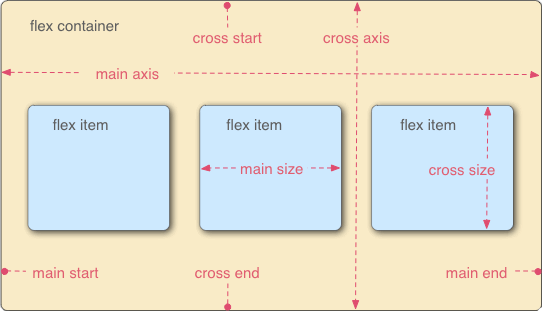
# CSS flex布局（弹性布局/弹性盒子）

Flex 是 Flexible Box 的缩写，意为“弹性布局”或者“弹性盒子”，是 CSS3 中的一种新的布局模式，可以简便、完整、响应式地实现各种页面布局，当页面需要适应不同的屏幕大小以及设备类型时非常适用。目前，几乎所有的浏览器都支持 Flex 布局。

**1. 基本概念**

采用 Flex 布局的元素，称为 Flex 容器（flex container），简称“容器”。它的所有子元素自动成为容器成员，称为 Flex 项目（flex item），简称“项目”。容器默认存在两根轴，分别为水平的主轴（main axis）和垂直的交叉轴又叫做侧轴（cross axis）。主轴的开始位置叫做 main start，结束位置叫做 main end；交叉轴的开始位置叫做 cross start，结束位置叫做 cross end。项目默认沿主轴排列。单个项目占据的主轴空间叫做 main size，占据的交叉轴空间叫做 cross size。如下图所示：

  
图：Flex 容器

提示：您可以通过将元素的 display 属性设置为 flex（生成块级 flex 容器）或 inline-flex（生成类似 inline-block 的行内块级 flex 容器）。

### 注意：当一个元素设置了 Flex 布局以后，其子元素的 float、clear 和 vertical-align 等属性将失效。

### 主轴和侧轴是可以通过代码来修改的。

CSS 中提供了以下属性来实现 Flex 布局：

|  |  |
| --- | --- |
| **属性** | **描述** |
| display | 指定 HTML 元素的盒子类型 |
| flex-direction | 指定弹性盒子中子元素的排列方式 |
| flex-wrap | 设置当弹性盒子的子元素超出父容器时是否换行 |
| flex-flow | flex-direction 和 flex-wrap 两个属性的简写 |
| justify-content | 设置弹性盒子中元素在主轴（横轴）方向上的对齐方式 |
| align-items | 设置弹性盒子中元素在侧轴（纵轴）方向上的对齐方式 |
| align-content | 修改 flex-wrap 属性的行为，类似 align-items，但不是设置子元素对齐，而是设置行对齐 |
| order | 设置弹性盒子中子元素的排列顺序 |
| align-self | 在弹性盒子的子元素上使用，用来覆盖容器的 align-items 属性 |
| flex | 设置弹性盒子中子元素如何分配空间 |
| flex-grow | 设置弹性盒子的扩展比率 |
| flex-shrink | 设置弹性盒子的收缩比率 |
| flex-basis | 设置弹性盒子伸缩基准值 |

### 按照作用范围的不同，这些属性可以分为两类：

#### 1）分别为容器属性（flex-direction、flex-wrap、flex-flow、justify-content、align-items、align-content）

#### 2）项目属性（order、align-self、flex、flex-grow、flex-shrink、flex-basis）。下面就来介绍一下这些属性的使用。

**2. 容器属性**

**1) flex-direction**

flex-direction 属性用来决定主轴的方向（即项目的排列方向），属性的可选值如下：

|  |  |
| --- | --- |
| **值** | **描述** |
| row | 默认值，主轴沿水平方向从左到右 |
| row-reverse | 主轴沿水平方向从右到左 |
| column | 主轴沿垂直方向从上到下 |
| column-reverse | 主轴沿垂直方向从下到上 |
| initial | 将此属性设置为属性的默认值 |
| inherit | 从父元素继承此属性的值 |

示例代码如下：

[复制纯文本复制](http://c.biancheng.net/css3/flex.html)

