HTML/CSS/JavsScript

极夜酱

目录

Part I

CSS

Chapter 1 CSS 简介

1.1 给 HTML 打扮——CSS 样式

1.1.1 层叠样式表（CSS, Cascading Style Sheets）

CSS 主要用于定义 HTML 内容在浏览器内的显示样式，如文字大小、颜色、字体加粗等。使用 CSS 样式的一个好处是通过定义某个样式，可以让不同网页位置的问题有着统一的字体、字号或颜色等。CSS 样式由选择器和声明组成，而声明又由属性和值组成。

选择器也称选择符，用于指明网页中要应用样式规则的元素。声明的内容写在大括号【{}】内，属性和值之前用冒号【:】分隔。当有多条声明时，中间可以用分号【;】分隔。最后一条声明可以没有分号，但是为了后续修改方便，一般也加上分号。为了使样式更加容易阅读，可以将每条声明单独成行。

修改字体大小和颜色

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>修改字体大小和颜色</title>
    <style type="text/css">
        p {
            font-size: 20px;
            color: red;
        }
    </style>
</head>
<body>
    <p>修改字体大小和颜色</p>
</body>
</html>
```

1.2 既然那么好，那就引入 CSS 吧——内联式 CSS

1.2.1 内联式 CSS

内联式 CSS 样式，也称行间样式，就是把 CSS 代码直接写在现有的 HTML 标签中。注意 CSS 样式必须写在元素的开始标签里，不能在结束标签里。

```
<开始标签 style="属性: 值;">文本</结束标签>
```

CSS 样式必须写在 style 属性的双引号中，如果有多条 CSS 样式代码可以设置可以写在一起，中间用【;】隔开。

内联式 CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>内联式CSS</title>
</head>
<body>
    <p style="color: red; font-size: 20px;">内联式CSS</p>
</body>
</html>
```

1.3 换个地方吧，行内太挤了——嵌入式 CSS

1.3.1 嵌入式 CSS

嵌入式 CSS 样式，也称页面级 CSS 样式，就是把 CSS 样式代码写在<style>之间，嵌入式 CSS 样式一般放在<head>内。

嵌入式 CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>嵌入式CSS</title>
    <style type="text/css">
        span {
            color: red;
        }
    </style>
</head>
<body>
    <p><span>乔恩</span>找到<span>艾洛莉</span></p>
</body>
</html>
```

1.4 还是把 HTML 和 CSS 分开吧——外部式 CSS

1.4.1 外部式 CSS

外部式 CSS 样式，也称外联式 CSS 样式，就是把 CSS 样式代码写一个单独的外部文件中，这个 CSS 样式文件以.css 为扩展名。在<head>内使用<link>将 CSS 外部样式文件链接到 HTML 文件内。

```
<link href="CSS样式文件名" rel="stylesheet" type="text/css" />
```

CSS 样式文件名以有意义的英文命名，<link>中 rel="stylesheet" type="text/css"是固定写法，不需要修改。

external_css.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>外部式CSS</title>
    <link type="text/css" rel="stylesheet"
        href="external_css.css">
</head>
<body>
    <div></div>
</body>
</html>
```

external_css.css

```
div {
    width: 100px;
    height: 100px;
    background-color: blue;
}
```

1.5 总有个先来后到吧——三种链接方式的优先级

1.5.1 CSS 引入方式优先级

如果有一种情况：对于同一个元素同时使用了三种方法设置 CSS 样式，那么哪种方式真正有效呢?

三种 CSS 引入方式是有优先级的：内联式 > 嵌入式 > 外部式。但是嵌入式 > 外部式有一个前提，那就是嵌入式 CSS 样式的位置一定在外部式的后面。在实际开发中也会将<link>写在<style>的前面。

总的来说，优先级遵循“就近原则”，离被设置元素越近优先级别越高。

但是以上总结的优先级有一个前提，那就是内联式、嵌入式、外部式样式表中 CSS 样式是在相同权值的情况下。那权值是什么呢?

