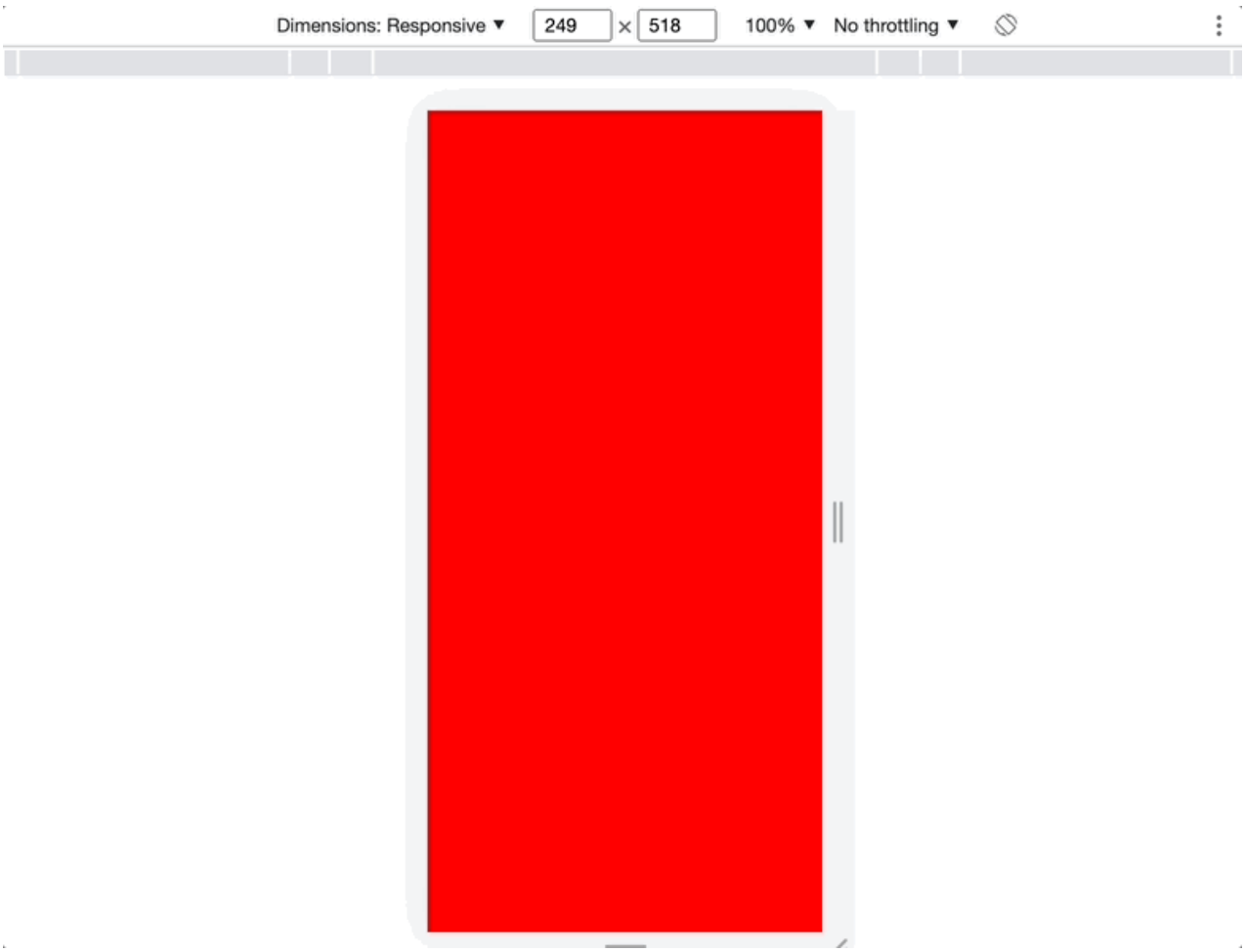
媒体特性	说明
height	定义输出设备中的页面可见区域高度
min-height	定义输出设备中的页面最小可见区域高度
max-height	定义输出设备中的页面最大可见区域高度
device-height	定义输出设备的屏幕可见高度
min-device-height	定义输出设备的屏幕的最小可见高度
max-device-height	定义输出设备的屏幕可见的最大高度
width	定义输出设备中的页面可见区域宽度
min-width	定义输出设备中的页面最小可见区域宽度
max-width	定义输出设备中的页面最大可见区域宽度
device-width	定义输出设备的屏幕可见宽度
min-device-width	定义输出设备的屏幕最小可见宽度
max-device-width	定义输出设备的屏幕最大可见宽度

