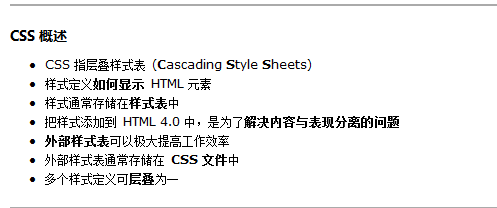
# 1.CSS技术



# 2.CSS初体验

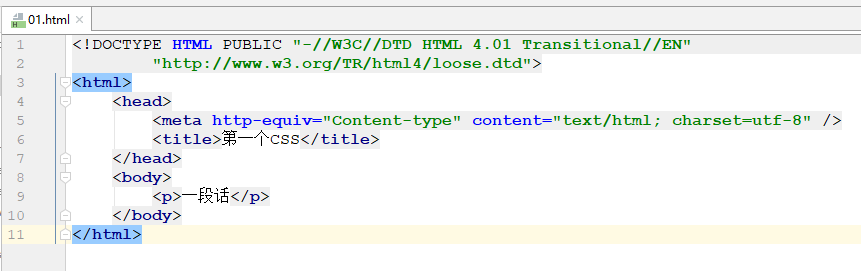
能不能叠加

样式能不能定义在html的外部----相分离

是否能提高工作效率

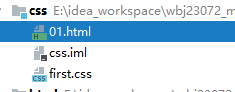
采用CSS实现 给定一段话，加粗、倾斜、变红修饰

定义html文件



html文件中，只包含主体的内容 部分 ---->没有任何修饰 --- >分离、结构简单，便于维护，效率高了

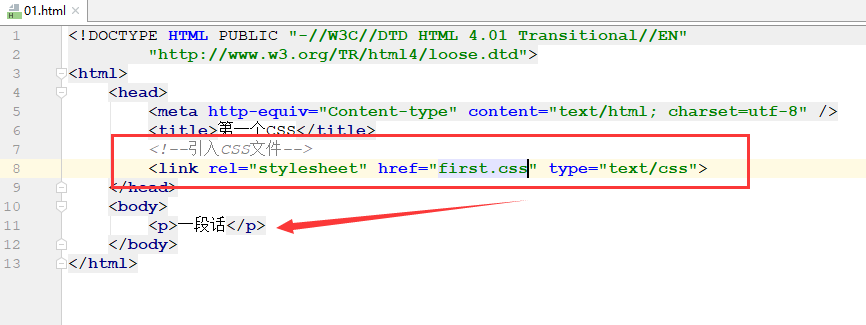
定义一个样式文件 first.css



具体的样式写在first.css的样式表中



html中引入css文件的引用



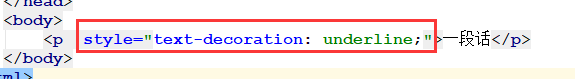
访问

# 3.书写位置

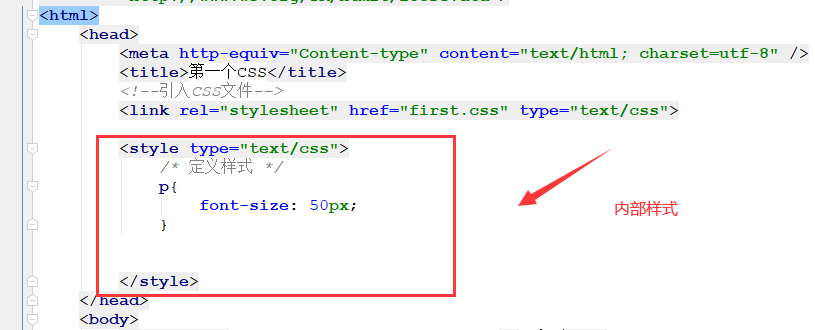
(1)可以写在外部的css文件中，html中引用即可。

(2)行内样式

所有的标签都有一个style属性



(3)内部样式



讨论：

有三种书写样式的位置，怎么用？

行内样式基本不用，因为内容和显示高度耦合

内部样式少用，内容和显示解耦的不彻底，一般测试或该页面中有少量特殊修饰的时候

建议多多的使用外部样式。

优先级的问题(自行验证)

行内样式 > 内部样式 > 外部样式

# 4.样式的语法

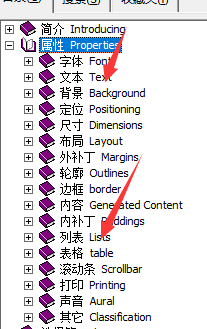
选择器{

属性 : 属性值 ;

属性 : 属性值 ;

