### 内联块状元素

**内联块状元素（**inline-block**）**就是同时具备内联元素、块状元素的特点，代码display:inline-block就是将元素设置为内联块状元素。(css2.1新增)，<img>、<input>标签就是这种内联块状标签。

inline-block元素特点：

1、和其他元素都在一行上；

2、元素的高度、宽度、行高以及顶和底边距都可设置。

## 水平居中设置

### 行内元素

我们在实际工作中常会遇到需要设置水平居中场景，今天我们就来看看怎么设置水平居中的。

如果被设置元素为文本、图片等行内元素时，水平居中是通过给**父元素**设置 text-align:center 来实现的。

### 定宽块状元素

当被设置元素为块状元素时用 text-align：center 就不起作用了，这时也分两种情况：**定宽**块状元素和**不定宽**块状元素。这一小节我们先来讲一讲定宽块状元素。

满足定宽和块状两个条件的元素是可以通过设置“左右margin”值为“auto”来实现居中的

### 不定宽块状元素方法

1. 加入 table 标签

第一步：为需要设置的居中的元素外面加入一个 table 标签 ( 包括 <tbody>、<tr>、<td> )。

第二步：为这个 table 设置“左右 margin 居中”（这个和定宽块状元素的方法一样）。

2. 第二种方法：改变块级元素的 dispaly 为 inline 类型，然后使用 text-align:center 来实现

居中效果。

3.方法三：通过给父元素设置 float，然后给父元素设置 position:relative 和 left:50%，子元素设置 position:relative 和 left:-50% 来实现水平居中。

## 垂直居中

### 父元素高度确定的单行文本

**父元素高度确定的单行文本**的竖直居中的方法是通过设置父元素的 height 和 line-height 高度一致来实现的

### 父元素高度确定的多行文本（方法一）

父元素高度确定的多行文本、图片、块状元素的竖直居中的方法有两种：

方法一：使用插入 table (包括tr、td)标签，同时设置 vertical-align：middle。

<style>

.wrap{height:300px;background:#ccc}

</style>

<table class="wrap"><tr><td>

<div>

<img src="http://img.mukewang.com/54ffac56000169c001840181.jpg" title="害羞的小女生"/>

</div>

</td></tr></table>

## 隐性改变display类型

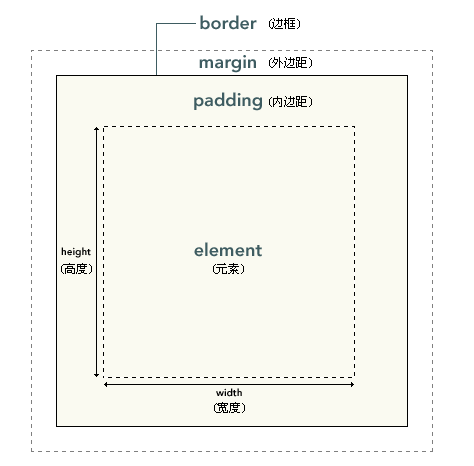
有一个有趣的现象就是当为元素（不论之前是什么类型元素，display:none 除外）设置以下 2 个句之一：

1. position : absolute
2. float : left 或 float:right

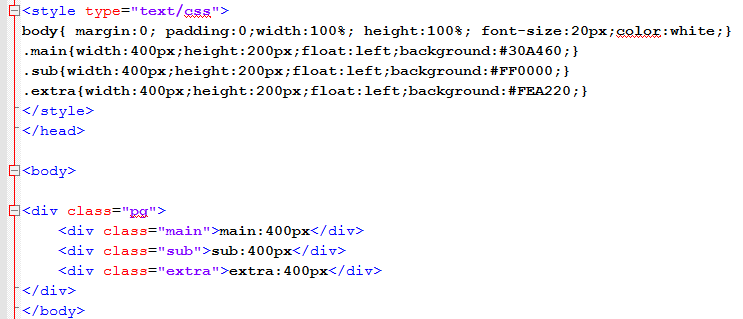
元素会自动变为以 display:inline-block 的方式显示，当然就可以设置元素的 width 和 height 了且默认宽度不占满父元素。

