5 CSS+DIV布局2

浙 江 师 范 大 学

主讲老师：徐晓丹（663601）

主要内容

- 盒子模型
- 应用实例

css盒子模型

- css中的盒子(box)模型用于描述为一个html元素形成的矩形盒子。盒子模型还涉及为各个元素调整外边距（margin)、边框(border)、内边距(padding)和内容的具体操作。

1. border-style边框样式属性

- 在CSS里，利用边框样式属性不仅统一设置四边框的样式，还可以对单边框进行设置。
- 基本语法：
  - border-style:样式值 //统一设置四条边框，以下为单独设置
  - border-top-style:样式值
  - border-bottom-style:样式值
  - border-left-style:样式值
  - border-right-style:样式值
- 语法说明：
  - **border-style是一个复合属性，复合属性的值有四种设置方法，其他4个都是单个边框的样式属性，只能取一个值。**
  - **设置一个值：四条边框宽度均使用一个值。**
  - **设置两个值：上边框和下边框宽度调用第一个值，左边框和右边框宽度调用第二个值。**
  - **设置三个值：上边框宽度调用第一个值，右边框与左边框宽度调用第二个值，下边框调用第三个值。**
  - **设置四个值：四条边框宽度的调用顺序为上、右、下、左。**

边框样式

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。
inherit	规定应该从父元素继承边框样式。

2. border-width边框宽度属性

- border-width
- border-top-width
- border-bottom-width
- border-left-width
- border-right-width
- 使用像素值作为单位。

3. border-color边框色彩属性

- border-color属性是用来控制边框显示的颜色，和边框宽度、边框样式的设置方法一样，既可以统一（border-color）设置，也可以分别设置每个边框（border-top- color、border-bottom- color、border-left- color、border-right- color）的颜色。

4. border属性的复合设置

- 在CSS中，border属性用来同时设置边框的样式、宽度和颜色，也可以另外对每个边界属性单独进行设置。
- 基本语法：
  - border:边框宽度 | 边框样式 | 边框颜色
  - border-top:上边框宽度 | 上边框样式 | 上边框颜色
  - border-bottom:下边框宽度 | 下边框样式 | 下边框颜色
  - border-left:左边框宽度 | 左边框样式 | 左边框颜色
  - border-right:右边框宽度 | 右边框样式 | 右边框颜色

填充

- 填充（padding，也可以叫内边距或内补白）主要是指边框和内部元素之间的空白距离，利用padding属性设置元素内的边界时，也包括5个属性，同样也有四种设置方法。
- 基本语法：
  - padding:长度 | 百分比
  - padding-top: 长度 | 百分比
  - padding-bottom: 长度 | 百分比
  - padding-left: 长度 | 百分比
  - padding-right: 长度 | 百分比
- 语法说明：
  - 长度包括长度值和长度单位，百分比是相对于上级元素宽度的百分比，不允许用负数，padding作为复合属性的四种取值方法请参考边框样式的取值方法。

边距

- 边距（margin）可以单独设置（margin-top、margin-right、margin-bottom、margin-left）四个边距，也可以简写为margin统一设置，按上、右、下、左顺时针方向书写。
- 基本语法：
  - margin-(top、right、bottom、left):长度单位 | 百分比单位 | auto
- 代码示例：
  - #nav { margin-top: 10px; margin-bottom: 10px;}
  - #main { margin: 0px auto 0px auto; }
- 采用margin速记方式，可以进一步简写代码，以上第2行还可以速记为“#main{margin:0 auto;}”，该样式除表示边距外，还表示具有块属性的元素相对其父元素来说是水平居中对齐。

边框高级属性

- CSS3对原来的盒模型功能进行了完善，增强了元素边框和背景样式的控制能力，还新增了不少UI特性，用以解决用户界面设置问题。

圆角边框

- 基本语法：
  - border-radius：[<length> | <percentage>]{1,4} [/ [<length> | <percentage>]{1,4}]?
  - 派生的相关子属性：border-top-right-radius、border-bottom-right-radius、border-bottom-left-radius、border-top-left-radius分别定义右上角、右下角、左下角、左上角的圆角。
- 语法说明：
  - <length> / <length>这两个参数分别定义角形状的四分之一椭圆的两个半径，每个参数允许设置1~4个参数值（对应那4个派生子属性）。第一个值表示圆角的水平半径，第二个值表示圆角的垂直半径，两个参数值通过斜线分隔。如果第二个值省略，则它等于第一个值，这时这个角就是一个四分之一圆角。如果任意一个值为0，则这个角是矩形，不会是圆的。值不允许是负值。
  - 四个子属性既可以单独使用，也可以采用border-radius的速记形式按照top-left、top-right、bottom-right、bottom-left的顺序来设置的，中间用空格分隔。如果bottom-left省略，那么它等于top-right；如果bottom-right省略，那么它等于top-left；如果top-right省略，那么它等于top-left。

盒阴影

- 基本语法：
  - box-shadow：none | <shadow> [, <shadow>]*
  - <shadow> = inset? && <length>{2,4} && <color>?
- 语法说明：
  - box-shadow这个属性应用的非常普遍，可以使你的元素立刻变得漂亮起来，只是记得不要把值设得太离谱。取值none表示无阴影，取值<shadow>可以使用公式表示为 inset? && [<length>{2,4} && <color>?]，其中，inset表示设置阴影的类型为内阴影，默认为外阴影；<length>是由浮点数字和单位标识符组成的1到4个长度值，用来定义阴影水平偏移、垂直偏移、阴影模糊值和阴影外延值；<color>表示阴影颜色（color of shadow）。
  - 参数<length>可取1到4个值：第1个长度值用来设置对象的阴影水平偏移值（x-offset可取正负值）；第2个长度值用来设置对象的阴影垂直偏移值（y-offset可取正负值）；如果提供了第3个长度值则用来设置对象的阴影模糊值（blur不允许负值）；如果提供了第4个长度值则用来设置对象的阴影外延值（可取正负值）。