}

样式中有哪些属性和属性值？要学

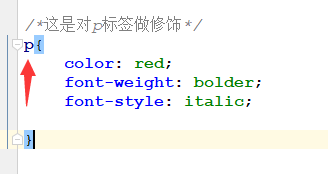


选择器：代表被修饰的对象

# 5.选择器

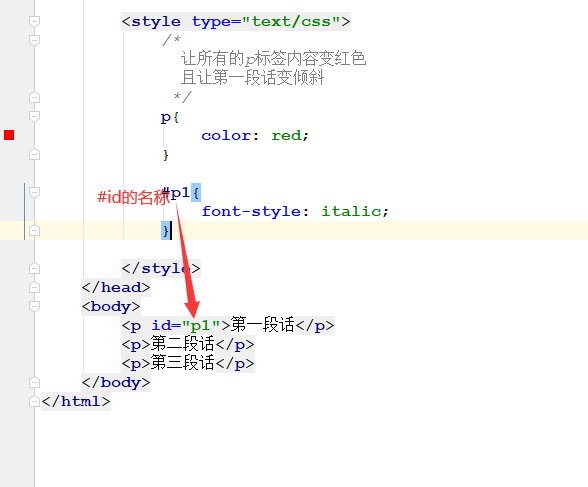
选择器有很多种

## 5.1 标签选择器



对所有的标签都有效

## 5.2 ID选择器



## 5.3 类选择器



改造



任务：自学其他的样式选择器





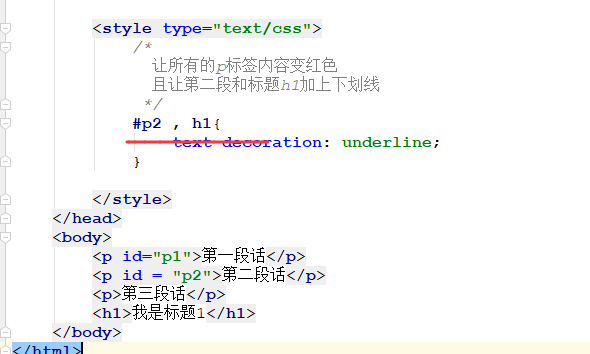
## 5.4 复合选择器

由标签选择器、ID选择器、类选择器组合而来。

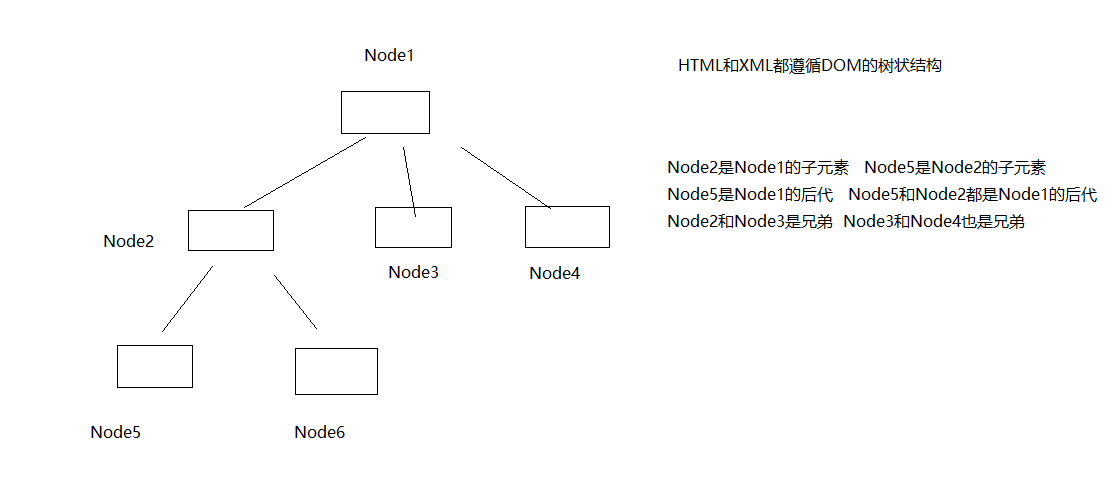
分组选择器:多个选择器使用相同的样式

选择器1,选择器2,…..选择器N{

}



树状结构



子元素选择器: A的子元素B

选择器A > 选择器B{

}



后代选择器：选择器1的所有后代元素选择器2

选择器1 选择器2{

