

5、盒模型的种类

在 CSS3 中使用 **display** 属性的属性值定义盒的类型，总体来说盒模型分为 **block 类型**（可以设置宽度和高度样式）和 **inline 类型**（不可以设置宽度和高度样式），比如 `div` 和 `p` 元素属于 **block 类型**，`span` 和 `a` 元素属于 **inline 类型**，再细分一下类型（即 **display 属性值**）有：

（1）inline-block 类型（很重要）：让本身属于 **block** 盒类型的元素以 **inline** 类型的方式显示，例如在 `div` 元素中将 `display` 分别设置为 `inline-block` 和 `inline` 的效果是一样的，但如果给它们指定宽度和高度，效果就不一样了，因为设置的宽度和高度只对 `inline-block` 类型起作用，对 `inline` 类型不起作用，所以 `inline-block` 类型的元素内容会溢出，而 `inline` 类型的正常显示，具体 demo 如下：

```
<style type="text/css">
    div{
        background: green;
        width: 50px;
        height: 50px;
    }
    .div1{display:inline-block;}
    .div2{display: inline;}
</style>
<div class="div1">在 CSS3 中使用 display 属性定义盒的类型</div>
<div class="div2">在 CSS3 中使用 display 属性定义盒的类型</div>
```

盒的基本类型

在
CSS3
中使用
display
属性定
义盒的
类型

效果图为：

在CSS3中使用display属性定义盒的类型

注意：inline-block 类型可以代替 float 属性，实现如下功能：

1）利用 inline-block 类型可以使多个 block 元素在一行中分列显示

CSS2.1 之前要在一行中并列显示多个 **block** 元素，需要用到 `float` 属性或者 `position` 属性。设置元素的样式为 `{float:left;}`，该元素就会变成左浮动，左浮动元素会“浮”在不浮动元素的左侧。如果两个左浮动元素的高度不一样，要使不浮动元素内容换行显示，还必须在不浮动元素中设置 `{clear:both}` 清除浮动，否则不浮动元素内容会跟在浮动元素后面显示，例如：

```
<style type="text/css">
    .div{
        width:400px;
    }
    /*div 元素如果不设置高度，会默认等于元素内容的高度，如果元素内容不一致，
       则 div 元素的高度也会不同*/
    .div1{
        background:green;
        float:left;
```

```

        width: 200px;
    }
    .div2{
        background: #f60;
        float: left;
        width: 200px;
    }
    /*div1 和 div2 表示的 div 元素都是左浮动，会并列显示，如果不清除浮动，div3 元素内容会紧跟在后面显示；若清除浮动，div3 元素内容就会换行显示*/
    .div3{
        background: #d829ff;
        clear:both;
    }
</style>
<div class="div">
    <div class="div1">利用 inline-block 类型可以使多个 block 元素在一行中分列显示，利用
inline-block 类型可以使多个 block 元素在一行中分列显示</div>
    <div class="div2">利用 inline-block 类型可以使多个 block 元素在一行中分列显示</div>
    <div class="div3">利用 inline-block 类型可以使多个 block 元素在一行中分列显示</div>
</div>
不清除浮动的效果图为：

```

利用inline-block类型分列显示

清除浮动的效果图为：

利用inline-block类型分列显示

但在 **CSS2.1** 中使用 **inline-block** 类型可以直接将两个 **div** 元素**进行并列显示**，不需要使用 **float** 和 **clear** 属性了。但默认情况下使用 **inline-block** 类型并列显示的**垂直对齐方式是底部对齐**，为了让对齐方式改为顶部对齐，还要给 **div** 元素的样式加入 **vertical-align:top**，例如：

```

<style type="text/css">
    /*要使 div1 和 div2 并列显示，可以将其 display 属性都设置为 inline-block 类型*/
    .div1{