媒体特性	说明
color	定义输出设备每一组彩色原件的个数。如果不是彩色设备，则值等于0
max-color	定义输出设备每一组彩色原件的最大个数
min-color	定义输出设备每一组彩色原件的最小个数
color-index	定义在输出设备的彩色查询表中的条目数。如果没有使用彩色查询表，则值等于0
min-color-index	定义在输出设备的彩色查询表中的最小条目数
max-color-index	定义在输出设备的彩色查询表中的最大条目数
aspect-ratio	定义输出设备中的页面可见区域宽度与高度的比率
min-aspect-ratio	定义输出设备中的页面可见区域宽度与高度的最小比率
max-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最大比率
device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的比率
min-device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最小比率
max-device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最大比率
grid	用来查询输出设备是否使用栅格或点阵
monochrome	定义在一个单色框架缓冲区中每像素包含的单色原件个数。如果不是单色设备，则值等于0
max-monochrome	定义在一个单色框架缓冲区中每像素包含的最大单色原件个数
min-monochrome	定义在一个单色框架缓冲区中每像素包含的最小单色原件个数
orientation	定义输出设备中的页面可见区域高度是否大于或等于宽度。
resolution	定义设备的分辨率。如：96dpi, 300dpi, 118dpcm
max-resolution	定义设备的最大分辨率
min-resolution	定义设备的最小分辨率
scan	定义电视类设备的扫描工序

4.案例：根据页面宽度改变背景色

页面宽度 <= 400时，背景是红色；
400 < 页面宽度 <= 600 时，背景是橙色；
页面宽度 > 600时，背景是黄色；

```
<style>
    @media screen and (max-width: 400px) {
        body {
            background-color: red;
        }
    }

    @media screen and (min-width: 401px) and (max-width: 600px) {
        body {
            background-color: orange;
        }
    }

    @media screen and (min-width: 601px) {
        body {
            background-color: yellow;
        }
    }
</style>
```