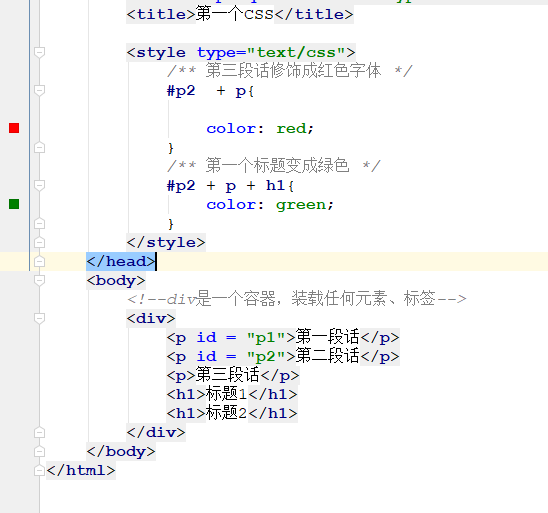
}



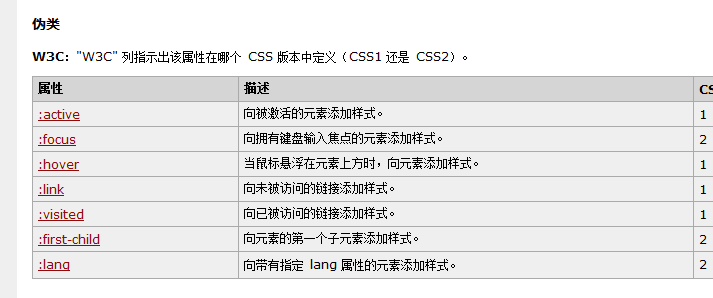
相邻的兄弟选择器 : 紧接着1的后一个兄弟2

选择器1 + 选择2 {

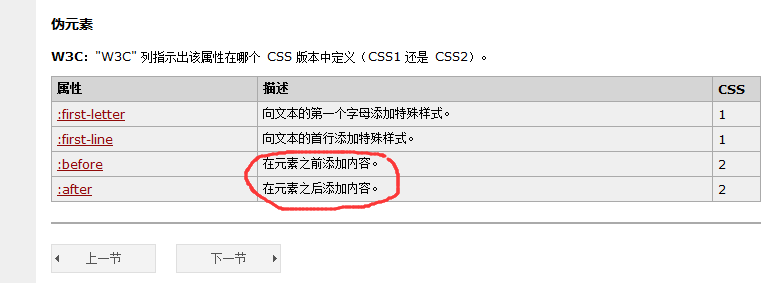
}

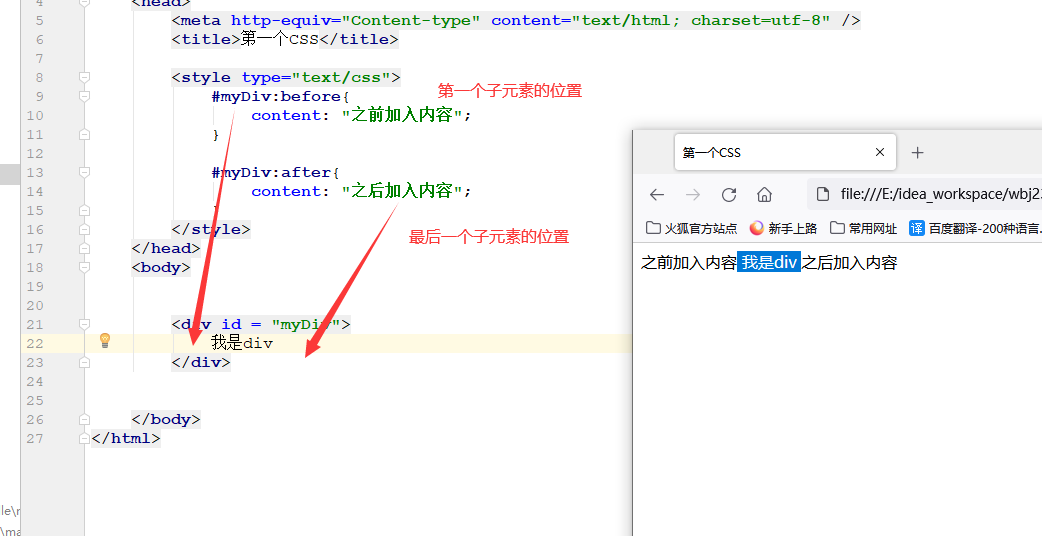


## 5.5伪类/伪元素









## 5.6 总结

ID选择器很少使用，类选择器被大量使用，其他的选择器灵活使用。

类样式的命名 .单词-单词

# 6.具体的样式

## 6.1 字体



## 6.2 文本



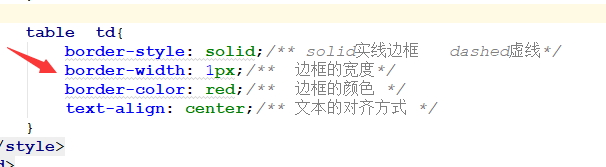


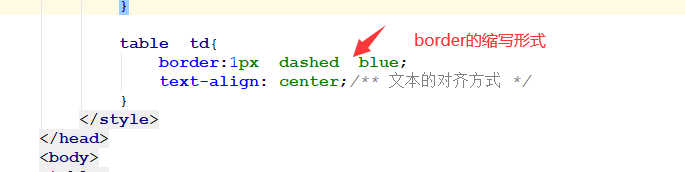
## 6.3 边框

有四个方向

上(top) 右(right) 下(bottom) 左(left)

