CSS 基础入门

第 3 天课堂笔记（本课程共 6 天）

前端与移动开发学院

http://web.itcast.cn

目录

一、上节课知识复习

css 属性，面试的时候会有笔试，笔试没有智能感应的：

● 加粗，倾斜，下划线：

```
font-weight:bold;
font-style:italic;
text-decoration:underline;
```

● 背景颜色、前景色：

```
background-color:red;
color:red;
```

● class 和 id 的区别

class 用于 css 的，id 用于 js 的。

1）class 页面上可以重复。id 页面上唯一，不能重复。

2）一个标签可以有多个 class，用空格隔开。但是 id 只能有 id。

● 选择器

说 IE6 层面兼容的： 标签选择器、id 选择器、类选择器、后代、交集选择器、并集选择器、通配符。

```
p
#box
.spec
div p
div.spec
div,p
*
```

IE7 能够兼容的：儿子选择器、下一个兄弟选择器

```
div>p
h3+p
```

IE8 能够兼容的：

```
ul li:first-child
ul li:last-child
```

● css 两个性质：

1） 继承性。有一些属性给祖先元素，所有的后代元素都集成上了。

哪些属性能继承：color、font-、text-、line-

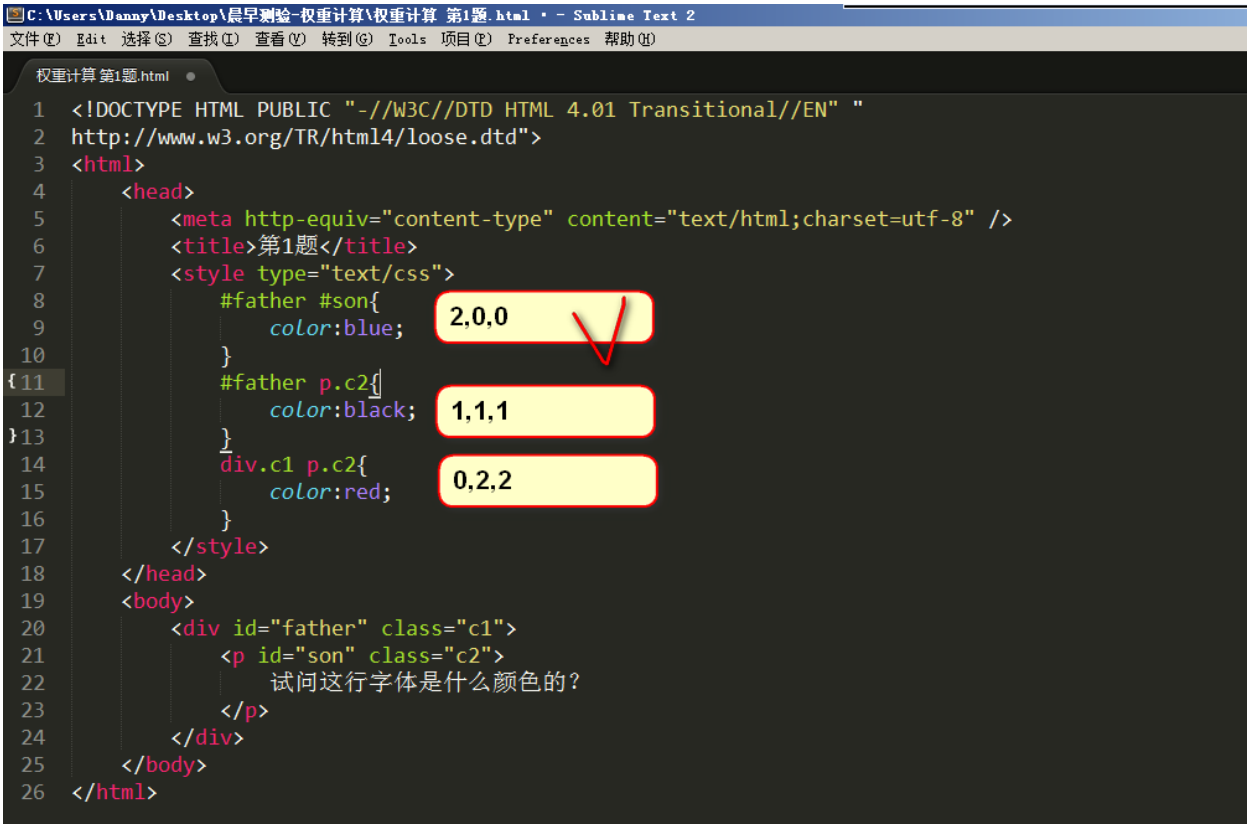
2） 层叠性。层叠性是一种能力，就是处理冲突的能力。当不同选择器，对一个标签的同一个样式，有不同的值，听谁的？这就是冲突。css 有着严格的处理冲突的机制：

■ 选择上了，数权重，(id 的数量，类的数量，标签的数量)。如果权重一样，谁写在后面听谁的。

■ 没有选择上，通过继承影响的，就近原则，谁描述的近听谁的。**如果描述的一样近，比如选择器权重，如果权重再一样重，谁写在后面听谁的。**

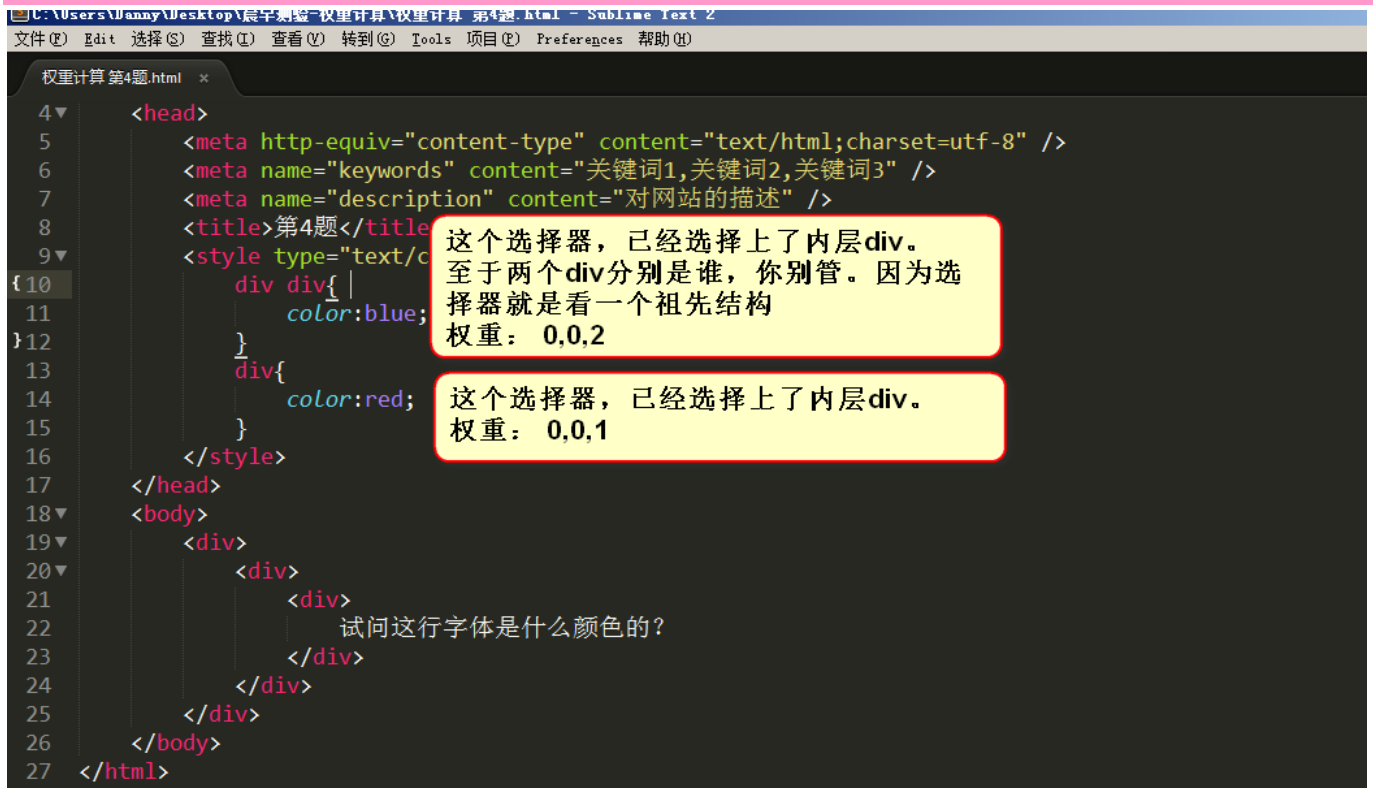
再看几道题目:

第 1 题：



第 2 题：

第 3 题：




```
<head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <meta name="keywords" content="关键词1,关键词2,关键词3" />
    <meta name="description" content="对网站的描述" />
    <title>第4题</title>
    <style type="text/c
        div div{
            color:blue;
        }
        div{
            color:red;
        }
    </style>
</head>
<body>
    <div>
        <div>
            <div>
                试问这行字体是什么颜色的？
            </div>
        </div>
    </div>
</body>
</html>
```

第 4 题：


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <meta http-equiv="Content-Ty                    TF-8">
    <title>Document</title>
    <style type="text/css">
        #box1 div{
            color:red;
        }
        #box3{
            color:blue;
        }
    </style>
</head>
<body>
    <div id="box1" class="c1">
        <div id="box2" class="c2">
            <div id="box3" class="c3">
                文字
            </div>
        </div>
    </div>
</body>
</html>
```

二、权重问题深入

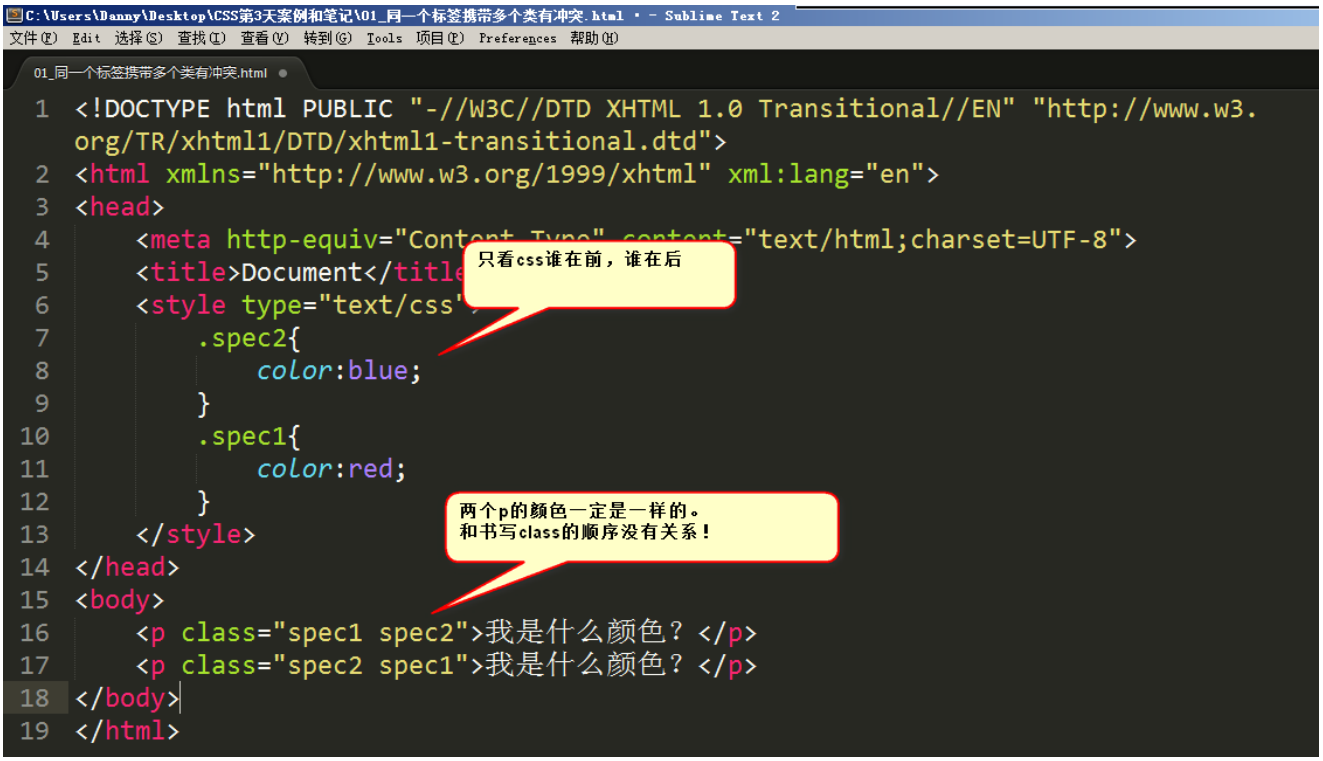
2.1 同一个标签，携带了多个类名，有冲突：

```
<p class="spec1 spec2">我是什么颜色？ </p>
<p class="spec2 spec1">我是什么颜色？ </p>
```

和在标签中的挂类名的书序无关，只和 css 的顺序有关：

```
        .spec2{
            color:blue;
        }
        .spec1{
            color:red;
        }
    </style>
```

红色的。因为 css 中 red 写在后面。




```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
    <title>Document</title>
    <style type="text/css">
        .spec2{
            color:blue;
        }
        .spec1{
            color:red;
        }
    </style>
</head>
<body>
    <p class="spec1 spec2">我是什么颜色？ </p>
    <p class="spec2 spec1">我是什么颜色？ </p>
</body>
</html>
```

2.2 !important 标记

```
    <style type="text/css">
        p{
            color:red !important;
        }
        #para1{
            color:blue;
        }
        .spec{
            color:green;
        }
    </style>
```

important 是英语里面的“重要的”的意思。我们可以通过语法：

```
k:v !important;
```

来给一个属性提高权重。这个属性的权重就是无穷大。

一定要注意语法：

正确的：

```
font-size:60px !important;
```

错误的：

```
font-size:60px; !important;        → 不能把!important 写在外面
font-size:60px important;          →  不能忘记感叹号
```

!important 需要强调 3 点：

1） !important 提升的是一个属性，而不是一个选择器