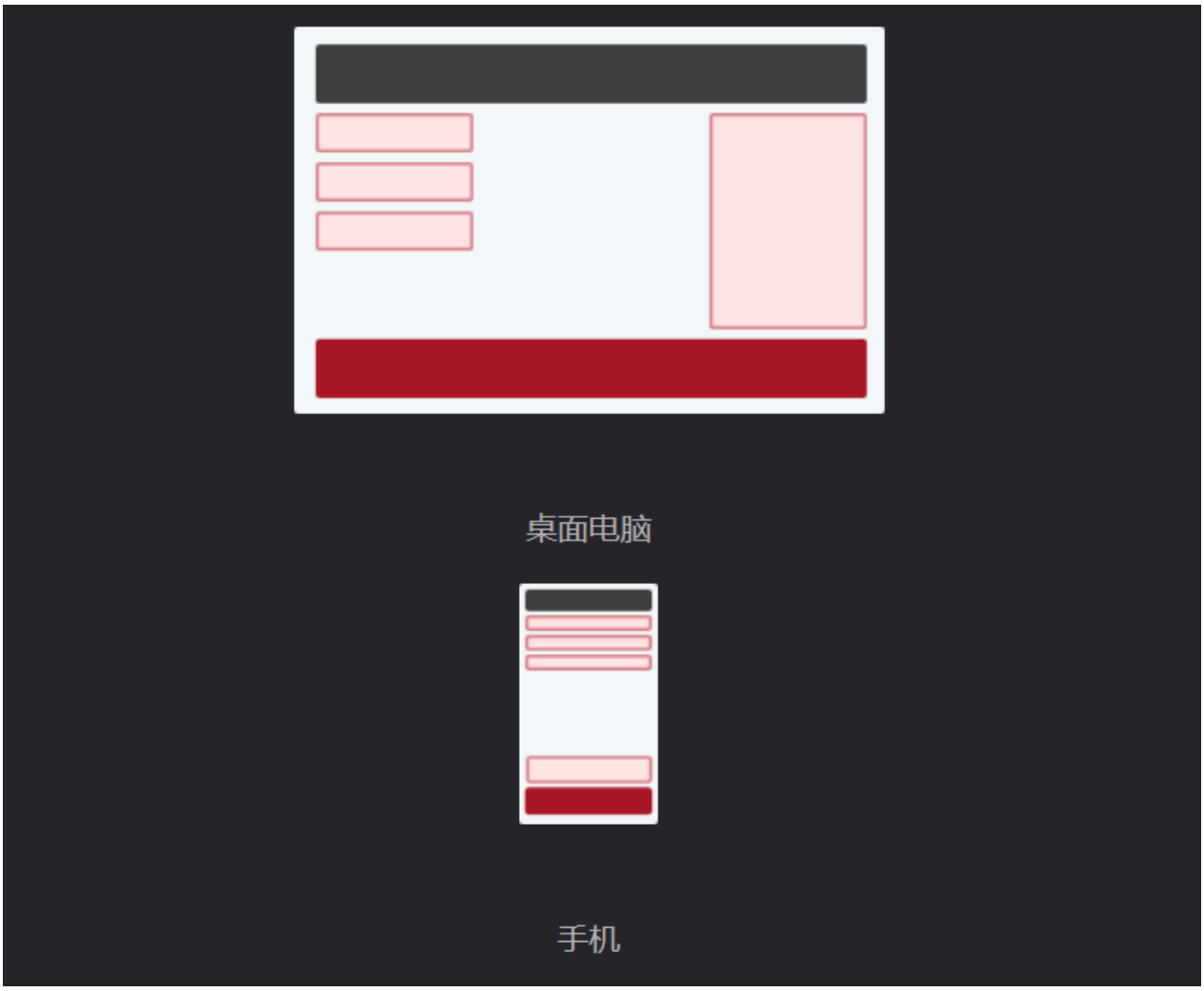
5.案例：浏览器窗口是 **600px** 或更小，则背景颜色为浅蓝色

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

6.添加断点

稍早前，我们制作了一张包含行和列的网页，但是这张响应式网页在小屏幕上看起来效果并不好。

媒体查询可以帮助我们。我们可以添加一个断点，其中设计的某些部分在断点的每一侧会表现得有所不同。



使用媒体查询在 768px 处添加断点：

实例:当屏幕（浏览器窗口）小于 768px 时，每列的宽度应为 100％

```
/* 针对桌面设备: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {
  /* 针对手机: */
  [class*="col-"] {
    width: 100%;
  }
}
```

7.始终移动优先设计

移动优先（Mobile First）指的是在对台式机或任何其他设备进行设计之前，优先针对移动设备进行设计（这将使页面在较小的设备上显示得更快）。

这意味着我们必须在 CSS 中做一些改进。

当宽度小于 768px 时，我们应该修改设计，而不是更改宽度。我们就这样进行了“移动优先”的设计：

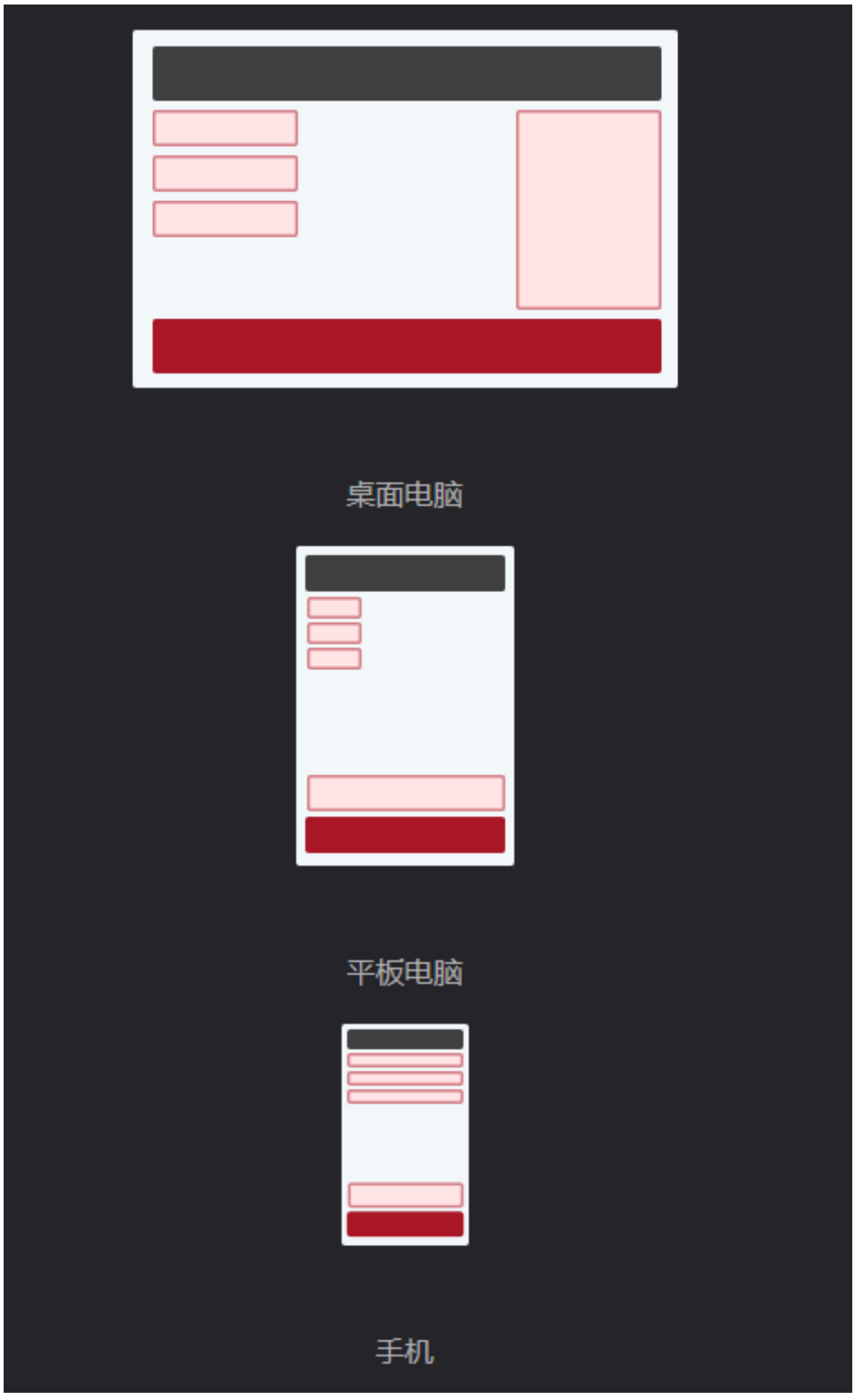
```
/* 针对手机: */
[class*="col-"] {
  width: 100%;
}

@media only screen and (min-width: 768px) {
  /* 针对桌面: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
```

```
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

8.另一个断点

可以添加任意多个断点。

我们还会在平板电脑和手机之间插入一个断点。



为此，我们添加了一个媒体查询（在 600 像素），并为大于 600 像素（但小于 768 像素）的设备添加了一组新类：