如果不指定方向，则代表四个方向





## 6.4 表格

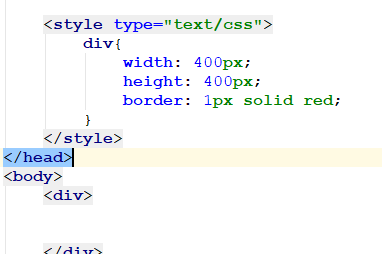
border-spacing :0

border-collapse : collapse

记住：修饰table的，所以要写在表格中

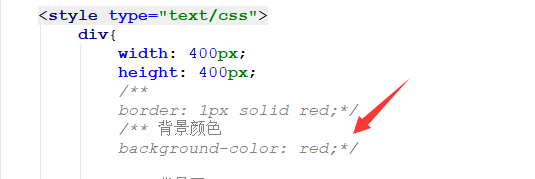


## 6.5 尺寸

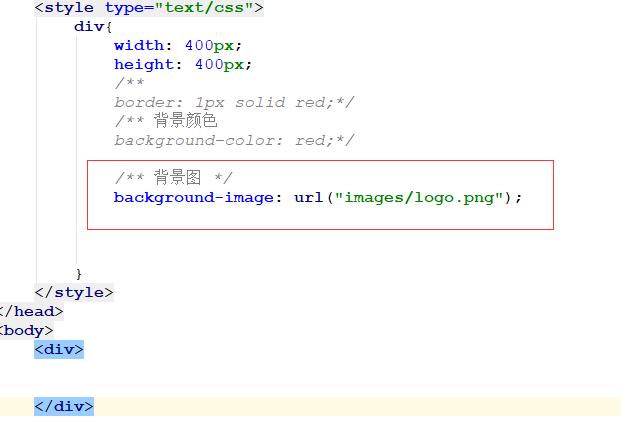


## 6.6 背景

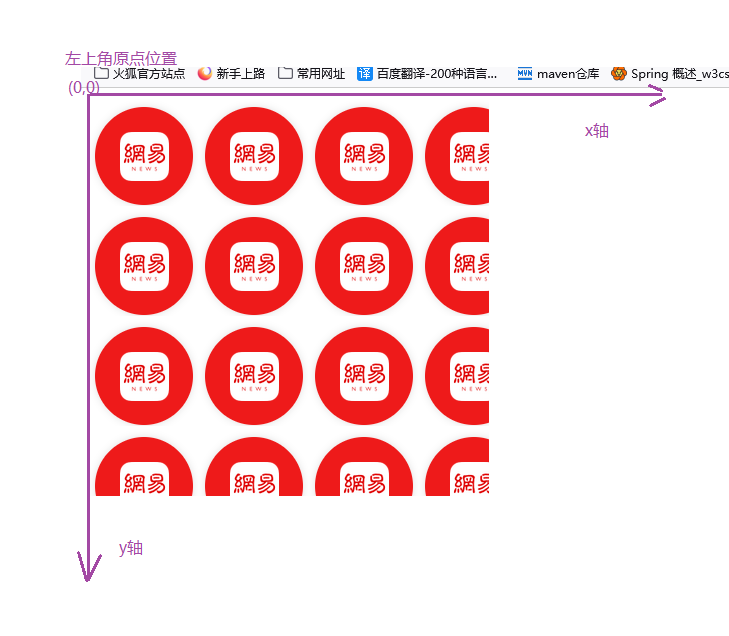
设置背景色：



设置背景图片



效果



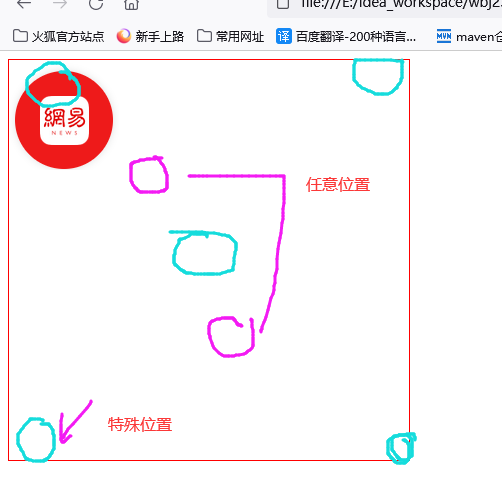
分析：当背景图片过小时，会在x轴和y轴产生平铺效果

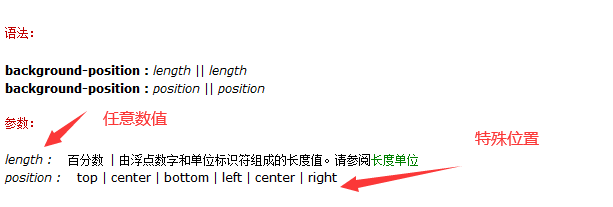
如果取消平铺？如何取消X轴平铺？如何取消Y轴平铺

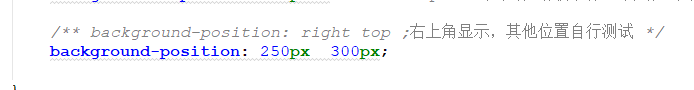


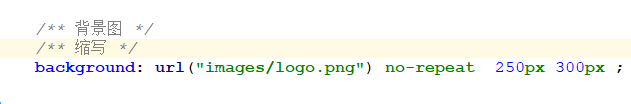


背景图的偏移技术









## 6.7 背景图偏移技术应用

雪碧图/精灵图

将很多个小图片合成为一张大图，为了减少和服务器的交互次数，提高性能。



获取 ”强”这个小背景

思路：

(1)估算出小背景的尺寸 28px \* 28px

(2)使用精灵图做背景图片

(3)偏移 x轴和y轴的正负性 - -

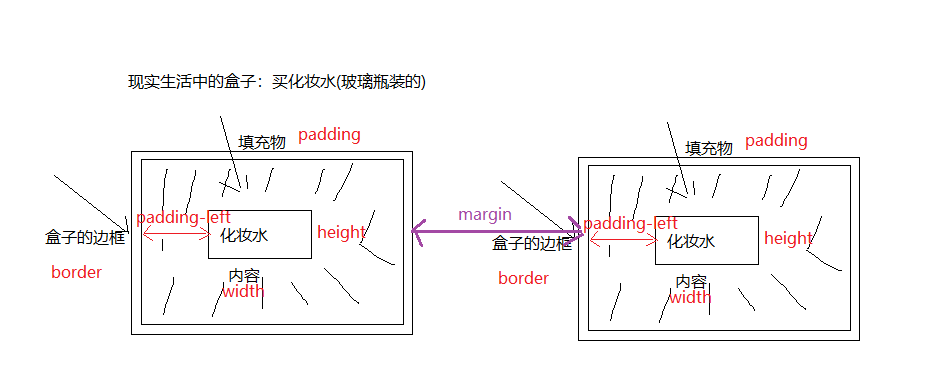


## 6.8 盒子模型

在网页布局中，将所有的标签都当成一个盒子(矩形)，布局的本质就是盒子与盒子之间的相对位置。



到底什么是盒子模型？



盒子之间的边框 ：border 四个方向

盒子内部的填充物：内补丁/内边距 padding 四个方向

盒子与盒子之间的距离：外补丁/外边距 margin表示 四个方向

盒子自身的内容也有空间：宽 width 高height

所以：网页上的一个盒子占浏览器 真实的空间

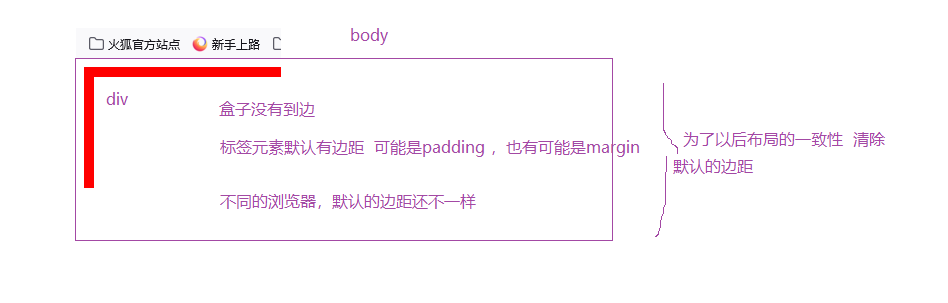
宽 = border-left + padding-left + width + padding-right + border-right

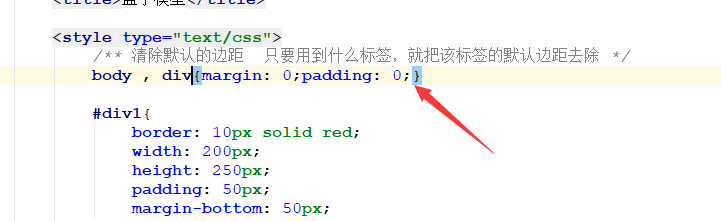
高 = border-top + padding-top + height + padding-bottom + border-bottom

