

# CSS

## 1. 字体

CSS 字体属性 (Font)		CS
属性	描述	CS
<u>font</u>	在一个声明中设置所有字体属性。	1
<u>font-family</u>	规定文本的字体系列。	1
<u>font-size</u>	规定文本的字体尺寸。	1
<u>font-size-adjust</u>	为元素规定 aspect 值。	2
<u>font-stretch</u>	收缩或拉伸当前的字体系列。	2
<u>font-style</u>	规定文本的字体样式。	1
<u>font-variant</u>	规定是否以小型大写字母的字体显示文本。	1
<u>font-weight</u>	规定字体的粗细。	1

## 2. 选择器

## 子元素选择器

子元素选择器语法

```
元素名>元素名 {  
    样式声明 ;  
}
```

```
div>a {  
    color: red;  
}
```

## 下周工作规划

01

父子元素之间用>相隔

02

选择的子元素必须为父元素的直系子元素

# 并集选择器

## 并集选择器语法

```
选择器1，选择器2，选择器3，选择器n{  
    样式声明；  
}
```

```
.nav,  
.p1,  
span a{  
    color: blue;  
}
```

01

选择器与选择器之间用 “，” 相隔

02

并集选择器之中可以填入任何形式的选择器

# 伪类选择器

:link 可以改变未访问过的链接样式

:visited 可以改变已经访问过的链接样式

:hover 可以改变鼠标经过链接时的样式

:active 可以改变在鼠标点击（为松开左键）链接时的样式

```
/* :link 是可以改变未访问过的链接样式 */
a:link {
    color: red;
}

/* :visited 可以改变已经访问过的链接样式 */
a:visited {
    color: yellow;
}

/* :hover 可以改变鼠标经过链接时的样式 */
a:hover {
    color: blue;
}

/* :active 可以改变在鼠标点击（为松开左键）链接时的样式 */
a:active {
    color: orange;
}
</style>
</head>
```

# CSS \* 选择器

选择所有元素，并设置其背景色：

```
*  
{  
background-color:yellow;  
}
```

## 定义和用法

- \* 选择器选择所有元素。
- \* 选择器也可以选择另一个元素内的所有元素：  
选择  
元素内的所有元素：

```
div *  
{  
background-color:yellow;  
}
```

## 3. 转换元素显示模式

### 语法

display: 元素显示模式;

可选选项：

block 块元素

inline 行内元素

inline-block 行内块元素

none 不显示（莫得了）

元素显示模式转换

```
me * 2.行内元素.html 3.行内块元素.html 4.元素显示模式转换.html  
px  
> DOCTYPE html>  
> <html>  
到内置浏览器(I)  
>   <head>  
4     <meta charset="utf-8">  
5     <title></title>  
6     <style type="text/css">  
7       div {  
8         height: 200px;  
9         width: 200px;  
10        background-color: red;  
11        margin-top: 10px; o  
12        display: inline;  
13      }  
14    </style>  
15  </head>  
16  <body>  
17    <div>123</div>  
18    <div>123</div>  
19  </body>  
20 </html>  
21
```

## 块级元素和行内元素

标签分为两种等级：

- 1, 行内元素。
- 2, 块级元素。

### 行内元素和块级元素的区别：

行内元素：

- 与其他行内元素并排
- 不能设置宽高，默认的宽度就是文字的宽度

块级元素：

- 霸占一行，不能与其他任何元素并列。

- 能接受宽高，如果不设置宽度，那么宽度将默认变为父级的100%。

## 块级元素和行内元素的分类：

在HTML的角度来讲，标签分为：

文本级标签：p , span , a , b , i , u , em

容器级标签：div , h系列 , li , dt , dd

p：里面只能放文字和图片和表单元素，p里面不能放h和ul，也不能放p。

从CSS的角度讲，CSS的分类和上面的很像，就p不一样：

行内元素：除了p之外，所有的文本级标签，都是行内元素。p是个文本级标签，但是是个块级元素。

块级元素：所有的容器级标签，都是块级元素，以及p标签。

## 块级元素和行内元素的相互转换：

我们可以通过display属性将块级元素(比如div)和行内元素进行相互转换。

display: inline;

那么这个标签将变为行内元素，即：

1，此时这个div将不能设置宽度和高度了。

2，此时这个div可以和其他行内元素并排了。

同样的到了我们也可以用display将行内元素(比如span)转行成块级元素。

display: block;