```
/* 针对手机:  */
[class*="col-"] {
  width: 100%;
}

@media only screen and (min-width: 600px) {
  /* 针对平板电脑:  */
  .col-s-1 {width: 8.33%;}
  .col-s-2 {width: 16.66%;}
  .col-s-3 {width: 25%;}
  .col-s-4 {width: 33.33%;}
  .col-s-5 {width: 41.66%;}
  .col-s-6 {width: 50%;}
  .col-s-7 {width: 58.33%;}
  .col-s-8 {width: 66.66%;}
  .col-s-9 {width: 75%;}
  .col-s-10 {width: 83.33%;}
  .col-s-11 {width: 91.66%;}
  .col-s-12 {width: 100%;}
}

@media only screen and (min-width: 768px) {
  /* 针对桌面:  */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

对于台式机：

第一和第三部分都会跨越 3 列。中间部分将跨越 6 列。

对于平板电脑：

第一部分将跨越 3 列，第二部分将跨越 9 列，第三部分将显示在前两部分的下方，并将跨越 12 列：

```
<div class="row">
  <div class="col-3 col-s-3">...</div>
  <div class="col-6 col-s-9">...</div>
  <div class="col-3 col-s-12">...</div>
</div>
```

9.典型的设备断点

高度和宽度不同的屏幕和设备不计其数，因此很难为每个设备创建精确的断点。为了简单起见，可以瞄准这五组：

实例：

```
/* 超小型设备（电话，600px 及以下） */
@media only screen and (max-width: 600px) {...}

/* 小型设备（纵向平板电脑和大型手机，600 像素及以上） */
@media only screen and (min-width: 600px) {...}

/* 中型设备（横向平板电脑，768 像素及以上） */
@media only screen and (min-width: 768px) {...}

/* 大型设备（笔记本电脑/台式机，992px 及以上） */
@media only screen and (min-width: 992px) {...}

/* 超大型设备（大型笔记本电脑和台式机，1200px 及以上） */
@media only screen and (min-width: 1200px) {...}
```

9.4 弹性盒子

CSS3新增了一种新型的弹性盒子模型。通过弹性盒子模型，我们可以轻松地创建自适应浏览器窗口的“流动布局”以及自适应字体大小的弹性布局，使得响应式布局的实现更加容易。

此外记住一点：在使用弹性盒子模型之前，你必须为父元素定义`display:flex;`或`display:inline-flex;`，这样父元素才具有弹性盒子模型的特点。

9.4.1 flex属性

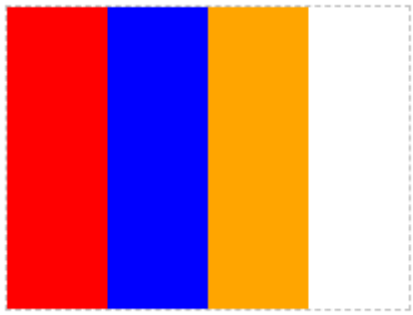
属性	说明
flex-grow	定义子元素的放大比例
flex-shrink	定义子元素的缩小比例
flex-basis	定义子元素的宽度，替代width属性
flex	复合属性，flex-grow、flex-shrink、flex-basis的简写
flex-direction	定义子元素的排列方向
flex-wrap	定义子元素是单行显示，还是多行显示
flex-flow	复合属性，flex-direction、flex-wrap
order	定义子元素的排列顺序
justify-content	定义子元素在“主轴”上的对齐方式
align-items	定义子元素在“纵轴”上的对齐方式

9.4.2 案例

1.子元素宽度之和小于父元素宽度

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                width: 200px;
                height: 150px;
                border: 1px dashed silver;
            }

            #box1,
            #box2,
            #box3 {
                width: 50px;
            }