图像边框

- border-image专门用于图像边框的处理，它的强大之处在于能够灵活地分割图像，并应用于边框。
- 基本语法：
  - border-image：<' border-image-source '> || <' border-image-slice '> [/ <' border-image-width '> | / <' border-image-width '>? / <' border-image-outset '>]? || <' border-image-repeat '>
- 子属性语法：
  - border-image-source：none | <image>
  - border-image-slice：[<number> | <percentage>]{1,4} && fill?
  - border-image-width：[<length> | <percentage> | <number> | auto]{1,4}
  - border-image-outset：[<length> | <number>]{1,4}
  - border-image-repeat：[stretch | repeat | round | space]{1,2}

- 语法说明：
  - border-image是个复合属性，设置或检索对象的边框样式使用图像来填充。
  - 应用范围：所有的元素，除了table的样式属性border-collapse是collapse时。
  - <border-image-source>：设置或检索对象的边框是否用图像定义样式或图像来源路径。None表示无背景图片，<image>使用绝对或相对地址指或者创建渐变色来确定图像。
  - <border-image-slice>：设置或检索对象的边框背景图的分割方式。<number>用浮点数指定宽度，不允许负值。<percentage>用百分比指定宽度，参照其包含块区域进行计算，不允许负值。fill保留裁减后的中间区域，其铺排方式遵循<border-image-repeat>的设定。该属性指定从上、右、下、左方位来分隔图像，将图像分成4个角，4条边和中间区域共9份，中间区域始终是透明的（即没图像填充），除非加上关键字fill。

- 语法说明：
  - <border-image-width>：设置或检索对象的边框厚度。该属性用于指定使用多厚的边框来承载被裁剪后的图像，该属性可省略，由外部的<border-width>来定义。<length>用长度值指定宽度，<percentage>用百分比指定宽度（参照其包含块进行计算），<number>用浮点数指定宽度，都不允许负值。如果设置为auto值，则<border-image-width>采用与<border-image-slice>相同的值。
  - <border-image-outset>：设置或检索对象的边框背景图的扩展。该属性用于指定边框图像向外扩展所定义的数值，即如果值为10px，则图像在原本的基础上往外延展10px再显示。<length>用长度值指定宽度，<number>用浮点数指定宽度，都不允许负值。
  - <border-image-repeat>：设置或检索对象的边框图像的平铺方式。该属性用于指定边框背景图的填充方式，可定义0-2个参数值，即水平和垂直方向。如果2个值相同，可合并成1个，表示水平和垂直方向都用相同的方式填充边框背景图；如果2个值都为stretch，则可省略不写。stretch指定用拉伸方式来填充边框背景图。repeat指定用平铺方式来填充边框背景图，当图片碰到边界时，如果超过则被截断。round指定用平铺方式来填充边框背景图，图片会根据边框的尺寸动态调整图片的大小直至正好可以铺满整个边框。space指定用平铺方式来填充边框背景图，图片会根据边框的尺寸动态调整图片的之间的间距直至正好可以铺满整个边框。

盒子的高度、宽度计算

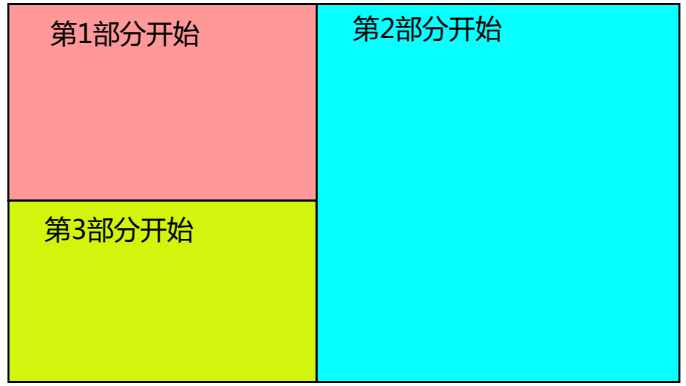
- 在css中设计一个块级元素的width和height属性时比如.box{width ：100px; height:100px}时，其中的width 和height仅仅是对content部分设置的。而不是内容，内边距，边框的总和 。
- 盒子的高度=内容+边距+边框

盒子模型高度举例

```
#st1 {
    clear: left;
    float: left;
    height: 200px;
    width: 200px;
    background-color:#9F3;
}
#st2 {
    clear: right;
    float: right;
    height: 400px;
    width: 200px;
    background-color:#F06;
}
```

见例子：hezi1.html

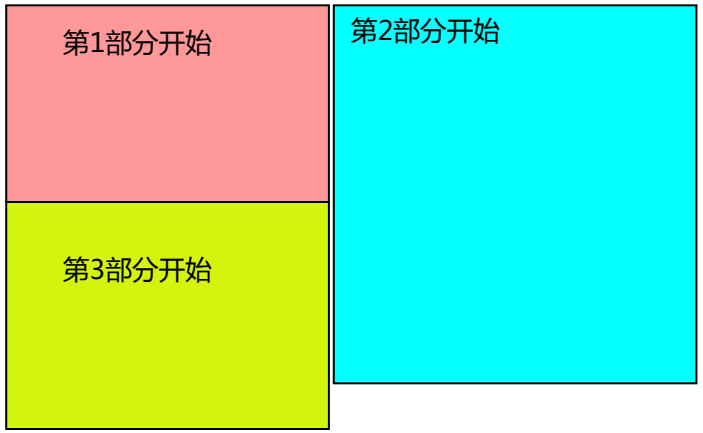
```
#st3 {
        clear: left;
        float: left;
        height:
200px;
        width: 200px;
        background-
color:#F99;
```




```
<div style="width:400px; height:400px">
<div id="st1">第1部分开始</div>
<div id="st2">第2部分开始</div>
<div id="st3">第3部分开始</div>
</div>
```