那么这个span标签将变为块级标签，即：

1，此时这个span能够设置宽度，高度。

2，此时这个span必须独占一行，其他元素无法与之并排。

3，如果不设置宽度，将占满父级。

标准流里面的限制非常多，导致很多页面效果无法实现，如果我们现在就要并排，并且就要设置宽度，就只有：脱离标准流。

CSS一共有三种手段，是一个元素脱离标准流文档：

• 浮动

---

• 绝对定位

---

• 固定定位

---

下面章节讲

## 4.三大特性

- 层叠性
- 继承性
- 优先级

## CSS三大特性 层叠性



如果两个相同的选择器之中有样式声明冲突的情况下  
后面选择器的样式会将前面选择器冲突的样式覆盖掉

如果相同选择器之中没有发生冲突的样式  
则会保留不变

## CSS三大特性 继承性



所谓的继承性就是字面理解的意思

就是父元素的一些样式会同时继承到它的子元素之中

这种特性可以很方便的帮我们避免代码重复

## CSS三大特性 继承性



1.样式的继承，只会继承font, text, line开头的样式和color样式

2.子元素继承的样式权重为0

# CSS三大特性 优先级

优先级也就是选择器的权重

简单来说，那个选择器的权重高就执行那个选择器

通配符选择器或者继承样式为0

标签选择器权重为1

类选择器权重为10

id选择器权重为100

内联样式权重为1000

!important 为无穷大

# CSS三大特性 优先级

```
<style type="text/css">
  ul li {
    color : red;
  }
</style>
```

标签选择器权重为1  
所以  $1+1 = 2$

```
.box span {
```

标签选择器权重为1  
类选择器权重为10  
所以  $10+1 = 11$

## 5.盒子模型

# 课程内容

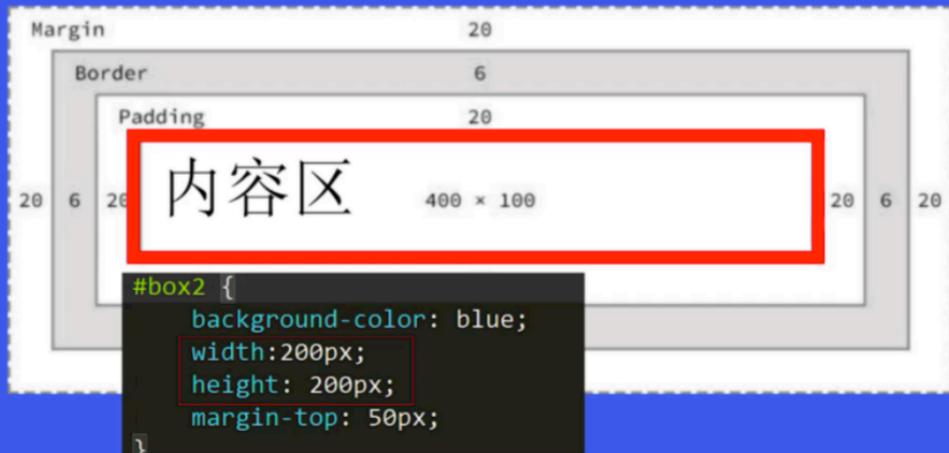
- 盒子模型初步
  - 边框
  - 内边距
  - 外边距

# 盒子模型

网页之中都是由一个个板块而搭建而成  
这些板块就是盒子模型  
所谓的搭建网站，就是在搭盒子  
而在盒子之中，还可以包含更多的盒子

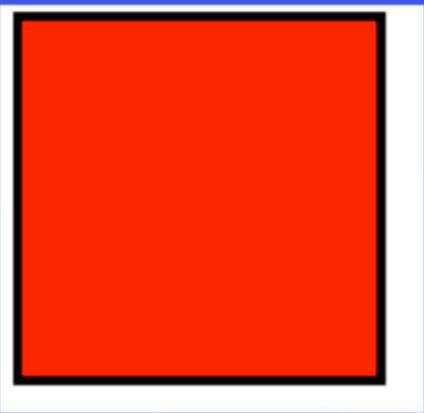


## 盒子模型组成部分



\*边框\*

# 边框 border



## border-width

用来调整盒子模型边框线宽

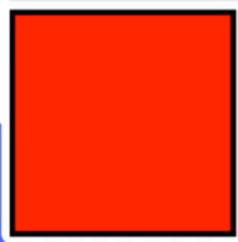
## border-style

用来调整边框线条样式

## border-color

用来调整边框线条颜色

# border-style



solid

实线



dashed

虚线



dotted

点线



none

无边框

## border符合写法

border: border-width border-style border-color;

比如: border: 1px solid black;

```
    padding: 0;
    margin: 0px;
}
#box1 {

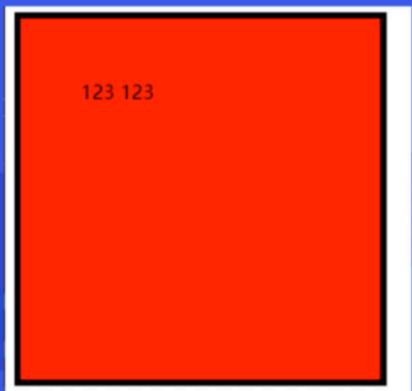
    background-color: red;
    margin: 50px;
    width: 200px;
    height: 200px;
    /* 边框会影响盒子模型实际大小 */
    border: 1px solid black;
}
```

```
        </style>
    </head>
    <body>
        <div id="box1">
        </div>
    </body>
</html>
```

## \*内边距\*

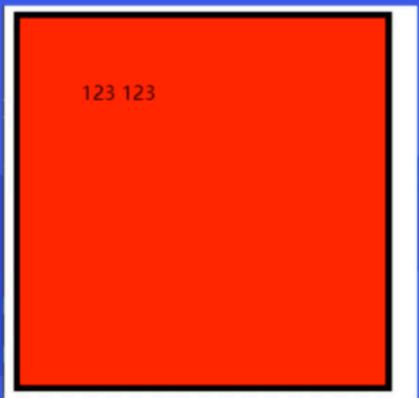
---

# 内边距 padding



内边距是内容区与边框之间的距离

内边距也会影响盒子模型的整体大小



# 内边距 padding

padding-top 上内边距

padding-right 右内边距

padding-bottom 下内边距

padding-left 左内边距

## padding符合写法

padding: 一个数值;

则表示上右下左四个内边距都是此数值的长度

比如: padding: 20px;

# padding符合写法

padding: 一个数值 两个数值;

第一个数值为上下内边距  
第二个数值为左右内边距

比如: padding: 20px  
          40px;

# padding符合写法

padding: 一个数值 两个数值 三个数值;

第一个数值为上内边距  
第二个数值为左右内边距  
第三个数值为下内边距

比如: padding: 20px  
          40px 60px;

# padding符合写法

padding：一个数值 两个数值 三个数值 四个数值；

第一个数值为上内边距

第二个数值为右内边距

第三个数值为下内边距

第四个数值为左内边距

比如：padding: 20px 40px 60px  
80px;

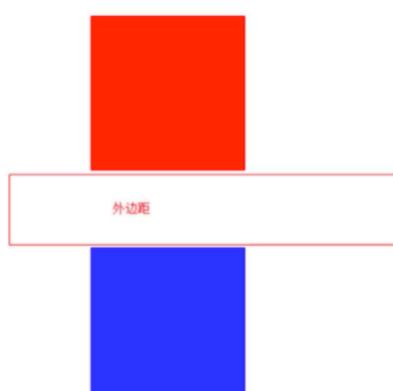
## \*外边距\*



## 外边距

外边距是指盒子模型与其他盒子模型相隔的距离

因为在盒子模型之外，所以不会影响盒子模型大小



## 外边距 定义方式



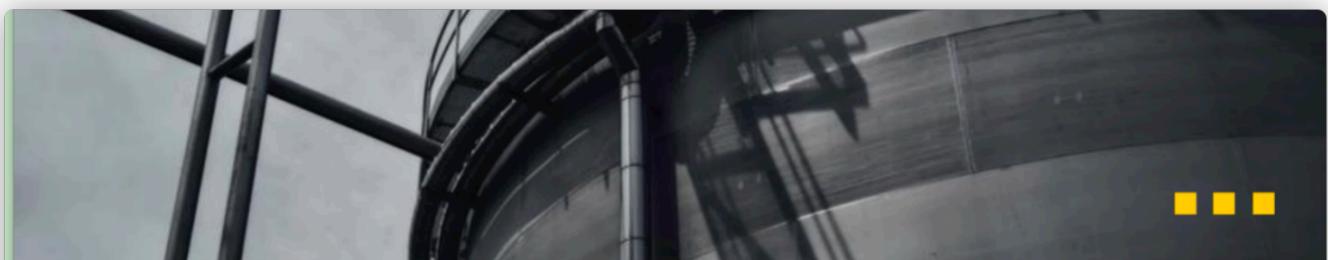
### 分向定义

margin-top  
margin-bottom  
margin-right  
margin-left



### 复合型定义

margin: 上右下左;