            #box1 {
                background: red;
            }
```

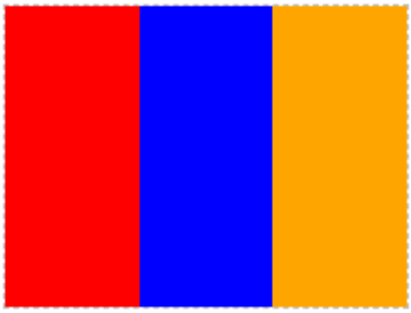
```
            #box2 {
                background: blue;
            }

            #box3 {
                background: orange;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1"></div>
            <div id="box2"></div>
            <div id="box3"></div>
        </div>
    </body>
</html>
```



2.子元素宽度之和大于父元素宽度

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                width: 200px;
                height: 150px;
                border: 1px dashed silver;
            }

            #box1,
            #box2,
            #box3 {
                width: 100px;
            }
```

```
            #box1 {
                background: red;
            }

            #box2 {
                background: blue;
            }

            #box3 {
                background: orange;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1"></div>
            <div id="box2"></div>
            <div id="box3"></div>
        </div>
    </body>
</html>
```



注：在这个例子中，弹性盒子（父元素）的宽度为200px，而所有子元素宽度之和为300px，此时子元素宽度之和大于父元素宽度。因此，子元素会按比例来划分宽度。这就是弹性盒子的特点。

9.4.3 flex-grow 放大比例

在CSS3中，我们可以使用flex-grow属性来定义弹性盒子内部子元素的放大比例。也就是当所有子元素宽度之和小于父元素的宽度时，子元素如何分配父元素的剩余空间。

默认值为 0，即不放大。当容器空间有剩余时，会按照各个项目设置的 `flex-grow` 值来分配剩余空间。如果一个项目的 `flex-grow` 设为 2，另一个项目设为 1，则前者占据的剩余空间是后者的两倍。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                width: 200px;
                height: 150px;
                border: 1px dashed silver;
            }

            #box1,
            #box2,
            #box3 {
                width: 50px;
            }

            #box1 {
                background: red;
                flex-grow: 0;
            }

            #box2 {
                background: blue;
                flex-grow: 1;
            }

            #box3 {
                background: orange;
                flex-grow: 2;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1"></div>
            <div id="box2"></div>
            <div id="box3"></div>
        </div>
    </body>
</html>
```

9.4.4 flex-shrink 缩小比例

在CSS3中，flex-shrink属性用于定义弹性盒子内部子元素的缩小比例。也就是当所有子元素宽度之和大于父元素的宽度时，子元素如何缩小自己的宽度。

默认值为 1，即缩小到最小值。当容器空间不足时，会按照各个项目设置的 `flex-shrink` 值来分配缺少的空间。与 `flex-grow` 对应的是，`flex-shrink` 数值越大的项目会被优先缩小。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                width: 200px;
                height: 150px;
                border: 1px dashed silver;
            }

            #box1,
            #box2,
            #box3 {
                width: 100px;
            }

            #box1 {
                background: red;
                flex-shrink: 0;
            }

            #box2 {
                background: blue;
                flex-shrink: 1;
```

```
            }

            #box3 {
                background: orange;
                flex-shrink: 2;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1"></div>
            <div id="box2"></div>
            <div id="box3"></div>
        </div>
    </body>
</html>
```

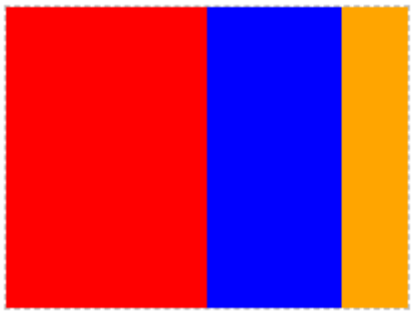
9.4.5 flex-basis 元素宽度

在CSS3中，我们可以定义弹性盒子内部的子元素在分配空间之前，该子元素所占的空间大小。浏览器会根据这个属性，计算父元素是否有多余空间。

说白了，flex-basis就是width的替代品，它们都用来定义子元素的宽度。只不过在弹性盒子中，flex-basis的语义会比width更好。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                width: 200px;
                height: 150px;
                border: 1px dashed silver;
            }

            #box1,
            #box2,
            #box3 {
                flex-basis: 100px;
            }

            #box1 {
                background: red;
                flex-shrink: 0;
```

```
            }

            #box2 {
                background: blue;
                flex-shrink: 1;
            }

            #box3 {
                background: orange;
                flex-shrink: 2;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1"></div>
            <div id="box2"></div>
            <div id="box3"></div>
        </div>
    </body>
</html>
```



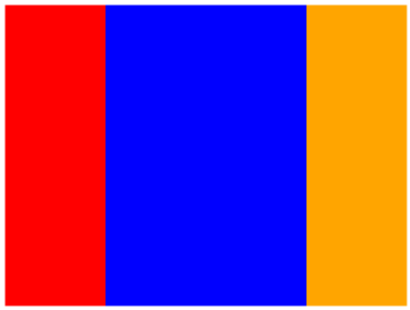
9.4.6 flex 复合属性

flex属性的默认值为“0 1 auto”。