Chapter 2 选择器

2.1 选一个标签——标签选择器

2.1.1 标签选择器

标签选择器其实就是 HTML 代码中的标签。

```
tag_selector {
    attribute: value;
}
```

标签选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>标签选择器</title>
    <style type="text/css">
        h2 {
            color: green;
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h2>CSS特点</h2>
    <p>CSS为HTML标记语言提供了一种样式描述。</p>
</body>
</html>
```

2.2 再选一个类——类选择器

2.2.1 类选择器

类选择器在 CSS 样式中是最常用的。类选择器使用【.】开头，后加类选择器的名称，CSS 样式代码会被作用到属于该类的 HTML 标签中。在标签中使用 class 属性为标签设置一个类。

类选择器与标签是多对多的关系，即类选择器名称可以多个标签共用，一个元素可以用多个 class，一个 class 值可以对应多个元素。多个 class 值之间使用空格分隔。

类选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>类选择器</title>
    <style type="text/css">
        .title {
            color: green;
        }
    </style>
</head>
<body>
    <h3 class="title">丰富的样式定义</h3>
    <p>CSS提供了丰富的文档样式外观。</p>
    <h3 class="title">易于使用和修改</h3>
    <p>CSS样式表可以将所有的样式声明统一存放，进行统一管理。</p>
</body>
</html>
```

2.3 取个唯一表示——ID 选择器

2.3.1 ID 选择器

使用 ID 选择器，必须给标签添加上 id 属性，即为标签设置 id 属性。ID 选择器名称的前面使用【#】。

ID 选择器与标签是一对一的关系，即一个元素只能有一个 id 值，一个 id 值只能对应一个元素。id 是全局唯一的，就像身份证号码一样。

ID 选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>ID选择器</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
        }

        #square1 {
            background-color: red;
        }
        #square2 {
            background-color: blue;
        }
    </style>
</head>
<body>
    <div id="square1"></div>
    <div id="square2"></div>
</body>
</html>
```

2.4 捡了个儿子——子选择器

2.4.1 子选择器

子选择器【>】，用于选择指定标签元素的第一代子元素。

子选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>子选择器</title>
    <style type="text/css">
        .food > li {
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h2>食物</h2>
    <ul class="food">
        <li>水果
            <ul>
                <li>苹果</li>
                <li>香蕉</li>
            </ul>
        </li>
        <li>蔬菜
            <ul>
                <li>白菜</li>
                <li>油菜</li>
            </ul>
        </li>
    </ul>
</body>
</html>
```

2.5 这么快就当爷爷了——后代选择器

2.5.1 后代选择器

后代选择器，也称包含选择器，用于选择指定标签元素的后辈元素。

```
ancestor_selector descendant_selector {
    attribute: value;
}
```

后代选择器与子选择器的区别在于，子选择器仅是指它的直接后代，而后代选择器是作用于所有子后代元素。

后代选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>后代选择器</title>
    <style type="text/css">
        .food li {
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h2>食物</h2>
    <ul class="food">
        <li>
            水果
            <ul>
                <li>苹果</li>
                <li>香蕉</li>
                <li>橘子</li>
            </ul>
        </li>
        <li>
```

```
24              蔬菜
25              <ul>
26                  <li>白菜</li>
27                  <li>油菜</li>
28                  <li>卷心菜</li>
29              </ul>
30          </li>
31      </ul>
32  </body>
33  </html>
```

2.6 我全都要——通配符选择器

2.6.1 通配符选择器

通配符选择器【*】，也称通用选择器，是功能最强大的选择器，用于匹配 HTML 中所有的标签元素，包括<html>、<body>等。

通配符选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>通配符选择器</title>
    <style type="text/css">
        * {
            background-color: yellow;
        }
    </style>
</head>
<body>

</body>
</html>
```

2.7 给选择器分个组——分组选择器

2.7.1 分组选择器

分组选择器【,】，用于为 HTML 中多个标签元素设置同一个样式。

分组选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>分组选择器</title>
    <style type="text/css">
        h1, h2, h3 {
            color: red;
        }
    </style>
</head>
<body>
    <h1>HTML</h1>
    <h2>CSS</h2>
    <h3>JavaScript</h3>
</body>
</html>
```

2.8 伪装者——伪类选择器

2.8.1 伪类选择器

伪类选择器【:】允许给 HTML 标签的某种状态设置样式，例如给一个标签元素的鼠标滑过的状态设置字体颜色。

伪类选择器	功能
:link	未访问
:visited	已访问
:hover	鼠标悬停
:active	鼠标按下
:enabled	可用的时候触发
:disabled	不可用的时候触发

表 2.1: 常用伪类选择器

到目前为止，可以兼容所有浏览器的伪类选择器就是在<a>上使用:hover。

伪类选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>伪类选择器</title>
    <style type="text/css">
        a:hover {
            color: orange;
        }
    </style>
</head>
<body>
    <a href="http://www.baidu.com">百度一下，你就知道</a>
</body>
</html>
```