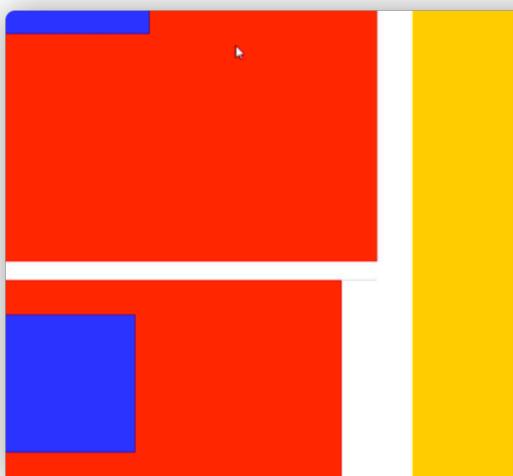


## 元素 居中 对齐

1.首先定义块元素宽高

2.在样式之中填写margin: 0 auto;

注意：只是将左右外边距调成auto即可，上下内边距可随意调整



## 嵌套元素塌陷

解决方法有三种：

- 1.给父元素添加边框
- 2.给父元素添加内边距
- 3.给父元素添加overflow: hidden;

## 6. 浮动

CSS一共有三种手段，是一个元素脱离标准流文档：

- 浮动
- 绝对定位
- 固定定位

### 传统网页布局

标准流

又称为文档流，按照元素的书写顺序依次排列。

浮动

脱离文档流，按照开发者的意愿进行浮动到指定方向。

定位

比浮动更加自由  
可以按照规定的数值，将网页元素定位至指定的方向。

在网页布局之中，经常用这三种方法掺杂着进行开发

# 浮动

元素脱离文档流，根据开发者的意愿漂浮到网页的任意方向。

```
<style type="text/css">
    #box {
        float: left;
        width: 200px;
        height: 200px;
        background-color: red;
    }
</style>
```

## 浮动语法

float: 对齐方向；

参数：

1.left: 向左对齐

2.right: 向右对齐

如：float: left;

### 1. 浮动：

浮动是CSS里面布局最多的一个属性。

float: 表示浮动的意思。

属性:

- none: 表示不浮动, 默认。
- left: 表示左浮动。
- right: 表示右浮动。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        *{
            /*margin: 0;*/
        }
        .box1{
            width: 300px;
            height: 300px;
            background-color: red;
            float: left;
        }
        .box2{
            width: 400px;
            height: 400px;
            background-color: green;
            float: right;
        }
        span{
            float: left;
            width: 100px;
            height: 200px;
            background-color: yellow;
        }
    </style>
</head>
<body>
    <div class="box1"></div>
    <div class="box2"></div>
    <span>路飞</span>
</body>
</html>
```



我们会发现，三个元素并排显示，.box1和span因为是左浮动，所有紧挨在一起，这种现象是贴边现象。.box2盒子因为是右浮动，所以紧靠着右边。

## 浮动的四大特性：

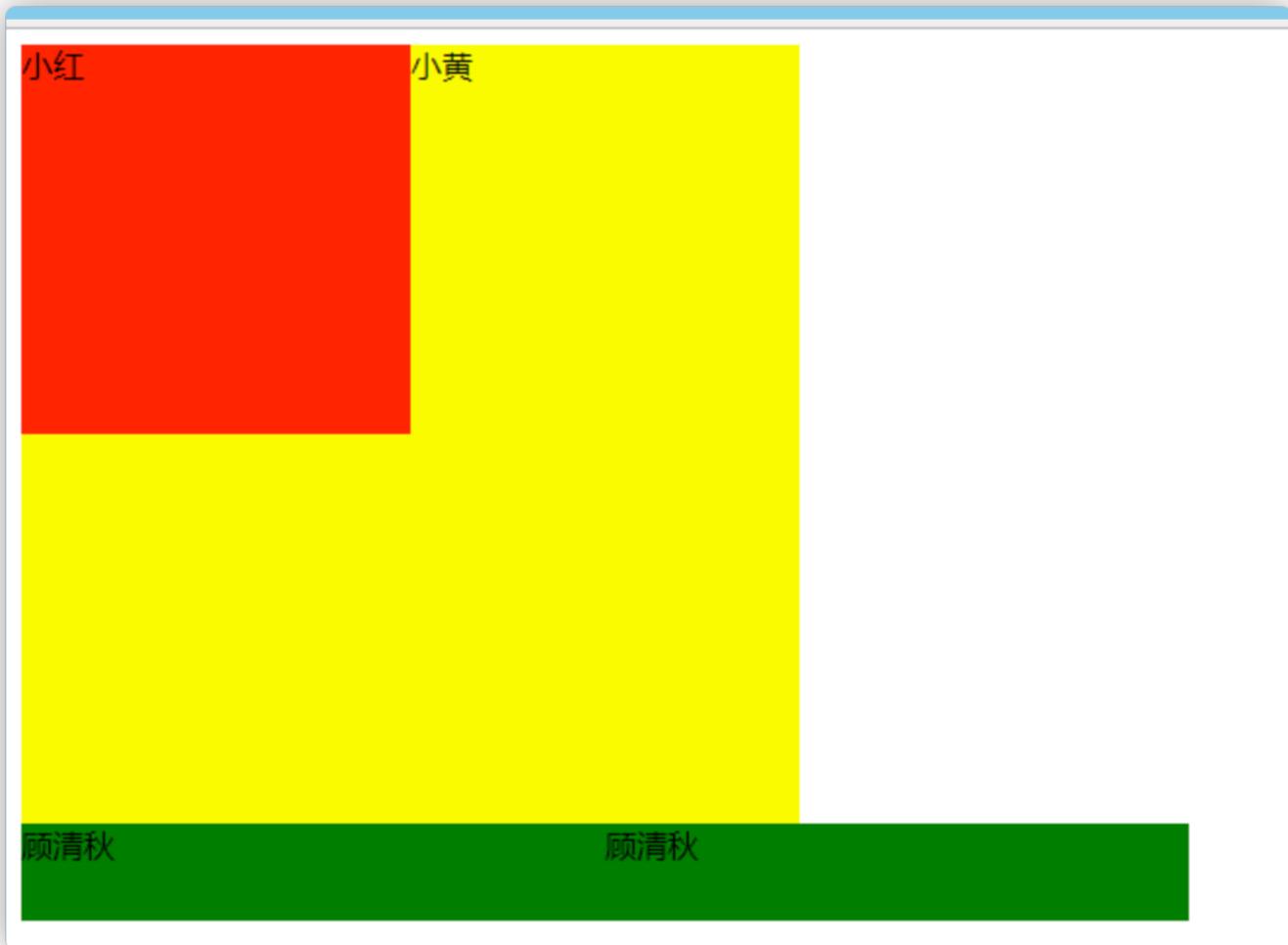
- 1，浮动的元素脱离标准流
- 2，浮动的元素互相贴靠。
- 3，浮动的元素由"子围"效果。
- 4，收缩的效果。

## 浮动元素的脱标：

脱标：就是脱离了标准文档流。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .box1{
            width: 200px;
            height: 200px;
            background-color: red;
            float: left;
        }
        .box2{
            width: 400px;
            height: 400px;
            background-color: yellow;
        }
        span{
            float: left;
            width: 300px;
            height: 50px;
            background-color: green;
        }
    </style>
</head>
<body>
    <div class="box1"></div>
    <div class="box2"></div>
    <span>路飞</span>
</body>
</html>
```

```
</style>
</head>
<body>
  <div class="box1">小红</div>
  <div class="box2">小黄</div>
  <span>顾清秋</span>
  <span>顾清秋</span>
</body>
</html>
```