演示盒子模型



仔细分析：





## 6.9 盒子模型的应用

(1)制作漂亮的表格



(2)制作漂亮的按钮

百度的登录按钮



# 7.布局技术

## 7.1 默认布局

默认布局遵从文档流布局。

三个规则：

(1)标签被浏览器自上而下显示

(2)遇到块级元素，独占一行

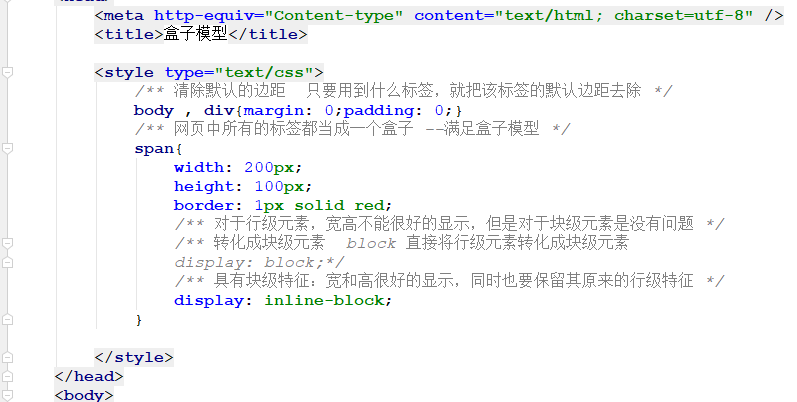
(3)遇到行级元素，在一行中共存，直到浏览器的右侧边缘时会自动换行。

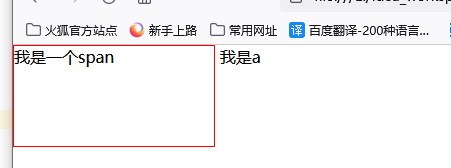
## 7.2 行级/块级转化

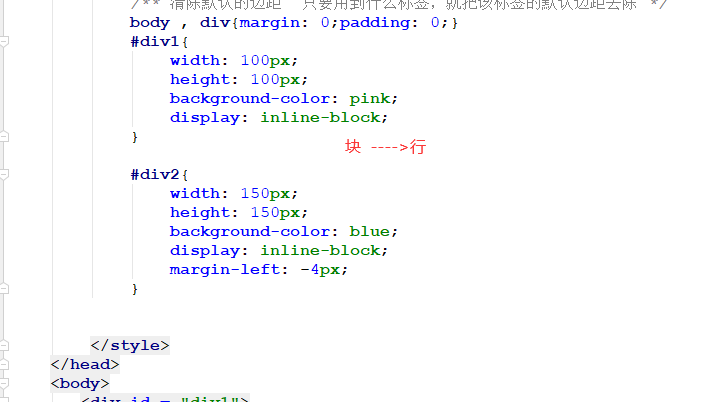
按照默认的规则，两者是不同的元素

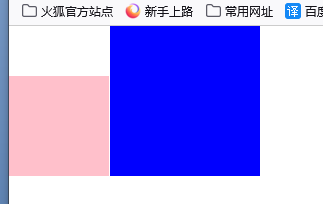
要想能转化，则必须打破默认的文档流布局，可以借助display

行 ----》块

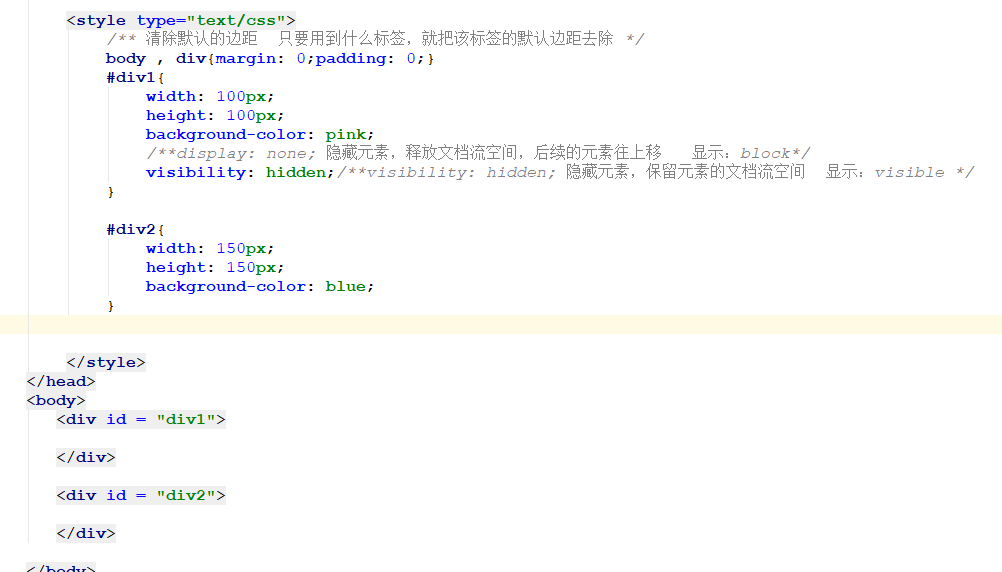




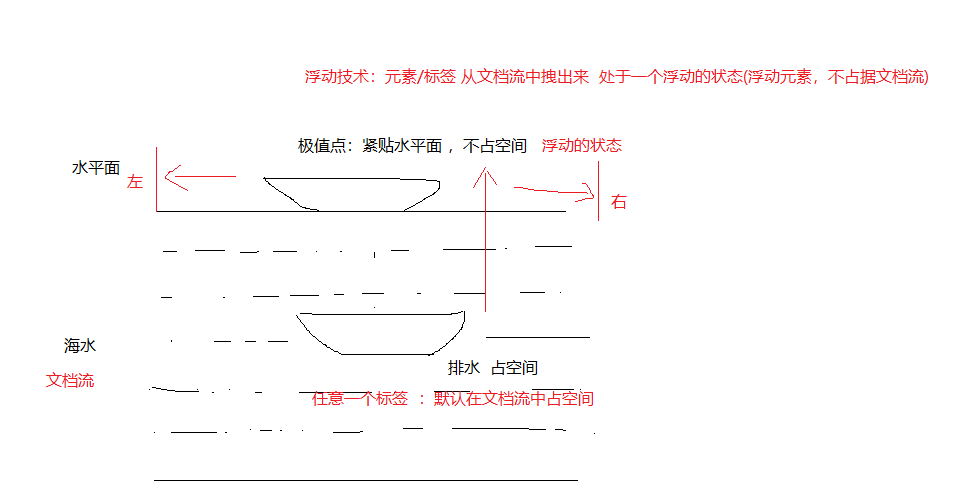