2.9 为所欲为——选择器最高层级!important

2.9.1 选择器优先级

每个选择器都是有优先级的，如果一个元素使用了多个选择器，则会按照选择器的优先级来给定样式。

选择器的优先级依次是：内联样式 > ID 选择器 > 类选择器 > 标签选择器 > 通配符选择器。

2.9.2 选择器权重

浏览器是根据权值来判断使用哪种 CSS 样式的，权值高的优先级更高。

选择器	权值
!important	∞
行间样式	1000
ID	100
class	10
标签	1
通配符	0

表 2.2: 选择器权重

2.9.3 !important

有些特殊情况需要为某些样式设置具有最高权值，这时候可以使用!important，注意!important要写在分号的前面。

!important

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>!important</title>
6     <style type="text/css">
```

```
        h1 {
            color: red !important;
        }

        h1 {
            color: blue;
        }
    </style>
</head>
<body>
    <h1>为所欲为</h1>
</body>
</html>
```

Chapter 3 字体、文本样式

3.1 字体样式

3.1.1 字体样式

使用 CSS 样式可以为网页中的文字设置字体。注意不要设置不常用的字体，因为如果用户本地电脑上没有安装该字体，就会显示浏览器默认的字体。

浏览器默认的字号为 16px，使用 font-size 可以修改字号大小。

为文字设置粗体是有单独的 CSS 样式来实现的，再也不用为了实现粗体样式而使用<h1>-<h6>或了。

font-weight 的默认值为 normal，通过设置属性值为 lighter、bold、bolder 或 100-900 之间的整百数值改变文字的粗细。注意，字体能否被 bolder 或 lighter 更改取决于字体包是否存在该样式。

font-style 可以设置字体样式，并且有 3 种设置方式：

1. 正常字体为 normal，也是 font-style 的默认值
2. italic 为字体设置为斜体，用于字体本身就有倾斜的样式
3. oblique 强制将字体倾斜

字体样式

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>字体样式</title>
6     <style type="text/css">
```

```
        p {
            font-family: "arial";
            font-size: 20px;
            font-weight: bold;
            font-style: italic;
        }
    </style>
</head>
<body>
    <p>Cascading Style Sheets (CSS) is a style sheet language
       used for describing the presentation of a document
       written in a markup language such as HTML.</p>
</body>
</html>
```

3.2 上个色——color

3.2.1 color

color 属性可以设置字体颜色。color 的值有 3 种设置方式：

1. 英文命令颜色。

```
color: red;
```

2. 十六进制颜色代码：使用 6 位十六进制数表示光学三原色“红绿蓝”。

R	H	B
00 - FF	00 - FF	00 - FF

表 3.1: 颜色代码

如果每两位十六进制数都相同，可简写。

颜色代码	颜色
#F00	红色
#0F0	绿色
#00F	蓝色
#000	黑色
#FFF	白色
#0FF	青色
#F40	淘宝红

表 3.2: 常见颜色代码

3. 颜色函数 rgb()：由光学三原色 RGB 的比例来配色。rgb() 函数中每一项的值可以是 0-255 之间的整数，也可以是 0%-100% 的百分数。

```
color: rgb(133, 45, 200);
color: rgb(20%, 33%, 25%);
```

3.3 文本样式

3.3.1 文本样式

text-decoration 可以设置添加到文本的修饰，默认值为 none。属性值为 underline 为文本添加下划线，属性值为 overline 为文本添加上划线，属性值为 line-through 为添加穿过文本的线，一般用于商品折扣价。

使用 line-height 可以设置段落中的行间距离（行高）。使用 text-align 可以为文本设置对齐方式，属性值包括 left、right 和 center。

