```
HTML + CSS +JavaScript
```

结构 + 表现 + 交互

1、什么是CSS

如何学习

1、CSS是什么

2、CSS怎么用（快速入门）

3、**CSS选择器（重点+难点）**

4、美化网页（文字，阴影，超链接，列表，渐变.....）

5、盒子模型

6、浮动

7、定位

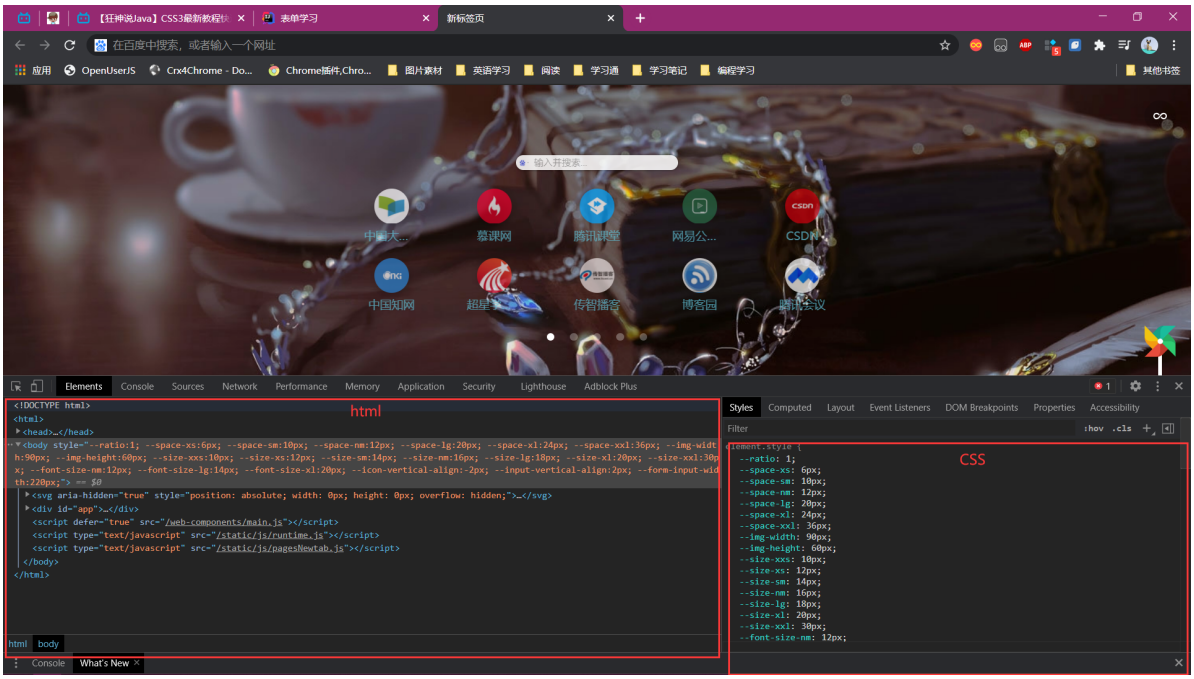
8、网页动画（特效效果）

1.1、什么是CSS

Cascading Style Sheets　层叠/级联样式表

CSS：表现（美化网页）

字体，颜色，边距，高度，宽度，背景图片，网页定位，网页浮动.....



1.2、发展史

CSS 1.0

CSS 2.0 DIV（块）+ CSS，HTML与CSS结构分离的思想，网页变得简单，SEO

CSS 2.1 浮动，定位

CSS 3.0 圆角，阴影，动画..... 浏览器兼容性~

1.3、快速入门

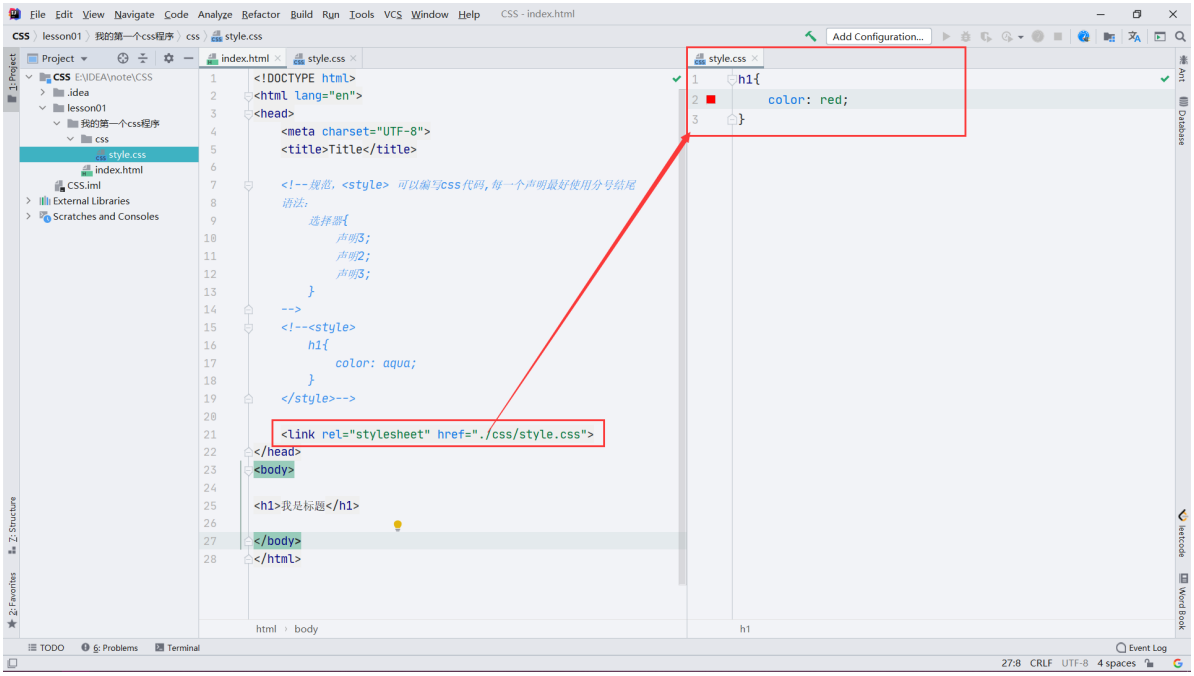
练习格式：

style

基本入门

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <!--规范，<style> 可以编写css代码,每一个声明最好使用分号结尾
    语法：
        选择器{
            声明3;
            声明2;
            声明3;
        }
    -->
    <style>
        h1{
            color: aqua;
        }
    </style>