```
flex: 1;`等价于`flex: 1 1 auto
flex: 2`等价于`flex: 2 1 auto
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                width: 200px;
                height: 150px;
            }
```

```
            #box1 {
                background: red;
                flex: 1;
            }

            #box2 {
                background: blue;
                flex: 2;
            }

            #box3 {
                background: orange;
                flex: 1;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1"></div>
            <div id="box2"></div>
            <div id="box3"></div>
        </div>
    </body>
</html>
```



9.4.7 flex-direction 排列方向

在CSS3中，我们可以使用flex-direction属性来定义弹性盒子内部“子元素”的排列方向。也就是定义子元素是横着排，还是竖着排。

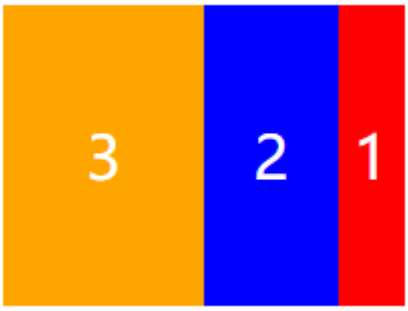
flex-direction属性取值

属性值	说明
row	默认值，横向排列
row-reverse	横向反向排列

属性值	说明
column	纵向排列
column-reverse	纵向反向排列

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                flex-direction: row-reverse;
                width: 200px;
                height: 150px;
            }

            #box1,
            #box2,
            #box3 {
                height: 150px;
                line-height: 150px;
                text-align: center;
                font-size: 30px;
                color: white;
            }

            #box1 {
                background: red;
                flex: 1;
            }

            #box2 {
                background: blue;
                flex: 2;
            }

            #box3 {
                background: orange;
                flex: 3;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1">1</div>
            <div id="box2">2</div>
            <div id="box3">3</div>
        </div>
```

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



9.4.8 flex-wrap 换行

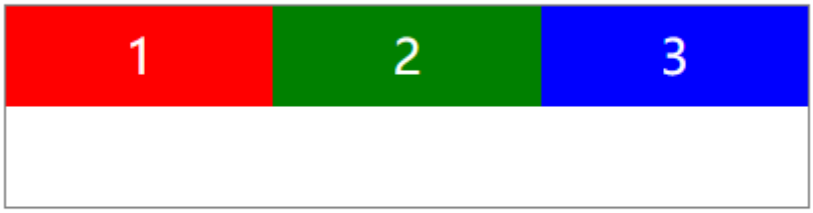
在CSS3中，我们可以使用flex-wrap属性来定义弹性盒子内部“子元素”是单行显示还是多行显示。

属性值	说明
nowrap	默认值，单行显示
wrap	多行显示
wrap-reverse	多行显示，但是是反向

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            /* 公用样式 */
            .wrapper1,
            .wrapper2,
            .wrapper3 {
                display: flex;
                color: white;
                font-size: 24px;
                width: 400px;
                height: 100px;
                line-height: 50px;
                border: 1px solid gray;
                text-align: center;
            }

            .wrapper1 div,
            .wrapper2 div,
            .wrapper3 div {
```

```
                height: 50%;
                width: 50%;
            }

            .red {
                background: red;
            }

            .green {
                background: green;
            }

            .blue {
                background: blue;
            }

            /* 弹性盒子样式 */
            .wrapper1 {
                flex-wrap: nowrap;
            }

            .wrapper2 {
                flex-wrap: wrap;
            }

            .wrapper3 {
                flex-wrap: wrap-reverse;
            }
        </style>
    </head>
    <body>
        <h3>1、flex-wrap:nowrap（默认值）</h3>
        <div class="wrapper1">
            <div class="red">1</div>
            <div class="green">2</div>
            <div class="blue">3</div>
        </div>
        <h3>2、flex-wrap:wrap</h3>
        <div class="wrapper2">
            <div class="red">1</div>
            <div class="green">2</div>
            <div class="blue">3</div>
        </div>
        <h3>3、flex-wrap:wrap-reverse</h3>
        <div class="wrapper3">
            <div class="red">1</div>
            <div class="green">2</div>
            <div class="blue">3</div>
        </div>
    </body>
</html>
```

1、flex-wrap:nowrap（默认值）



2、flex-wrap:wrap

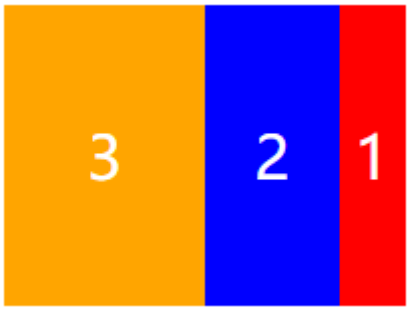
3、flex-wrap:wrap-reverse

9.4.9 flex-flow 复合属性

在CSS3中，我们可以使用flex-flow属性来同时设置flex-direction、flex-wrap这两个属性。说白了，flex-flow属性就是一个简写形式，就是一个“语法糖”。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
                flex-flow: row-reverse nowrap;
                width: 200px;
                height: 150px;
            }

            #box1,
            #box2,
            #box3 {
                height: 150px;
```

```
                line-height: 150px;
                text-align: center;
                font-size: 30px;
                color: white;
            }

            #box1 {
                background: red;
                flex: 1;
            }

            #box2 {
                background: blue;
                flex: 2;
            }

            #box3 {
                background: orange;
                flex: 3;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1">1</div>
            <div id="box2">2</div>
            <div id="box3">3</div>
        </div>
    </body>
</html>
```