```

```

        background:green;
        width: 200px;
        display:inline-block;
    }
    .div2{
        /*分列显示默认的垂直对齐方式是底部对齐，要改为顶部对齐，必须加上
        vertical-align 属性*/
        background: #f60;
        width: 200px;
        display:inline-block;
        vertical-align: top;
    }
    .div3{
        /*div3 元素换行显示，将 div3 元素的宽度设置为 div1 和 div2 宽度的总和*/
        background: #d829ff;
        width:400px;
    }
</style>
<div class="div">
    <!-- 前两个分列显示的 div 元素之间如果有空格，必须使第二个 div 紧邻第一个 div 其
    后才能消除空格-->
    <div class="div1">利用 inline-block 类型可以使多个 block 元素在一行中分列显示， 利用
    inline-block 类型可以使多个 block 元素在一行中分列显示</div><div class="div2">利用 inline-
    block 类型可以使多个 block 元素在一行中分列显示</div>
    <div class="div3">利用 inline-block 类型可以使多个 block 元素在一行中分列显示</div>
</div>

```

2) 使用 inline-block 类型来显示水平菜单

大多数水平菜单是基于 ul 和 li 元素来实现的，CSS2.1 之前要实现水平菜单，需要用到 float 属性，具体 demo 如下：

```

<style type="text/css">
    ul{
        /*去掉无序列表整体的外边距和内边距*/
        margin: 0px;
        padding: 0px;
    }
    li{
        padding: 10px 20px;
        background-color: #2292ff;
        border-right:solid 1px #2066c7;/*设置右边框样式*/
        width:100px;
        text-align: center;
        list-style: none;/*去掉无序列表前的小圆点*/
        float:left;/*使列表项显示在一行中*/
    }

```

```

a{
    color:#fff;
    text-decoration: none;/*去掉链接自带的下划线*/
}
</style>
<ul>
    <li><a href="index.html">首页</a></li>
    <li><a href="index.html">首页</a></li>
    <li><a href="index.html">首页</a></li>
</ul>

```

使用inline-block类型来显示水平菜单



但在 **CSS2.1** 中使用 **inline-block** 类型可以直接显示为水平菜单，不需要使用 float 属性了。具体 demo 如下：

```

<style type="text/css">
    ul{
        /*去掉无序列表整体的外边距和内边距*/
        margin: 0px;
        padding: 0px;
    }
    li{
        display: inline-block;/*使列表项显示在一行中*/
        padding: 10px 20px;
        background-color: #2292ff;
        border-right:solid 1px #2066c7;/*设置右边框样式*/
        width:100px;
        text-align: center;
    }
    a{
        color:#fff;
        text-decoration: none;/*去掉链接自带的下划线*/
    }
</style>
<ul>
    <!-- 显示在一行的 li 元素之间如果有空格，必须使下一个元素紧跟上一个元素才能消除空格 -->
    <li><a href="index.html">首 页 </a></li><li><a href="index.html">首 页 </a></li><li><a href="index.html">首 页 </a></li>
</ul>

```

使用inline-block类型来显示水平菜单



效果图为：

(2) inline-table 类型：让表格以 inline 类型的方式显示，默认垂直对齐方式是顶部对齐，为了让对齐方式改为底部对齐，还要给 table 元素的样式加入 vertical-align:bottom，例如实现表格前后都有文字围绕：

```
<style type="text/css">
    table{
        border:solid 3px #ccc;
        display: inline-table;/*表格以 inline 的方式显示*/
        vertical-align: bottom;
    }
    td{
        border:solid 3px #898989;
    }
</style>
```

```
<body>
<h3>inline-table 类型</h3>
这里是文字内容
```

```
<table>
    <tr>
        <td>1</td>
        <td>2</td>
    </tr>
    <tr>
        <td>1</td>
        <td>2</td>
    </tr>
</table>
```

```
这里是文字内容
</body >
```

效果图为：

inline-table类型



这里是文字内容

(3) list-item 类型：可以将多个元素作为列表来显示，同时在元素的开头加上列表标记，具体 demo 及效果图如下：

```
<style type="text/css">
    div{
        display:list-item;
        list-style-type: circle;
        margin-left: 20px;
    }
</style>
<div>list-item 类型</div>
<div>list-item 类型</div>
<div>list-item 类型</div>
```