文本样式

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文本样式</title>
    <style type="text/css">
        h2, p {
            text-decoration: underline;
            line-height: 2em;   /* 两倍行间距 */
            text-align: center;
        }
    </style>
</head>
<body>
    <h2>《望庐山瀑布》</h2>
    <p>唐·李白</p>
    <p>
        日照香炉生紫烟，<br/>
        遥看瀑布挂前川。<br/>
        飞流直下三千尺，<br/>
        疑是银河落九天。
    </p>
</body>
</html>
```

Chapter 4 盒子模型与布局模型

4.1 元素分类

4.1.1 我要独占一行——块级元素

在 HTML 中 、<p>、<h1>、<form>、<ul>、<li>等都是块级元素。每个块级元素都从新的一行开始，并且其后的元素也另起一行。块级元素的高度、宽度、行高遗迹顶和底边距都可以设置，宽度在不设定的情况下，是它本身父容器的 100%（和父元素的宽度一致）。

块级元素

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>块级元素</title>
</head>
<body>
    <div>这是一个div标签</div>
    <p>这是一个p标签</p>
    <h1>这是一个h1标签</h1>
</body>
</html>
```

通过设置display：block可以将元素显示为块级元素。

4.1.2 我要和你站一起——内联元素

在 HTML 中，、<a>、<label>、、等都是内联元素（行内元素）。内联元素和其它元素都在一行上，元素的高度、宽度及顶部和底部编剧不可设置，元素的宽度就是它包含的文字或图片的宽度，不可改变。

块级元素也可以设置display：inline 将元素设置为内联元素。

内联元素与块级元素转换

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>内联元素与块级元素转换</title>
    <style type="text/css">
        a {
            display: block;
        }

        div {
            display: inline;
        }
    </style>
</head>
<body>
    <a>我要变成块级元素</a>
    <a>我也要变成块级元素</a>