```
p{
    color:red !important;     → 只写了这一个!important，所以就字体颜色属性提升权重
    font-size: 100px ;        → 这条属性没有写!important，所以没有提升权重
}
#para1{
    color:blue;
    font-size: 50px;
}
.spec{
    color:green;
    font-size: 20px;
}
```

所以，综合来看，字体颜色是 red（听 important 的）；字号是 50px（听 id 的）；

2） !important 无法提升继承的权重，该是 0 还是 0

比如 HTML 结构：

```
<div>
    <p>哈哈哈哈哈哈哈哈</p>
</div>
```

有 CSS 样式：

```
div{
    color:red !important;
}
p{
    color:blue;
}
```

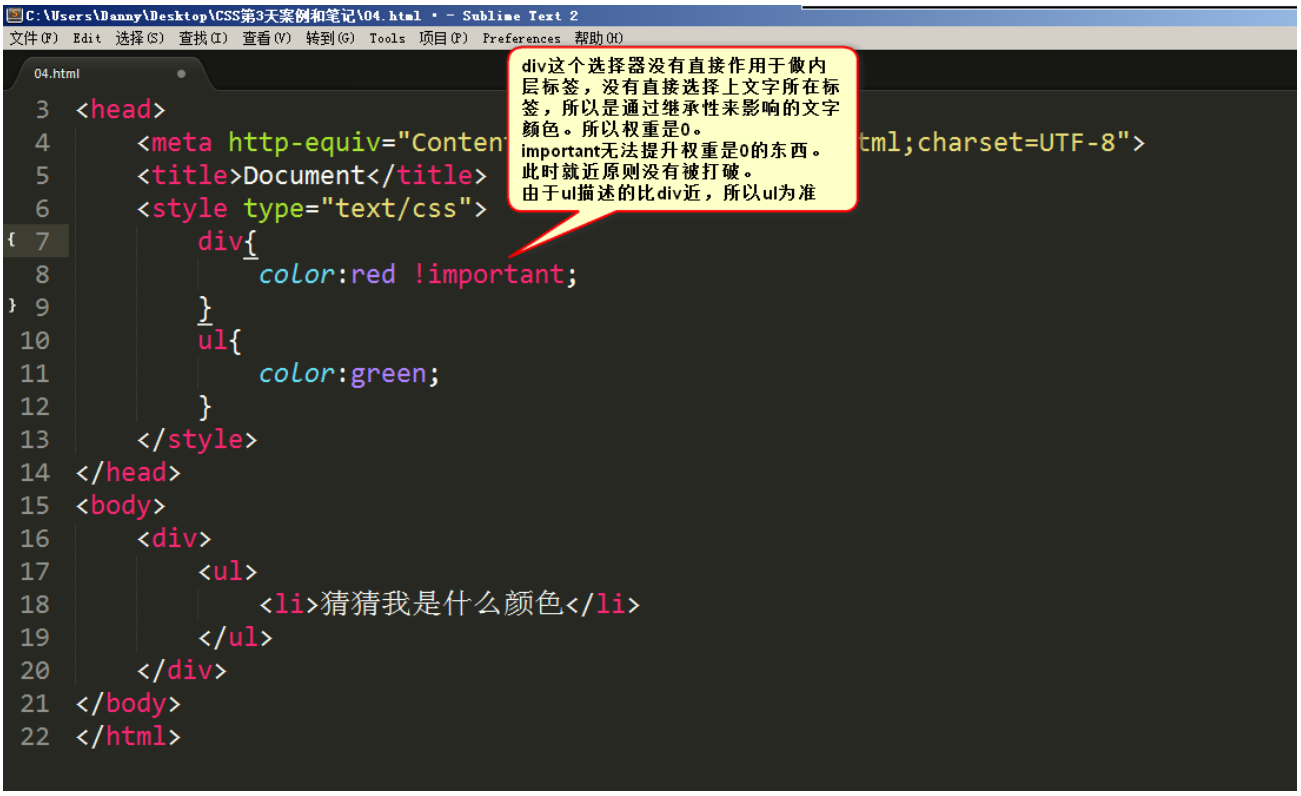
由于 div 是通过继承性来影响文字颜色的，所以!important 无法提升它的权重，权重依然是 0。

干不过 p 标签，因为 p 标签是实实在在选中了，所以字是蓝色的（以 p 为准）。

3）!important 不影响就近原则

如果大家都是继承来的，按理说应该按照“就近原则”，那么 important 能否影响就近原则呢？

答案是：不影响。远的，永远是远的。不能给远的写一个 important，干掉近的。

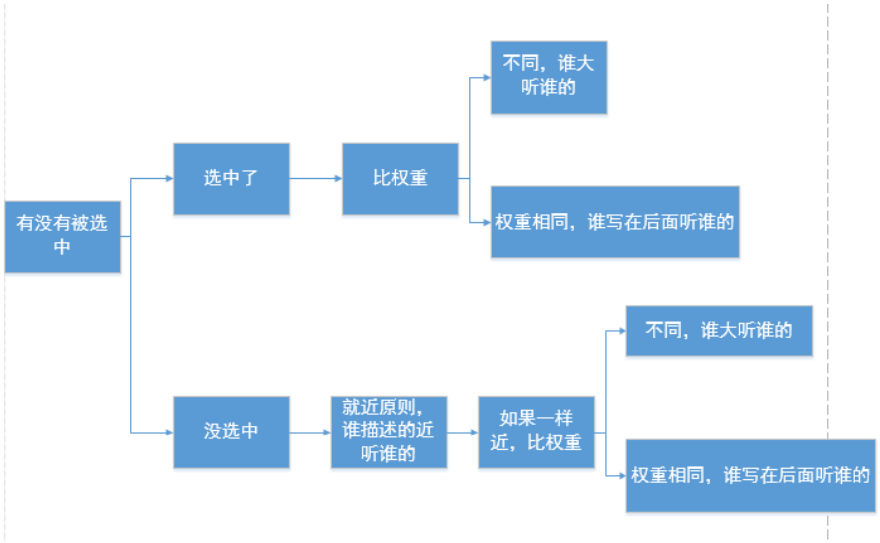


! important 做站的时候，不允许使用。因为会让 css 写的很乱。

现在，我们知道层叠性能比较很多东西：

选择器的写法权重，谁离的近，谁写在下面。

2.3 权重计算的总结



还要知道　! important 的性质。

三、盒模型

3.1 盒子中的区域

一个盒子中主要的属性就 5 个：width、height、padding、border、margin。

width 是“宽度”的意思，CSS 中 width 指的是内容的宽度，而不是盒子的宽度。

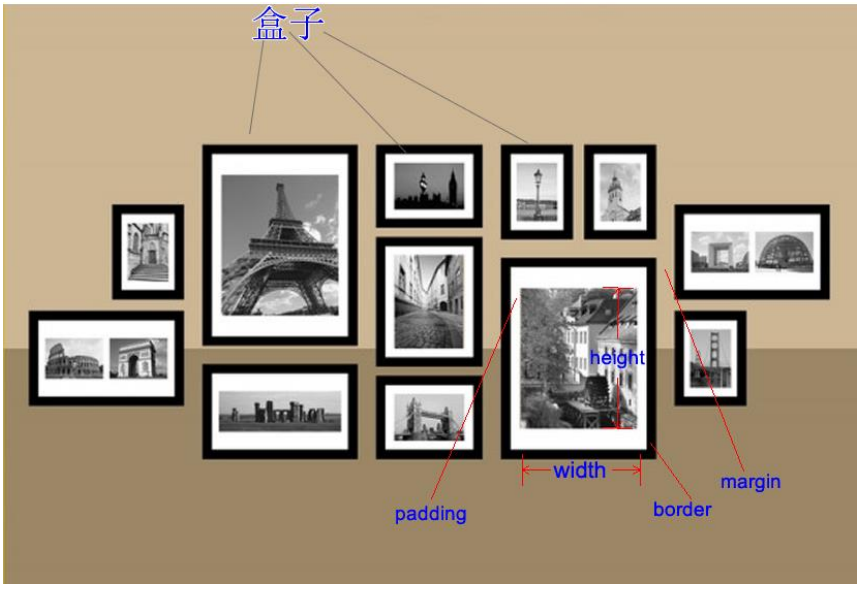
height 是“高度”的意思，CSS 中 height 指的是内容的高度，而不是盒子的高度

padding 是“内边距”的意思

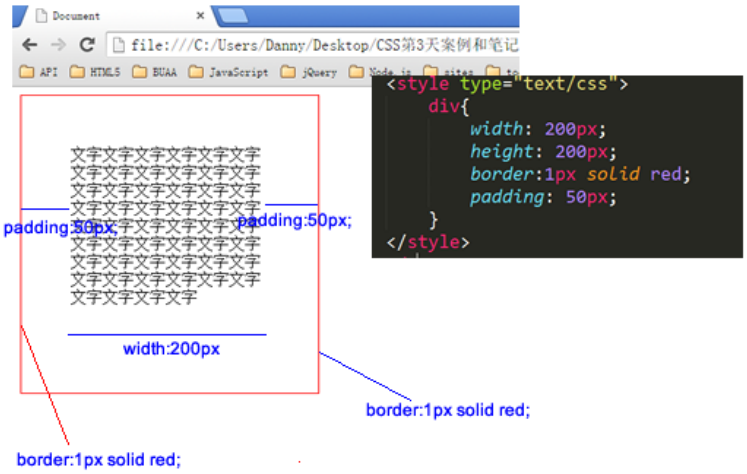
border 是“边框”

margin 是“外边距”

盒模型的示意图：



代码演示：



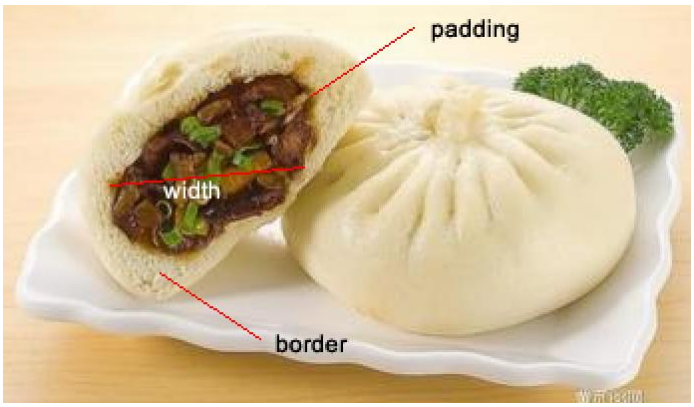
这个盒子 width:200px; height:200px; 但是真实占有的宽高是 302*302。 这是因为还要加上 padding、border。

宽度和真实占有宽度，不是一个概念！！

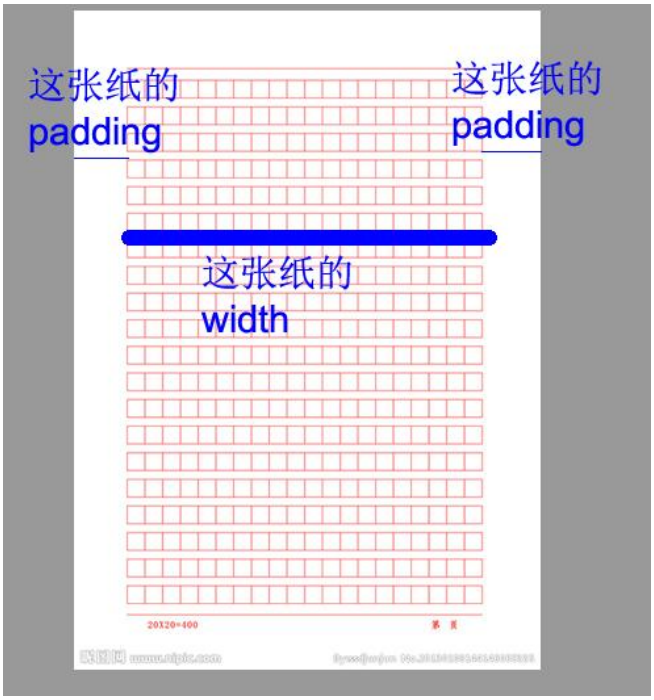
3.2 认识 width、height

一定要知道，在前端开发工程师眼中，世界中的一切都是不同的：

比如丈量一个包子多宽？前端开发工程师，只会丈量包子馅：



丈量稿纸，前端开发工程师只会丈量内容宽度：



丈量人脸，只会丈量五官：

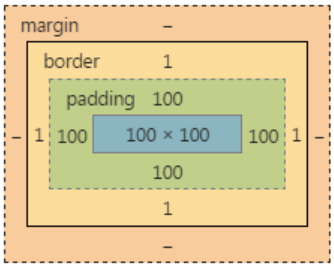
下面这两个盒子，真实占有宽高，完全相同，都是 302*302：

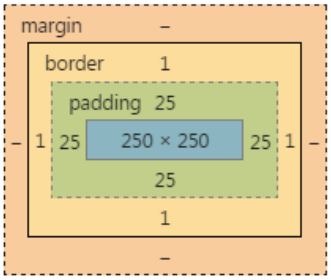
```
.box1{
    width: 100px;
    height: 100px;
    padding: 100px;
    border: 1px solid red;
}