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<style>**

**#main {**

**border: 1px solid #CCC;**

**padding: 5px;**

**position: relative;**

**}**

**.row, .row\_reverse, .column, .column\_reverse{**

**display: flex;**

**margin-bottom: 5px;**

**}**

**.row {**

**flex-direction: row;**

**}**

**.row\_reverse {**

**flex-direction: row-reverse;**

**}**

**.column {**

**flex-direction: column;**

**}**

**.column\_reverse {**

**flex-direction: column-reverse;**

**position: absolute;**

**top: 120px;**

**left: 400px;**

**}**

**.row div, .row\_reverse div, .column div, .column\_reverse div {**

**width: 50px;**

**height: 50px;**

**border: 1px solid black;**

**}**

**</style>**

**</head>**

**<body>**

**<div id="main">**

**<div class="row">**

**<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>**

**</div>**

**<div class="row\_reverse">**

**<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>**

**</div>**

**<div class="column">**

**<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>**

**</div>**

**<div class="column\_reverse">**

**<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>**

**</div>**

**</div>**

**</body>**

**</html>**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

#main {

border: 1px solid #CCC;

padding: 5px;

position: relative;

}

.row, .row\_reverse, .column, .column\_reverse{

display: flex;

margin-bottom: 5px;

}

.row {

flex-direction: row;

}

.row\_reverse {

flex-direction: row-reverse;

}

.column {

flex-direction: column;

}

.column\_reverse {

flex-direction: column-reverse;

position: absolute;

top: 120px;

left: 400px;

}

.row div, .row\_reverse div, .column div, .column\_reverse div {

width: 50px;

height: 50px;

border: 1px solid black;

}

</style>

</head>

<body>

<div id="main">

<div class="row">

<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>

</div>

<div class="row\_reverse">

<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>

</div>

<div class="column">

<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>

</div>

<div class="column\_reverse">

<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>

</div>

</div>

</body>

</html>

运行结果如下图所示：

|  |
| --- |
| flex-direction 属性演示 |

图：flex-direction 属性演示

**2) flex-wrap**

flex-wrap 属性用来设置当弹性盒子的子元素（项目）超出父容器时是否换行，属性的可选值如下：

|  |  |
| --- | --- |
| **值** | **描述** |
| nowrap | 默认值，表示项目不会换行 |
| wrap | 表示项目会在需要时换行 |
| wrap-reverse | 表示项目会在需要时换行，但会以相反的顺序 |
| initial | 将此属性设置为属性的默认值 |
| inherit | 从父元素继承属性的值 |

示例代码如下：

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<style>**

**#main {**

**border: 1px solid #CCC;**

**padding: 5px;**

**}**

**.nowrap, .wrap, .wrap\_reverse {**

**display: flex;**

**flex-direction: row;**

**margin-bottom: 15px;**

**}**

**.nowrap {**

**flex-wrap: nowrap;**

**}**

**.wrap {**

**flex-wrap: wrap;**

**}**

**.wrap\_reverse {**

**flex-wrap: wrap-reverse;**

**}**

**.nowrap div, .wrap div, .wrap\_reverse div {**

**width: 70px;**

**height: 50px;**

**border: 1px solid black;**

**}**

**</style>**

**</head>**

**<body>**

**<div id="main">**

**<div class="nowrap">**

**<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>**

**</div>**

**<div class="wrap">**

**<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>**

**</div>**

**<div class="wrap\_reverse">**

**<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>**

**</div>**

**</div>**

**</body>**

**</html>**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

#main {

border: 1px solid #CCC;

padding: 5px;

}

.nowrap, .wrap, .wrap\_reverse {

display: flex;

flex-direction: row;

margin-bottom: 15px;

}

.nowrap {

flex-wrap: nowrap;

}

.wrap {

flex-wrap: wrap;

}

.wrap\_reverse {

flex-wrap: wrap-reverse;

}

.nowrap div, .wrap div, .wrap\_reverse div {

width: 70px;

height: 50px;

border: 1px solid black;