盒子模型高度举例

```
#st1 {
    clear: left;
    float: left;
    height: 200px;
    width: 200px;
    background-color:#9F3;
}
#st2 {
    clear: right;
    float: right;
    height: 400px;
    width: 200px;
    background-color:#F06;
}
#st3 {
    clear: left;
    float: left;
    height: 200px;
    width: 200px;
    background-color:#F99;
      padding-top:50px;  }
```




```
<div style="width:400px; height:400px">
<div id="st1">第1部分开始</div>
<div id="st2">第2部分开始</div>
<div id="st3">第3部分开始</div>
</div>
```

盒子模型的高度

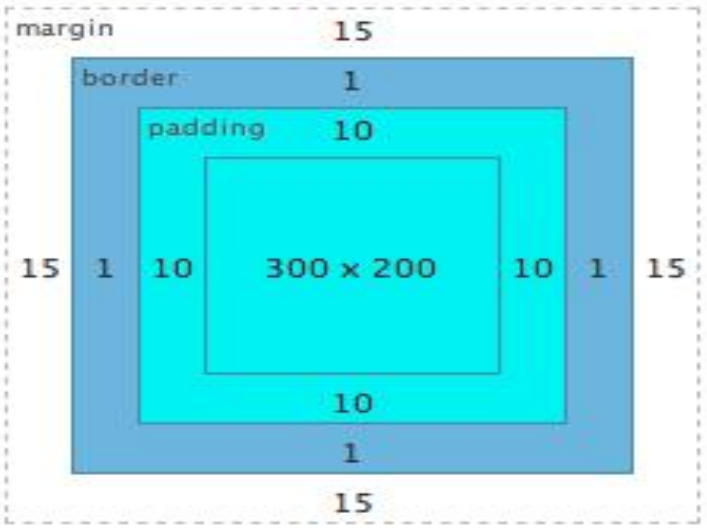
width： 300px;

height： 200px;

padding： 10px;

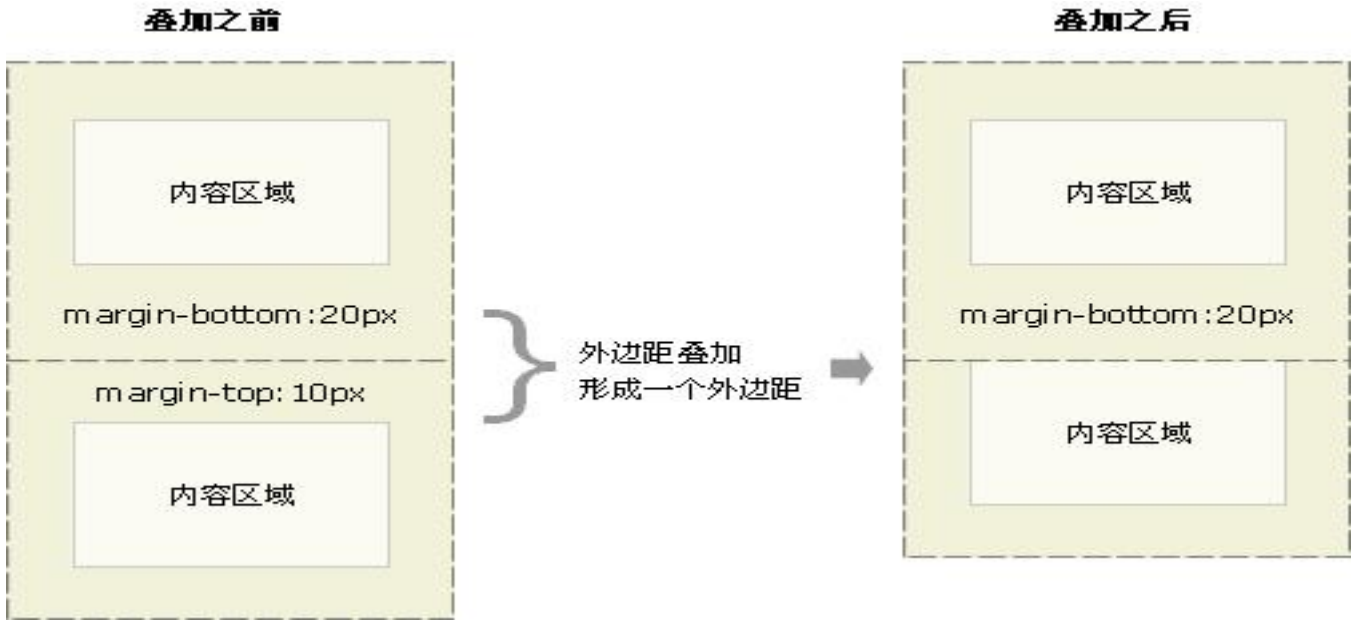
border： 1px solid #000;

margin： 15px;



如图所示：该盒子总的尺寸为352*252

- 纵向的无定位（static）元素的相邻外边距会叠加合成为其中一个较大宽度的外边距的值。



盒子的定位

- position属性：static；relative；absolute；fixed
- 放置一个块级元素于页面上时，如果不设置它的定位属性即position:static，或者position:relative,块的宽度是自动延伸填充其父元素的宽度区域

```
{
background:black;
color:White;
height:100px;
padding:10px;
border:20px solid Red;
margin:30px;
}
```

- 但是浮动floated元素和绝对定位元素，他们会收缩以致包裹紧贴内容

```
{
background:black;
color:White;
height:100px;
padding:10px;
border:20px solid Red;
margin:30px;
position:absolute;
}
```

position的属性

- static:元素框正常生成，默认设置。
- absolute:绝对定位，相对于包含块（父层）定位。
- relative：相对定位，相对于元素在文档流中初始位置。