.box2{
    width: 250px;
    height: 250px;
    padding: 25px;
    border:1px solid red;
}
```

真实占有宽度=　左 border　+　左 padding　+　width　+　右 padding　+　右 border

这两个盒子的盒模型图，见下表：



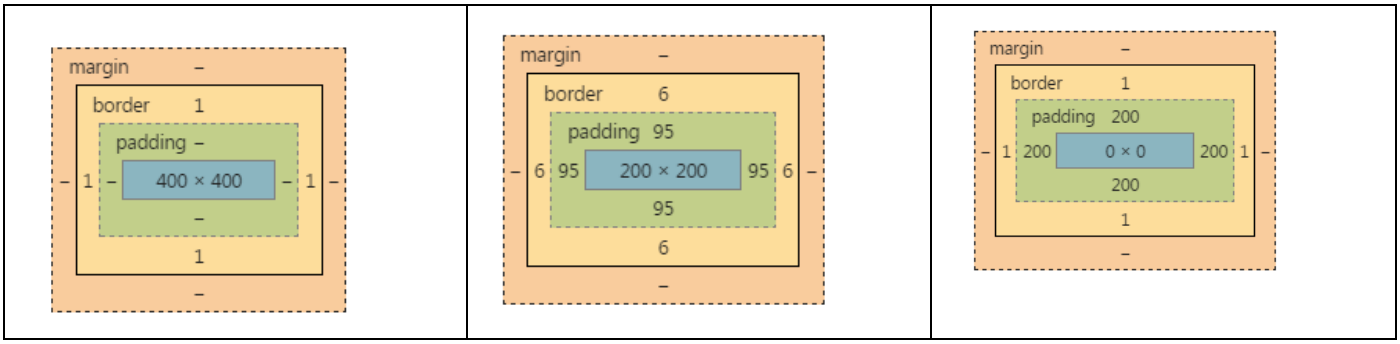


小练习，大家自己写三个 402*402 的盒子

答案（答案有无穷多种，我们只写其中三种）：