## Padding的一点说明

Css中box model示意图如下：



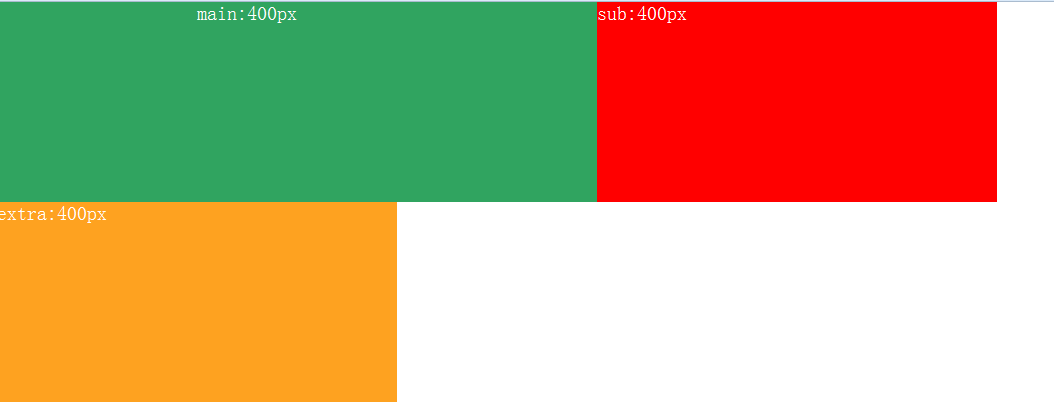
根据直觉，padding的设置只会影响content相对于边框的位置，并不会影响box的大小。可是，由padding的翻译——“内边框”可以看出，其实padding是靠增加box的大小来达到定位content的目的。示例如下：



这段代码效果是依次float排列三个宽度为400px的元素



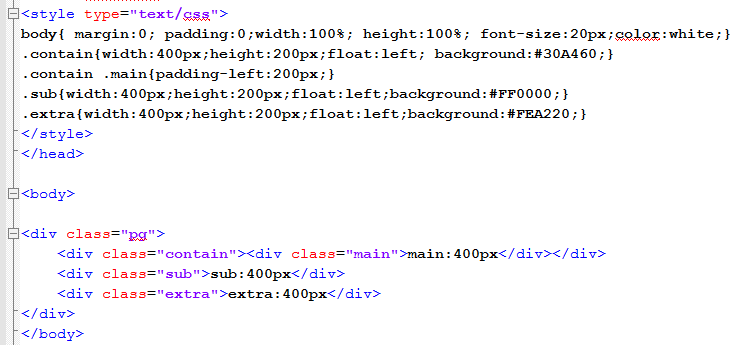
当设置main元素的padding时，效果如下：



可见，padding将main元素硬生生地撑大200px，来达到将main元素的content移至距左边框200px的位置的。因此padding的设置将改变元素的大小。