</head>
<body>

<h1>我是标题</h1>

</body>
</html>
```

建议使用这种规范：



CSS优势：

1、内容与表现分离

2、网页结构表现统一，可以实现复用

3、样式十分丰富

4、建议使用独立于html的css文件

5、利用SEO（Search Engine Optimization），容易被搜索引擎收录！　vue不容器被收录

1.4、CSS的四种导入方式

- 行内样式：style属性

```
<!--行内样式：在标签元素中，编写一个style属性，编写样式即可-->
<h1 style="color: aqua;">我是标题</h1>
```

- 内部样式：style标签

```
<!--内部样式-->
    <style>
        h1{
            color: red;
        }
    </style>
```

- 外部样式：链接式-link标签

```
<!--外部样式：链接式-->
    <link rel="stylesheet" href="./css/style.css">
```

- 外部样式：导入式-@import url("")（弊端：网页元素多时，可能会先展现结构，后渲染）

@import 是 CSS 2.1 特有的！

```
<!--外部样式：导入式-->
    <style>
        @import url("./css/style.css");
    </style>
```

总览：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <!--内部样式-->
    <style>
        h1{
            color: red;
        }
    </style>

    <!--外部样式：链接式-->
    <link rel="stylesheet" href="./css/style.css">

    <!--外部样式：导入式-->
    <style>
        @import url("./css/style.css");
    </style>

</head>
<body>

<!--样式优先级：就近原则-->

<!--行内样式：在标签元素中，编写一个style属性，编写样式即可-->
<h1 style="color: aqua;">我是标题</h1>

</body>
</html>
```

样式加载优先级：就近原则

2、选择器

> 作用：选择页面上的某一个或某一类元素

2.1、基本选择器

1、标签选择器：选择一类标签　标签{}

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        /*标签选择器，会选择到页面上所有的此标签的元素*/
        h1{
            color: #ffffff;
            background: #33d2d2;
            border-radius: 5px;
        }
        p{
            font-size: 50px;
        }
    </style>

</head>
<body>

<h1>hello</h1>
<h1>world</h1>
<p>come</p>

</body>
</html>
```

2、类选择器 class：选中所有相同class属性的标签，可以跨标签　.class名{}

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        /*类选择器的格式　　.class的名称{}
        好处，可以多个标签归类，是同一个class，可以复用
         */
        .c1{
            color: beige;
        }
        .c2{
            color: darkseagreen;
        }
    </style>

</head>
<body>
```

```
<h1 class="c1">标题1</h1>
<h1 class="c1">标题2</h1>
<h1 class="c2">标题3</h1>

<p class="c2">p标签</p>

</body>
</html>
```

3、id选择器：全局唯一！　#id名{}

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        /*id选择器 ： id必须保证全局唯一
        #id名称{}
        优先级:
            不遵循就近原则，固定的
            id选择器 > class选择器 > 标签选择器
         */
        #bt1{
            color: red;
        }
        .c1{
            color: blue;
        }
        h1{
            color: green;
        }
    </style>

</head>
<body>

<h1 class="c1" id="bt1">标题1</h1>
<h1 class="c1">标题2</h1>
<h1 class="c1">标题3</h1>
<h1>标题4</h1>
<h1>标题5</h1>

</body>
</html>
```

优先级：id > class > 标签

2.2、层次选择器

1、后代选择器：在某个元素的后面

```
/*后代选择器*/
/*li 后的所有 p标签*/
li p{
    background: red;
}
```

2、子选择器，一代，儿子（所有儿子选择器）

```
/*子选择器*/
body>p{
    background: blue;
}
```

3、相邻兄弟选择器（最近的弟弟选择器）

```
/*相邻兄弟选择器：只有一个，相邻（向下）*/
.active+p{
    background: pink;
}
```

4、通用选择器（所有弟弟选择器）

```
/*通用兄弟选择器，当前选中元素向下的所有兄弟元素*/
.active~p{
    background: aqua;
}
```

选择器	描述	返回	示例
$("ancestor descendant")	选取ancestor元素里的所有descendant（后代）元素	集合元素	$("div span")选取里的所有的元素 span
$("parent > child")	选取parent元素下的child（子）元素，与（" $ancestor descendant$ "）有区别，("ancestor descendant")选择的是后代元素	集合元素	$("div > span")选取元素下元素名是的子元素
$("prev + next")	选取紧接在prev元素后的next元素	集合元素	$(".one + div")选取class为one的下一个同辈元素
$("prev ~ siblings")	选取prev元素之后的所有siblings元素	集合元素	$("#two ~ div")选取id为two的元素后面的所有同辈元素

2.3、结构伪类选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <!--避免使用  class，id选择器 -->
    <style>
        /*ul的第一个子元素*/
        ul>li:first-child{
            background: red;
        }
        /*ul的最后一个子元素*/
        ul>li:last-child{
            background: green;
        }

        /*选中 p1:定位到父元素，选择当前父元素的第 x 个子元素
        选中父级元素的第 x 个子元素，并且是当前元素（同类标签）才生效      （顺序）
        */
        p:nth-child(2){
            background: aqua;
        }

        /*选中父级元素的子元素中与p元素属性相同的第2个元素       （属性）*/
        p:nth-of-type(2){
            background: pink;
        }
    </style>
</head>
<body>

<h1>h1</h1>
<p>p1</p>
<p>p2</p>
<p>p3</p>
<ul>
    <li>li1</li>
    <li>li2</li>
    <li>li3</li>
</ul>
</body>
</html>
```

h1

p1

p2

p3

- li1
- li2
- li3

2.4、属性选择器（常用）

id + class 结合~

正则表达式里面的通配符

- = 绝对等于
- *= 包含这个元素
- ^= 以这个开头
- $= 以这个结尾
- ……

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        .demo>a{
            float: left;
            display: block;
            width: 50px;
            height: 50px;
            border-radius: 25px;
            margin-right: 10px;
            background: blue;
            text-align: center;
            line-height: 50px;
            color: white;
            text-decoration: none;
        }
        /*属性名，属性名=属性值（正则）
        =   绝对等于
        *=  包含这个元素
        ^=  以这个开头
        $=  以这个结尾
        */
        /*存在id属性的元素*/
        a[id]{
            background: aqua;
        }