例子1：hezi4.html

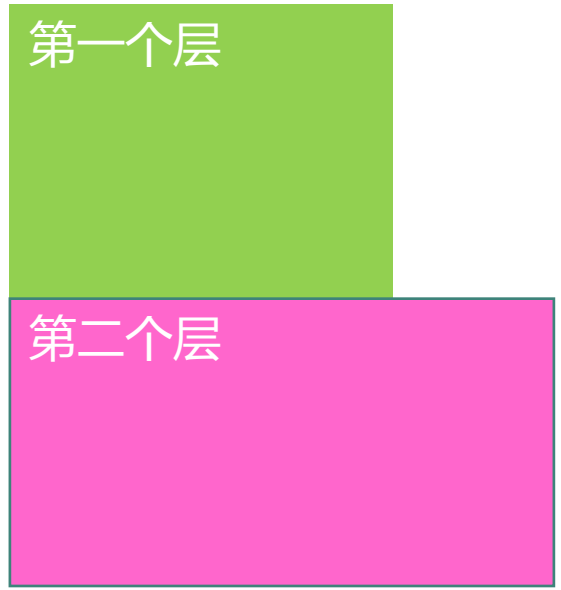
要设置层居中显示，如果使用绝对定位就容易出错，因为浏览器窗口一直是随着分辨率的大小自动适应的

例子2：heizi5.html

abolute从文档流中脱离出来，可以实现层的重叠，relative还占据文档流中的位置，层不能重叠。

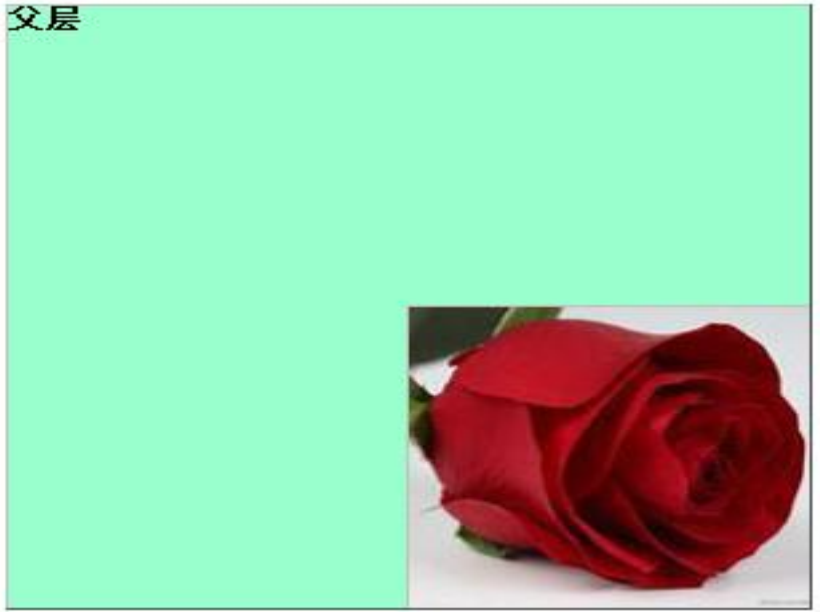
position的属性

```
.s1 {
        background-color: #9FC;
        height: 200px;
        width: 200px;}
.s2 {
        background-color: #F9C;
        height: 200px;
        width: 300px;
          top: 100px;
        position:absolute;-->
<div class="s1">第1个层</div>
<div class="s2">第2个层</div>
```