}

</style>

</head>

<body>

<div id="main">

<div class="nowrap">

<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>

</div>

<div class="wrap">

<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>

</div>

<div class="wrap\_reverse">

<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>

</div>

</div>

</body>

</html>

运行结果如下图所示：

|  |
| --- |
| flex-wrap 属性演示 |

图：flex-wrap 属性演示

**3) flex-flow**

flex-flow 属性是 flex-direction 和 flex-wrap 两个属性的简写，语法格式如下：

flex-flow: flex-direction flex-wrap;

示例代码如下：

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<style>**

**.flex\_flow {**

**display: flex;**

**flex-flow: row-reverse wrap;**

**}**

**.flex\_flow div {**

**width: 60px;**

**height: 60px;**

**margin-bottom: 5px;**

**border: 1px solid black;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="flex\_flow">**

**<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>**

**</div>**

**</body>**

**</html>**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

.flex\_flow {

display: flex;

flex-flow: row-reverse wrap;

}

.flex\_flow div {

width: 60px;

height: 60px;

margin-bottom: 5px;

border: 1px solid black;

}

</style>

</head>

<body>

<div class="flex\_flow">

<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div>

</div>

</body>

</html>

运行结果如下图所示：

|  |
| --- |
| flex-flow 属性演示 |

图：flex-flow 属性演示

**4) justify-content**

justify-content 属性用于设置弹性盒子中元素在主轴（横轴）方向上的对齐方式，属性的可选值如下：

|  |  |
| --- | --- |
| **值** | **描述** |
| flex-start | 默认值，左对齐 |
| flex-end | 右对齐 |
| center | 居中 |
| space-between | 两端对齐，项目之间的间隔是相等的 |
| space-around | 每个项目两侧的间隔相等 |
| space-evenly | 均分富余空间 |
| initial | 将此属性设置为属性的默认值 |
| inherit | 从父元素继承属性的值 |

#### 注意：如果需要实现弹性容器子元素的水平和垂直方向都居中，需要在弹性容器中设置下面两项

#### justify-content：center;

#### align-items：center：

#### 实例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <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>弹性布局实例</title>      <style>          .box{              border: 1px solid yellow;              list-style: none;              width: 500px;              height: 300px;              margin: 0;              padding: 0;              /\* 设置弹性布局注意必须在父容器上面设置 \*/              display: flex;              /\* flex-direction设置弹性布局方向，也就是切换主侧轴，默认是row \*/              /\* flex-direction: column; \*/              /\* flex-direction: row-reverse; \*/              /\* 设置弹性容器的子元素是否换行，默认是不换行 \*/              flex-wrap: nowrap;              /\* flex-wrap: wrap; \*/              /\* flex-wrap: wrap-reverse; \*/              /\* 设置子元素的对齐方式 \*/              justify-content: center;              /\* justify-content: flex-end; \*/              /\* justify-content: flex-start; \*/              /\* justify-content: space-between; \*/              /\* justify-content: space-around; \*/              /\* justify-content: space-evenly; \*/              /\* 设置弹性容器子元素在侧轴方向的对齐方式 \*/              /\* align-items: flex-start; \*/              /\* align-items: stretch; \*/              /\* align-items: flex-end; \*/              align-items: center;          }            .box .item{              border: 1px dashed green;              width: 100px;              height: 100px;          }      </style>  </head>  <body>      <ul class="box">          <li class="item">1</li>          <li class="item">2</li>          <li class="item">3</li>          <li class="item">4</li>          <!-- <li class="item">5</li> -->          <!-- <li class="item">6</li> -->      </ul>  </body>  </html> |
| 效果 |

#### 可见使用flex布局设置元素居中是很方便的

### 实例2

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <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>弹性布局实例</title>      <style>            .flex-box{              width: 500px;              height: 300px;              border: 1px solid deeppink;              /\* 弹性布局 \*/              display: flex;              /\* 设置儿子水平居中 \*/              justify-content: center;              /\* 设置儿子垂直居中 \*/              align-items: center;          }          .flex-item{              /\* border: 1px dashed green; \*/              background-color: lime;              width: 100px;              height: 100px;          }      </style>  </head>  <body>        <div class="flex-box">          <div class="flex-item">孩子</div>      </div>  </body>  </html> |
| 效果： |