```
.box1{
    width: 400px;
    height: 400px;
    border: 1px solid red;
}
.box2{
    width: 200px;
    height: 200px;
    border: 6px solid red;
    padding: 95px;
}
.box3{
    width: 0px;
    height: 0px;
    padding: 200px;
    border: 1px solid red;
}
```

这三个盒子的盒模型图：

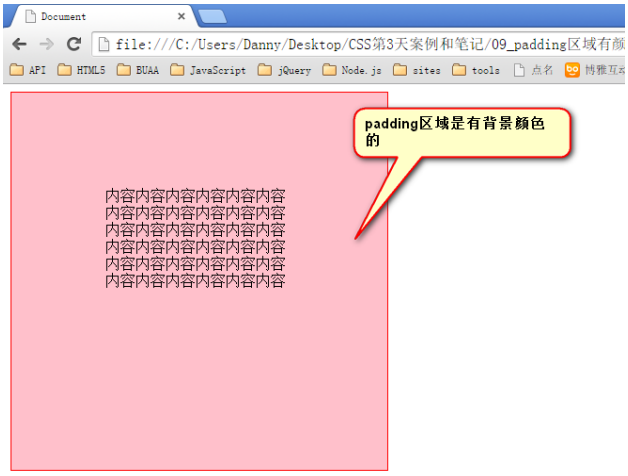


如果想保持一个盒子的真是占有宽度不变，那么加 width 就要减 padding。加 padding 就要减 width。

3.3 认识 padding

padding 就是内边距。padding 的区域有背景颜色，css2.1 前提下，并且背景颜色一定和内容区域的相同。

也就是说，background-color 将填充所有 boder 以内的区域。



padding 是 4 个方向的，所以我们能够分别描述 4 个方向的 padding。

方法有两种，第一种写小属性；第二种写综合属性，用空格隔开。

小属性：

```
1    padding-top: 30px;
2    padding-right: 20px;
3    padding-bottom: 40px;
4    padding-left: 100px;
```

top 上、right 右、bottom 下、left 左。

这种属性，就是复合属性。比如不写 padding-left 那么就是没有左内边距。

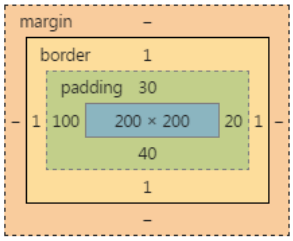
快捷键就是 pdt、pdr、pdb、pdl 然后按 tab。

综合属性：

如果写了 4 个值：

```
1    padding:30px 20px 40px 100px;
```

上、右、下、左



空格隔开的，四个数字就是上、右、下、左。

也就是说，前端开发工程师眼中的顺序不一样。

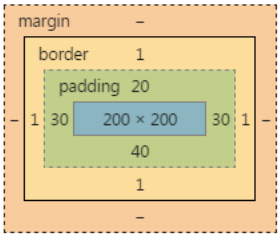
老百姓：上下左右

强调开发工程师：上、右、下、左

如果只写 3 个值：

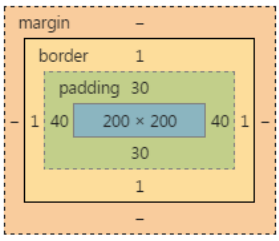
```
padding: 20px 30px 40px;
```

上、右、下、??和右一样



如果只写 2 个值：

```
padding: 30px 40px;
```



也就是说，

```
padding: 30px 40px;
```

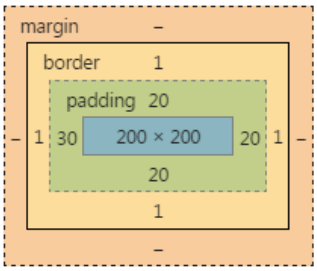
等价于：

```
padding-top: 30px;
padding-bottom: 30px;
padding-left: 40px;
padding-right: 40px;
```

要懂得，用小属性层叠大属性：

```
padding: 20px;
padding-left: 30px;
```

对应的盒模型图：



下面的写法错误：

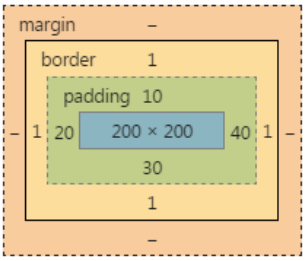
```
padding-left: 30px;
padding: 20px;
```

不能把小属性，写在大属性前面。

下面的题，会做了，说明你听懂了：

题目 1，说出下面盒子真实占有宽高，并画出盒模型图：

```
div{
    width: 200px;
    height: 200px;
    padding: 10px 20px 30px;
    padding-right: 40px;
    border: 1px solid #000;
}
```

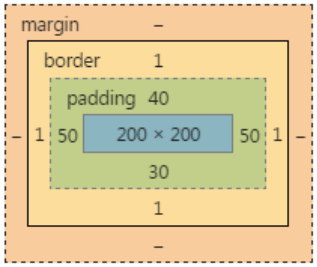


真实占有宽度 = 200 + 20 + 40 + 1 + 1 = 262px

题目 2，说出下面盒子真实占有宽高，并画出盒模型图：

```
div{
    width: 200px;
    height: 200px;
    padding-left: 10px;
    padding-right: 20px;
    padding:40px 50px 60px;
    padding-bottom: 30px;
    border: 1px solid #000;
}
```

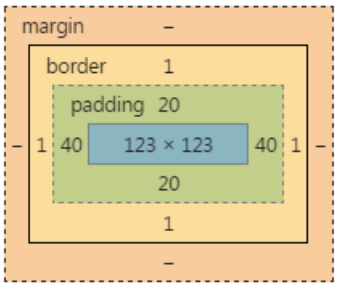
padding-left:10px；　和 padding-right:20px；没用，因为后面的 padding 大属性，层叠掉了他们。



强调一点，padding-left 不是 padding-left-width

```
padding-left:10px;    √
padding-left-width:30px;  ×
```

第 3 题，我现在给你盒模型图，请写出代码，试着用最最简单的方法写



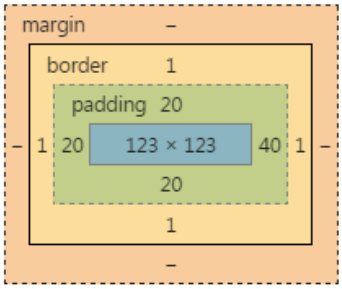
width:123px;

height:123px;

padding:20px 40px;

border:1px solid red;

第 4 题：



width:123px;

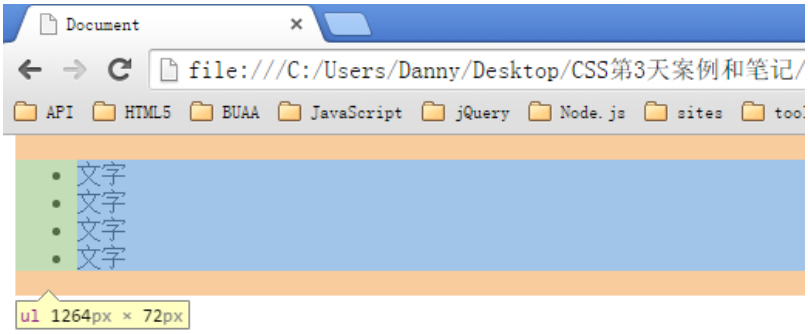
height:123px;

padding:20px;

padding-right:40px;

border:1px solid red;

一些元素，默认带有 padding，比如 ul 标签。



所以，我们为了做站的时候，便于控制，总是喜欢清除这个默认的 padding：

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

*的效率不高，所以我们使用并集选择器，罗列所有的标签（不用背，有专业的清除默认样式的样式表，今后学习）：

