

# css第03天

感觉还是看视频跟着做案例好一些 而且更接近实际操作

## 一、css三大特性

### 1、层叠性(后来居上)

相同选择器给设置相同的样式，此时一个样式就会覆盖（层叠）另一个冲突的样式。层叠性主要解决样式冲突的问题

☆ 也可以使用层叠性来先设置大多数人的样式 然后再叠上自己的样式

层叠性原则：

- 样式冲突，遵循的原则是就近原则，哪个样式离结构近，就执行哪个样式
- 样式不冲突，不会层叠
- 后来居上 从上往下读（通过f12可以看到被划掉了）

```
<style type="text/css">
  div{
    color:red;
    font-size:18px;
  }
  div{
    color:green;
  }
</style>
```

### 2、继承性 (只继承某一些)

CSS中的继承：子标签会继承父标签的某些样式，如文本颜色和字号。恰当地使用继承可以简化代码，降低CSS样式的复杂性。

```
<style type="text/css">
  div{
    color:red;
    font-size:18px;
  }
</style>
</head>
<body>
  <div>
    <p>文本颜色</p>
  </div>
</body>
```

子元素可以继承父元素的样式：

(text-, font-, line-这些元素开头的可以继承，以及color属性)

继承性口诀：龙生龙，凤生凤，老鼠生的孩子会打洞

## 行高的继承性(好用 自动调整)

```
body {  
    font:12px/1.5 Microsoft YaHei;  
}
```

- 行高可以跟单位也可以不跟单位(含义不同)
- 如果子元素没有设置行高，则会继承父元素的行高为 1.5
- 单独设置了子元素的行高 就会按照新设置的这个
- 此时子元素的行高是：当前子元素的文字大小 \* 1.5
- body 行高 1.5 这样写法最大的优势就是☆里面子元素可以根据自己文字大小自动调整行高

## ☆☆3、优先级(与层叠性区分)

当同一个元素指定多个选择器，就会有优先级的产生。

- 选择器相同，则执行层叠性

(比如都是标签选择器 就近原则)

- 选择器不同，则根据选择器权重执行
- 行内>id>类>元素>\*>继承

选择器优先级计算表格：

选择器	选择器权重
继承 或者 *	0,0,0,0
元素选择器	0,0,0,1
类选择器，伪类选择器	0,0,1,0
ID选择器	0,1,0,0
行内样式 style=""	1,0,0,0
!important 重要的	∞ 无穷大

自写：

从上到下权重依次变大

```
div{  
    color:pink!important      /权重最高  
}
```

eg.类选择器比元素选择器级别更高

优先级注意点：

1. 权重是有4组数字组成,但是不会有进位。
2. 可以理解为类选择器永远大于元素选择器, id选择器永远大于类选择器,以此类推..
3. ☆等级判断从左向右, 如果某一位数值相同, 则判断下一位数值。
4. ☆可以简单记忆法: 通配符和☆继承权重为0, 标签选择器为1, 类(伪类)选择器为10, id选择器100, 行内样式表为1000, !important 无穷大
5. 注意不会进位!!!! eg.0,0,0,11就算有十个标签选择器也比不上类选择器!!!!  
所以也不能总是按1/10/100来考虑!!!!
6. ☆继承的权重是0, 如果该元素没有直接选中, 不管父元素权重多高, 子元素得到的权重都是0。

```
<style>
#father {
    color: red;
}
p {
    color: pink;
}
</style>
</head>
<body>
<div id="father">
    <p>你还是很好看</p>
</div>
</body>
</html>
```

因为继承的权重是0, 父亲权重很高, 但是子孙拿到的权重是0, 权力没法继承咯  
所以认的是标签选择器

看标签到底执行哪个样式 先看标签有没有被直接写出来6

6. a链接比较特殊 浏览器默认指定样式 蓝色有下划线

```
body {
    color: red;
}
/* a链接浏览器默认制定了一个样式 蓝色的 有下划线 */
a {color: blue; />
</style>
</head>
<body>
<div id="father">
    <p>你还是很好看</p>
</div>
<a href="#">我是单独的样式</a>
</body>
</html>
```

相当于把标签单独选出来写了个样式 自然不会继承body的  
单独指定样式:

```
        }
        /* a 链接浏览器默认制定了一个样式 蓝色的 有下划线 */
        a {
            color: green;
        }
    
```