如果想要移动content的位置而又不改变元素的大小，其中一种方案是：

在main标签外再嵌套一个contain标签，将padding的设置交给main，其他设置统统扔给contain。代码如下：

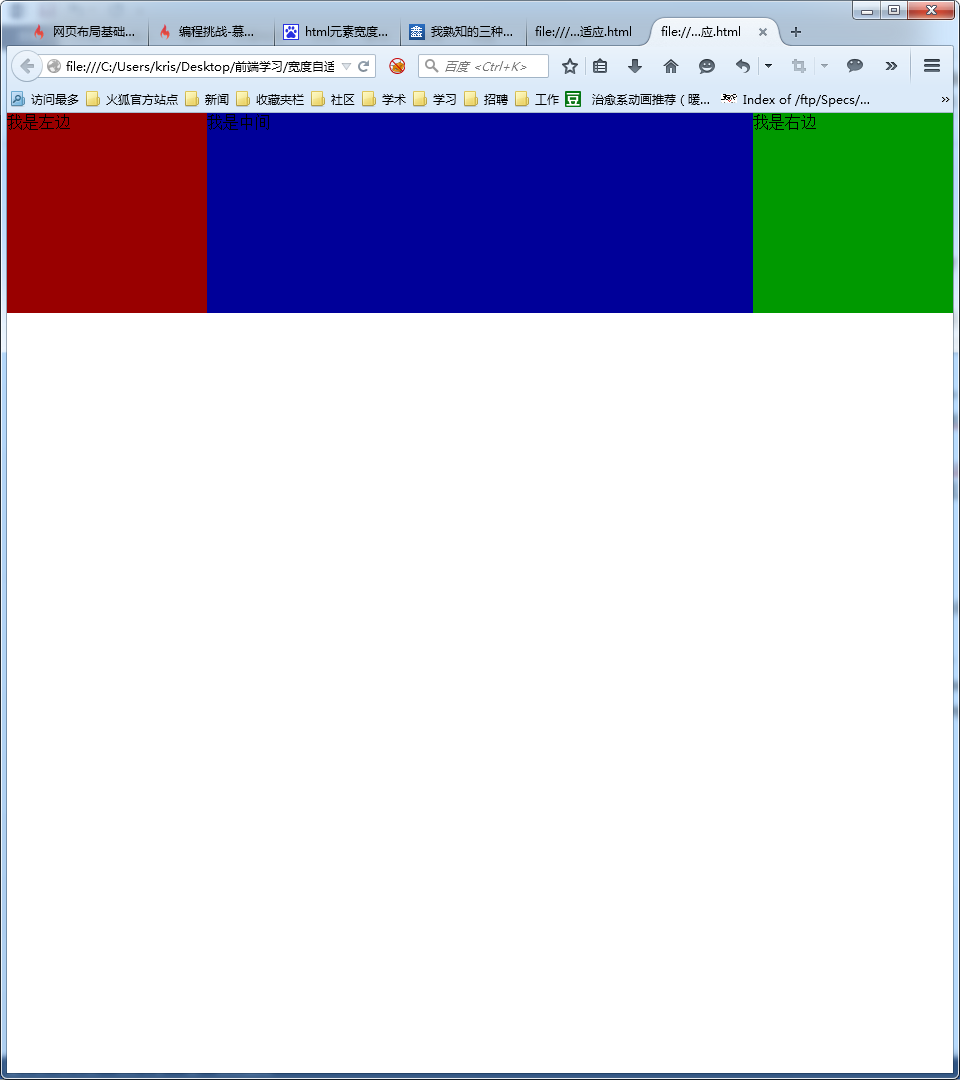


其效果如下：



## 宽度自适应

实现左右栏宽度固定，中间栏宽度自适应，效果如下：

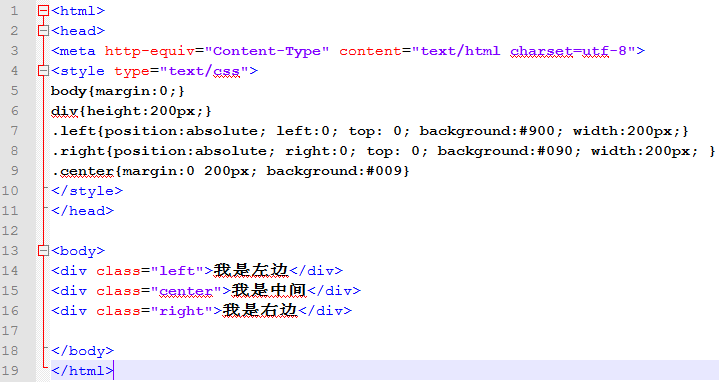


* 利用position:absolute

1. 设置左右栏position为absolute，left、right分别为0，然后top全都设置为0，这点别忘了！！！

2. 设置中间栏margin为margin: 0 210px;

具体代码如下



此方法的缺点在于必须设置top值，否则右边栏会掉到下面一行。

* 利用float

1. 左右栏分别设置float为left和right。

2. 设置margin: 0 210px;

具体代码如下：



此方法的确定在于：必须将自适应的元素摆在其他元素后面。另外：不足在于：中间主体存在克星，clear:both属性。如果要使用此方法，需避免明显的clear样式。

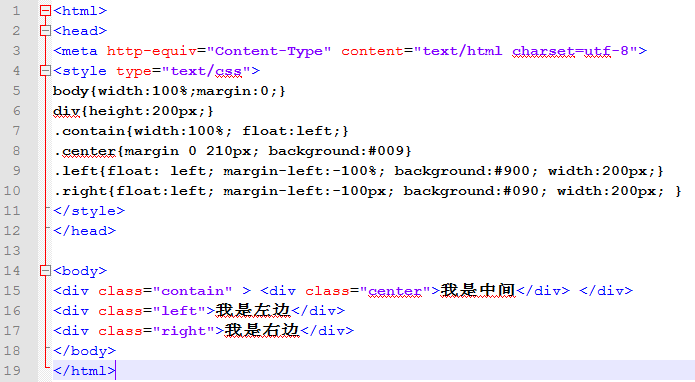
* margin负值

1. 中间的主体要使用双层标签。外层div宽度100%显示，并且浮动（本例左浮动，下面所述依次为基础），内层div为真正的主体内容，含有左右200像素的margin值。