        /*id等于second的元素*/
        a[id=second]{
```

```
            background: green;
        }

        /*class属性中以item结尾的元素*/
        a[href$=pdf]{
            background: pink;
        }

        /*href属性中包含ma的元素*/
        a[href*=baidu]{
            background: yellow;
        }

        /*href属性中以image开头的元素*/
        a[href^=image]{
            background: red;
        }
    </style>

</head>
<body>

<p class="demo">
    <a href="https://www.baidu.com" class="links item first" id="first">1</a>
    <a href="" class="links item second" target="_blank" id="second">2</a>
    <a href="image/123.jpg" class="links item">3</a>
    <a href="image/123.png" class="links item">4</a>
    <a href="a.pdf" class="links item">5</a>
    <a href="b.pdf" class="links item">6</a>
    <a href="a.doc" class="links item">7</a>
    <a href="b.doc" class="links item last">8</a>
</p>

</body>
</html>
```

CSS，JS，Query，Vue

3、美化网页元素

3.1、为什么要美化网页

1、传递页面信息

2、美化网页，页面漂亮，才能吸引用户

3、凸显页面主题

4、提高用户体验

span标签：重点要突出的字，用span标签套起来

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        span{
            font-size: 50px;
        }
    </style>
</head>
<body>

欢迎学习<span>Java</span>

</body>
</html>
```

3.2、字体样式

- font-family: 字体
- font-size: 字体大小
- font-weight: 字体粗细
- color: 字体颜色
- font: oblique bold 30px/50px 楷体; */斜体 粗体 字体大小/行高 字体/*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <!--
    font-family:    字体
    font-size:      字体大小
    font-weight:    字体粗细
    color:          字体颜色
    -->
    <style>
        body{
            font-family: 楷体
        }
        h1{
            background: #33d2d2;
        }
        p{
            font-size: 20px;
            font-weight: bold;
            color: white;
            background: aquamarine;
        }
        p:nth-of-type(2){
            /*斜体 粗体 字体大小/行高 字体*/
            font: oblique bold 30px/50px 楷体;
        }
    </style>
```

```
</head>
<body>

<h1><em>故事介绍</em></h1>
<p>
    平静安详的元泱境界，每隔333年，总会有一个神秘而恐怖的异常生物重生，它就是魁拔！魁拔的每一次出现，都会给元泱境界带来巨大的灾难！即便是天界的神族，也在劫难逃。在天地两界各种力量的全力打击下，魁拔一次次被消灭，但又总是按333年的周期重新出现。魁拔纪元1664年，天神经过精确测算后，在魁拔苏醒前一刻对其进行毁灭性打击。但谁都没有想到，由于一个差错导致新一代魁拔成功地逃脱了致命一击。很快，天界魁拔司和地界神圣联盟均探测到了魁拔依然生还的迹象。因此，找到魁拔，彻底消灭魁拔，再一次成了各地热血勇士的终极目标。
</p>
<p>
    在偏远的兽国窝窝乡，蛮大人和蛮吉每天为取得象征成功和光荣的妖侠纹耀而刻苦修炼，却把他们生活的村庄搅得鸡犬不宁。村民们绞尽脑汁把他们赶走。一天，消灭魁拔的征兵令突然传到窝窝乡，村长趁机怂恿蛮大人和蛮吉从军参战。然而，在这个一切都凭纹耀说话的世界，仅凭蛮大人现有的一块冒牌纹耀，不要说参军，就连住店的资格都没有。受尽歧视的蛮吉和蛮大人决定，混上那艘即将启程去消灭魁拔的巨型战舰，直接挑战魁拔，用热血换取至高的荣誉。
</p>

</body>
</html>
```

3.3、文本样式

1、颜色 color rgb rgba

2、文本对齐方式 text-align = center;

3、首行缩进 text-indent: 2em;

4、行高 line-height: 单行文字上下居中：line-height=height

5、装饰 text-decoration:

6、文字图片水平对齐： vertical-align: middle

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <!--
    color:颜色：
        单词
        RGB
        RGBA    A:0~1   透明度
    text-align:对齐
    text-indent:段落首行缩进  em字符
    height：块的高度
    line-height：行高  行高 和 块的高度一致，就可以上下居中
    -->
    <style>
        h1{
```

```
            /*color: red;
            color: #ff0000;
            color: rgb(255,0,0);*/
            color: rgba(255,0,0,0.5);
            text-align: center;
        }
        p:nth-of-type(1){
            text-indent: 2em;
        }
        p:nth-of-type(2){
            background: aquamarine;
            height: 200px;
            line-height: 100px;
        }
        /*下划线*/
        .l1{
            text-decoration: underline;
        }
        /*中划线*/
        .l2{
            text-decoration: line-through;
        }/*上划线*/
        .l3{
            /*text-decoration-line: overline;
            text-decoration-color: red;*/
            text-decoration: overline red;
        }
        /*水平对齐~参照物--a，b*/
        img,span{
            vertical-align: middle;
        }
    </style>

</head>
<body>
<h1><em>故事介绍</em></h1>
<p>
    平静安详的元泱境界，每隔333年，总会有一个神秘而恐怖的异常生物重生，它就是魁拔！魁拔的每一次出现，都会给元泱境界带来巨大的灾难！即便是天界的神族，也在劫难逃。在天地两界各种力量的全力打击下，魁拔一次次被消灭，但又总是按333年的周期重新出现。魁拔纪元1664年，天神经过精确测算后，在魁拔苏醒前一刻对其进行毁灭性打击。但谁都没有想到，由于一个差错导致新一代魁拔成功地逃脱了致命一击。很快，天界魁拔司和地界神圣联盟均探测到了魁拔依然生还的迹象。因此，找到魁拔，彻底消灭魁拔，再一次成了各地热血勇士的终极目标。
</p>
<p>
    在偏远的兽国窝窝乡，蛮大人和蛮吉每天为取得象征成功和光荣的妖侠纹耀而刻苦修炼，却把他们生活的村庄搅得鸡犬不宁。村民们绞尽脑汁把他们赶走。一天，消灭魁拔的征兵令突然传到窝窝乡，村长趁机怂恿蛮大人和蛮吉从军参战。然而，在这个一切都凭纹耀说话的世界，仅凭蛮大人现有的一块冒牌纹耀，不要说参军，就连住店的资格都没有。受尽歧视的蛮吉和蛮大人决定，混上那艘即将启程去消灭魁拔的巨型战舰，直接挑战魁拔，用热血换取至高的荣誉。
</p>

<p class="l1">hahahah</p>
<p class="l2">hahahah</p>
<p class="l3">hahahah</p>

<p>
    <img src="./images/1.jpg" width="20%" alt="picture">
```

```
    <span>小女孩图片</span>
</p>
</body>
</html>
```