list-item类型

- list-item类型
- list-item类型
- list-item类型

(4) run-in 类型与 compact 类型

如果设置为 run-in 类型或 compact 类型的元素后面还有 block 类型的元素，则 run-in 类型的元素会被包含在后面 block 类型元素的内部，而 compact 类型的元素会被放置在后面 block 类型元素的左边，具体 demo 如下：

```
<style type="text/css">
    dl.run-in dt{
        display: run-in;
        border: solid 2px #f60;
        background-color: #ccc;
    }
    dl.compact dt{
        display: compact;
        border: solid 2px #f60;
        background-color: #ccc;
    }
</style>
<dl class="run-in">
    <dt>run-in 类型</dt>
    <dd>run-in 类型的元素会被包含在 block 类型元素的内部</dd>
</dl>
<dl class="compact">
    <dt>compact 类型</dt>
    <dd>compact 类型的元素会被放置在 block 类型元素的左边</dd>
</dl>
```

run-in类型与compact类型

run-in类型 run-in类型的元素会被包含在block类型元素的内部

compact类型

效果图为： compact类型的元素会被放置在block类型元素的左边

(5) 表格相关类型：可以将多个元素的 display 属性设置为与表格相关的属性值，这些元素就可以显示为表格的形式，常用的相关属性如下：

HTML5 中的元素	对应的 display 属性值	说明
table	table	代表整个表格
caption	table-caption	代表整个表格的标题
tr	table-row	代表表格中的一行
td	table-cell	代表单元格
th	table-cell	代表表头

(6) none 类型：元素指定该类型以后就不会被显示，例如结合 UI 元素状态伪类选择器实现鼠标移动上去时元素被隐藏，移开后显示：

```
<style type="text/css">
    #a{
        width: 500px;
```

```

        height: 500px;
        background-color: #ccc;
    }
    #b{
        width: 200px;
        height: 200px;
        background-color: green;
    }
    /*鼠标移动到 a 元素上时，b 元素被隐藏，移开时显示*/
    #a:hover #b{
        display: none;
    }
</style>
<div id='a'>
    <div id="b"></div>
</div>

```

6、盒模型相关的属性

(1) overflow 属性 (内容溢出): 在盒模型的样式中设置宽度和高度之后，就有可能出现内容溢出的情况，可以使用 **overflow** 属性指定如何显示盒模型溢出的内容，可取的属性值有：**hidden** (溢出的内容被隐藏)、**scroll** (无论是否溢出都添加水平和垂直滚动条)、**auto** (只有当内容溢出时，根据需要自动添加水平或垂直滚动条)、**visible** (和不使用 **overflow** 的效果一样，即溢出内容原样显示)，例如：

```

<style type="text/css">
    div{
        width: 100px;
        height: 100px;
        background-color: green;
        overflow: auto;
    }
</style>

```

<div>在盒模型的样式中设置宽度和高度之后，就有可能出现内容溢出的情况</div>

注意：

- 1) 可以使用 **overflow-x** 属性和 **overflow-y** 属性单独指定水平或垂直方向上内容溢出的显示方式
- 2) 可以通过设置 **text-overflow** 属性的值为 **ellipsis** 在盒内容的末尾添加一个省略号 "...", 但该属性只有当盒内容在水平方向溢出时才有效，例如通过设置 **white-space: nowrap** 使得盒右端内容不能换行显示，这样就达到了水平方向溢出的效果，然后就可以使用 **text-overflow** 属性了，demo 如下：

```

<style type="text/css">
    div{
        white-space: nowrap;/*使盒内容不能换行显示，即所有内容都显示在一行中*/
        width: 300px;
        border: solid 1px green;
    }

```

```

        overflow:hidden;/*先隐藏溢出的内容*/
        text-overflow: ellipsis;/*然后在盒内容的末尾添加省略号*/
    }
</style>
<div>在盒模型的样式中设置宽度和高度之后，就有可能出现内容溢出的情况</div>

```

text-overflow属性

效果图为: 在盒模型的样式中设置宽度和高度之后...