2. 左栏与右栏都是采用margin负值定位的，左栏左浮动，margin-left为-100%，由于前面的div宽度100%与浏览器，所以这里的-100%margin值正好使左栏div定位到了页面的左边；

3. 右边栏也是左浮动，其margin-left也是负值，大小为其本身的宽度即200像素。刚好覆盖中间栏的最右边。

具体代码如下：



注意：想实现中间栏宽度自适应，右边栏的margin-left必须大于等于本身宽度，只是大于时，右边会露出一些中间栏。如下图：

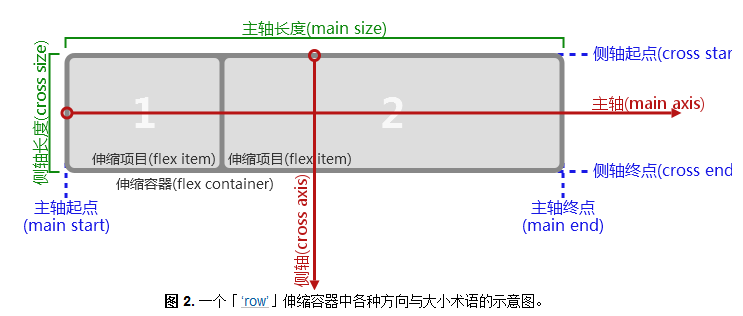


此方法三栏相互关联，能实现真正的自适应，但是理解上有些难度，且出现百分比宽度，过多的负值定位，如果出现布局的bug，排查不易。

## 伸缩布局

一个设有「display:flex」或「display:inline-flex」的元素是一个伸缩容器，伸缩容器的子元素被称为为伸缩项目，这些子元素使用伸缩布局模型来排版。

与布局计算偏向使用书写模式方向的块布局与行内布局不同，伸缩布局偏向使用[*伸缩流*](http://www.w3.org/html/ig/zh/css-flex-1/#flex-flow-property)方向。为了让描述伸缩布局变得更容易，本章节定义一系列相对于伸缩流的术语。「[*flex-flow*](http://www.w3.org/html/ig/zh/css-flex-1/#flex-flow)」的值决定了这些术语如何对应到物理方向（顶／右／底／左）、物理轴（垂直／水平）、物理大小（宽度／高度）。



### 伸缩容器 ―「display」值「[*flex*](http://www.w3.org/html/ig/zh/css-flex-1/#display-flex)」与「[*inline-flex*](http://www.w3.org/html/ig/zh/css-flex-1/#display-inline-flex)」

「*flex*」

这个值让元素产生块级[*伸缩容器*](http://www.w3.org/html/ig/zh/css-flex-1/#flex-container)盒。

「*inline-flex*」

这个值让元素产生行内级[*伸缩容器*](http://www.w3.org/html/ig/zh/css-flex-1/#flex-container)盒。

### flex-flow

flex-flow是用来伸缩行换行，flex-flow属性是同时设定“flex-direction(伸缩流的方向)”和“flex-wrap（伸缩行换行）”属性的缩写

### flex-direction

flex-direction属性可以用来设定伸缩容器的主轴的方向，这也决定了用户代理配置伸缩项目的方向。主要适用于伸缩容器，主要包括以下几个值：

|  |  |
| --- | --- |
| row | flex-direction的默认值，表示伸缩容器的主轴与当前书写模式的行内轴（文字布局的主要主向）。主轴起点与主轴终点方向分别等同于当前书写模式的始与结方向 |
| row-reverse | 表示的是除了主轴起点与主轴终点方向交换以外同row属性值的作用 |
| column | 表示的是伸缩容器的主轴与当前书写模式的块轴（块布局的主要方向）同向。主轴起点与主轴终点方向分别等同于当前书写模式的前与后方向。简单的可以理解为列布局 |
| column-reverse | 表示的是除了主轴起点与主轴终点方向交换以外同“column”的属性值作用 |