3.4、超链接伪类

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        img{
            width: 10%;
        }
        /*去掉装饰*/
        a{
            text-decoration: none;
        }

        /*指向未被访问页面的链接*/
        a:link{
            color: grey;
        }
        /*用于设置指向已被访问的页面的链接*/
        a:visited{
            color: red;
        }
        /*鼠标悬浮状态  hover 必须位于 :link 和 :visited 之后*/
        a:hover{
            color: orange;
        }
        /*鼠标按住未释放的状态*/
        a:active{
            color: green;
            font-size: 30px;
        }

        /*文本阴影：水平阴影位置  垂直阴影位置  模糊距离  阴影颜色  */
        span{
            text-shadow: 2px 2px 3px grey;
        }
    </style>

</head>
<body>

<p>
    <img src="./images/1.jpg" alt="Java编程思想">
</p>
<p>
    <a href="#">Java编程思想</a>
</p>
```

```
<p>
    <a href="#">作者：Bruce Eckel</a>
</p>
<p>
    <span>￥89.2</span>
</p>

</body>
</html>
```

:link	a:link	选择所有未被访问的链接。
:visited	a:visited	选择所有已被访问的链接。
:active	a:active	选择活动链接。
:hover	a:hover	选择鼠标指针位于其上的链接。

3.5、阴影

```
/*文本阴影：水平阴影位置  垂直阴影位置  模糊距离  阴影颜色  */
        span{
            text-shadow: 2px 2px 3px grey;
        }
```

3.6、 列表

```
/*ul>li*/
/*
list-style:
    disc:实心圆点（默认）
    none:不显示
    decimal:数字
    circle: 空心圆点
    square:实心正方形
*/
ul>li{
    list-style: none;
    /*text-indent: 1em;*/
}
```

3.7、背景

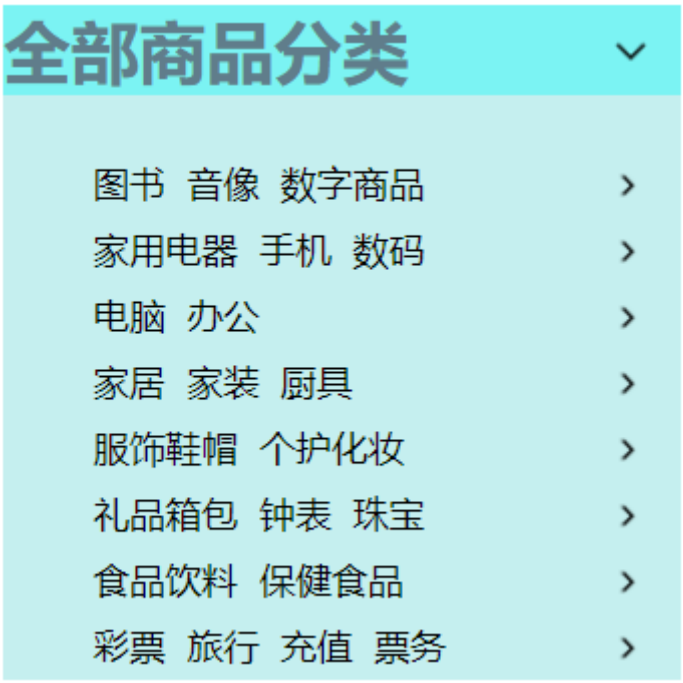
背景颜色

背景图片

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        div[class]{
            width: 1000px;
            height: 700px;
            /*border:边框宽度 边框样式 边框颜色
            border-width:
            border-style:     （dotted-点，solid-实线，double-双线，dashed-虚线）
            border-color:
            */
            border: 1px solid red;
            margin-bottom: 20px;
            background-image: url("./images/4.png");
        }
        .div1{
            background-repeat: repeat-x;
        }
        .div2{
            background-repeat: repeat-y;
        }
        .div3{
            background-repeat: no-repeat;
        }
    </style>

</head>
<body>

<div class="div1"></div>
<div class="div2"></div>
<div class="div3"></div>

</body>
</html>
```

练习：

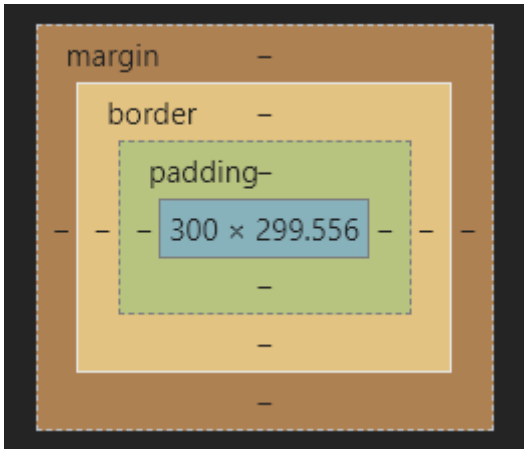


3.8、渐变

```
background-color: #FFDEE9;
background-image: linear-gradient(0deg, #FFDEE9 0%, #B5FFFC 100%);
```