#### 还可以使用flex布局实现多重居中

### 实例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <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>弹性布局实例</title>      <style>             html,body{               height: 100%;               display: flex;               /\* 设置儿子水平居中 \*/              justify-content: center;              /\* 设置儿子垂直居中 \*/              align-items: center;           }          .flex-box{              width: 500px;              height: 300px;              border: 1px solid deeppink;              /\* 弹性布局 \*/              display: flex;              /\* 设置儿子水平居中 \*/              justify-content: center;              /\* 设置儿子垂直居中 \*/              align-items: center;          }          .flex-item{              /\* border: 1px dashed green; \*/              background-color: lime;              width: 100px;              height: 100px;              display: flex;               /\* 设置儿子水平居中 \*/              justify-content: center;              /\* 设置儿子垂直居中 \*/              align-items: center;          }      </style>  </head>  <body>        <div class="flex-box">          <div class="flex-item">孩子</div>      </div>  </body>  </html> |
| 效果 |

#### flex布局的一个非常有用的应用是处理图片和文本对齐 另外的示例代码如下：

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<style>**

**.flex {**

**display: flex;**

**flex-flow: row wrap;**

**margin-top: 10px;**

**}**

**.flex div {**

**width: 60px;**

**height: 60px;**

**margin-bottom: 5px;**

**border: 1px solid black;**

**}**

**.flex-start {**

**justify-content: flex-start;**

**}**

**.flex-end {**

**justify-content: flex-end;**

**}**

**.center {**

**justify-content: center;**

**}**

**.space-between {**

**justify-content: space-between;**

**}**

**.space-around {**

**justify-content: space-around;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="flex flex-start">**

**<div>A</div><div>B</div><div>C</div><div>D</div>**

**</div>**

**<div class="flex flex-end">**

**<div>A</div><div>B</div><div>C</div><div>D</div>**

**</div>**

**<div class="flex center">**

**<div>A</div><div>B</div><div>C</div><div>D</div>**

**</div>**

**<div class="flex space-between">**

**<div>A</div><div>B</div><div>C</div><div>D</div>**

**</div>**

**<div class="flex space-around">**

**<div>A</div><div>B</div><div>C</div><div>D</div>**

**</div>**

**</body>**

**</html>**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

.flex {

display: flex;

flex-flow: row wrap;

margin-top: 10px;

}

.flex div {

width: 60px;

height: 60px;

margin-bottom: 5px;

border: 1px solid black;

}

.flex-start {

justify-content: flex-start;

}

.flex-end {

justify-content: flex-end;

}

.center {

justify-content: center;

}

.space-between {

justify-content: space-between;

}

.space-around {

justify-content: space-around;

}

</style>

</head>

<body>

<div class="flex flex-start">

<div>A</div><div>B</div><div>C</div><div>D</div>

</div>

<div class="flex flex-end">

<div>A</div><div>B</div><div>C</div><div>D</div>

</div>

<div class="flex center">

<div>A</div><div>B</div><div>C</div><div>D</div>

</div>

<div class="flex space-between">

<div>A</div><div>B</div><div>C</div><div>D</div>

</div>

<div class="flex space-around">

<div>A</div><div>B</div><div>C</div><div>D</div>

</div>

</body>

</html>

运行结果如下图所示：

|  |
| --- |
| justify-content 属性演示 |

图：justify-content 属性演示

**5) align-items**

align-items 属性用来设置弹性盒子中元素在侧轴（纵轴）方向上的对齐方式，属性的可选值如下：

|  |  |
| --- | --- |
| **值** | **描述** |
| stretch | 默认值，项目将被拉伸以适合容器 |
| center | 项目位于容器的中央 |
| flex-start | 项目位于容器的顶部 |
| flex-end | 项目位于容器的底部 |
| baseline | 项目与容器的基线对齐 |
| initial | 将此属性设置为属性的默认值 |
| inherit | 从父元素继承属性的值 |