```
body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,code,form,fieldset,legend,input,textarea,p,blockquote,th,td{
    margin:0;
    padding:0
}
```

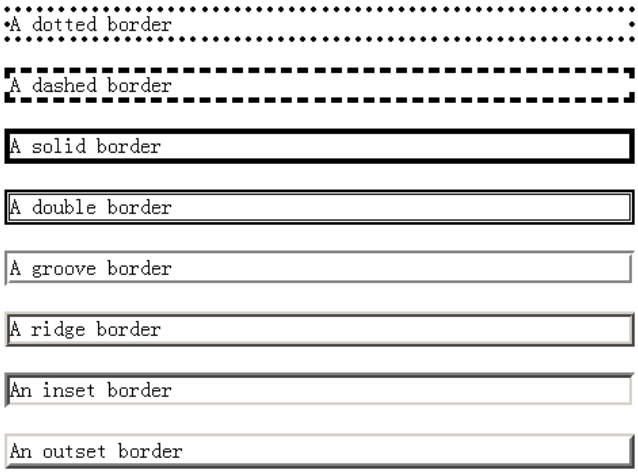
3.4 border

就是边框。边框有三个要素：粗细、线型、颜色。

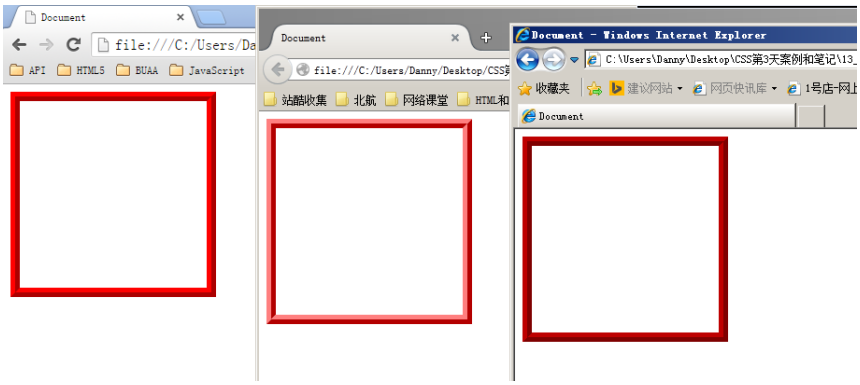
颜色如果不写，默认是黑色。另外两个属性不写，要命了，显示不出来边框。

```
1    border: 1px dashed red;
```

所有的线型：



比如，border:10px ridge red;　在 chrome 和 firefox、IE 中有细微差别：



如果公司里面的设计师，处女座的，追求极高的页面还原度，那么不能使用 css 来制作边框。

就要用到图片，就要切图了。所以，比较稳定的就几个：solid、dashed、dotted

border 是一个大综合属性，

```
1    border:1px solid red;
```

就是把 4 个边框，都设置为 1px 宽度、线型实线、red 颜色。

border 属性能够被拆开，有两大种拆开的方式：

1） 按 3 要素:border-width、border-style、border-color

2） 按方向：border-top、border-right、border-bottom、border-left

按 3 要素拆开：

```
1    border-width:10px;      → 边框宽度
2    border-style:solid;     → 线型
3    border-color:red;       → 颜色。
```

等价于：

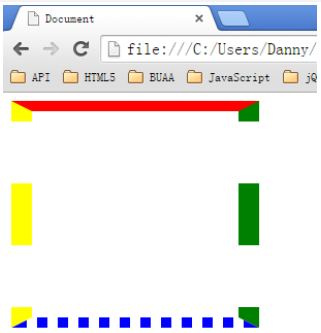
```
1    border:10px solid red;
```

现在心里要明白，**原来一个 border 是由三个小属性综合而成：**

border-width　border-style　border-color。

如果某一个小要素后面是空格隔开的多个值，那么就是上右下左的顺序：

```
1    border-width:10px 20px;
2    border-style:solid dashed dotted;
3    border-color:red green blue yellow;
```



按方向来拆

```
1    border-top:10px solid red;
2    border-right:10px solid red;
3    border-bottom:10px solid red;
4    border-left:10px solid red;
```

等价于

```
1    border:10px solid red;
```

按方向还能再拆一层，就是把每个方向的，每个要素拆开，一共 12 条语句：

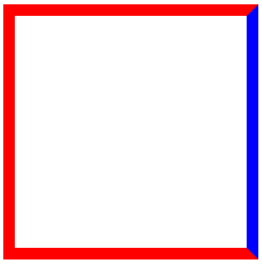
```
1    border-top-width:10px;
2    border-top-style:solid;
3    border-top-color:red;
4    border-right-width:10px;
5    border-right-style:solid;
6    border-right-color:red;
7    border-bottom-width:10px;
```

```
8   border-bottom-style:solid;
9   border-bottom-color:red;
10  border-left-width:10px;
11  border-left-style:solid;
12  border-left-color:red;
```

等价于