4、盒子模型

4.1、什么是盒子模型



- margin：外边距
- border：边框
- padding：内边距

4.2、边框

1、边框粗细 border-width

2、边框样式 border-style

3、边框颜色 border-color

4、边框圆角 border-radius

border: border-width border-style border-color

4.3、内外边距

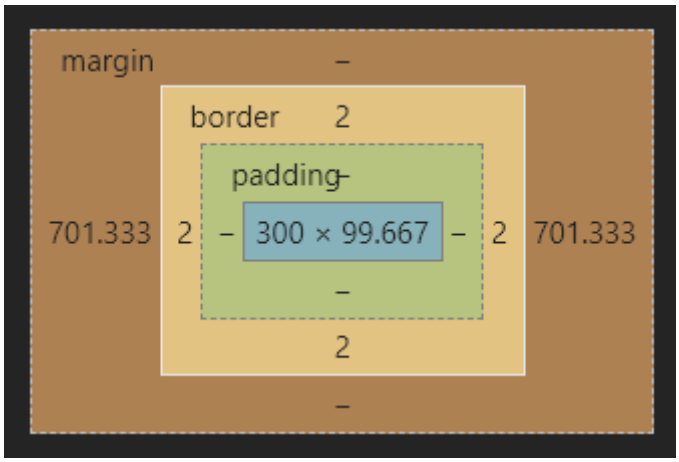
外边距：margin

```
/*所有外边距为0*/
margin: 0;
/*margin: 上下为0，左右自动*/
margin: 0 auto;
/*margin: 上 右 下 左*/
margin: 5px 10px 15px 20px;
```

内边距：padding

```
/*所有内边距为0*/
padding: 0;
/*padding: 上下为5px，左右为10px*/
padding: 5px 10px;
/*padding: 上 右 下 左*/
padding: 5px 10px 15px 20px;
```

盒子的计算方式：元素到底多大?

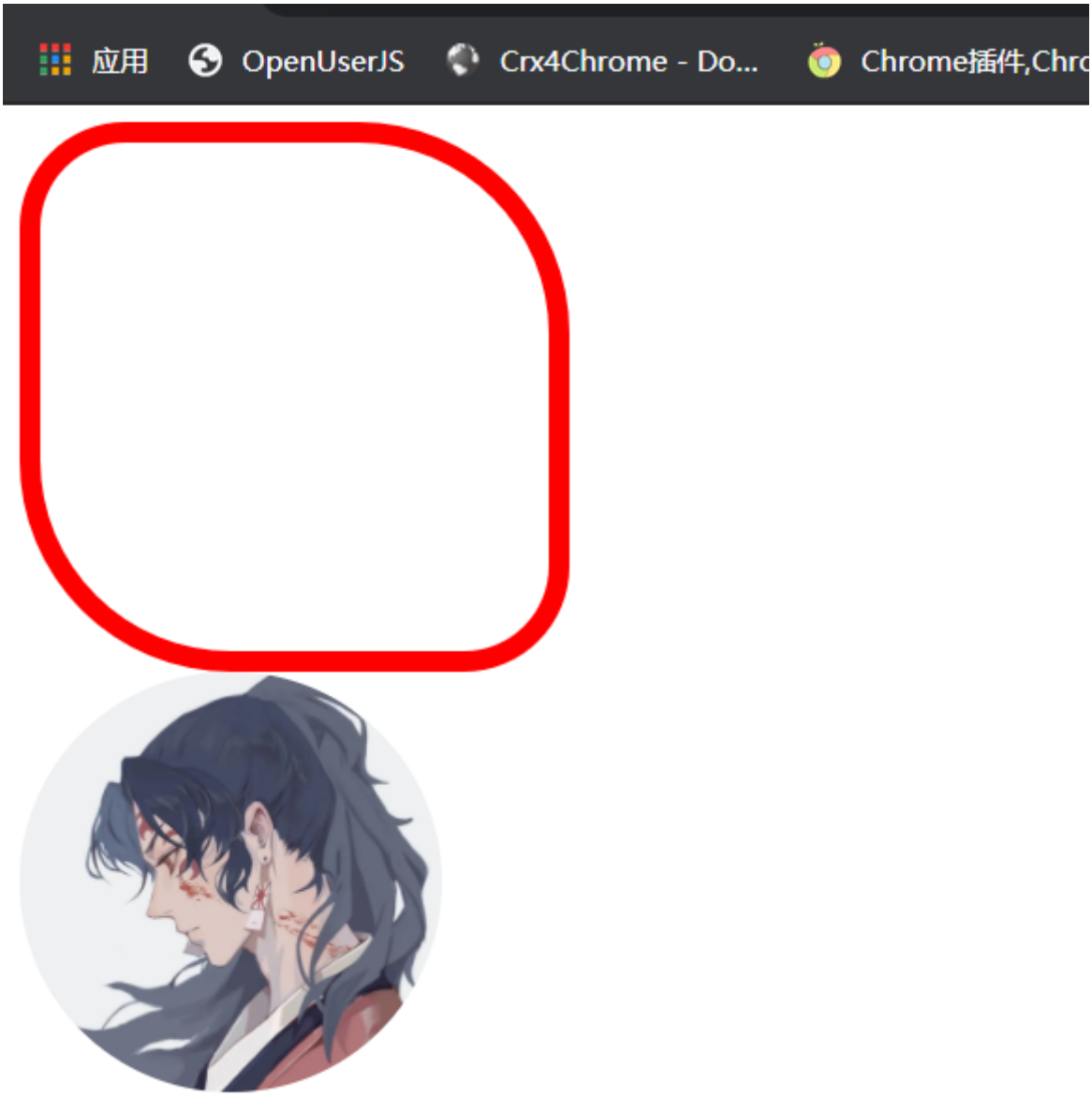


元素大小 = margin + border + padding + 内容大小

4.4、圆角边框

```
/*正方形->圆形：border-radius=border+padding+width*/
border-radius: 130px;
/*border-radius: 左上和右下  右上和左下*/
border-radius: 50px 100px;
/*border-radius: 左上 右上 右下 左下 （顺时针）*/
border-radius: 50px 100px 50px 100px;
```