几个属性值的效果如下图所示：

|  |
| --- |
| align-items 属性演示 |

图：align-items 属性演示

### align-item和align-content属性的区别

|  |
| --- |
|  |

#### align-content属性只有在有多行的情况下才有作用，只有一行的画是没有作用的

#### 注意：stretch属性值只有在子元素没有设置高度的时候才会拉伸，否则没有效果

**6) align-content**

align-content 属性与 justify-content 属性类似，可以在弹性盒子的侧轴还有多余空间时调整容器内项目的对齐方式，属性的可选值如下：

|  |  |
| --- | --- |
| **值** | **描述** |
| stretch | 默认值，将项目拉伸以占据剩余空间 |
| center | 项目在容器内居中排布 |
| flex-start | 项目在容器的顶部排列 |
| flex-end | 项目在容器的底部排列 |
| space-between | 多行项目均匀分布在容器中，其中第一行分布在容器的顶部，最后一行分布在容器的底部 |
| space-around | 多行项目均匀分布在容器中，并且每行的间距（包括离容器边缘的间距）都相等 |
| initial | 将此属性设置为属性的默认值 |
| inherit | 从父元素继承该属性的值 |

几个属性值的效果如下图所示：

|  |
| --- |
| align-content 属性演示 |

图：align-content 属性演示

**3. 项目属性**

**1) order**

order 属性用来设置项目在容器中出现的顺序，您可以通过具体的数值来定义项目在容器中的位置，属性的语法格式如下：

order: number;

其中 number 就是项目在容器中的位置，默认值为 0。  
  
示例代码如下：

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<style>**

**.flex {**

**display: flex;**

**flex-flow: row wrap;**

**margin-top: 10px;**

**}**

**.flex div {**

**width: 60px;**

**height: 60px;**

**margin-bottom: 5px;**

**border: 1px solid black;**

**}**

**.flex div:nth-child(1) {**

**order: 5;**

**}**

**.flex div:nth-child(2) {**

**order: 3;**

**}**

**.flex div:nth-child(3) {**

**order: 1;**

**}**

**.flex div:nth-child(4) {**

**order: 2;**

**}**

**.flex div:nth-child(5) {**

**order: 4;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="flex">**

**<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>**

**</div>**

**</body>**

**</html>**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

.flex {

display: flex;

flex-flow: row wrap;

margin-top: 10px;

}

.flex div {

width: 60px;

height: 60px;

margin-bottom: 5px;

border: 1px solid black;

}

.flex div:nth-child(1) {

order: 5;

}

.flex div:nth-child(2) {

order: 3;

}

.flex div:nth-child(3) {

order: 1;

}

.flex div:nth-child(4) {

order: 2;

}

.flex div:nth-child(5) {

order: 4;

}

</style>

</head>

<body>

<div class="flex">

<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>

</div>

</body>

</html>

运行结果如下图所示：

|  |
| --- |
| order 属性演示 |

图：order 属性演示

##### 注意：order的值是从0开始的，数值越大约靠右，order可以是负数

**2) align-self**

align-self 属性允许您为某个项目设置不同于其它项目的对齐方式，该属性可以覆盖 align-items 属性的值。align-self 属性的可选值如下：

|  |  |
| --- | --- |
| **值** | **描述** |
| auto | 默认值，表示元素将继承其父容器的 align-items 属性值，如果没有父容器，则为“stretch” |
| stretch | 项目将被拉伸以适合容器 |
| center | 项目位于容器的中央 |
| flex-start | 项目位于容器的顶部 |
| flex-end | 项目位于容器的底部 |
| baseline | 项目与容器的基线对齐 |
| initial | 将此属性设置为属性的默认值 |
| inherit | 从父元素继承属性的值 |

示例代码如下：

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<style>**

**.flex {**

**display: flex;**

**flex-flow: row wrap;**

**align-items: flex-end;**

**border: 1px solid #CCC;**

**height: 150px;**

**}**

**.flex div {**

**width: 60px;**

**height: 60px;**

**border: 1px solid black;**

**}**

**.flex div:nth-child(4) {**

**align-self: flex-start;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="flex">**

**<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>**

**</div>**

**</body>**

**</html>**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

.flex {

display: flex;

flex-flow: row wrap;

align-items: flex-end;

border: 1px solid #CCC;

height: 150px;