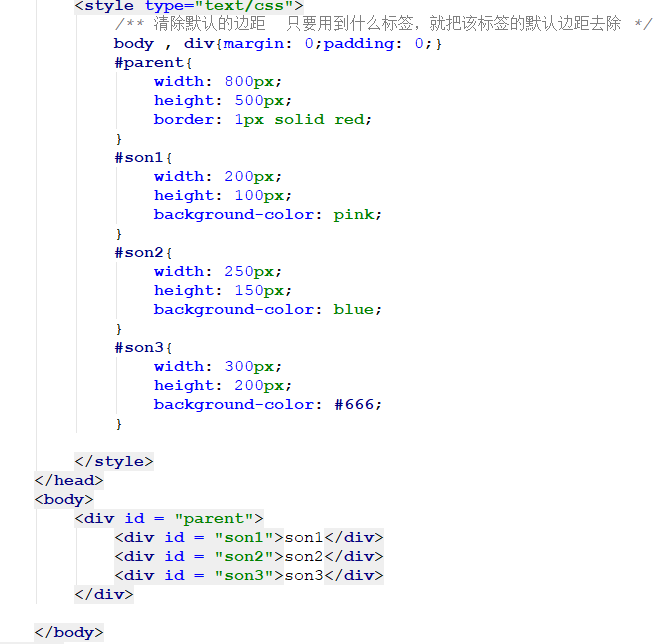
## 7.3 显示和隐藏

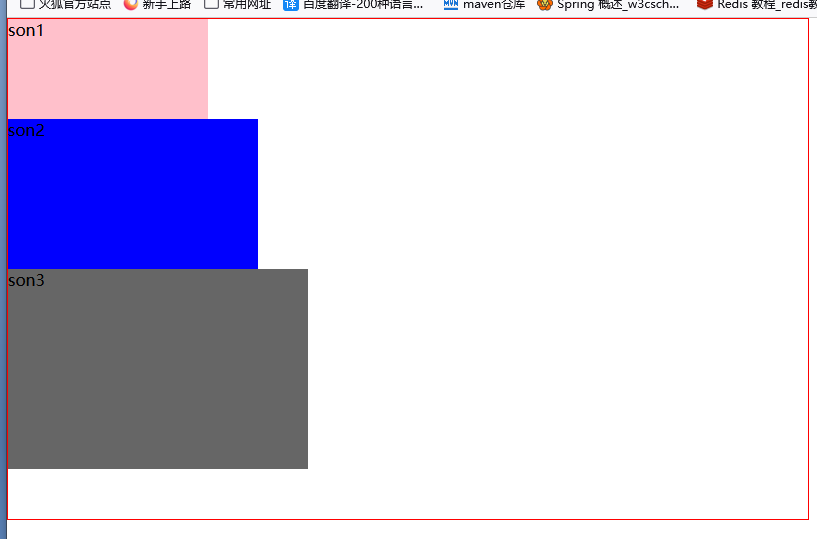


## 7.4 浮动技术

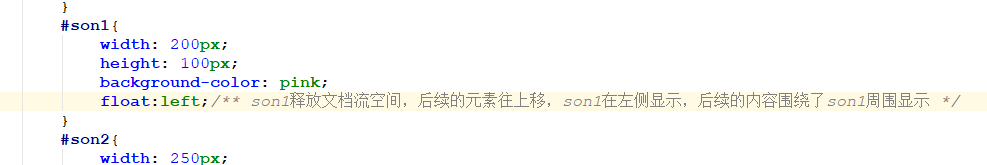


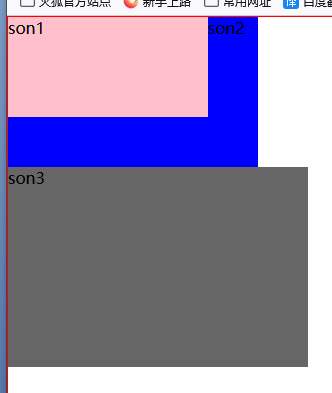
原始素材



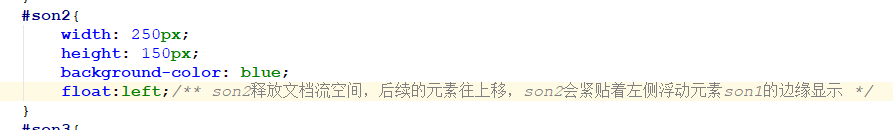


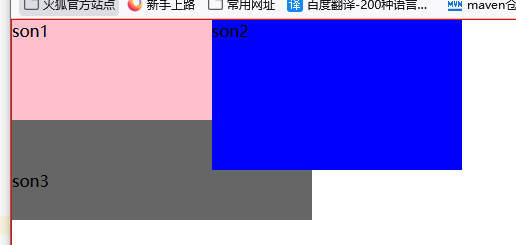
(1)让son1变成浮动元素，往左浮动



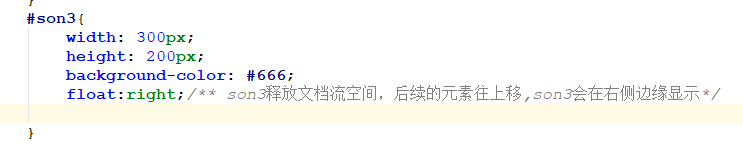


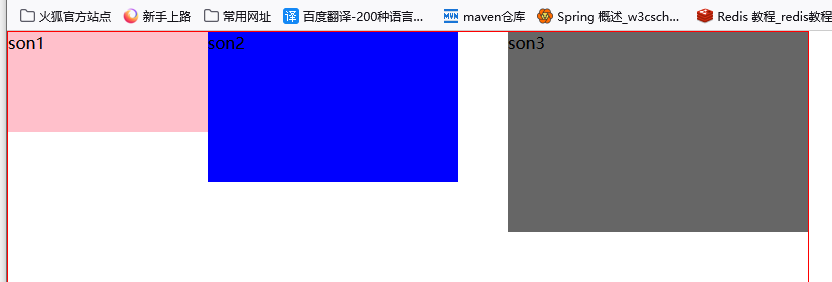
(2)再将son2设置为浮动元素，往左浮



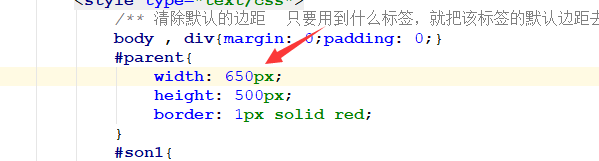


(3)son3也变成浮动元素，往右浮动