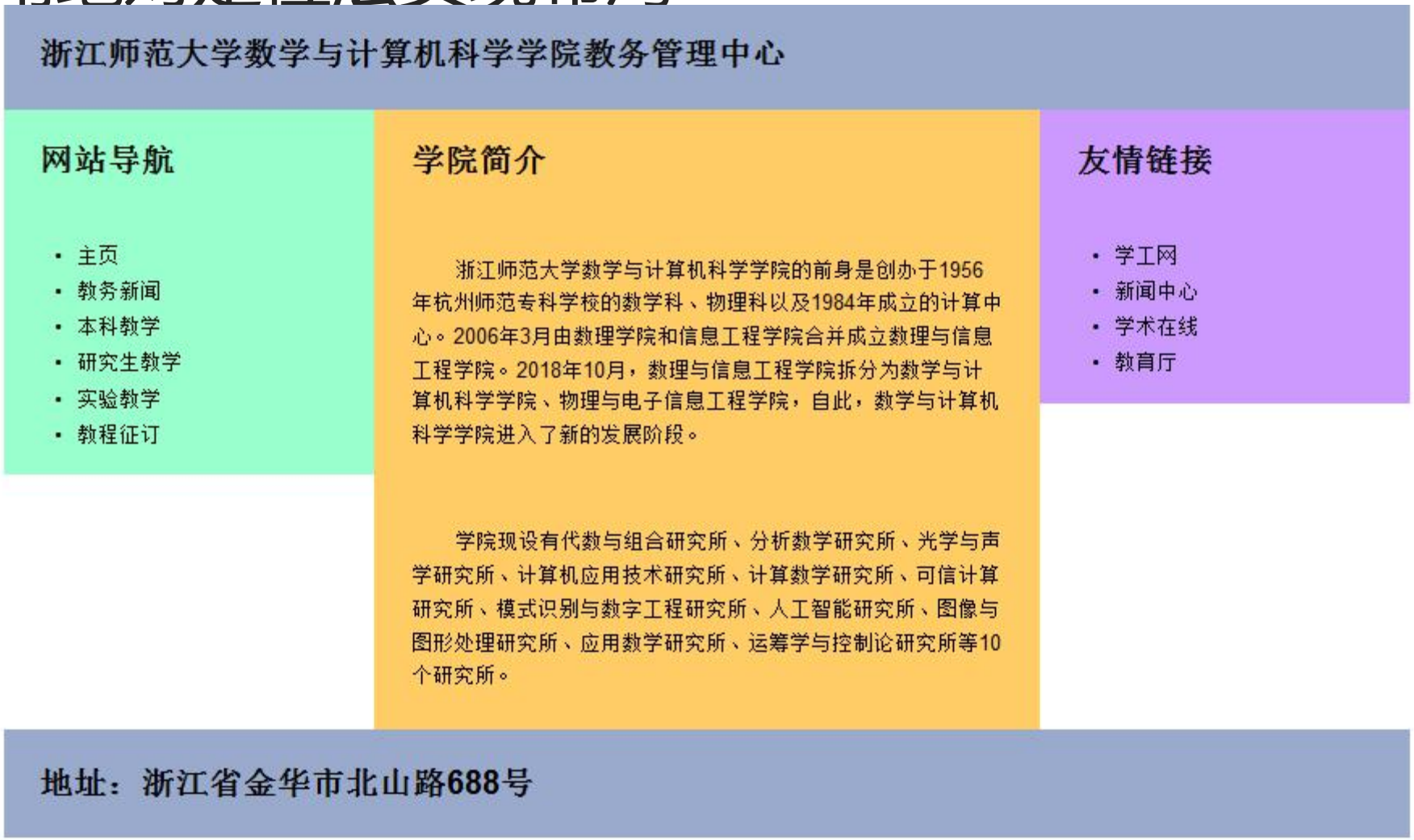
absolute

- Absolute是相对于父元素的定位。



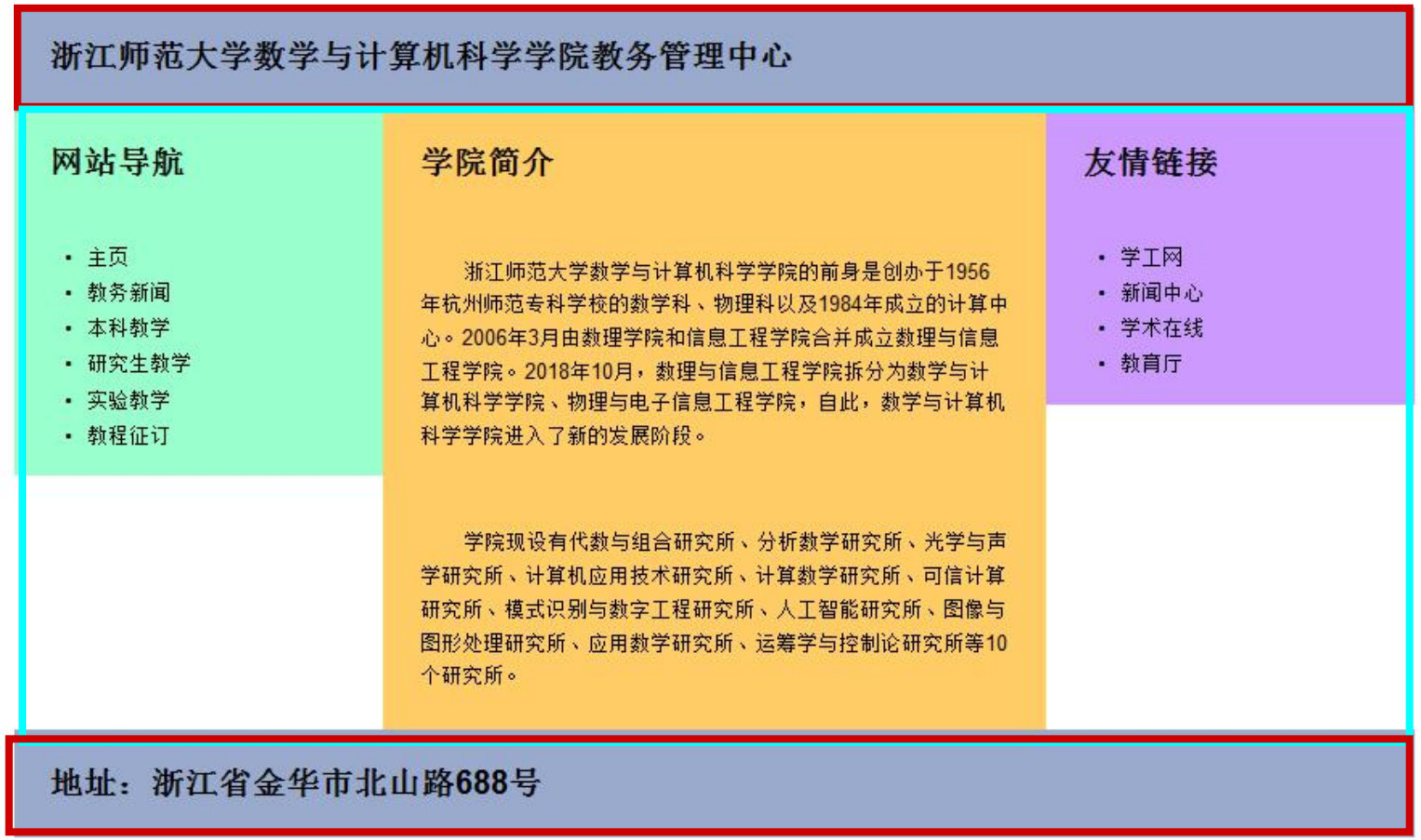
例子1多个层布局

- 使用绝对定位法实现布局



例子1多个层布局

- 使用绝对定位法实现布局



```
<div style="width:760px; margin:auto">
<div id= “#header">浙江师范大学</div>
<div id= “#container">
<div id= “#left” >左边栏</div>
<div id= “#middle” >中间内容</div>
<div id= “#right” >右边</div>
</div>
<div id= “#footer” >booter栏</div>
</div>
```

```
#header,#footer {
  background:#99AACC;
  width:760px;
  }
#container {
        position: relative;
        width:760px;
        margin:0 auto;
        }
```

```
#left {
    width: 200px;
    position: absolute;
    left: 0px;
    top: 0px;
    background:#99FFCC;
}
#middle {
    margin-right: 200px;
    margin-left: 200px;
    background:#ffcc66;
    }
#right {
    width: 200px;
    position: absolute;
    right: 0px;
    top: 0px;
    background:#CC99FF;
    }
```

Left,right脱离了标准流，middle保持标准流的模式，盒子在标准流中按照从左到右，从上到下的顺序排放。
利用父元素为relative,子元素为absolute实现定位

```
h2{
    margin:0;
    padding:20px;
    }
p{
    padding:20px;
    text-indent:2em;
    margin:0;
    }
```