}

.flex div {

width: 60px;

height: 60px;

border: 1px solid black;

}

.flex div:nth-child(4) {

align-self: flex-start;

}

</style>

</head>

<body>

<div class="flex">

<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>

</div>

</body>

</html>

运行结果如下图所示：

|  |
| --- |
| align-self 属性演示 |

图：align-self 属性演示

**3) flex**

flex 属性是 flex-grow、flex-shrink 和 flex-basis 三个属性（也叫弹性因子）的简写，语法格式如下：

flex: flex-grow flex-shrink flex-basis;

参数说明如下：

* flex-grow：（必填参数）一个数字，用来设置项目相对于其他项目的增长量（份额）数值越大分的越多，默认值为 0；只有容器还有富余空间才有用；

计算公式：父容器的总富余空间/flex-grow总份数\*每一个元素对应的份数=每一个元素的增长量

* flex-shrink：（选填参数）一个数字，用来设置项目相对于其他项目的收缩量，默认值为 1；

计算公式：总收缩量/flex-shrink总份数 \* 每一个元素对应的份数=每一个元素的收缩量

* flex-basis：（选填参数）项目的长度，合法值为 auto（默认值，表示自动）、inherit（表示从父元素继承该属性的值） 或者以具体的值加 "%"、"px"、"em" 等单位的形式。这个属性可以覆盖父容器项目的宽度设置

也就是说复合属性flex的默认值是 0 1 auto，另外，flex 属性还有两个快捷值，分别为 auto（1 1 auto）和 none（0 0 auto）。

注意：{flex:1;}这一句的意思是{flex-grow:1;flex-shrink:1;flex-basis:0%;}

示例代码如下：

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<style>**

**.flex {**

**display: flex;**

**flex-flow: row wrap;**

**align-items: flex-end;**

**border: 1px solid #CCC;**

**}**

**.flex div {**

**width: 60px;**

**height: 60px;**

**border: 1px solid black;**

**}**

**.flex div:nth-child(2) {**

**flex:0;**

**}**

**.flex div:nth-child(4) {**

**flex:1 1 auto;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="flex">**

**<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>**

**</div>**

**</body>**

**</html>**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

.flex {

display: flex;

flex-flow: row wrap;

align-items: flex-end;

border: 1px solid #CCC;

}

.flex div {

width: 60px;

height: 60px;

border: 1px solid black;

}

.flex div:nth-child(2) {

flex:0;

}

.flex div:nth-child(4) {

flex:1 1 auto;

}

</style>

</head>

<body>

<div class="flex">

<div>A</div><div>B</div><div>C</div><div>D</div><div>E</div>

</div>

</body>

</html>

运行结果如下图所示：

|  |
| --- |
| flex 属性演示 |

图：flex 属性演示

另外，除了可以使用 flex 属性外，您也可以使用 flex-grow、flex-shrink、flex-basis 几个属性来分别设置项目的增长量、收缩量以及项目长度，例如：

**.flex div:nth-child(4) {**

**flex-grow: 1;**

**flex-shrink: 1;**

**flex-basis: auto;**

**/\* 等同于 flex:1 1 auto; \*/**

**}**

### 4.Flex布局的特点以及和float的区别

### 1.)如果就应该容器的使用孩子都设置float，而容器又没有设置高度，容器会发生高度坍塌。而flex布局容器不会坍塌，而是由孩子的高度撑开

### 2）使用float时，如果容器一行排列不下会换行，而flex会缩小孩子使得他们能够放在一行里面。

### 3）弹性布局会把行内元素转换为块级元素显示

### 4）如果flex容器的孩子没有设置宽高，孩子的宽度由内容决定，高度默认占满父容器

### 5）容器的flex布局只对儿子元素有效果，对孙子是没有效果的

### 6）容器设置flex布局后对他的兄弟元素没有影响而float会影响兄弟元素

### 可见，flex布局解决了float的所有问题，有float的优点又没有float的缺点