### flex-wrap

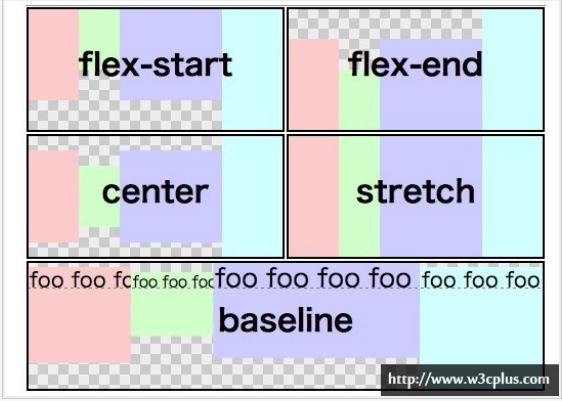
flex-wrap属性主要用来控制伸缩容器是单行还是多行，也决定了侧轴方向一新的一行的堆放方向。主要适用于伸缩容器，主要包括以下几个值：

|  |  |
| --- | --- |
| nowrap | flex-wrap的默认值，表示的是伸缩容器为单行。侧轴起点方向等同于当前书写模式的起点或前/头在侧轴的那一边，而侧轴终点方向是侧轴起点的相反方向 |
| wrap | 表示的是伸缩容器为多行。侧轴起点方向等同于当前书写模式的起眯或前/头在侧轴的那一边，而侧轴终点方向是侧轴起点的相反方向。 |
| wrap-reverse | 除了侧轴起点与侧轴终点方向交换以外同wrap所起作用相同 |

### 侧轴对齐伸缩项目——align-items

“align-items”属性允许您调整伸缩项目在侧轴的对齐方式，主要包括以下几个值：

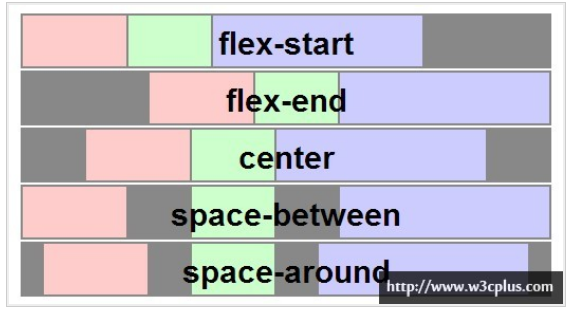
* flex-start/baseline：伸缩项目在侧轴起点边的外边距紧靠住该行在侧轴起点的边。
* flex-end：伸缩项目在侧轴终点边的外边距靠住该行在侧轴终点的边。
* center：伸缩项目的外边距盒在该行的侧轴上居中放置。（如果伸缩行的尺寸小于伸缩项目，则伸缩项目会向两个方向溢出相同的量）。
* stretch：伸缩项目拉伸，填满整个侧轴



### 主轴对齐伸缩项目——justify-content

这个属性主要用来设置伸缩项目沿主轴的对齐方式，从而调整伸缩项目之间的间距。设置了这个属性，在主轴方向上设置的任何margin都不会起作用。其值主要有

* flex-start：伸缩项目向一行的起始位置靠齐。该行的第一个伸缩项目在主轴起点边的外边距与该行在主轴起点的边对齐，同时所有后续的伸缩项目与其前一个项目对齐。
* flex-end：伸缩项目向一行的结束位置靠齐。该行的最后一个伸缩项目在主轴终点边的外边距与该行在主轴终点的边对齐，同时所有前面的伸缩项目与其后一个项目对齐。
* center：伸缩项目向一行的中间位置靠齐。该行的伸缩项目将相互对齐并在行中居中对齐，同时第一个项目与该行的在主轴起点的边的距离等同与最后一个项目与该行在主轴终点的边的距离（如果剩余空间是负数，则保持两端溢出的长度相等）。
* space-between：伸缩项目会平均地分布在一行里。如果剩余空间是负数，或该行只有一个伸缩项目，则此值等效于「flex-start」。在其它情况下，第一个项目在主轴起点边的外边距会与该行在主轴起点的边对齐，同时最后一个项目在主轴终点边的外边距与该行在主轴终点的边对齐，而剩下的伸缩项目在确保两两之间的空白空间相等下平均分布。
* space-around: 每个项目两端都有空间