    <div>我要变成内联元素</div>
    <div>我也要变成内联元素</div>
</body>
</html>
```

4.1.3 我还要占个大位置——内联块状元素

内联块状元素就是同时具备内联元素和块级元素的特点。内联块状元素的特点是和其它元素都在一行上，但元素的高度、宽度、行高以及顶和底边距都可以设置。

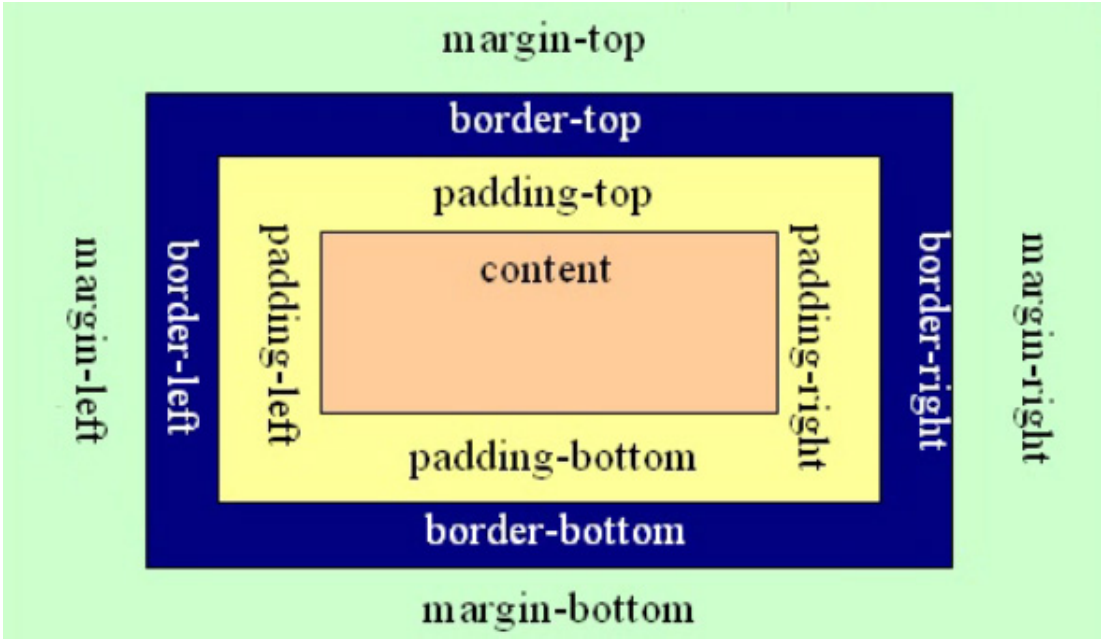
通过设置display：inline－block就可以将元素设置为内联块状元素。如<img>、<input>就是这种内联块状标签。

4.2 盒子模型

4.2.1 盒子模型

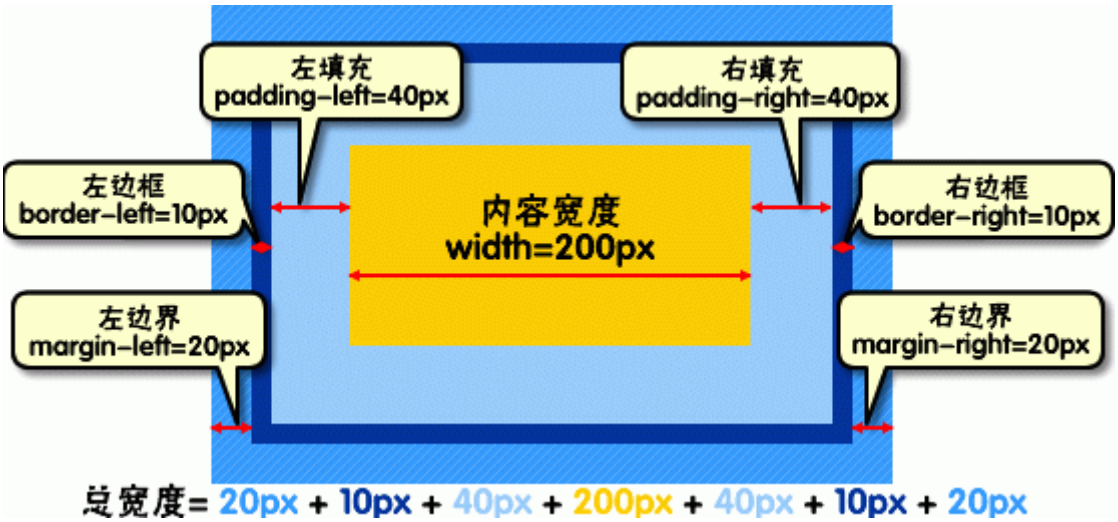
盒子模型包含 4 个部分：

1. 外边距（margin）
2. 外边框（border）
3. 内边距（padding）
4. 内容（content）



盒模型的宽度和高度和平常所说的物体的宽度和高度的理解是不一样的，CSS 内定义的宽和高，指的是盒模型中内容的宽和高。

元素实际的宽度（盒子的宽度）= 左外边距 + 左边框 + 左内边距 + 内容宽度 + 右内边距 + 右边框 + 右外边距



4.2.2 外边框（Border）

盒子模型的外边框可以设置粗细、样式和颜色：

1. border-width：设置边框的宽度，常用单位为像素（px）
2. border-style：边框样式包括实线 solid、虚线 dashed、点线 dotted
3. border-color：设置边框颜色

当三个属性都需要被设置时，也可以使用 border 属性简写。例如设置一个粗细为 1px 实心的黑色边框：

```
1 border: 1px solid black;
```

CSS 中也允许只为一个方向的边框设置属性，使用 border-top、border-bottom、border-left、border-right 可以单独设置上、下、左、右四条边框。

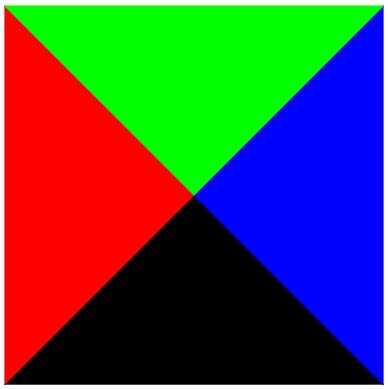
元素边框的圆角效果可以使用 border-radius 属性来设置。设置圆角的值的顺序为左上、右上、右下、左下。如果 border-radius 属性只给出一个值表示四个圆角都被设置成该值。如果只给出两个值表示左上角和右下角设置为第一个值，右上角和左下角设置为第二个值。当给一个正方形的圆角效果值设置为其宽度一半时，显示效果为圆形。

圆形