效果：红色的盒子盖住了黄色的盒子，一个行内的span标签，竟然能够设置宽高了。

原因1：小红设置了浮动，而小黄并没有设置浮动，小红脱离了标准文档流，其实质是他不在页面中占据位置了，此时浏览器认为小黄是标准文档流的第一个盒子，所以就渲染到了页面中的第一个位置上，这种现象，也有一种叫法：浮动元素“飘起来了”。

原因2：所有的标签一旦设置浮动，就能够并排且不区分块元素或行内元素，换言之，能够设置宽高了。

## 浮动元素互相贴靠：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .box1{
            width: 100px;
            height: 400px;
            background-color: red;
            float: left;
        }
        .box2{
            width: 150px;
            height: 450px;
            float: left;
            background-color: yellow;
        }
        .box3{
            width: 300px;
            height: 300px;
            float: left;
            background-color: green;
        }
    </style>
</head>
<body>
    <div class="box1">小红</div>
    <div class="box2">小黄</div>
    <div class="box3">小绿</div>
</body>
</html>
```

效果发现：如果父元素有足够的空间，那么小绿就会紧靠小黄，小黄紧靠小红，小红靠着边。  
如果没有足够的空间，小绿那么就会靠着小红，若没有足够的空间靠着小红，就会自己靠边。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .box{
            width: 300px;
            border:1px solid black;
        }
    </style>
```

```

.box1{
    width: 100px;
    height: 400px;
    background-color: red;
    float: left;
}

.box2{
    width: 150px;
    height: 450px;
    float: left;
    background-color: yellow;
}

.box3{
    width: 80px;
    height: 300px;
    float: left;
    background-color: green;
}

</style>
</head>
<body>
<div class="box">
    <div class="box1">小红</div>
    <div class="box2">小黄</div>
    <div class="box3">小绿</div>
</div>
</body>
</html>

```

例子

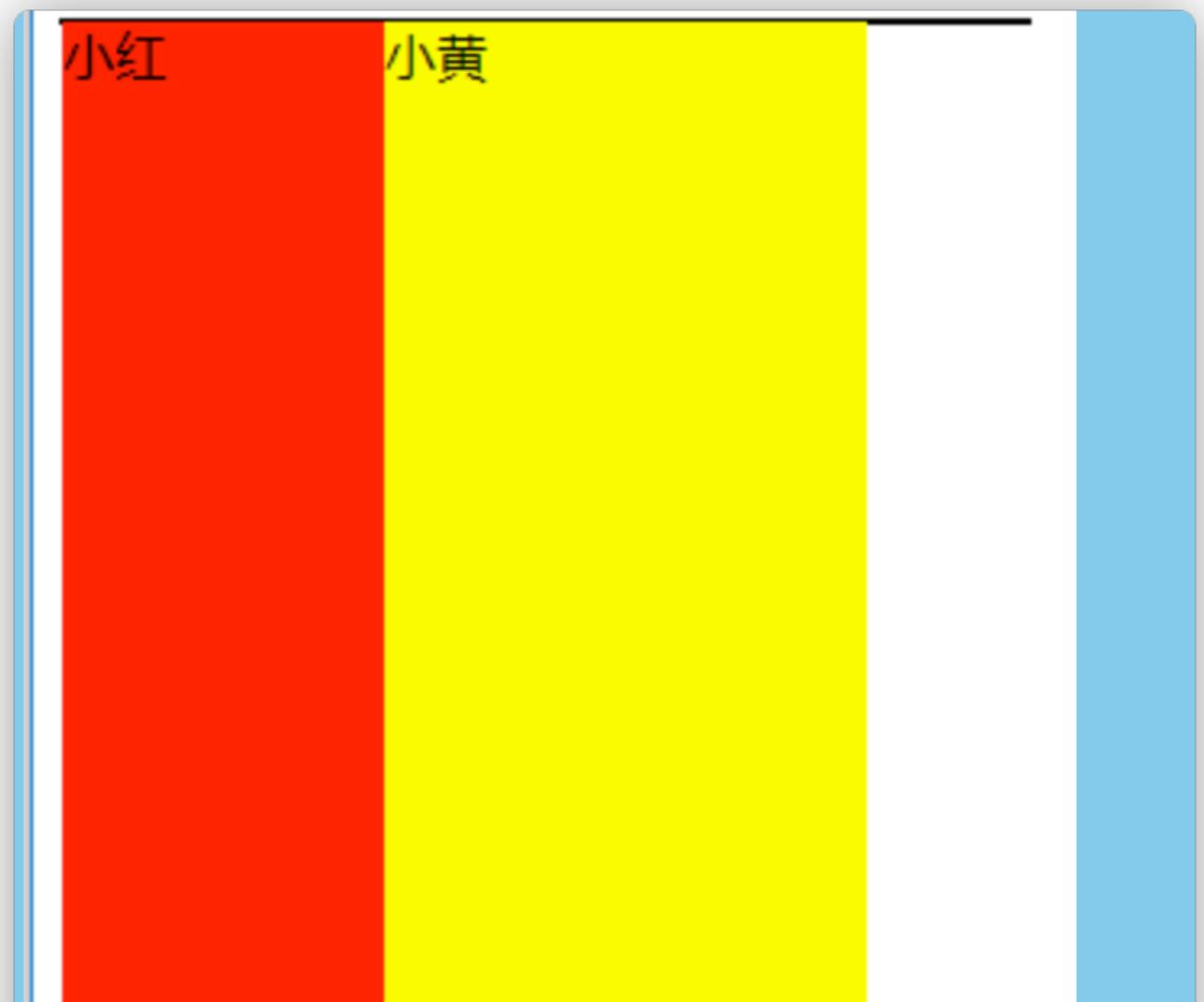
效果发现：如果父元素有足够的空间，那么小绿就会紧靠小黄，小黄紧靠小红，小红靠着边。如果没有足够的空间，小绿那么就会靠着小红，若没有足够的空间靠着小红，就会自己靠边。