相当于使用层叠性

权重叠加：如果是复合选择器，则会有权重叠加，需要计算权重。有4组数字组成，但是不会有进位。

- div ul li -----> 0,0,0,3
- .nav ul li -----> 0,0,1,2
- a:hover -----> 0,0,1,1
- .nav a -----> 0,0,1,1

```
3     <style>
4         li {
5             color: red;
6         }
7         ul li {
8             color: green;
9         }
10    </style>
11    </head>
12    <body>
13        <ul>
14            <li>大猪蹄子</li>
15            <li>大肘子</li>
16            <li>猪尾巴</li>
17        </ul>
18    </body>
```

```
19    <style>
20        /* 复合选择器会有权重叠加的问题 */
21        /* ul li 权重 0,0,0,1 + 0,0,0,1 = 0,0,0,2 */
22        ul li {
23            color: green;
24        }
25        /* li 的权重是 0,0,0,1 */
26        li {
27            color: red;
28        }
29    </style>
30    </head>
31    <body>
32        <ul>
33            <li>大猪蹄子</li>
34            <li>大肘子</li>
35            <li>猪尾巴</li>
36        </ul>
37    </body>
```

```

        /* li 的权重是 0,0,0,1    1 */
        li {
            color: red;
        }
        /* .nav li 权重      0,0,1,0 + 0,0,0,1 = 0,0,1,1    11 */
        .nav li {
            color: pink;
        }
    
```

与顺序无关

是因为复合选择器要叠加

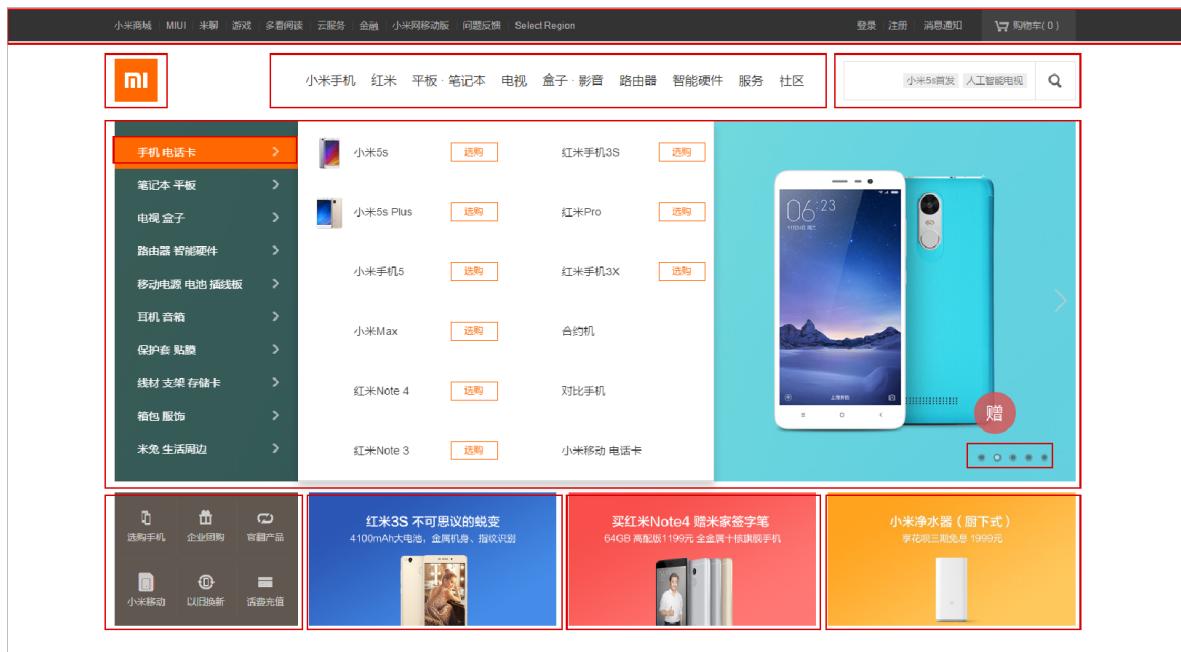
注意不会进位!!!!!! eg.0,0,0,11就算有十个标签选择器也比不上类选择器!!!!