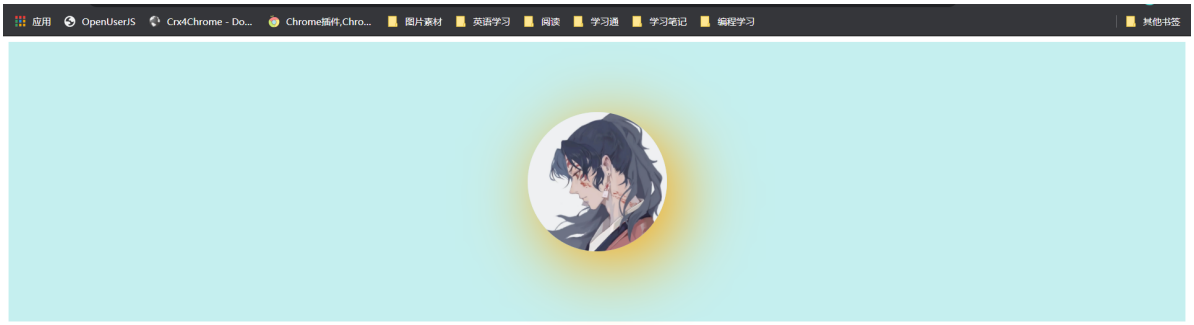
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        div:nth-of-type(1){
            border: 10px solid red;
            width: 200px;
            height: 200px;
            padding: 20px;
            /*正方形->圆形：border-radius=border+padding+width*/
            border-radius: 130px;
            /*border-radius: 左上和右下  右上和左下*/
            border-radius: 50px 100px;
            /*border-radius: 左上 右上 右下 左下 （顺时针）*/
            border-radius: 50px 100px 50px 100px;
        }
        /*方形头像变圆形头像*/
        img{
            width: 200px;
            border-radius: 100px;
        }
    </style>

</head>
<body>

<div></div>
<div><img src="./images/head.jpg" alt="head"></div>
</body>
</html>
```



4.5、盒子阴影

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        div{
            height: 400px;
            text-align: center;
            background: #c5efef;
        }
        img{
            /*居中效果需要其父级元素设置过text-align: center*/
            margin: 100px auto;
            width: 200px;
            border-radius: 100px;
            box-shadow: 20px 20px 100px orange;
```

```
        }
    </style>

</head>
<body>
<div>
    <img src="./images/head.jpg" alt="head">
</div>
</body>
</html>
```



5、浮动

5.1、标准文档流

文档流指的是元素排版布局过程中，元素会**默认**自动从左往右，从上往下的**流式排列方式**。并最终窗体自上而下分成一行行，并在每行中从左至右的顺序排放元素。

块级元素：
1).霸占一行，不能与其他任何元素并列
2).能接受宽、高
3).如果不设置宽度，那么宽度将默认变为父亲的100%，即和父亲一样宽
行内元素：
1).与其他元素并排
2).不能设置宽、高。默认的宽度就是文字的宽度

在HTML中，标签分为：文本级和容器级；
文本级：p、span、a、b、i、u、em
容器级：div、h系列、li、dt、dd

所有的**文本级标签**都是**行内元素**，*除了p*；**p是个文本级，但是是个块级元素**；
所有的**容器级标签**都是**块级元素**

行内元素可以包在块级元素中

块级元素不能包在行内元素中

5.2、display

- display: block;显示为块元素
- display: inline;显示为行内元素
- display: inline-block;显示为 行内块元素
- display: none;不显示

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <!--
    display: block;显示为块元素
    display: inline;显示为行内元素
    display: inline-block;显示为 行内块元素
    display: none;不显示
    -->
    <style>
        body{
            margin: 0px;
        }
        div{
            background: #c5efef;
            width: 100px;
            height: 100px;
            display: inline-block;
        }
        span{
            background: orange;
            width: 100px;
            height: 100px;
            display: inline-block;
        }
    </style>

</head>
<body>

<div>div块元素</div>
<span>span行内元素</span>

</body>
</html>
```

display 也是一种实现行内元素排列的方式，但是我们很多情况都是使用float

5.3、float

定义和用法

float 属性定义元素在哪个方向浮动。以往这个属性总应用于图像，使文本围绕在图像周围，不过在 CSS 中，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。

如果浮动非替换元素，则要指定一个明确的宽度；否则，它们会尽可能地窄。

注释：假如在一行之上只有极少的空间可供浮动元素，那么这个元素会跳至下一行，这个过程会持续到某一行拥有足够的空间为止。

- float: left 向左浮动
- float: right 向右浮动

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>float</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>

<div id="father">
    <div class="layer01"><img src="images/1.jpg" alt=""></div>
    <div class="layer02"><img src="images/2.jpg" alt=""></div>
    <div class="layer03"><img src="images/3.jpg" alt=""></div>
    <div class="layer04">浮动的盒子可以向左浮动，也可以向右浮动，直到它的外边缘碰到包含框或另一个浮动盒子为止</div>
</div>

</body>
</html>
```

```
div{
    margin: 10px;
    padding: 5px;
}
#father{
    border: 1px solid #000;
}
.layer01{
    border: 1px solid red;
    display: inline;
    float: left;
}
.layer02{
    border: 1px solid blue;
    display: inline;
    float: left;
}
.layer03{
    border: 1px solid green;
    display: inline;
    float: left;
}
.layer04{
    border: 1px #666 dashed;
```

```
    font-size: 12px;
    line-height: 23px;
    display: inline;
    float: left;
}
.layer01>img{
    width: 200px;
}
.layer02>img{
    width: 150px;
}
.layer03>img{
    width: 100px;
}

.layer01,.layer02,.layer03,.layer04{
    display: inline;
    clear: both;
    /*clear: bottom;*/

}
```