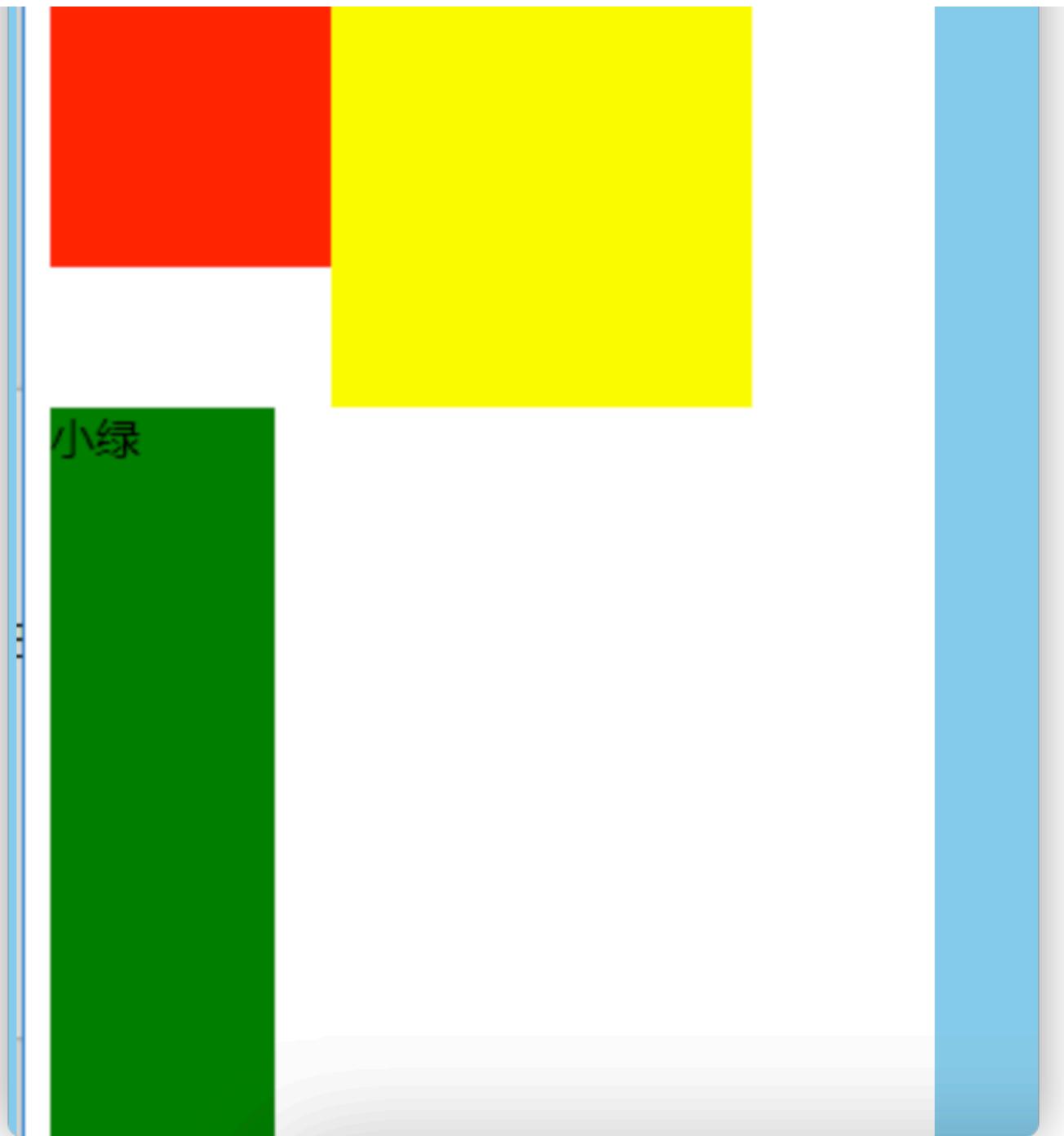
```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .box{
            width: 300px;
            border: 1px solid black;
        }
        .box1{
            width: 100px;
            height: 400px;
            background-color: red;
            float: left;
        }
        .box2{

```

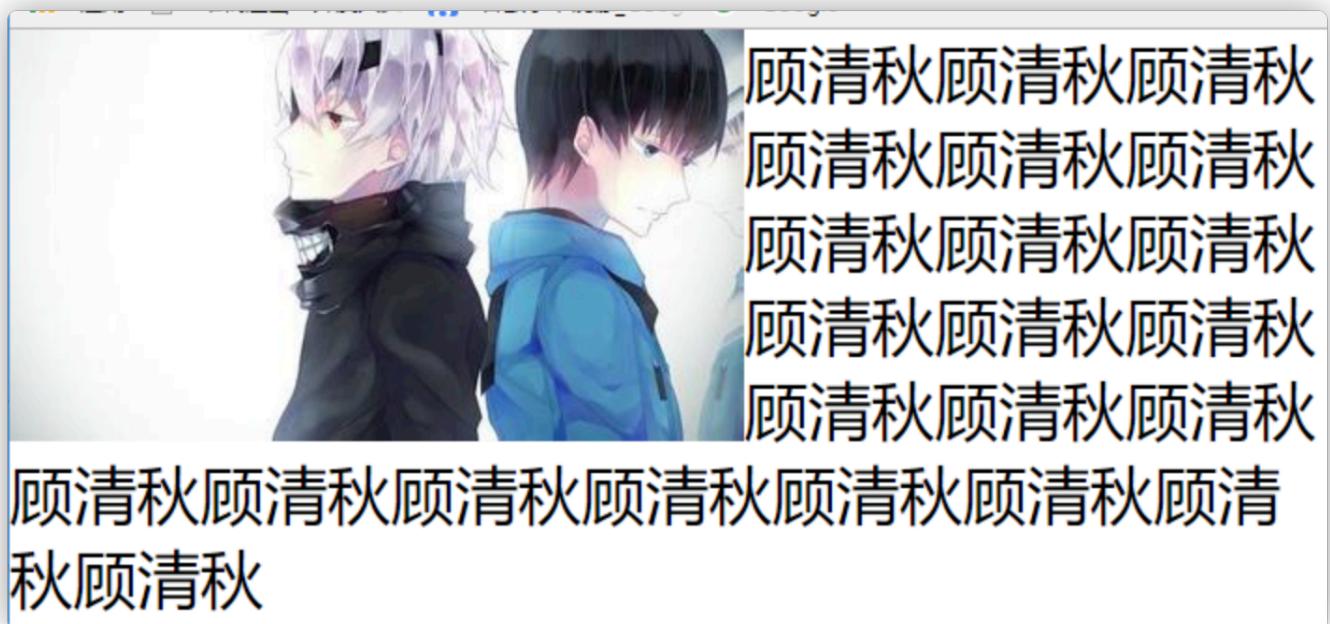
```
        width: 150px;
        height: 450px;
        float: left;
        background-color: yellow;
    }
.box3{
    width:     80px;
    height: 300px;
    float:     left;
    background-color: green;
}
</style>
</head>
<body>
<div class="box">
    <div class="box1">小红</div>
    <div class="box2">小黄</div>
    <div class="box3">小绿</div>
</div>
</body>
</html>
```