```
1   border:10px solid red;
```

工作中到底用什么？很简答：什么简单用什么？



写法：

```
1   border:10px solid red;
2   border-right-color:blue;
```

写法：

```
1   border:10px solid red;
2   border-style:solid dashed;
```

border 可以没有，

```
1   border:none;
```

某一条边没有：

```
1   border-left: none;
```

也可以调整左边边框的宽度为 0：

```
1   border-left-width: 0;
```

四、标准文档流

宏观的讲，我们的 web 页面和 photoshop 等设计软件有本质的区别：web 页面的制作，是个“流”，必须从上而下，像“织毛衣”。而设计软件，想往哪里画个东西，都能画。

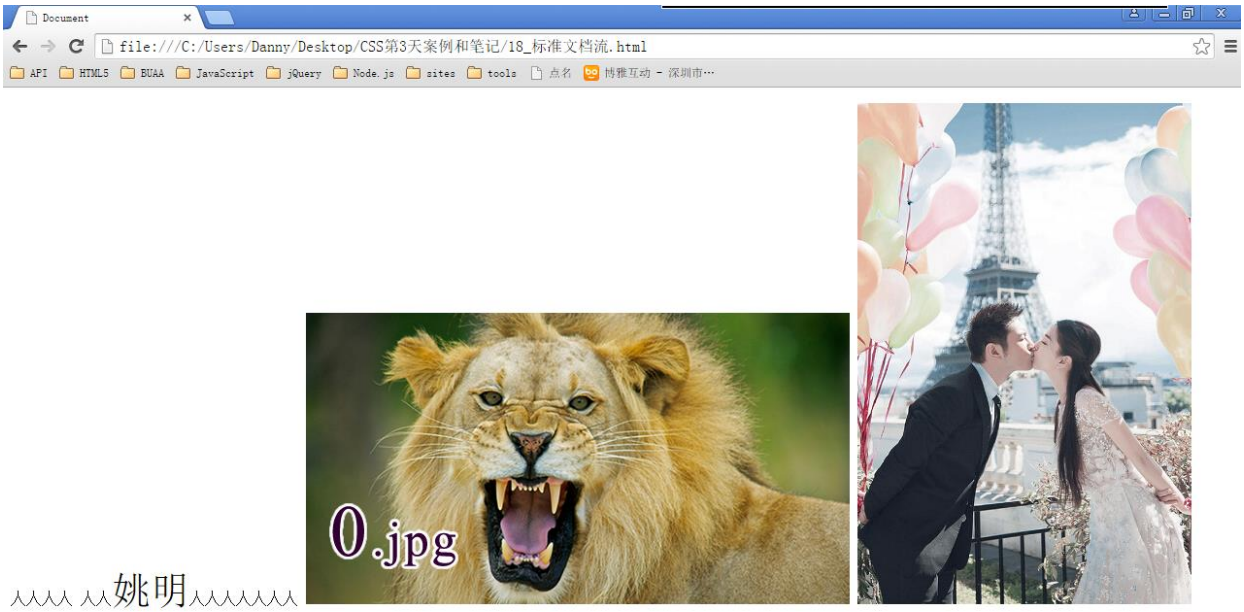
我们要看看标准流有哪些微观现象：

1） 空白折叠现象：

比如，如果我们想让 img 标签之间没有空隙，必须紧密连接：

```
1    <img src="images/0.jpg" /><img src="images/1.jpg" /><img src="images/2.jpg" />
```

2） 高矮不齐，底边对齐：



3） 自动换行，一行写不满，换行写。

4.1 块级元素和行内元素

学习的初期，你就要知道，标准文档流等级森严。标签分为两种等级：

1） 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亲的 100%。

2） 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。

在 HTML 中，我们已经将标签分过类，当时分为了：文本级、容器级。

文本级：p、span、a、b、i、u、em

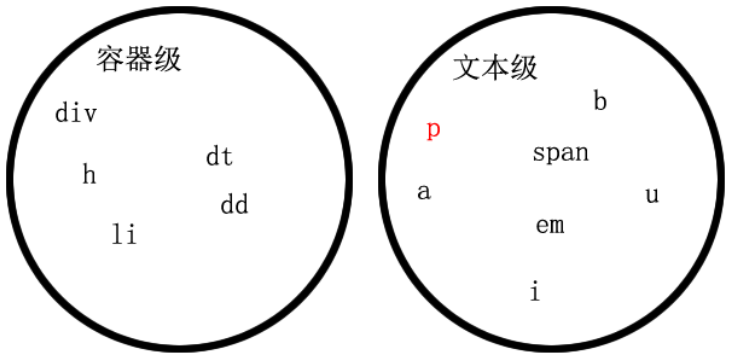
容器级：div、h 系列、li、dt、dd

CSS 的分类和上面的很像，就 p 不一样：

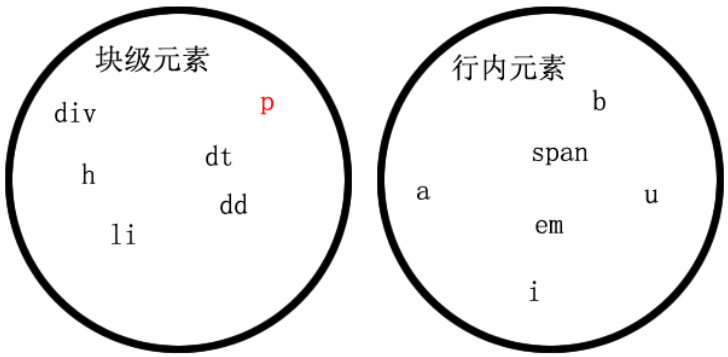
所有的文本级标签，都是行内元素，除了 p，p 是个文本级，但是是个块级元素。

所有的容器级标签都是块级元素。

HTML将标签分为容器级和文本级



CSS将标签分为块级元素和行内元素



4.2 块级元素和行内元素的相互转换

块级元素可以设置为行内元素

行内元素可以设置为块级元素

```
div{
    display: inline;
    background-color: pink;
    width: 500px;
    height: 500px;
}
```

display 是“显示模式”的意思，用来改变元素的行内、块级性质

inline 就是“行内”。

一旦，给一个标签设置

```
display: inline;
```

那么，这个标签将立即变为行内元素。此时它和一个 span 无异：

- 此时这个 div 不能设置宽度、高度；
- 此时这个 div 可以和别人并排了

同样的道理，

```
span{
    display: block;
    width: 200px;
    height: 200px;
    background-color: pink;
}
```

“block”是“块”的意思

让标签变为块级元素。此时这个标签，和一个 div 无异：

- 此时这个 span 能够设置宽度、高度
- 此时这个 span 必须霸占一行了，别人无法和他并排
- 如果不设置宽度，将撑满父亲

标准流里面限制非常多，标签的性质恶心。比如，我们现在就要并排、并且就要设置宽高。

所以，移民！脱离标准流！

css 中一共有三种手段，使一个元素脱离标准文档流：

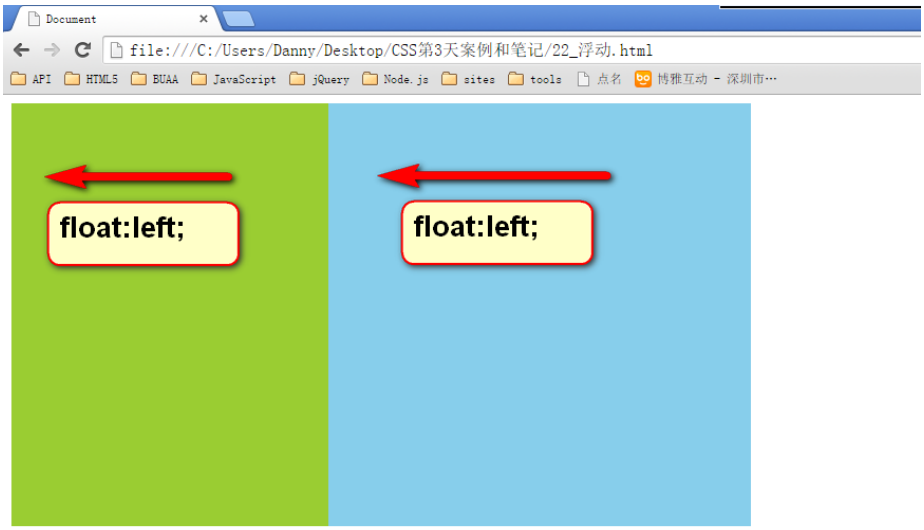
1） 浮动

2） 绝对定位

3） 固定定位

五、浮动

浮动是 css 里面布局用的最多的属性。



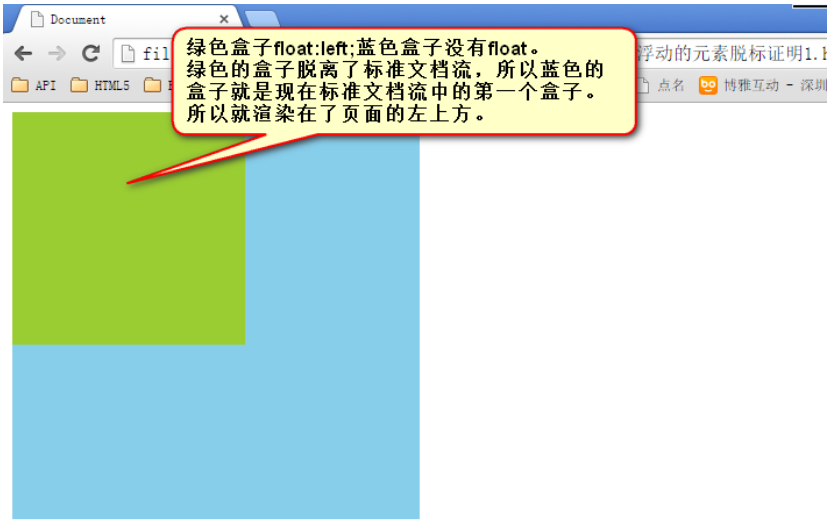