9.4.10 order 排列顺序

在CSS3中，我们可以使用order属性来定义弹性盒子内部“子元素”的排列顺序。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
```

```
        <title></title>
        <style type="text/css">
            #wrapper {
                display: flex;
            }

            #box1,
            #box2,
            #box3 {
                width: 150px;
                height: 150px;
                line-height: 150px;
                text-align: center;
                font-size: 30px;
                color: white;
            }

            #box1 {
                background: red;
                order: 2;
            }

            #box2 {
                background: blue;
                order: 3;
            }

            #box3 {
                background: orange;
                order: 1;
            }
        </style>
    </head>
    <body>
        <div id="wrapper">
            <div id="box1">1</div>
            <div id="box2">2</div>
            <div id="box3">3</div>
        </div>
    </body>
</html>
```

9.4.11 justify-content 水平对齐

在CSS3中，我们可以使用justify-content属性来定义弹性盒子内部子元素在“横轴”上的对齐方式。

justify-content属性取值

属性值	说明
flex-start	默认值，所有子元素在左边
center	所有子元素在中间
flex-end	所有子元素在右边
space-between	所有子元素平均分布
space-around	所有子元素平局分布，两边留有间距

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            /*定义整体样式*/
            .flex {
                display: flex;
                flex-flow: row nowrap;
                background-color: lightskyblue;
                margin-bottom: 5px;
            }

            .item {
                width: 80px;
                padding: 10px;
                text-align: center;
                background-color: hotpink;
                box-sizing: border-box;
            }

            /*定义justify-content*/
            .start {
                justify-content: flex-start;
            }

            .center {
                justify-content: center;
            }
```

```
            .end {
                justify-content: flex-end;
            }

            .between {
                justify-content: space-between;
            }

            .around {
                justify-content: space-around;
            }
        </style>
    </head>
    <body>
        <h3>1、flex-start:</h3>
        <div class="flex start">
            <div class="item">1</div>
            <div class="item">2</div>
            <div class="item">3</div>
            <div class="item">4</div>
        </div>
        <h3>2、center:</h3>
        <div class="flex center">
            <div class="item">1</div>
            <div class="item">2</div>
            <div class="item">3</div>
            <div class="item">4</div>
        </div>
        <h3>3、flex-end:</h3>
        <div class="flex end">
            <div class="item">1</div>
            <div class="item">2</div>
            <div class="item">3</div>
            <div class="item">4</div>
        </div>
        <h3>4、space-between:</h3>
        <div class="flex between">
            <div class="item">1</div>
            <div class="item">2</div>
            <div class="item">3</div>
            <div class="item">4</div>
        </div>
        <h3>5、space-around:</h3>
        <div class="flex around">
            <div class="item">1</div>
            <div class="item">2</div>
            <div class="item">3</div>
            <div class="item">4</div>
        </div>
    </body>
</html>
```

1、flex-start:

2、center:

3、flex-end:

4、space-between:

5、space-around:

9.4.12 align-items 垂直对齐

在CSS3中，我们可以使用align-items属性来定义弹性盒子内部子元素在“纵轴”上的对齐方式。

align-items属性取值

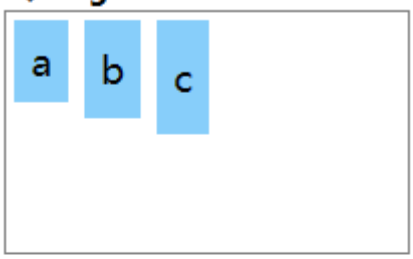
属性值	说明
flex-start	默认值，所有子元素在上边
center	所有子元素在中间
flex-end	所有子元素在下边
baseline	所以子元素在父元素的基线上
stretch	拉伸子元素适应父元素的高度

```
<!DOCTYPE html>
<html>
    <head>
```