## 浮动元素字围效果：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>
  <style>
    *{
      padding: 0;
      margin:0;
    }
    div{
```



效果发现：所谓的字围效果，当div浮动，p不浮动，div遮盖了p,div的层级提高，但是p中的文字不会被遮盖，此时就形成了字围效果。

## 浮动元素紧凑效果：

收缩：一个浮动元素，如果没有设置width，那么就自动收缩为文字的宽度。（和行内元素很像）

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>緊湊效果</title>
<style>
    div{
        float: left;
        background-color: red;
    }
}
```

```
</style>
</head>
<body>
    <div>顾清秋</div>
</body>
</html>
```



顾清秋

## 2.清除浮动

---

谨记：关于浮动，一定要遵循一个原则，永远不是一个盒子单独浮动，要浮动就要一起浮动。另外，有浮动，一定要清楚浮动。

# 清除浮动

在网页开发时，常常不能通过限制父元素的高度，从而发生高度塌陷的问题。

为了解决这个问题，出现了清除浮动这一操作。

视频之中演示的方法为双伪元素清除浮动法。



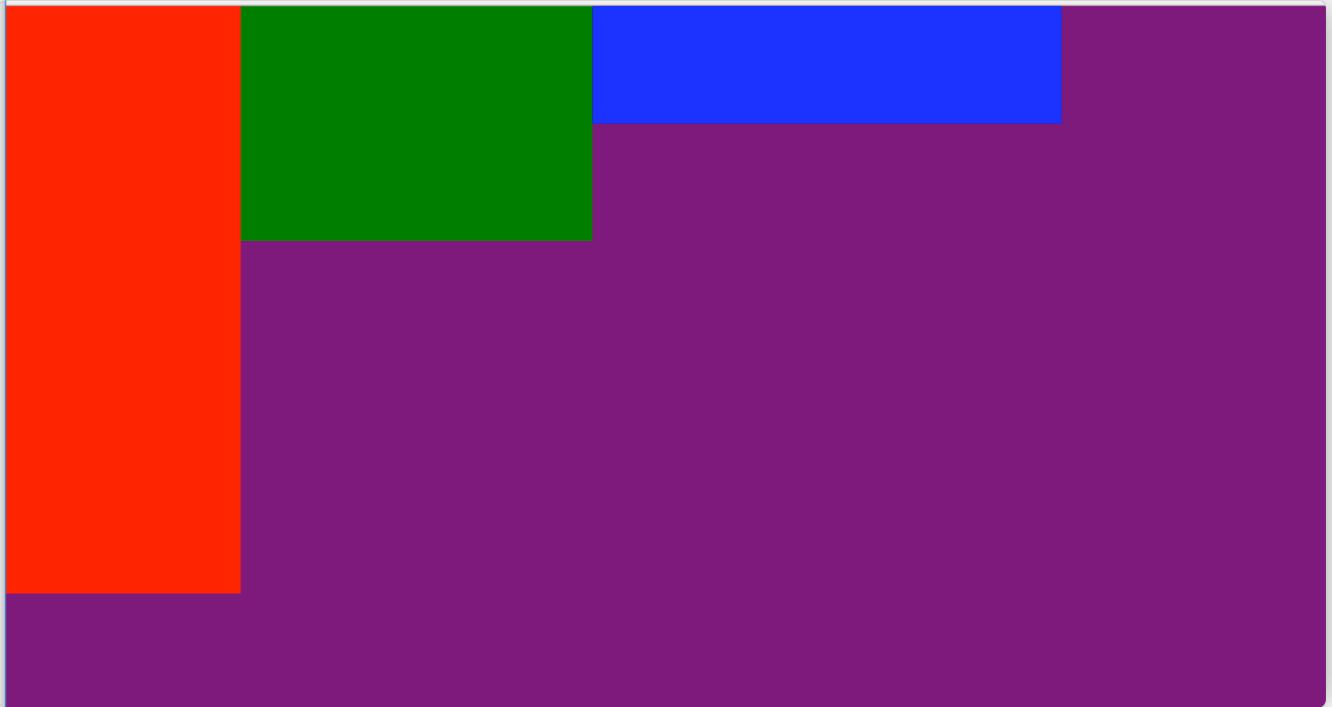
## 为什么要清楚浮动：

在页面布局的时候，每个结构中的父元素的高度，我们一般不会设置。

大家想，如果我第一版的页面的写完了，感觉非常爽，突然隔了一个月，老板说页面某一块的区域，我要加点内容，或者我觉得图片要缩小一下。这样的需求在工作中非常常见的。真想打他啊。那么此时作为一个前端小白，肯定是去每个地方加内容，改图片，然后修改父盒子的高度。那问题来了，这样不影响开发效率吗？答案是肯定的。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    *{
      padding: 0;
      margin: 0;
    }
  </style>
</head>
<body>
  <div>
    <h1>Hello World</h1>
    <img alt="A small image of a person's face." data-bbox="100px 100px 200px 200px"/>
  </div>
</body>
</html>
```