```
1 <!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>圆形</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
            background-color: red;
            border-radius: 50%;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过单独设置每条边的属性，可以在正方形内展现不同的颜色。

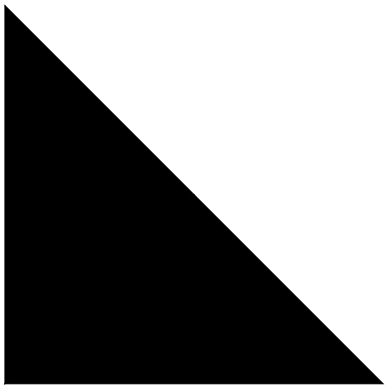


四色正方形

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>四色正方形</title>
    <style type="text/css">
        div {
            width: 0;
```

```
            height: 0;
            border: 100px solid black;
            border-left-color: red;
            border-top-color: green;
            border-right-color: blue;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过设置透明色（transparent），可以绘制出三角形。



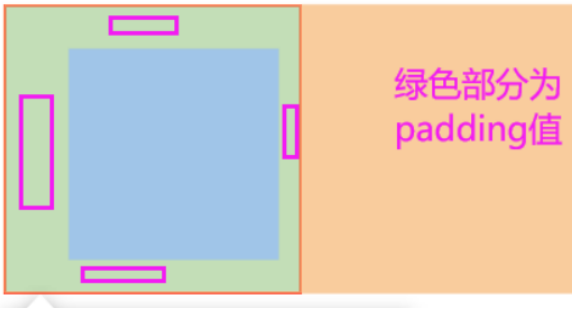
三角形

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>三角形</title>
    <style type="text/css">
        div {
            width: 0;
            height: 0;
            border: 100px solid black;
            border-left-color: black;
            border-top-color: transparent;
            border-right-color: transparent;
```

```
14          }
15      </style>
16  </head>
17  <body>
18      <div></div>
19  </body>
20  </html>
```

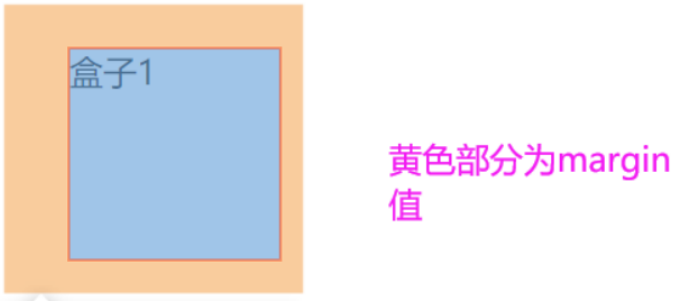
4.2.3 内边距（Padding）

元素内容与边框之间是可以设置距离的，称之为内边距或填充。内边距也可分为上、右、下、左（顺时针）。



4.2.4 外边距（Margin）

元素与其他元素之间的距离可以使用外边距来设置，外边距也可分为上、右、下、左。



margin 与 padding 的区别在于，padding 在外边框里，margin 在外边框外。

4.3 布局模型

4.3.1 布局模型

布局模型建立在盒子模型的基础上，在网页中，元素有 3 种布局模型：

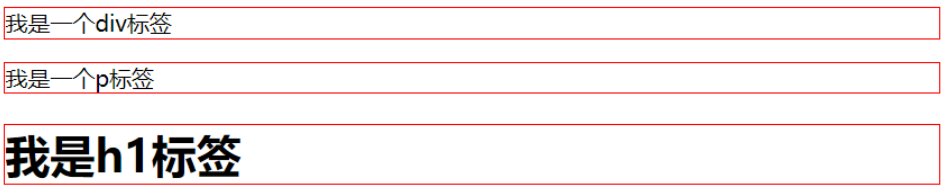
1. 流动模型（flow）
2. 浮动模型（float）
3. 层模型（layer）

4.3.2 流动模型

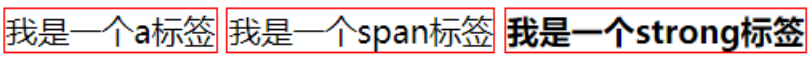
流动模型是默认的网页布局模式，也就是说网页在默认状态下的 HTML 网页元素都是根据流动模型来分布网页内容的。

流动布局模型有 2 个典型的特征：

1. 块级元素都会在所处的包含元素内自上而下按顺序垂直延伸分布，因为在默认状态下，块级元素的宽度都为 100%，即都会以行的形式占据位置。



2. 内联元素都会在所处的包含元素内从左到右水平分布显示。



4.3.3 浮动模型

块级元素这么霸道都是独占一行，如果想让两个块级元素并排显示，可以设置元素浮动。任何元素在默认情况下是不能浮动的，但可以设置 float 属性定义为浮动，如 、<p>、<table>、<img>等元素都可以被定义为浮动。

浮动模型

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>浮动模型</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
            border: 1px solid red;
        }

        #div1 {
            float: left;
        }

        #div2 {
            float: right;
        }
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>
</html>
```