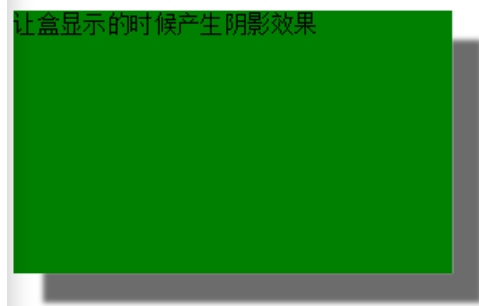
(2) box-shadow 属性 (盒阴影): 让盒显示的时候产生阴影效果，语法与 text-shadow 属性类似，同样是 `box-shadow: length length length color`，前两个 `length` 分别表示阴影离开盒的横向位移（正数表示向右）和纵向位移（正数表示向下），第三个 `length` 表示阴影模糊半径（数字越大越模糊，当设置为 0 时，阴影不会向外模糊），`color` 表示阴影颜色，具体 demo 及效果图如下：

```

<style type="text/css">
    div{
        background-color: green;
        width: 300px;
        height: 180px;
        box-shadow: 20px 20px 5px #6c6c6c;
    }
</style>
<div>让盒显示的时候产生阴影效果</div>

```

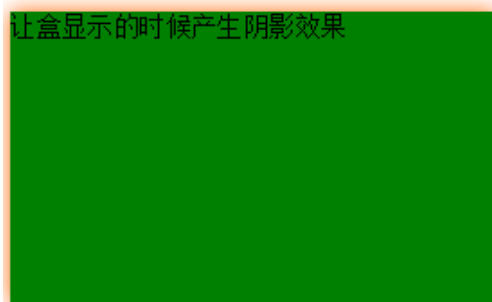
box-shadow属性



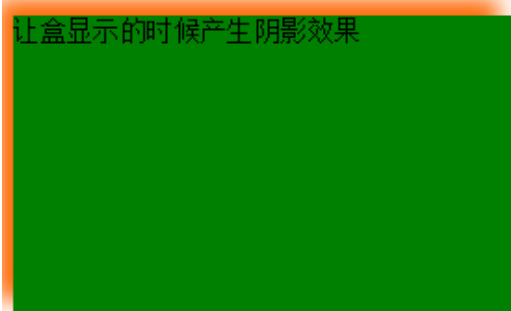
注意事项:

1) 将阴影离开盒子的横向位移和纵向位移都设置为 0 时，会在盒子的周围绘制阴影，例如设置 `box-shadow: 0px 0px 20px #f60` 的效果如左下图所示：

box-shadow属性



box-shadow属性



2) 横向位移设置为负数时，向左绘制阴影；纵向位移设置为负数时，向上绘制阴影，例如设置 `box-shadow: -10px -10px 20px #f60` 的效果如右上图所示：

3) 可以单独对盒内的子元素设置阴影，例如一个 `div` 元素内部有一个 `span` 元素，使用 `box-shadow` 属性让子元素 `span` 具有阴影效果：

```

<style type="text/css">
    span{
        background-color: green;
    }

```




```

        box-shadow: 10px -10px 10px #6c6c6c;
    }
</style>
<div><span>box-shadow 属性</span>让盒显示的时候产生阴影效果</div>

```

box-shadow属性

效果图为： box-shadow属性让盒显示的时候产生阴影效果

4) 结合 first-line 或 first-letter 伪元素选择器可以只让第一行或第一个字具有阴影，例如：


```

<style type="text/css">
    div:first-line{
        background-color: green;
        box-shadow: 10px -10px 10px #6c6c6c;
    }
</style>

```

<div>box-shadow 属性让盒显示的时候产生阴影效果
box-shadow 属性让盒显示的时候产生阴影效果</div>

box-shadow属性

效果图为： box-shadow属性让盒显示的时候产生阴影效果

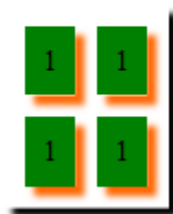
5) 对表格及单元格使用阴影，具体 demo 及效果图如下：

```

<style type="text/css">
    table{
        border-spacing: 12px;/*设置边框与单元格之间的空白*/
        box-shadow: 5px 5px 5px #000;/*设置整个表格的阴影*/
    }
    td{
        background-color: green;
        box-shadow: 5px 5px 5px #f60;/*设置每个单元格的阴影*/
        padding:10px;/*设置表格内边距（内补白）*/
    }
</style>
<table>
    <tr>
        <td>1</td>
        <td>1</td>
    </tr>
    <tr>
        <td>1</td>
        <td>1</td>
    </tr>
</table>