使用float属性布局

- 修改样式为

```
#container {
    width:760px;
    }
#left {
    width: 200px;
    background:#99FFCC;
}           Float:left;
#middle {
            background:#ffcc66;
    }           width:360px;
                    Float:left;
#right {
    width: 200px;
    background:#CC99FF;
            Float:right;
}
```


```
#header,#footer {
        background:#99AACC;
        width:760px;
                clear:both;
}
```

伸缩盒

- CSS3引入一种新的布局模式——Flexbox布局，即伸缩盒（Flexible Box）模型，用来提供一个更加有效的方式制定、调整和分布一个容器里的项目布局，即使它们的大小是未知或者动态的。

Flexbox布局模式

- CSS3引入的布局模式Flexbox布局，主要思想是让容器有能力让其子项目能够改变其宽度和高度（甚至顺序），以最佳方式填充可用空间（主要是为了适应所有类型的显示设备和屏幕大小）。Flex容器会使子项目（伸缩项目）扩展来填满可用空间，或缩小它们以防止溢出容器。
- 最重要的是Flexbox布局方向不可预知，而原先常规的页面布局（块就是从上到下，内联就从左到右）对于支持大型或者复杂的应用程序（特别是涉及取向改变、缩放和收缩等）又缺乏灵活性。

Flexbox模型的功能

- Flexbox布局对于设计比较复杂的页面非常有用，可以轻松地实现屏幕和浏览器窗口大小发生变化时保持元素的相对位置和大小不变，同时减少了依赖于浮动布局实现元素位置的定义以及重置元素的大小。

Flexbox模型的功能

- Flexbox布局在定义伸缩项目大小时伸缩容器会预留一些可用空间，可以调节伸缩项目的相对大小和位置。例如可以确保伸缩容器中的多余空间平均分配多个伸缩项目。当然，如果伸缩容器没有足够大的空间放置伸缩项目时，浏览器会根据一定的比例减少伸缩项目的大小，使其不溢出伸缩容器。

Flexbox布局功能特点：

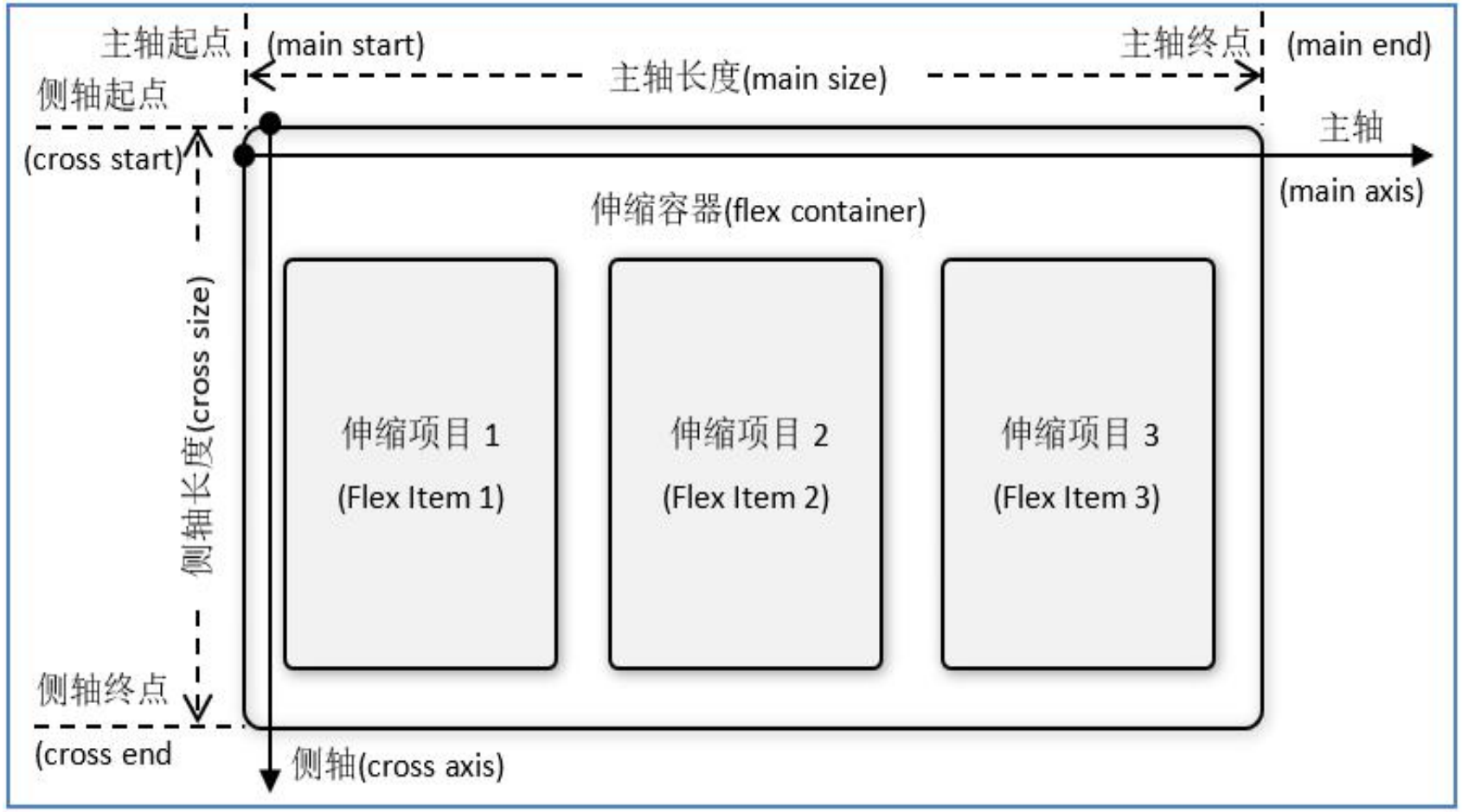
- 屏幕和浏览器窗口大小发生改变也可以灵活调整布局。
- 指定伸缩项目沿着主轴或侧轴按比例分配伸缩容器额外空间，以调整伸缩项目的大小。
- 指定伸缩项目沿着主轴或侧轴将伸缩容器额外空间分配到伸缩项目之前、之后或之间。
- 指定如何将垂直于元素布局的额外空间分布到该元素的周围。
- 控制元素在页面上的布局方向。
- 按照不同于文档对象模型（DOM）所指定排序方式对屏幕上的元素重新排序。也就是说可以在浏览器渲染中不按照文档流先后顺序重排伸缩项目顺序。