4.4 定位

4.4.1 层模型

层布局模型就像是图像软件 PhotoShop 中非常流行的图层编辑功能一样，每个图形能够精确定位操作（positioning）。由于有些元素是定点展示的，定位技术可以让元素在特定的位置出现。

层模型有 3 种形式：

1. 绝对定位（absolute）
2. 相对定位（relative）
3. 固定定位（fixed）

4.4.2 万事无绝对——绝对定位

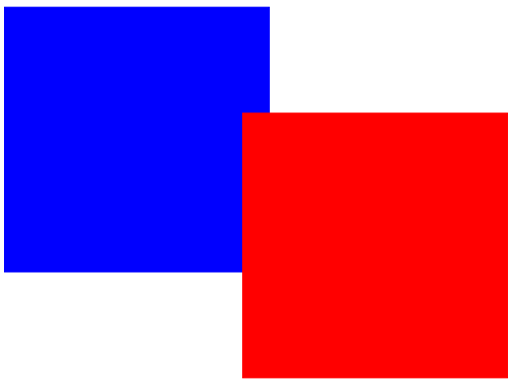
如果想为元素设置层模型中的绝对定位，需要设置position：absolute，这条语句的作用是将元素从文档流中拖出来，然后使用 left、right、top、bottom 属性相对于其最接近的一个具有定位属性的父包含块进行绝对定位。如果不存在这样的包含块，则相对于 body 元素，即相对于浏览器窗口。

当一个元素成为绝对定位元素，它就脱离了原来的层面，原来的位置会空出，下面的元素都会上来。

绝对定位

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>绝对定位</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
        }
```

```
11
12          #div1 {
13              background-color: red;
14              position: absolute;
15              top: 50px;      /* 距离窗口上边50px */
16              left: 100px;    /* 距离窗口左边100px */
17          }
18
19          #div2 {
20              background-color: blue;
21          }
22      </style>
23  </head>
24  <body>
25      <div id="div1"></div>
26      <div id="div2"></div>
27  </body>
28  </html>
```



4.4.3 相对于自己的位置——相对定位

如果想为元素设置层模型中的相对定位，需要设置position：relative，它通过left、right、top、bottom 属性确定元素在正常文档流中的偏移位置。相对定位完成的过程是首先按 static (float) 方式生成一个元素，然后相对于以前的位置移动，并且偏移前的位置保留不动，因此下面的元素无法上来。

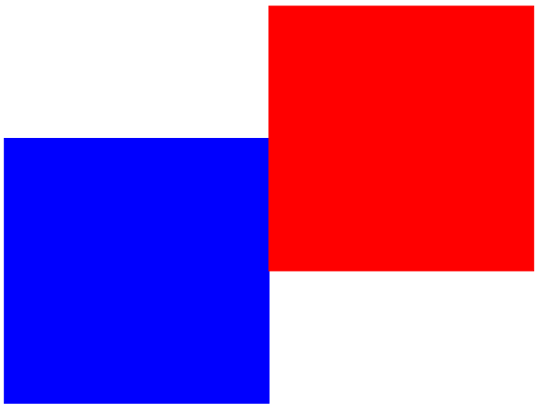
相对定位

```
1  <!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>相对定位</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
        }

        #div1 {
            background-color: red;
            position: relative;
            top: 50px;
            left: 100px;
        }