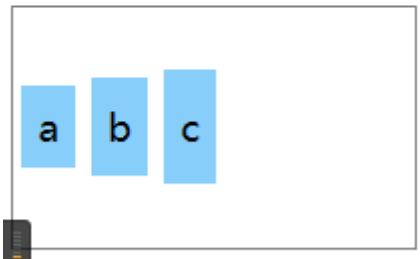
```
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
            .box {
                /*去除默认样式*/
                list-style-type: none;
                margin: 0;
                padding: 0;
                /*定义flex布局*/
                display: flex;
                width: 250px;
                height: 150px;
                border: 1px solid gray;
                font-size: 24px;
            }

            h3 {
                margin-bottom: 3px;
            }

            /*定义子元素样式*/
            .box li {
                margin: 5px;
                background-color: lightskyblue;
                text-align: center;
            }

            .box li:nth-child(1) {
                padding: 10px;
            }

            .box li:nth-child(2) {
                padding: 15px 10px;
            }

            .box li:nth-child(3) {
                padding: 20px 10px;
            }

            /*定义align-items*/
            #box1 {
                align-items: flex-start;
            }

            #box2 {
                align-items: center;
            }

            #box3 {
                align-items: flex-end;
            }
```

```
            #box4 {
                align-items: baseline;
            }

            #box5 {
                align-items: stretch;
            }
        </style>
    </head>
    <body>
        <h3>1、align-items:flex-start</h3>
        <ul id="box1" class="box">
            <li>a</li>
            <li>b</li>
            <li>c</li>
        </ul>
        <h3>2、align-items:center</h3>
        <ul id="box2" class="box">
            <li>a</li>
            <li>b</li>
            <li>c</li>
        </ul>
        <h3>3、align-items:flex-end</h3>
        <ul id="box3" class="box">
            <li>a</li>
            <li>b</li>
            <li>c</li>
        </ul>
        <h3>4、align-items:baseline</h3>
        <ul id="box4" class="box">
            <li>a</li>
            <li>b</li>
            <li>c</li>
        </ul>
        <h3>5、align-items:stretch</h3>
        <ul id="box5" class="box">
            <li>a</li>
            <li>b</li>
            <li>c</li>
        </ul>
    </body>
</html>
```

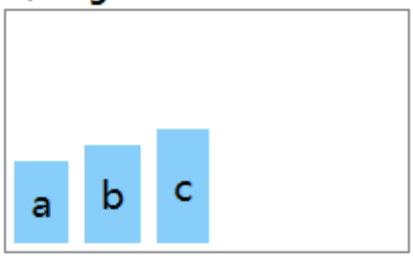
1、align-items:flex-start



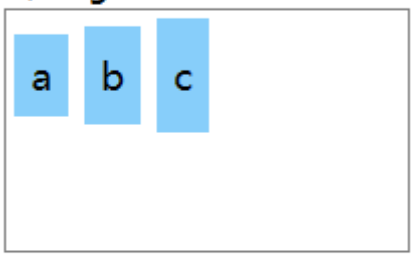
2、align-items:center



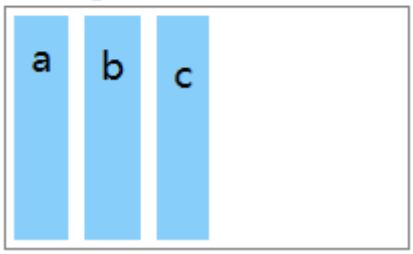
3、align-items:flex-end



4、align-items:baseline



5、align-items:stretch



9.5 综合案例（弹性盒子+媒体查询）-伸缩菜单

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title></title>
        <style type="text/css">
```

```css
/*定义整体样式*/
.nav {
    /*去除默认样式*/
    list-style-type: none;
    margin: 0;
    padding: 0;
    /*定义弹性盒子*/
    display: flex;
    background-color: hotpink;
}

.nav a {
    /*去除默认样式*/
    text-decoration: none;
    display: block;
    padding: 16px;
    color: white;
    text-align: center;
}

.nav a:hover {
    background-color: lightskyblue;
}

/*设备大于800px时*/
@media (min-width:800px) {

    /*所有子元素在右边*/
    .nav {
        justify-content: flex-end;
    }

    li {
        border-left: 1px solid silver;
    }
}

/*设备大于600px且小于800px时*/
@media (min-width:600px) and (max-width:800px) {

    /*所有子元素平分*/
    .nav li {
        flex: 1;
    }

    li+li {
        border-left: 1px solid silver;
    }
}

/*设备小于600px时*/
@media (max-width: 600px) {
```

```
                /*所有子元素纵向排列*/
                .nav {
                    flex-flow: column wrap;
                }

                li+li {
                    border-top: 1px solid silver;
                }
            }
        </style>
    </head>
    <body>
        <ul class="nav">
            <li><a href="#">首页</a></li>
            <li><a href="#">前端</a></li>
            <li><a href="#">后端</a></li>
            <li><a href="#">下载</a></li>
        </ul>
    </body>
</html>
```