Flexbox模型相关术语

- Flexbox不是一个属性，而是一个模块，包括多个CSS3属性，涉及很多东西，包括整个组属性。

1. 主轴和侧轴

- 用户代理沿着一个伸缩容器的主轴配置伸缩项目，主轴是主轴方向的延伸。伸缩项目主要沿着伸缩容器的主轴进行布局。但它不一定是水平的，这主要取决于justify-content属性，如果其取值为colunm主轴方向则变为垂直方向。



2. 伸缩容器和伸缩项目(1)

- 通过display属性显式地给一个元素设置flex或者inline-flex，这个容器就是一个伸缩容器。伸缩容器会为其内容创建新的伸缩格式化上下文，其中设置为flex的容器被渲染为一个块级元素，而设置为inline-flex的容器则渲染为一个行内元素。若元素display的指定值是inline-flex且元素是一个浮动或绝对定位元素，则display的计算值是flex。

2. 伸缩容器和伸缩项目(2)

- 一个伸缩容器的内容具有零个以上的伸缩项目——伸缩容器的每个子元素（除了需要修复的元素之外）都会成为一个伸缩项目，且用户代理会将任何直接在伸缩容器里的连续文字块包起来成为匿名伸缩项目。

3. 伸缩容器的属性

- Flexbox伸缩布局中伸缩容器和伸缩项目是伸缩布局模块中的重要部分，其中每一部分都具有各自的属性。
- 1）伸缩流方向。
- 2）伸缩行换行。
- 3）伸缩方向与换行。
- 4）主轴与侧轴对齐。
- 5）堆栈伸缩行。

4. 伸缩项目的属性

- 一个伸缩项目是一个伸缩容器的子元素，伸缩容器中的文本也被视为一个伸缩项目，伸缩项目中内容与普通流一样，伸缩项目都有一个主轴长度和侧轴长度。
- 1）显示顺序。
- 2）侧轴对齐。
- 3）伸缩性。

5. 伸缩行

- 伸缩项目沿着伸缩容器内的一个伸缩行定位，伸缩容器可以是单行的，也可以是多行的，主要由flex-wrap（旧版为box-lines）属性决定。单行的伸缩容器会将其所有子元素在单独的一行上进行布局，这种情况之下有可能导致内容溢出伸缩容器；多行的伸缩容器会将其伸缩项目配置在多个伸缩行上，这类似于文本的排列。当文本过宽导致一行无法容纳时，内容会断开并移至新的一行。当用户代理创建新的伸缩行时，这些伸缩行会根据flex-wrap属性沿着侧轴进行堆叠。除非伸缩容器本身是空的，每一个伸缩行至少包含一个伸缩项目。

Flexbox模型的使用

- 新版Flexbox模型使用的属性有：
- flex、flex-grow、flex-shrink、flex-basis、flex-flow、flex-direction、flex-wrap、align-content、align-items、align-self、justify-content、order。

1.伸缩容器设置display

- 伸缩容器为其内容创建新的伸缩格式化上下文（Flex formatting context），除了伸缩排版用来代替块布局以外，创建一个伸缩格式化上下文与创建一个块格式化上下文是一样的。
- 要改变元素的显示模式，需要使用display属性
- display: flex | inline-flex
- flex将对象作为弹性伸缩盒显示，inline-flex将对象作为内联块级弹性伸缩盒显示。CSS的columns、float、clear和vertical-align在伸缩项目上没有效果。

2. 伸缩流方向flex-direction

- flex-direction属性通过定义flex容器的主轴方向来决定flex子项在flex容器中的位置，这将决定flex需要如何进行排列。
  - flex-direction：row | row-reverse | column | column-reverse
- 默认值为row。row表示主轴与行内轴方向作为默认的书写模式，即横向从左到右排列（左对齐）；row-reverse对齐方式与row相反；column主轴与块轴方向作为默认的书写模式，即纵向从上往下排列（顶对齐）；column-reverse对齐方式与column相反。该属性的反转取值不影响元素的绘制，语音和导航顺序，只改变流动方向。

3. 伸缩换行flex-wrap

- flex-wrap属性控制flex容器是单行或者多行，主轴的方向决定了新行堆叠的方向。
  - flex-wrap：nowrap | wrap | wrap-reverse
- 默认值为nowrap。nowrap表示flex容器为单行，该情况下flex子项可能会溢出容器；wrap表示flex容器为多行，该情况下flex子项溢出的部分会被放置到新行，子项内部会发生断行；wrap-reverse表示反转wrap排列。

4. 伸缩流方向与换行flex-flow

- 复合属性flex-flow设置或检索弹性盒模型对象的子元素排列方式。
  - flex-flow：<' flex-direction '> || <' flex-wrap '>
- <' flex-direction '>定义弹性盒子元素的排列方向。
- <' flex-wrap '>控制flex容器是单行或者多行。

5. 主轴对齐justify-content（1）

- justify-content设置或检索Flex元素在主轴方向上的对齐方式。
  - justify-content：flex-start | flex-end | center | space-between | space-around
- 默认值为flex-start表示弹性盒子元素将向行起始位置对齐。该行的第一个子元素的主起始位置的边界将与该行的主起始位置的边界对齐，同时所有后续的伸缩盒项目与其前一个项目对齐。
- flex-end表示弹性盒子元素将向行结束位置对齐。该行的第一个子元素的主结束位置的边界将与该行的主结束位置的边界对齐，同时所有后续的伸缩盒项目与其前一个项目对齐。
- center表示弹性盒子元素将向行中间位置对齐。该行的子元素将相互对齐并在行中居中对齐，同时第一个元素与行的主起始位置的边距等同与最后一个元素与行的主结束位置的边距（如果剩余空间是负数，则保持两端相等长度的溢出）。