        #div2 {
            background-color: blue;
        }
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>
</html>
```



使用position：absolute可以实现元素相对于浏览器（body）设置定位，那可不可以相对于其它元素进行定位呢？当然可以，但是必须要满足以下的 3 点条件：

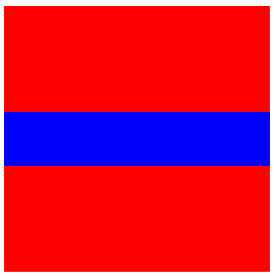
1. 参照定位的元素必须是相对定位元素的前辈元素。
2. 参照定位的元素必须加入position： relative 。
3. 定位元素加入 ‘position: absolute‘后便可以使用 top、bottom、left、right 来进行偏移了。

相对父元素定位

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>相对父元素定位</title>
    <style type="text/css">
        #div1 {
            width: 100px;
            height: 100px;
            background-color: red;
            position: relative;
        }

        #div2 {
            width: 100px;
            height: 20px;
            background-color: blue;
            position: absolute;
            top: 40px;
        }
    </style>
</head>
<body>
    <div id="div1">
        <div id="div2"></div>
    </div>
</body>
```

```
28 </html>
```



4.4.4 我就在那不动了——固定定位

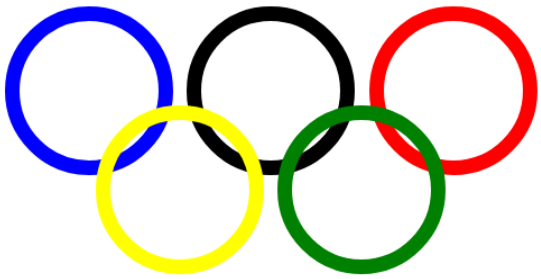
通过设置position：fixed让一个元素固定在一个位置，它的相对移动的坐标是视图（屏幕内的网页窗口）本身。由于视图本身是固定的，它不会随着浏览器窗口的滚动条滚动而变化，除非在屏幕中移动浏览器窗口的屏幕位置，或改变浏览器窗口的显示大小。因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流影响。

固定定位

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>固定定位</title>
6      <style type="text/css">
7          img {
8              position: fixed;
9              top: 15%;
10             left: 20%;
11         }
12     </style>
13 </head>
14 <body>
15     <img src="https://gimg2.baidu.com/image_search/src=http%3A%2F
           %2Fc-ssl.duitang.com%2Fuploads%2Fitem%2F201702%2F02%2
           F20170202151743_rvLWi.jpeg&refer=http%3A%2F%2
           Fc-ssl.duitang.com&app=2002&size=f9999,10000&q=a80&n=0&
           g=0n&fmt=jpeg?sec=1618588336&
```

```
        t=2612dc782456b9fdb8eeac0d0a215a85" alt="灰原哀.jpg"
        title="灰原哀">
16      <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br>
17  </body>
18  </html>
```

奥运五环



olympic.html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>奥运五环</title>
6       <link rel="stylesheet" type="text/css" href="olympic.css">
7   </head>
8   <body>
9       <!-- 展示奥运五环的区域 -->
10      <div class="stage">
11          <div id="circle1"></div>
12          <div id="circle2"></div>
13          <div id="circle3"></div>
14          <div id="circle4"></div>
15          <div id="circle5"></div>
16      </div>
17  </body>
18  </html>
```

olympic.css

```
* {
        margin: 0;
        padding: 0;
    }

    .stage {
        position: absolute;
        left: 50%;
        top: 50%;
        margin-left: -190px;
        margin-top: -93px;
        height: 185px;
        width: 380px;
    }

    #circle1,
    #circle2,
    #circle3,
    #circle4,
    #circle5 {
        position: absolute;
        width: 100px;
        height: 100px;
        border: 10px solid black;
        border-radius: 50%;
    }

    #circle1 {
        border-color: blue;
        left: 0;
        top: 0;
    }

    #circle2 {
        border-color: black;
        left: 130px;
```

```
        top: 0;
    }

    #circle3 {
        border-color: red;
        left: 260px;
        top: 0;
    }

    #circle4 {
        border-color: yellow;
        left: 65px;
        top: 70px;
    }

    #circle5 {
        border-color: green;
        left: 195px;
        top: 70px;
    }
```