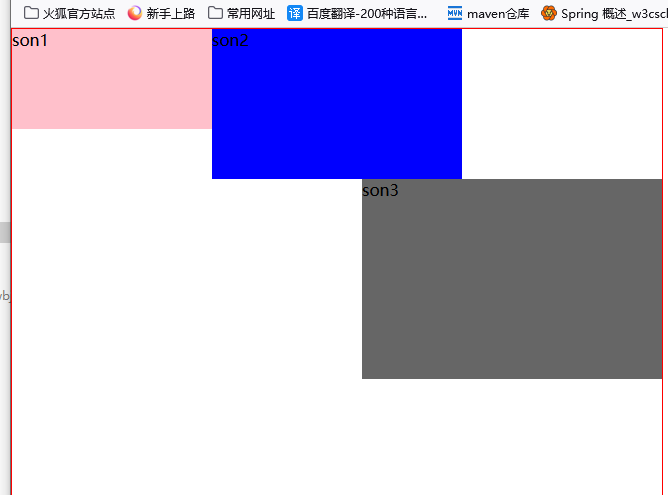




(4)改变父级元素的宽度



当父级元素宽度不够时，浮动元素会自动下移，直到合适的位置。



## 7.5 浮动的塌陷问题

观察一个现象：

元素素材

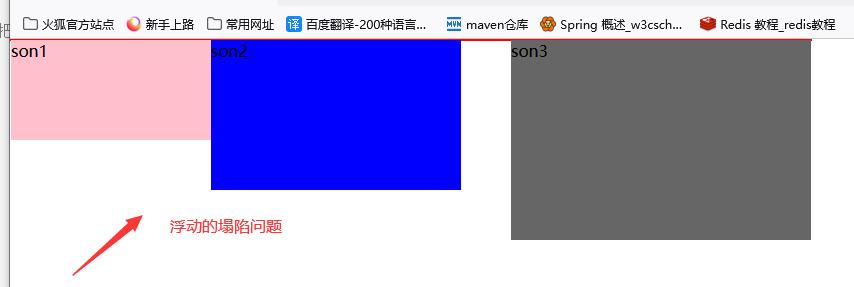


分别为子元素设置浮动

son1为浮动，父容器的高度 = son2的高度 + son3高度

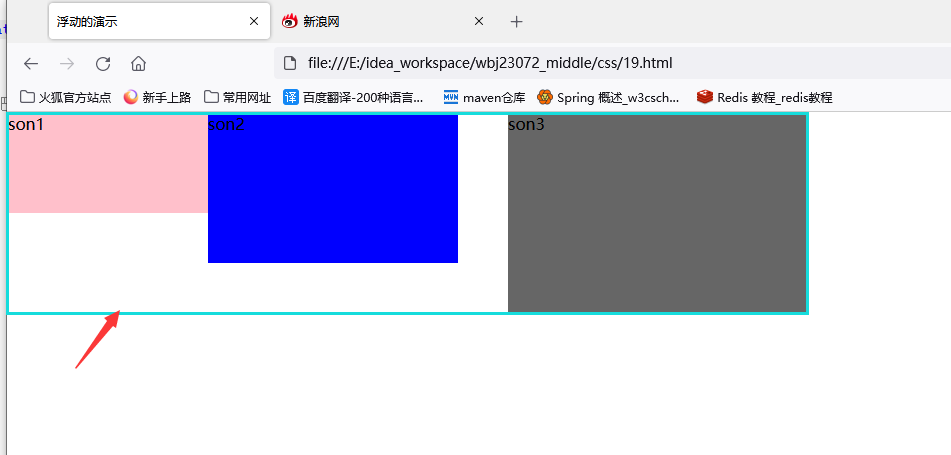
son2为浮动，父容器的高度 = son3高度

son3为浮动，父容器的高度 = 0 ---》退化成一条线



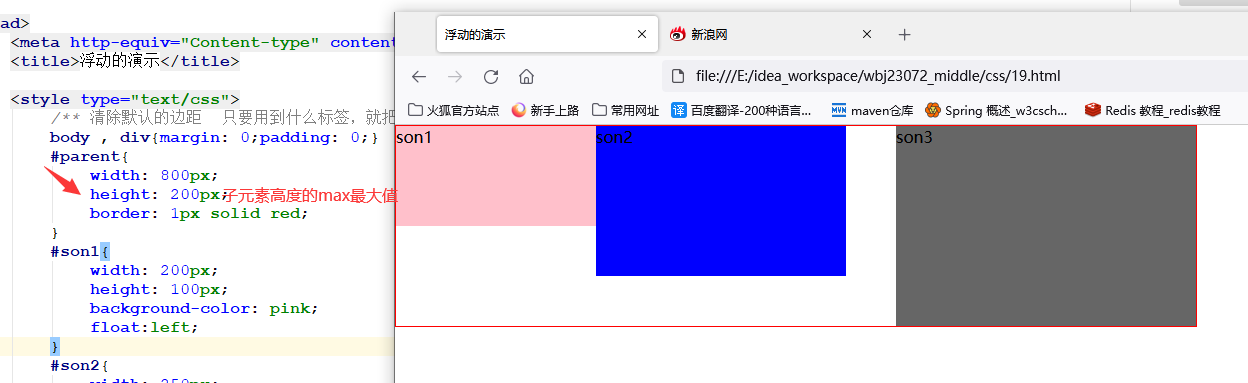
造成塌陷现象的根本原因：浮动带来的负面效果。

期望：父容器的高度不管是否有浮动元素，至少能包裹所有的子元素



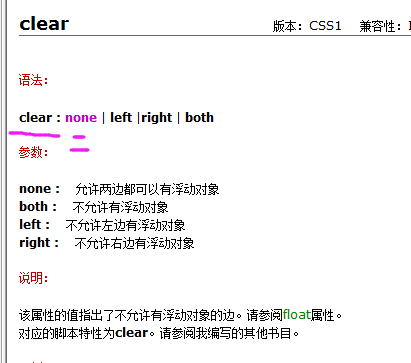
如何解决？

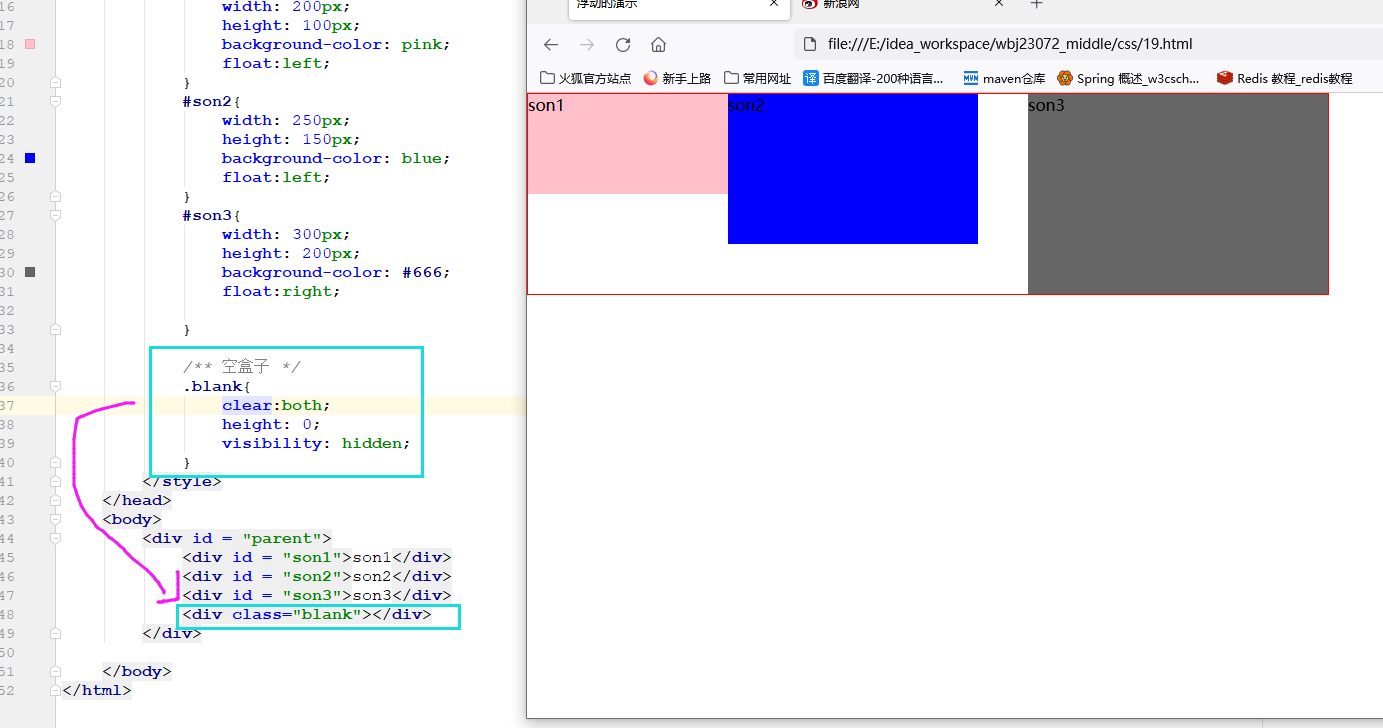
(1)指定父容器的高度