5.4、对比

- display

  方向不可以控制

- float

  浮动起来的话会脱离标准文档流，所以要解决浮动塌陷问题

5.5、解决浮动塌陷问题

方式一、增加一个空的div，放置在其他浮动元素之后，设置为块级元素，并且不允许左右有浮动元素。

```
#father>div:last-child{
    margin: 0;
    padding: 0;
    display: block;
    clear: both;
}
```

方式二、设置父级元素的高度大于最高的子元素

```
#father{
    border: 1px solid #000;
    height: 200px;
}
```

方式三、overflow

在浮动元素的父级设置overflow

可能的值

值	描述
visible	默认值。内容不会被修剪，会呈现在元素框之外。
hidden	内容会被修剪，并且其余内容是不可见的。
scroll	内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。
auto	如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。
inherit	规定应该从父元素继承 overflow 属性的值。

```
#father{
    border: 1px solid #000;
    height: 200px;
    overflow: auto;
}
```

方式四、伪类选择器:after（默认）

浮动元素的父级设置 伪类选择器after-`:after`

```
#father:after{
    content: "";/*空元素*/
    display: block;/*置为块级元素*/
    clear: both;/*元素左右不允许浮动*/
}
```

总结:

方式一:

缺点：需要在改变内容结构。

方式二:

缺点：新增元素可能会超过原来设置的高度，需要再次设置高度

方式三:

缺点：超出部分隐藏或滚动条显示，不美观

方式四:

默认，无副作用，一劳永逸。

6、定位

值	描述
absolute	生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。 元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。
fixed	生成绝对定位的元素，相对于浏览器窗口进行定位。 元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。
relative	生成相对定位的元素，相对于其正常位置进行定位。 因此，"left:20" 会向元素的 LEFT 位置添加 20 像素。
static	默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。
inherit	规定应该从父元素继承 position 属性的值。

6.1、相对定位

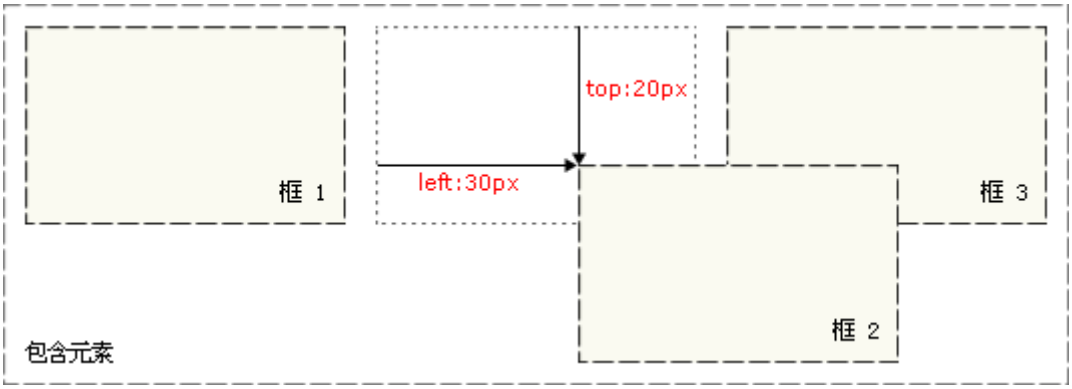
设置为相对定位的元素框会偏移某个距离。元素仍然保持其未定位前的形状，*它原本所占的空间仍保留*。

如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点进行移动。

如果将 top 设置为 20px，那么框将在原位置顶部下面 20 像素的地方。如果 left 设置为 30 像素，那么会在元素左边创建 30 像素的空间，也就是将元素向右移动。

```
#box_relative {
  position: relative;
  left: 30px;
  top: 20px;
}
```

如下图所示：



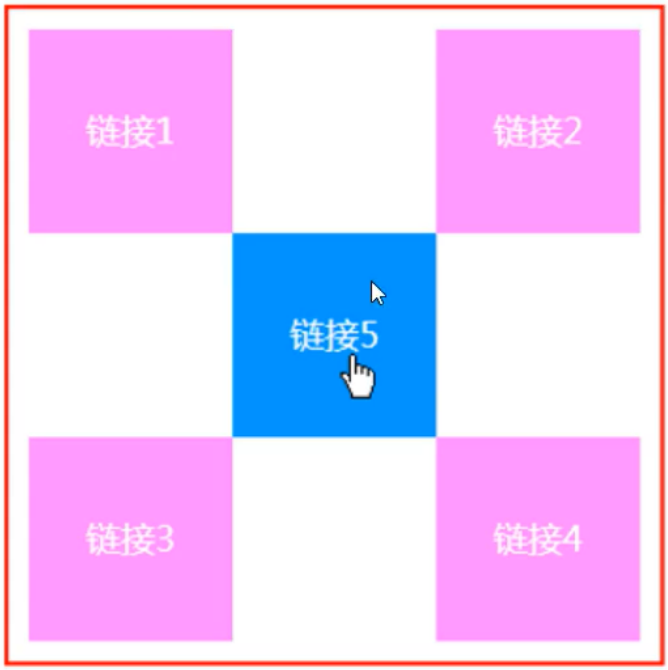
注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        div{
            margin: 10px;
```

```
            padding: 5px;
            font-size: 12px;
            line-height: 25px;
        }
        #father{
            border: 1px solid black;
            padding: 0px;
        }
        #div1{
            border: 1px solid red;
            background: red;
            position: relative;/*相对定位*/
            top: 20px;/*原位置在新位置的top20px处--即向下偏移20px*/
        }
        #div2{
            border: 1px solid green;
            background: green;
            position: relative;
        }
        #div3{
            border: 1px solid blue;
            background: blue;
            position: relative;
        }
    </style>

</head>
<body>

<div id="father">
    <div id="div1">第一个盒子</div>
    <div id="div2">第二个盒子</div>
    <div id="div3">第三个盒子</div>
</div>