所以也不能总是按1/10/100来考虑!!!!!!

## 二、盒子模型

### 1、网页布局的本质

网页布局的核心本质：就是利用 CSS 摆盒子。



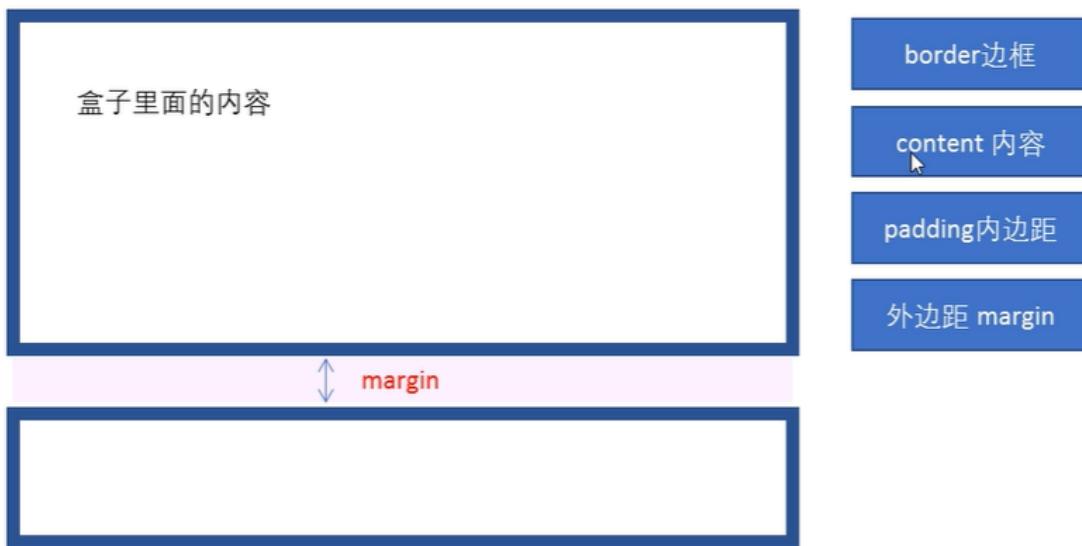
网页布局过程：

1. 先准备好相关的网页元素，网页元素基本都是盒子 Box。
2. 利用 CSS 设置好盒子样式，然后**摆放到相应位置**。
3. **往盒子里面装内容**

### 2、盒子模型 (Box Model) 组成

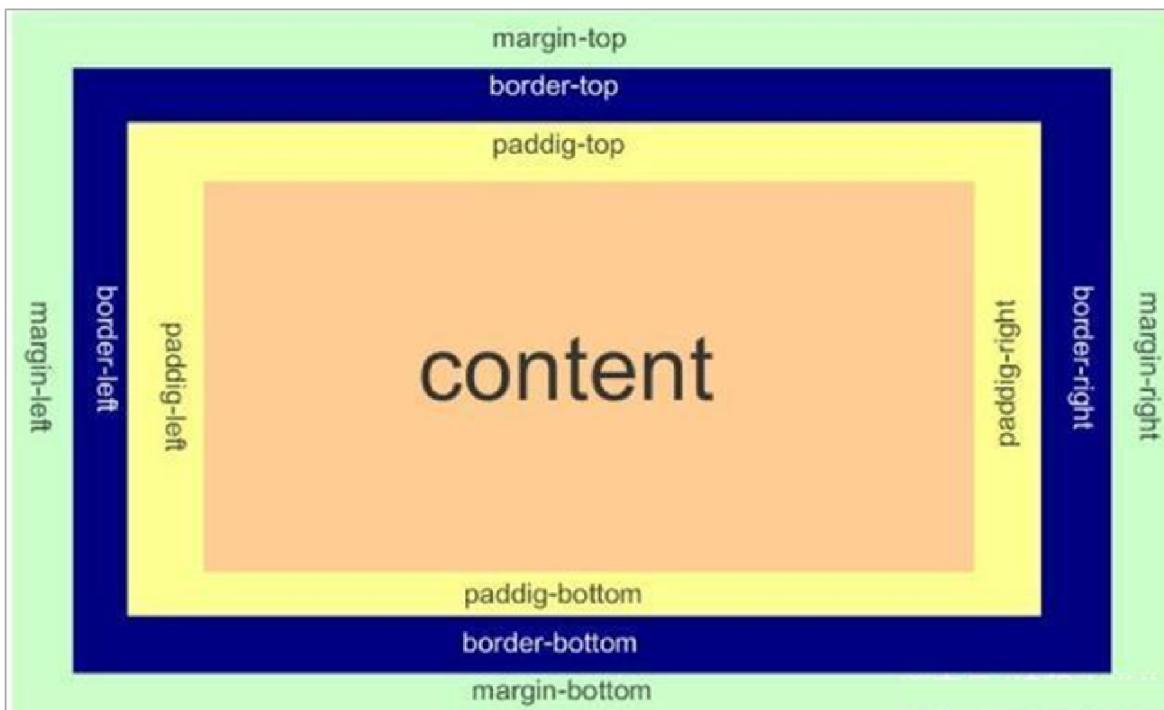
盒子模型：把 HTML 页面中的布局元素看作是一个矩形的盒子，也就是一个盛装内容的容器。