### 改变元素布局的顺序

通过“order”属性来修改伸缩项目的布局顺序（在不调整结构前提之下）

默认情况之下，所有的伸缩项目的顺序组都是“0”。我们可以很容易的给每个伸缩项目设置不同的顺序值。更高的值会排在后面，而原来的HTML结构并不会有任何变化。

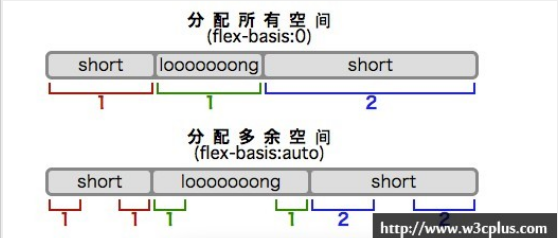
### 使你的元素具有弹性

Flexbox最强大的特性是能够通过“flex-flow”属性设置伸缩项目的流动方向，或者可以通过“flex”属性设置一个子元素可用的空间。flex的取值可以有三个部分

flex: [flex-grow] [flex-shrink] [flex-basis]

如flex:1 200px;

* flex-grow：此属性值为正数值，用来设置扩展比率，也就是剩余空间是正值的时候此伸缩项目相对于伸缩容器里其他伸缩项目能分配到空间比例。若省略则会被设置为“1”。
* flex-shrink：此属性值为正数值，用来设置收缩比率，也就是剩余空间是负值的时候此伸缩项目相对于伸缩容器里其他伸缩项目能收缩的空间比例。若省略则会被设置为“1”，在收缩的时候收缩比率会以伸缩基准值加权。
* flex-basis：与width属性使用相同的值，可以用来设置flex-basis长写并指定伸缩基准值，也就是根据可伸缩比率计算 出剩余空间的分布之前，伸缩项目主轴长度的起始数值。若在flex缩写省略了此属性设置，则flex-basis的指定值是“0”，若flex-basis的指定值是“auto”，则伸缩基准值的指定值是元素主轴长度属性的值。



上图是一个显示「绝对」伸缩（以零为基准值开始）与「相对」伸缩（以项目的内容大小为基准值开始）差异的图解。这三个项目的伸缩比例分别是「1」、「1」、「2」

多行flxbox(flex-flow:row wrap)和伸缩长度flex（如:flex:1 0 7rem）以及媒体查询，实现了一个[完美的效果](http://www.w3cplus.com/css3/flexbox-basics.html)。在不同视窗宽度下，伸缩项目在伸缩容器中可以平滑的进行变化。

flex:auto”时，伸缩容器中的伸缩项目（相当于flex: 1 1 auto）将使用其大小根据任何width/height或者min-width/min-height设定，它将扩展占用一个比例的任何自由空间可用，但在没有额外的自由空间将缩小以适应其内容。

flex:initial”(相当于flex:0 1 auto)，你会看到，当有多余的空间时，第三个子元素安容器大小不再增加，但仍然需要收缩。

**在使用flexbox之前将页面所有元素（比如：nav、header、footer等）设置成“display:block”，在支持flexbox的 浏览器会将内容自动伸缩，以适应整个设备的宽度。在不识别flexbox的浏览器中会按顺序整块的排列，但并不会影响你的阅读。**

### 伸缩项目的默认最小长度

为了让伸缩项目有合理的最小长度，本规范为 CSS 2.1 定义的「min-width」与「min-height」属性增加一个新的「auto」值作为这些属性的初始值。[CSS21]

名称： min-width, min-height

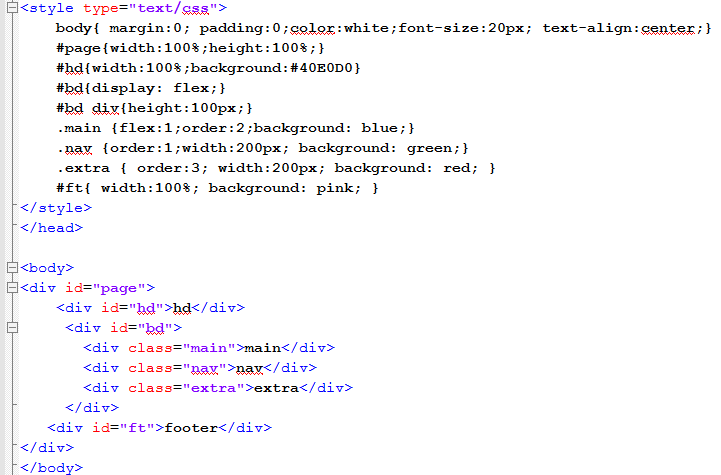
新值： auto

新初始值： auto

新计算值： 指定的百分比率或绝对长度或关键字

例子——用flexbox实现双飞翼

代码如下：



显示效果如下：