5. 主轴对齐justify-content(2)

- space-between表示弹性盒子元素会平均地分布在行里。如果最左边的剩余空间是负数，或该行只有一个子元素，则该值等效于flex-start。在其它情况下，第一个元素的边界与行的主起始位置的边界对齐，同时最后一个元素的边界与行的主结束位置的边距对齐，而剩余的伸缩盒项目则平均分布，并确保两两之间的空白空间相等。
- space-around表示弹性盒子元素会平均地分布在行里，两端保留子元素与子元素之间间距大小的一半。如果最左边的剩余空间是负数，或该行只有一个伸缩盒项目，则该值等效于'center'。在其它情况下，伸缩盒项目则平均分布，并确保两两之间的空白空间相等，同时第一个元素前的空间以及最后一个元素后的空间为其他空白空间的一半。

6. 侧轴对齐align-items

- align-items属性定义flex子项在flex容器的当前行的侧轴方向上的对齐方式。
  - align-items：flex-start | flex-end | center | baseline | stretch
- 默认值为stretch。flex-start表示弹性盒子元素的侧轴起始位置的边界紧靠住该行的侧轴起始边界；flex-end表示弹性盒子元素的侧轴起始位置的边界紧靠住该行的侧轴结束边界；center表示居中放置；
- baseline表示如弹性盒子元素的行内轴与侧轴为同一条，则该值与flex-start等效，其它情况下该值将参与基线对齐；stretch表示如果指定侧轴大小的属性值为auto，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照min/max-width/height属性的限制。

7. 单独侧轴对齐align-self

- 定义flex子项单独在侧轴方向上的对齐方式。
  - align-self：auto | flex-start | flex-end | center | baseline | stretch
- 默认值为auto。auto表示如果align-self的值为auto，则其计算值为元素的父元素的align-items值，如果其没有父元素，则计算值为stretch；flex-start表示弹性盒子元素的侧轴起始位置的边界紧靠住该行的侧轴起始边界；flex-end表示弹性盒子元素的侧轴起始位置的边界紧靠住该行的侧轴结束边界；center表示弹性盒子元素在该行的侧轴上居中放置（如果该行的尺寸小于弹性盒子元素的尺寸，则会向两个方向溢出相同的长度）；baseline表示如弹性盒子元素的行内轴与侧轴为同一条，则该值与flex-start等效，其它情况下，该值将参与基线对齐；stretch表示如果指定侧轴大小的属性值为auto，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照'min/max-width/height'属性的限制。

8. 堆栈伸缩行align-content

- 当伸缩容器的侧轴还有多余空间时，本属性可以用来调准“伸缩行”在伸缩容器里的对齐方式。
  - align-content：flex-start | flex-end | center | space-between | space-around | stretch
- 默认值为stretch。本属性在只有一行的伸缩容器上没有效果。
- flex-start表示各行向弹性盒容器的起始位置堆叠。
- flex-end表示各行向弹性盒容器的结束位置堆叠。
- center表示各行向弹性盒容器的中间位置堆叠。
- space-between表示各行在弹性盒容器中平均分布。
- space-around表示各行在弹性盒容器中平均分布，两端保留子元素与子元素之间间距大小的一半。
- stretch表示各行将会伸展以占用剩余的空间。

9. 伸缩性flex

- 复合属性flex设置或检索弹性盒模型对象的子元素如何分配空间。
  - flex：none | <' flex-grow '> <' flex-shrink >'? || <' flex-basis '>
- none关键字的计算值为： 0 0 auto。
- <' flex-grow '>用来指定扩展比率，即剩余空间是正值时此“flex子项”相对于“flex容器”里其他“flex子项”能分配到空间比例。在“flex”属性中该值如果被省略则默认为“1”。
- <' flex-shrink '>用来指定收缩比率，即剩余空间是负值时此“flex子项”相对于“flex容器”里其他“flex子项”能收缩的空间比例。在收缩的时候收缩比率会以伸缩基准值加权。在“flex”属性中该值如果被省略则默认为“1”。
- <' flex-basis '>用来指定伸缩基准值，即在根据伸缩比率计算出剩余空间的分布之前，“flex子项”长度的起始数值。在“flex”属性中该值如果被省略则默认为“0%”

9. 伸缩性flex

- 在“flex”属性中该值如果被指定为“auto”，则伸缩基准值的计算值是自身的width设置，如果自身的宽度没有定义，则长度取决于内容。
- 特殊例子：缩写“flex: 1”，其计算值为“1 1 0%”。缩写“flex: auto”，其计算值为“1 1 auto”。缩写“flex: none”，其计算值为“0 0 auto”。缩写“flex: 0 auto”或者“flex: initial”，其计算值为“0 1 auto”，即“flex”初始值。

10. 显示顺序order

- order属性设置或检索弹性盒模型对象的子元素出现的顺序。
  - order：<integer>
- <integer>用整数值来定义排列顺序，数值小的排在前面，可以为负值，默认值为0。
- 以上各属性的示例源码请参考电子版，下面用一个综合案例来看看它们的使用方法。

11. flexbox综合案例（三列布局）

图 6-3 伸缩盒三列布局效果

小结

- 在CSS中主要有inline、inline-block、block、table、absolute position和float以及一些新的盒模型，浏览器把每个元素都看做一个盒模型，每一个盒模型是由以下几个属性组合所决定的：display、position、float、width、height、margin、padding和border等，不同类型的盒模型会产生不同的布局，比如内联盒模型中以上部分属性是无效的。CSS3引入的伸缩盒模型，提供了一个更加有效的方式制定、调整和分布一个容器里的项目布局，即使它们的大小是未知或者动态的。大家要习惯伸缩盒模型的布局方式。

谢　谢　！

web开发

主讲老师：徐晓丹

浙 江 师 范 大 学