```

对表格及单元格使用阴影



(3) box-sizing 属性 (宽高计算): 指定元素宽度和高度的计算方法, 因为样式中指定的 width 和 height 属性表示的是可能包含了 padding 内边距 (内补白) 及 border、也可能没有包含 padding 及 border 的宽度和高度, 因此使用 box-sizing 属性可以很好地控制元素的总宽度和总高度。可选属性值有两个: **content-box** (表示元素**不包含 padding 及 border** 的宽度和高度)、**border-box** (表示元素**包含 padding 及 border** 的宽度和高度), **在不指定 box-sizing 属性值时, 默认使用 content-box 属性值**, 具体 demo 效果图如下:

```
<style type="text/css">
```

```
div{
    width:300px;
    padding: 10px;
    border: solid 10px #6c6c6c;
    background-color: green;
    margin:10px;
}
div#a{
    box-sizing: content-box;
}
div#b{
    box-sizing: border-box;
}
```

```
</style>
```

```
<div id='a'>指定元素宽度和高度的计算方法</div>
```

```
<div id='b'>指定元素宽度和高度的计算方法</div>
```

box-sizing属性



7、CSS3 新增的背景样式

CSS3 新增的背景属性有四个, 分别是 background-clip、background-origin、background-size、background-break, 下面进行详细讲解

(1) background-clip 属性: 指定背景的显示区域 (超出部分将被裁减掉), 可选属性值有 **border-box** (**默认值**, 背景从 border 区域向外裁剪, 即超出 border 区域的部分将被裁减掉)、**padding-box** (背景从 padding 区域向外裁剪)、**content-box** (背景从 content 区域向外裁剪), 这三种属性值的对比 demo 及效果图如下:

```
<style type="text/css">
```

```
div{
    background-color: #f60;
    width:150px;
    height: 50px;
    border:dashed 10px green;/*边框设置为虚线*/
    padding:10px;
}
#a{
    /*背景颜色从 border 区域向外裁剪*/
    background-clip: border-box;
}
#b{
```

background-clip背景属性



```

        /*背景颜色从 padding 区域向外裁剪*/
        background-clip: padding-box;
    }
    #c{
        /*背景颜色从 content 区域向外裁剪*/
        background-clip: content-box;
    }
</style>
<div id="a">background-clip 属性</div><br/>
<div id="b">background-clip 属性</div><br/>
<div id="c">background-clip 属性</div>

```

(2) background-origin 属性: 指定绘制背景图像的起始位置, 可选属性值与 background-clip 一样: **border-box** (指定 background-position 起始位置从 border 的外边缘开始显示背景图片)、**padding-box** (**默认值**, 指定 background-position 起始位置从 padding 的外边缘开始显示背景图片)、**content-box** (指定 background-position 起始位置从 content 的外边缘开始显示背景图片), 这三种属性值的对比 demo 及效果图如下:

```
<style type="text/css">
```

```

    div{
        width:100px;
        height: 50px;
        background-image: url('HTML5.png');
        background-repeat: no-repeat;
        border:dashed 20px green;
        padding: 20px;
    }
    #a{
        /*背景图片起始位置从 border 的外边缘开始*/
        background-origin: border-box;
    }
    #b{
        /*背景图片起始位置从 padding 的外边缘开始*/
        background-origin: padding-box;
    }
    #c{
        /*背景图片起始位置从 content 的外边缘开始*/
        background-origin: content-box;
    }

```

```

</style>
<div id="a"></div><br/>
<div id="b"></div><br/>
<div id="c"></div>

```

background-origin背景属性



(3) background-size 属性: 指定背景图像的尺寸, 可选属性值有: **auto** (**默认值**, 保持背景图片的原始宽度和高度)、**像素值** (设置背景图片的宽度和高度, 如果不指定, 都为默认值)、**百分比** (以容器元素的百分比设置背景图片的宽度和高度)、**cover** (图片过小时, 将