</body>
</html>
```

题目：

- 使用 和超链接<a>布局页面
- 每个超链接宽度和高度都是100px，背景颜色是粉色，鼠标指针移上去时变为蓝色
- 使用相对定位改变每个超链接的位置



解答：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        div{
            border: 2px solid red;
            padding: 10px;
            width: 300px;
            height:300px;
        }
        a{
            width: 100px;
            height: 100px;
            display: inline-block;
            background: pink;
            color: white;
            text-align: center;
            line-height: 100px;
            margin-right: -5px;
            text-decoration: none;
            position: relative;/*相对定位*/
        }
        a:hover{
            background: blue;
        }
        a:nth-of-type(1){
        }
```

```
        a:nth-of-type(2){
            left: 100px;
        }
        a:nth-of-type(3){
            top: 200px;
            right: 200px;
        }
        a:nth-of-type(4){
            top: 100px;
            left: 200px;
        }
        a:nth-of-type(5){
        }
    </style>

</head>
<body>

<div>
    <a href="#">链接1</a>
    <a href="#">链接2</a>
    <a href="#">链接3</a>
    <a href="#">链接4</a>
    <a href="#">链接5</a>
</div>

</body>
</html>
```

6.2、绝对定位

1、没有父季尾宿定位的前提下，相对于浏览器定位

2、祖先及元素存在定位，我们通常会基于祖先定位

3、在祖先元素范围内移动

相对于祖先级或浏览器的位置，进行指定便宜，绝对定位不在标准文档流中，原来的位置不会被保留。

绝对定位的元素的位置相对于***最近的已定位祖先元素***，如果元素没有已定位的祖先元素，那么它的位置相对于*最初的包含块*。

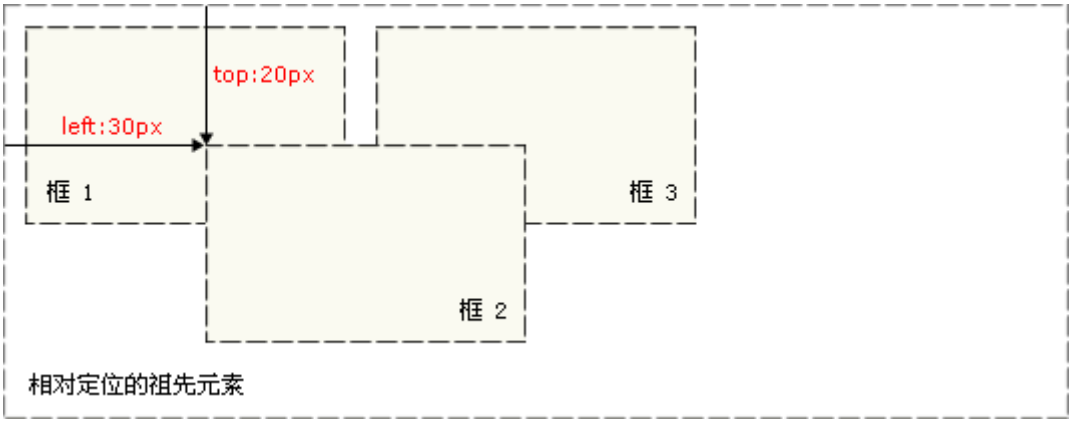
父级元素无定位--会向上寻找父级的父级，以此类推，直至找到浏览器边界（HTML）。

绝对定位的元素框会从文档流中完全删除，元素原先在正常文档流中所占的空间会关闭，定位后生成一个块级框。

示例：

```
#box_relative {
  position: absolute;
  left: 30px;
  top: 20px;
}
```

如下图所示：



如果同时设定 left 和 right，或者 top 和 bottom，会使元素拉伸，占据left 到 right之间的整个宽度，或者 top 到 bottom 之间的整个高度

如：

```
border: 1px solid red;
background: red;
position: absolute;
right: 0px;
left: 0px;
```

6.3、固定定位

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        body{
            height: 1000px;
        }
       div:nth-of-type(1){
           width: 100px;
           height: 100px;
           background: #33d2d2;
           position: absolute;/*绝对定位*/
           bottom: 0;
```

```
            right: 0;
        }
        div:nth-of-type(2){
            width: 50px;
            height: 50px;
            background: pink;
            position: fixed;/*固定定位*/
            bottom: 0;
            right: 0;
        }
    </style>

</head>
<body>

<div>div1</div>
<div>div2</div>

</body>
</html>
```

6.4、clip属性

clip 属性剪裁绝对定位和固定定位元素。

说明:

这个属性用于定义一个剪裁矩形。对于一个绝对定义元素，在这个矩形内的内容才可见。出了这个剪裁区域的内容会根据 overflow 的值来处理。剪裁区域可能比元素的内容区大，也可能比内容区小。

shape	设置元素的形状。唯一合法的形状值是：rect (*top*, *right*, *bottom*, *left*)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        img{
            position: absolute;
            right: 0;
            bottom: 0;
            clip: rect(0px,1000px,500px,0px);
        }
    </style>

</head>
<body>

<img src="./images/1.jpg" alt="">

</body>
</html>
```

rect()细节

6.5、z-index

z-index 属性设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面

注意：z-index 仅能在定位元素上奏效

auto	默认。堆叠顺序与父元素相等。
number	设置元素的堆叠顺序。
inherit	规定应该从父元素继承 z-index 属性的值。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>
        div:nth-of-type(1){
            text-align: center;
            line-height: 100px;
            width: 100px;
            height: 100px;
            background: #7bf3f3;
            position: absolute;
            left: 0;
            top: 0;
            /*z-index: 1;*/
        }
        div:nth-of-type(2){
            text-align: center;
            line-height: 200px;
            width: 200px;
            height: 200px;
            background: pink;
            position: absolute;
            left: 0;
            top: 0;
            opacity: 0.5;/*不透明度*/
            /*z-index: 0;*/
        }
    </style>

</head>
<body>

<div>div1</div>
<div>div2</div>

</body>
</html>
```

opacity：不透明度

7、动画

8、总结