```
.box1{
    float: left;
    width: 300px;
    height: 400px;
    background-color: yellowgreen;
}
.box2{
    float: left;
    width: 400px;
    height: 400px;
    background-color: skyblue;
}
```

两个元素并排了，并且两个元素都能够设置宽度、高度了（这在刚才的标准流中，不能实现）。

浮动想学好，一定要知道三个性质。

5.1 浮动的元素脱标

证明 1：



证明 2：

一个 span 标签不需要转成块级元素，就能够设置宽度、高度了。所以能够证明一件事儿，就是所有标签已经不区分行内、块了。也就是说，**一旦一个元素浮动了，那么，将能够并排了，并且能够设置宽高了。无论它原来是个 div 还是个 span。**

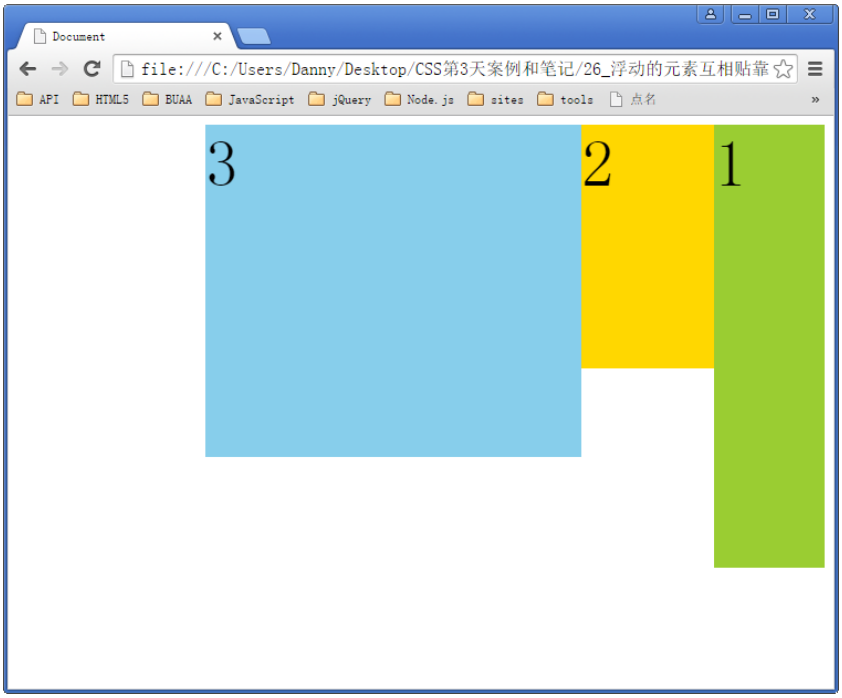
```
span{
    float: left;
    width: 200px;
    height: 200px;
    background-color: orange;
}
```

5.2 浮动的元素互相贴靠

如果有足够空间，那么就会靠着 2 哥。如果没有足够的空间，那么会靠着 1 号大哥。

如果没有足够的空间靠着 1 号大哥，自己去贴左墙。

右浮动：　　float:right;

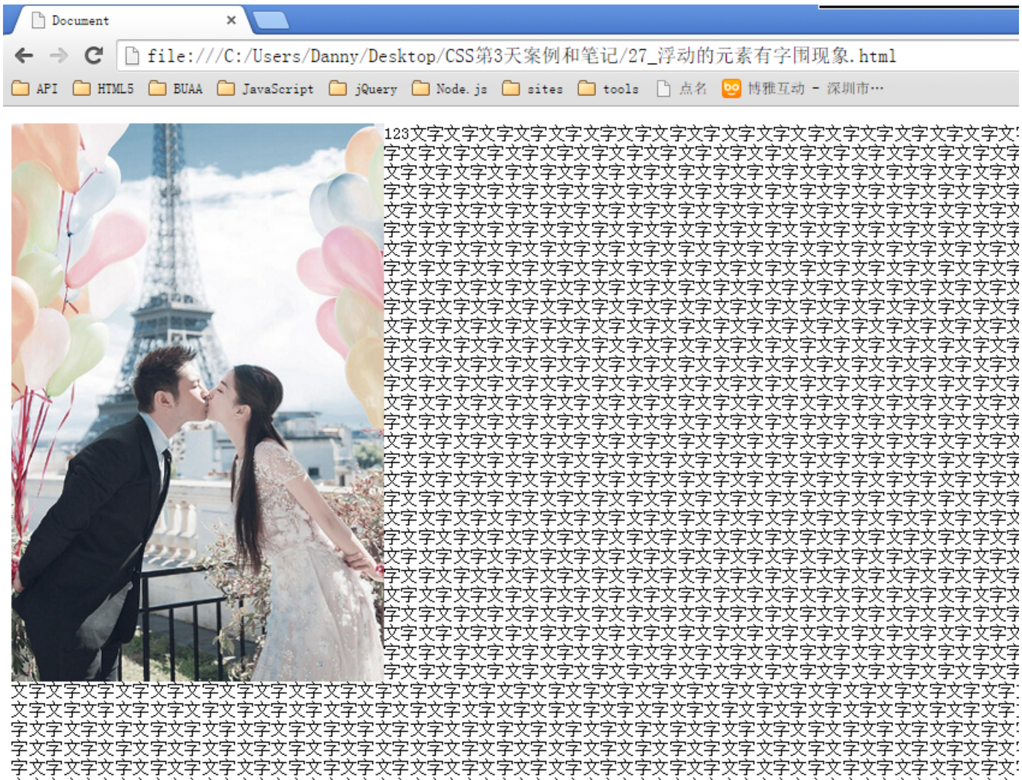


5.3 浮动的元素有“字围”效果

HTML：

```
<div>
    <img src="images/1.jpg" alt="" />
</div>
<p>123 文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字文字
</p>
```

让 div 浮动，p 不浮动：



div 挡住了 p，但是 p 中的文字不会被挡住，形成“字围”效果。

关于浮动我们要强调一点，浮动这个东西，我们在初期一定要遵循一个原则：

永远不是一个东西单独浮动，浮动都是一起浮动，要浮动，大家都浮动。