图片放大以铺满整个容器，但会使图片失真）、**contain**（图片过大时，将图片缩小以铺满整个容器，也会使图片失真），例如：

```
div{
    width:100px;/*content-box 的宽度和高度*/
    height: 50px;
    background-image: url('HTML5.png');
    background-repeat: no-repeat;/*注意 background-repeat 默认值为 repeat, 即背景图片重复铺满整个容器*/
    border:dashed 20px green;
    padding: 20px;
    background-origin: content-box;/*设置背景图片的绘制起点*/
    background-size: 100px 50px;/*背景图片大小正好和 div 元素的大小一致*/
}
```

background-size背景属性



效果图为：

（4）**background-break** 属性：指定内联元素的背景图像进行平铺时的循环方式，能够控制背景在不同区域显示，可选属性值有 **continuous**（默认值，忽视各个区域之间的间隔，当做一个大区域）、**bounding-box**（重新考虑区域之间的间隔）、**each-box**（对每一个独立的区域进行重新划分）

（5）在一个元素中重叠显示多个背景图片

利用 `background-image:url(1.png),url(2.png),url(3.png)`，图层叠放顺序是从上往下指定的，即指定的第一个图片放在图层的最上面，最后一个图片放在图层的最下面，具体 demo 如下：

`<style type="text/css">`

```
body{
    margin :0;
    padding:0;
    width:1000px;/*content-box 的宽度和高度*/
    height: 800px;
    /*指定图层叠放的顺序，从上往下叠放，即 bj4.png 放在最上面，bj1.jpg 放在最下面*/
    background-image: url('bj4.png'), url('bj3.png'), url('bj2.png'), url('bj1.jpg');
    background-repeat: no-repeat;
    background-position: center;/*指定背景图片的位置*/
}
</style>
```



效果图为：

8、CSS3 新增的边框样式

（1）圆角边框：利用 **border-radius** 属性绘制，属性值为圆角半径（数值越大弧度越大），具体 demo 及效果图如下：

<style type="text/css">

```
div{
    width:200px;
    height:100px;
    background-color: #f60;
    border:solid 20px green;
    border-radius: 30px;
}
```

</style>

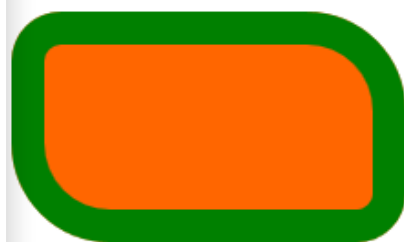
border-radius属性



知识点扩展：

1) **border-radius** 属性可以指定两个半径，第一个半径作为边框左上角与右下角的圆角半径，第二个半径作为边框左下角与右上角的圆角半径，比如 **border-radius: 30px 60px** 的效果如左下图所示：

border-radius属性

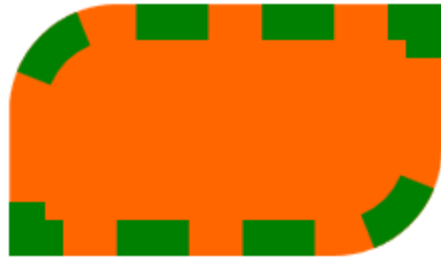


2) 可以绘制 4 个不同半径的圆角边框，利用 **border-top-left-radius** 属性（指定左上角半径）、**border-top-right-radius** 属性（指定右上角半径）、**border-bottom-left-radius** 属性（指定左下角半径）、**border-bottom-right-radius** 属性（指定右下角半径），具体 demo 及效果图如下：

```
<style type="text/css">
```

```
div{
    width:200px;
    height:100px;
    background-color: #f60;
    border:dashed 20px green;
    border-top-left-radius: 60px;
    border-bottom-right-radius: 60px;
}
```

border-radius属性



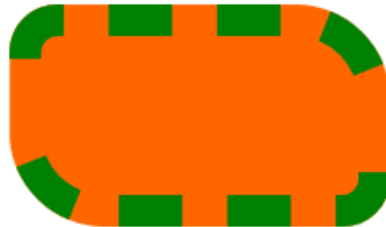
```
</style>
```

3) 如果使用了 border-radius 属性, 但没有设置 border 边框样式时, 浏览器会自动把元素背景区域的四个角绘制成圆角, 如左下图所示:

border-radius属性



border-radius属性



4) 不管将边框设置成什么样式, 只要使用了 border-radius 属性, 边框都会变成圆角的, 例如设置边框为虚线 border:dashed 20px green 的效果如右上图所示:

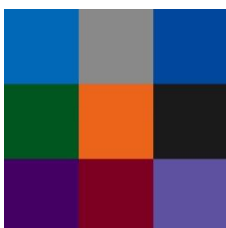
(2) 图像边框: 利用 border-image 属性, 给元素指定一张图像, 先对图像进行分割然后再填充到元素边框中, 常规语法是 border-image:url("边框图像的路径") 上边距 右边距 下边距 左边距 (注意这五个参数都是必需的, 边距值不带任何单位)。如果四个边距的值完全一样, 语法可以缩写为 border-image:url("边框图像的路径") 边距; 如果要考虑浏览器兼容性, 最好在 border-image 前面加上前缀, 比如 -webkit- (代表 Chrome 和 Safari 的引擎)、-moz- (代表 Firefox 的引擎)、-o- (代表 Opera 的引擎), 例如:

```
<style type="text/css">
```

```
div{
    width:300px;
    height:300px;
    border-width:100px;
    /*给边框指定一张图片 1.jpg, 将图片先按照上右下左边距各 100 像素的方式
    分割成 9 部分, 然后将其中四部分进行伸缩后填充到对应的 border 边框中*/
    -webkit-border-image:url("1.jpg") 100 ;
}
```

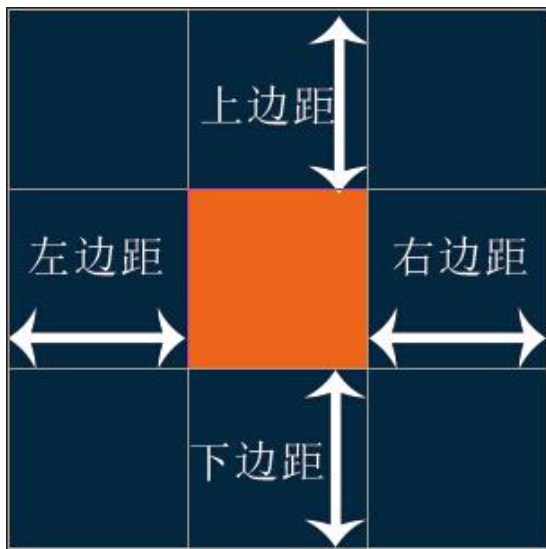
```
</style>
```

左下图是 1.jpg 的原图, 右下图是使用图像边框 border-image 拉伸后的效果图:

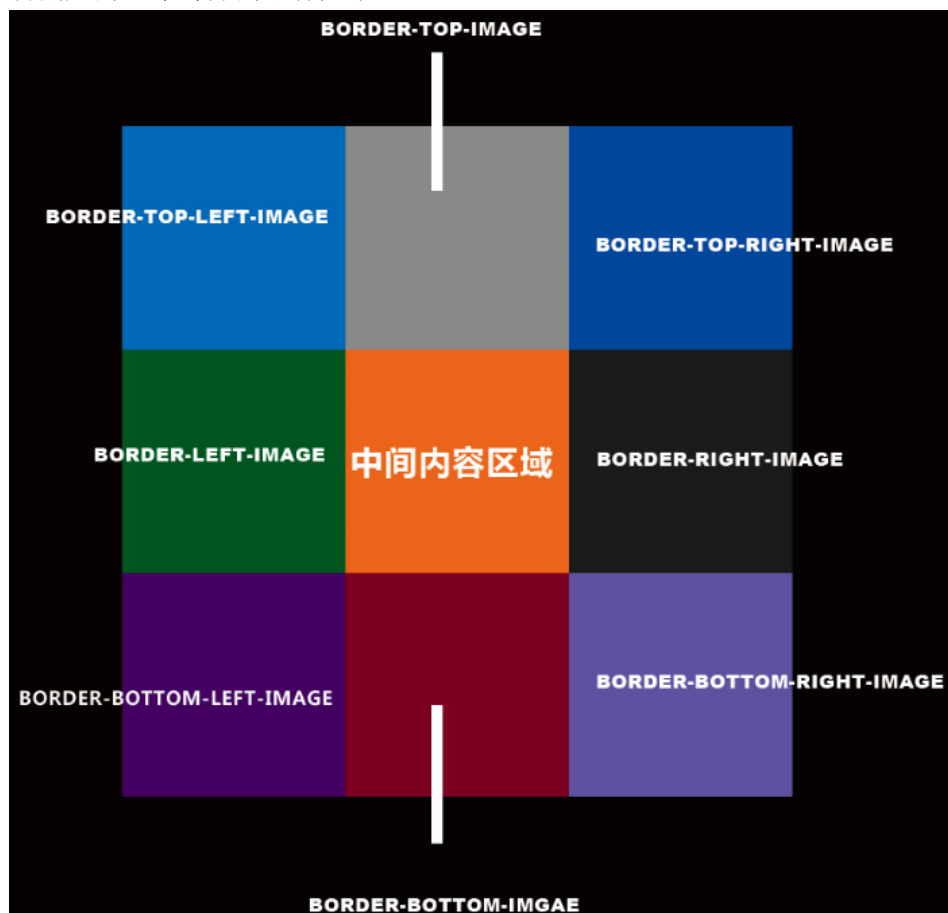


知识点扩展：

1) border-image 属性指定了图像的上、右、下、左边距以后，浏览器就会按照指定的边距对原背景图像进行切割，切割方式如下图所示：



切割后的 9 个部分的名称如下：

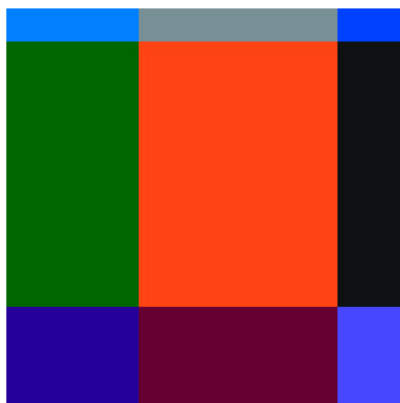


在浏览器中显示的时候，其中 4 个角 **border-top-left-image**、**border-top-right-image**、**border-bottom-left-image**、**border-bottom-right-image** 的图像不会进行任何的拉伸，而 **border-top-image**、**border-right-image**、**border-bottom-image**、**border-left-image** 的图像会分别作为 border 上边框、右边框、下边框、左边框的背景图像进行拉伸后填充显示

2) 利用 **border-image** 属性除了可以指定图像、边距以外，还可以指定**边框宽度**，语法是 **border-image:url("边框图像的路径") 上边距 右边距 下边距 左边距/border 上宽度 border 右宽度 border 下宽度 border 左宽度**。如果四个边距的值完全一样，边框宽度语法可以缩写为 **border-image:url("边框图像的路径") 边距/border 宽度**，例如：

```
<style type="text/css">
    div{
        width:300px;
        height:300px;
        /*将图片按照上右下左边距各 100 像素的方式分割成 9 部分，然后将其中四部分进行伸缩后填充到对应的 border 边框中*/
        -webkit-border-image:url("1.jpg") 100/25px 50px 75px 100px;
    }
</style>
```

border-image属性



效果图为：

3) 利用 **border-image** 属性指定填充到元素四条边框中的图像是以**拉伸**的方式显示还是以**平铺**的方式显示，语法是 **border-image:url("边框图像的路径") 上边距 右边距 下边距 左边距 /border 宽度 上下 border 中图像的显示方式 左右 border 中图像的显示方式**，其中显示方式可以指定的值有 **repeat**（平铺方式，每个分割单元的图像大小始终不变，可能图像不会完整显示）、**stretch**（默认值，拉伸方式）、**round**（先平铺显示，如果分割单元的图像不能完整显示，就进行拉伸显示），例如；

```
<style type="text/css">
    div{
        width:300px;
        height:300px;
    }
    #a{
        /*将图片按照上右下左边距各 100 像素的方式分割成 9 部分，然后填充到 50 像素的边框中，上下边框、左右边框中的图片都以拉伸的方式填充*/
        -webkit-border-image:url("3.jpg") 100/50px stretch stretch;
    }
    #b{
        /*上下边框、左右边框中的图片都以 repeat 的方式显示*/
        -webkit-border-image:url("3.jpg") 100/50px repeat repeat;
    }
</style>
```



```

    }
    #c{
        /*上下边框、左右边框中的图片都以 round 的方式显示*/
        -webkit-border-image:url("3.jpg") 100/50px round round;
    }
</style>

```

三种显示方式的对比效果图如下：