缺点：max最大值需要人为找，且可能会发生变化 ---->去修改父容器的高度

(2)清除浮动的负面效应



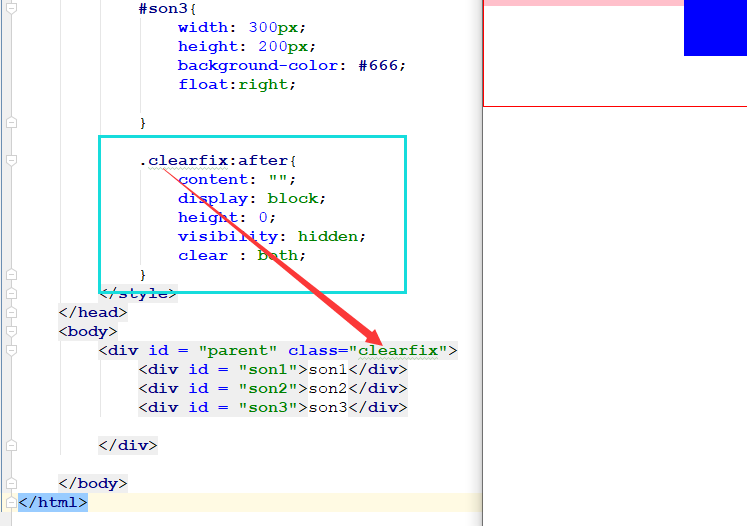


缺点：当网页上多个地方使用浮动时，都需要新增空盒子标签，比较麻烦。

有点像 “静态代理”

(3)动态增加空标签

空标签应该动态增加到父元素的最后一个子元素位置。

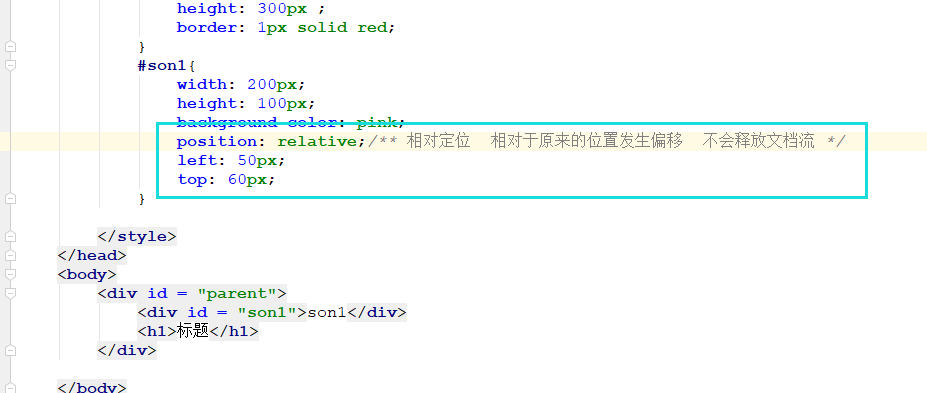


# 8.定位技术

相对定位

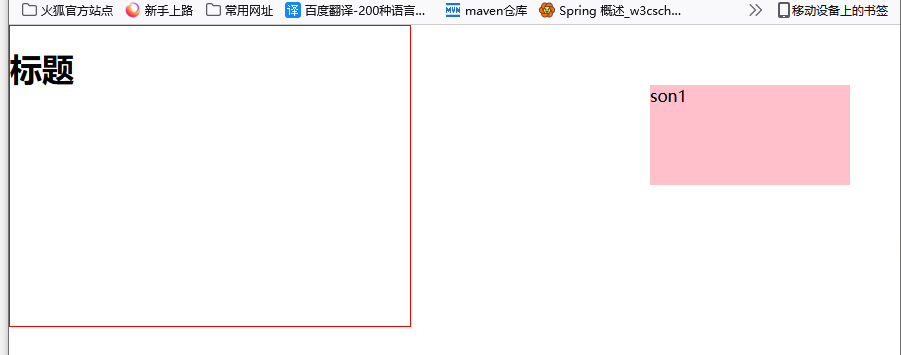
素材





绝对定位





关于绝对定位的参考位置问题：

(1)如果该元素的父元素或祖先元素采用了定位(不管是绝对还是相对定位)，会参考父元素或祖先元素

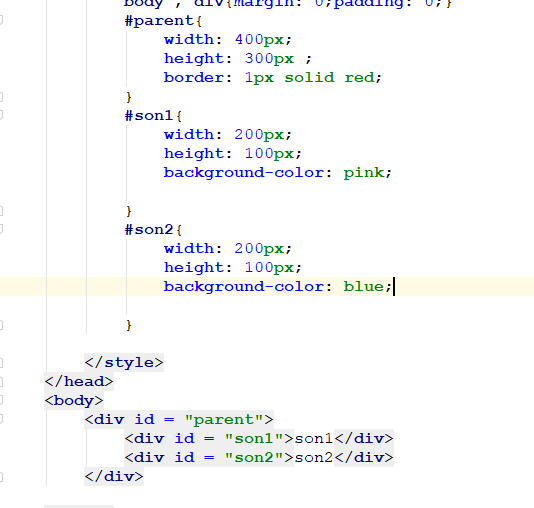


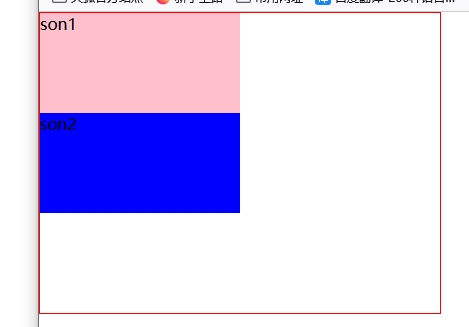




(2)否则参考浏览器

## 9.定位的叠加问题





son1和son2发生偏移 ---->叠加