```
.father{
    width: 1126px;
    /*子元素浮动，父级一般不设置高度*/
}
.box1{
    width: 200px;
    height: 500px;
    float: left;
    background-color: red;
}
.box2{
    width: 300px;
    height: 200px;
    float: left;
    background-color: green;
}
.box3{
    width: 400px;
    height: 100px;
    float: left;
    background-color: blue;
}
.father{
    width: 1126px;
    height: 600px;
    background-color: purple;
}
</style>
</head>
<body>
    <div class="father">
        <div class="box1"></div>
        <div class="box2"></div>
        <div class="box3"></div>
    </div>
    <div class="father2"></div>
</body>
</html>
```



效果发现：如果不给.father一个高度，那么浮动子元素是不会填充父盒子的高度，那么.father2的盒子就会占据第一个位置，影响页面布局。

那么我们知道，浮动元素确实能实现我们页面元素并排的效果，这是它的好处，同时它还带来了页面布局极大的错乱！！！所以我们要清除浮动！

## 清除浮动的方法：

- 1, 给父盒子设置高度。
- 2, clear: both;
- 3, 伪元素清除法
- 4, overflow: hidden;

## 给父盒子设置高度：

使用不灵活，会固定父盒子的高度。

### **clear: both;**

clear: 意思就是清楚的意思。

有三个值：

left: 当前元素左边不允许有浮动元素，

right: 当前元素右边不允许有浮动元素。

both: 当前元素的左右两边都不允许有浮动元素。

给浮动元素后面加一个空的div，并且该元素不浮动，然后设置clear: both。

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

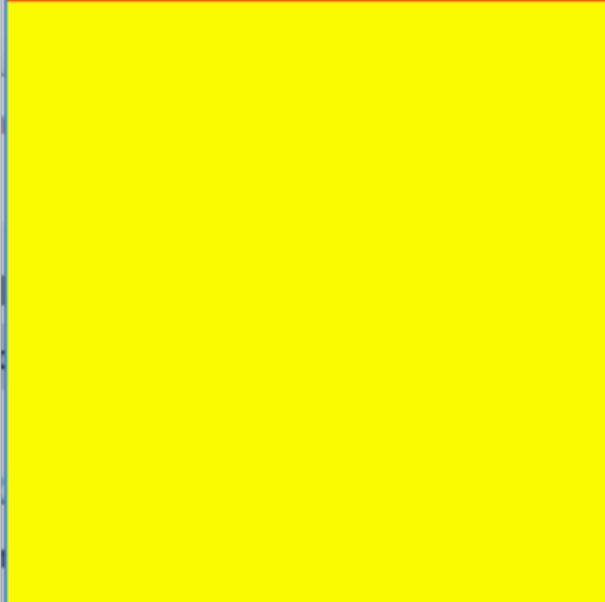
```
<meta charset="UTF-8">
<title>Document</title>
<style>
    * {
        padding: 0;
        margin: 0;
    }
    ul{
        list-style: none;
    }
    div{
        width: 400px;
    }
    div ul li{
        float: left;
        width: 100px;
        height: 40px;
        background-color:red;
    }
    .clear{
        width: 200px;
        height: 200px;
        background-color: yellow;
        clear:both;
    }
</style>
</head>
<body>
    <div>
        <ul>
            <li>python</li>
            <li>web</li>
            <li>linux</li>
        </ul>
    </div>
    <div class="clear"></div>
</body>
</html>
```

例子

python

web

linux



## 伪元素清除法(常用):

给浮动子元素的父盒子，也就是不浮动元素，添加一个clearfix的类，然后设置：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        * {
            padding: 0;
            margin: 0;
        }
        ul{
            list-style: none;
        }
        div{
            width: 400px;
        }
        div ul li{
            float: left;
            width: 100px;
        }
    </style>

```

```
height: 40px;
background-color:red;
}
.clear{
width: 200px;
height: 200px;
background-color: yellow;
/*clear:both;*/
}
.clearfix:after{
content: '.';
height: 0;
clear:both;
display: block;
visibility: hidden;
}

```

</style>

</head>

<body>

```
<div class="clearfix">
<ul>
<li>python</li>
<li>web</li>
<li>linux</li>
</ul>
</div>
<div class="clear"></div>

```

</body>

</html>

例子

python

web

linux

## overflow:hidden(常用):

overflow属性规定当内容溢出元素框时，发生的事情。

属性值：

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利查看其余内容。

inherit：规定应该从父元素继承overflow属性的值。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        body{
            overflow: scroll;
        }
        div{
            width: 100px;
            height: 100px;
```

```
        overflow: inherit;  
    }  
  </style>  
  
</head>  
<body>  
  <div>顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋  
清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋  
顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋  
顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋  
顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋顾清秋  
</div>  
</body>  
</html>
```

例子

7 定位

定位

相对比浮动和标准流  
定位会更加的自♂由  
可以非常自如的把元素放在整个页面的  
任意区域。

```
<style type="text/css">
#box {
    float: left;
    width: 200px;
    height: 200px;
    background-color: red;
}
</style>
```



position: 定位模式；

常用定位模式：

1.relative：相对定位

2.absolute：绝对定位

3.fixed：固定定位

# 定位语法

```
<html>
<head>
<style type="text/css">
h2.pos_abs
{
position:absolute;
left:100px;
top:150px;
}
</style>
</head>

<body>
<h2 class="pos_abs">这是带有绝对定位的标题</h2>
<p>通过绝对定位，元素可以放置到页面上的任何位置。下面的标题距离页面左侧 100px，距离页面顶部 150px。</p>
</body>

</html>
```

通过绝对定位，元素可以放置到页面上的任何位置。下面的标题距离页面左侧 100px，距离页面顶部 150px。

这是带有绝对定位的标题

# 定位模式

static 静态定位

relative 相对定位

absolute 绝对定位

fixed 固定定位

sticky 粘性定位

# 常用的定位模式

## 相对定位

`position: relative;`  
以定位元素的原始位置为  
基准，最后进行偏移。

## 绝对定位

`position: absolute;`  
根据最近一级带有定位模  
式的父元素为基准，进行  
偏移。

## 固定定位

`position: fixed;`  
属于特殊的绝对定位  
无视父元素是否有定位  
只会按照浏览器视口为基  
准偏移

## 相对定位

# 相对定位

position: relative;

1. 相对定位会根据该定位元素原始位置为基准，进行偏移

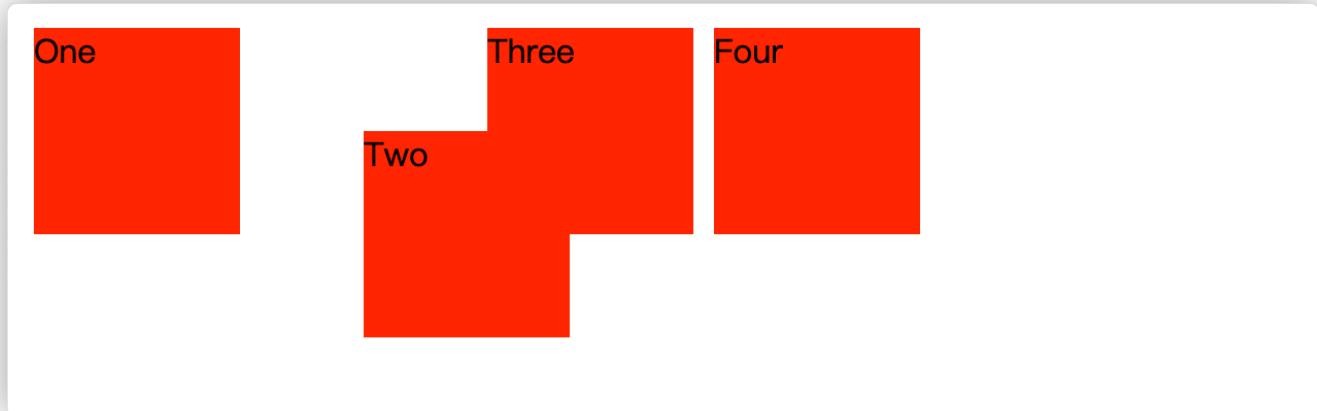
2. 相对定位的元素不会脱离文档流，会保留自身的原始位置。

相对定位元素不可层叠，依据left、right、top、bottom等属性在正常文档流中偏移自身位置。同样可以用z-index分层设计。

```
<head>
  <style type="text/css">
    .box {
      background: red;
      width: 100px;
      height: 100px;
      float: left;
      margin: 5px;
    }
    .two {
      position: relative;
      top: 50px;
      left: 50px;
    }
  </style>
</head>
<body>
```

```
<div class="box" >One</div>
<div class="box two" >Two</div>
<div class="box" >Three</div>
<div class="box">Four</div>
</body>
```

将class="two"的div定位到距离它本来位置的顶部和左侧分别50px的位置。不会改变其他元素的布局，会在此元素本来位置留下空白。



## 绝对定位

---

# 绝对定位

`position: absolute;`

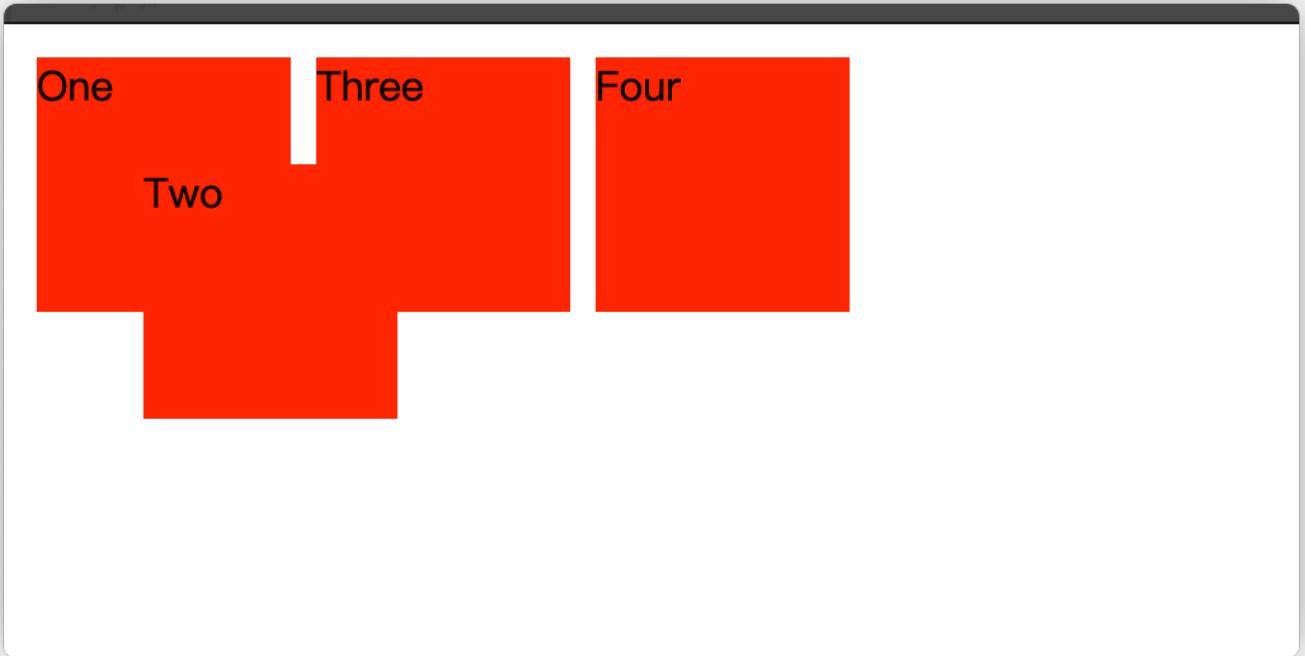
1. 绝对定位元素会根据最近一级带有定位的父元素为基准，进行偏移

2. 绝对定位的元素会脱离文档流，不会保留自身原始位置。

绝对定位的元素从文档流中拖出，使用left、right、top、bottom等属性相对于其最接近的一个最有定位设置的父级元素进行绝对定位，如果元素的父级没有设置定位属性，则依据 body 元素左上角作为参考进行定位。绝对定位元素可层叠，层叠顺序可通过 z-index 属性控制，z-index值为无单位的整数，大的在上面，可以有负值。  
绝对定位的定位方法：如果它的父元素设置了除static之外的定位，比如position:relative或position:absolute及position:fixed，那么它就会相对于它的父元素来定位，位置通过left , top ,right ,bottom属性来规定，如果它的父元素没有设置定位，那么就得看它父元素的父元素是否有设置定位，如果还是没有就继续向更高层的祖先元素类推，总之它的定位就是相对于设置了除static定位之外的定位的第一个祖先元素，如果所有的祖先元素都没有以上三种定位中的其中一种定位，那么它就会相对于文档body来定位（并非相对于浏览器窗口，相对于窗口定位的是fixed）。

```
<head>
<style type="text/css">
.box {
background: red;
width: 100px;
height: 100px;
float: left;
margin: 5px;
}
.two {
```

```
position: absolute;
top: 50px;
left: 50px;
}
</style>
</head>
<body>
<div class="box" >One</div>
<div class="box two" >Two</div>
<div class="box" >Three</div>
<div class="box">Four</div>
</body>
```



## 固定定位

---

# 固定定位

`position: fixed;`

1. 固定定位的元素只会以浏览器视口为基准，无视任何祖先元素。

2. 固定定位的元素会脱离文档流，不会保留自身原始位置。



CSS中的定位使用来布局的熟练应用对页面美化有很好的帮助，下面就进行详细介绍：定位分为静态定位，相对定位，绝对定位，固定定位这四种，定位有不同的参数，例如：`left`、`right`、`top`、`bottom`、`z-index`等。

固定定位与绝对定位类似，但它是相对于浏览器窗口定位，并且不会随着滚动条进行滚动。

固定定位的最常见的一种用途是在页面中创建一个固定头部、固定脚部或者固定侧边栏，不需使用`margin`、`border`、`padding`。

## 1、静态定位（static）

一般的标签元素不加任何定位属性都属于静态定位，在页面的最底层属于标准流。

## 2、绝对定位vs相对定位

绝对定位好像把不同元素安排到了一栋楼的不同楼层(除首层，文本流放在首层)，它们之间互不影响；相对定位元素在首层，与文本流一起存放，它们之间互相影响。

被设置了绝对定位的元素，在文档流中是不占据空间的，如果某元素设置了绝对定位，那么它在文档流中的位置会被删除，它浮了起来，其实设置了相对定位也会让元素浮起来，但它们的不同点在于，相对定位不会删除它本身在文档流中占据的空间，其他元素不可以占据该空间，而绝对定位则会删除掉该元素在文档流中的位置，使其完全从文档流中抽了出来，其他元素可以占据其空间，可以通过z-index来设置它们的堆叠顺序。