CSS 盒子模型本质上是一个盒子，封装周围的 HTML 元素，它包括：**边框**、**外边距**、**内边距**、和**实际内容**



padding:默认文字是抵着边框左上角的 这样可以隔开content和border

margin:盒子之间的距离



### 3、边框 (border)

#### 3.1、边框的使用

- 1、border可以设置元素的边框。边框有三部分组成：边框宽度(粗细) 边框样式 边框颜色；
- 2、语法：

```
border : border-width || border-style || border-color;
```

属性	作用
border-width	定义边框粗细，单位是px
border-style	边框的样式
border-color	边框颜色

边框样式 border-style 可以设置如下值：

- none: 没有边框即忽略所有边框的宽度 (默认值)
- solid: 边框为单实线(最为常用的)
- dashed: 边框为虚线
- dotted: 边框为点线



### 3、边框的合写分写

边框简写：

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

边框分开写法：

```
border-top: 1px solid red; /* 只设定上边框， 其余同理 */
```

- 例子：

```
/* 请给一个 200*200 的盒子，设置上边框为红色，其余边框为蓝色 */
div {
    width: 200px;
    height: 200px;
    /* border-top: 1px solid red;
    border-bottom: 1px solid blue;
    border-left: 1px solid blue;
    border-right: 1px solid blue; */
    /* border 包含四条边 */
    border: 1px solid blue;
    /* 层叠性 只是层叠了 上边框啊 */
    border-top: 1px solid red;
}
```

不能换位置！！！因为是层叠不是优先级哈！！！

先写大的再写小的(特殊的!!!!)

### 3.2、表格的细线边框(border-collapse)

通过css而非html来改表格边框

1、border-collapse 属性控制浏览器绘制表格边框的方式。它控制相邻单元格的边框。

2、语法：

```
border-collapse: collapse;
```

collapse 单词是合并的意思

border-collapse: collapse; 表示相邻边框合并在一起

- 例子1

```
<style>
    table,
    td {
        border: 1px solid pink;
    }
</style>
```

如果只有table的话 就是只给表格外边框 没给单位格子

- 例子2

```
<style>
    table {
        width: 500px;
        height: 249px;
    }
    table,
    td {
        border: 1px solid pink;
    }
</style>
```

只给表格改大小

- 例子3

1	鬼吹灯	◆	456	123	贴吧 图片 百科
3	西游记	◆	456	123	贴吧 图片 百科

单元格边框挨在一起 中间变粗了 所以希望可以合在一起

border-collapse

合并相邻的边框

```
table {  
    width: 500px;  
    height: 249px;  
}  
th {  
    height: 35px;  
}  
table,  
td, th {  
    border: 1px solid pink;  
    /* 合并相邻的边框 */  
    border-collapse: collapse;  
    font-size: 14px;  
    text-align: center;  
}
```

- 注意学前端要有把不同级别的东西分开的自觉！！！这样方便一个一个改

### 3.3、☆边框会影响盒子实际大小

边框会**额外增加盒子的实际大小**。因此我们有两种方案解决：

- 测量盒子大小的时候,不量边框。(最后再把border加上就正好)
- 如果测量的时候包含了边框,则需要内部的**width/height 减去边框宽度**

## 4、内边距 (padding)

### 4.1、☆内边距的使用方式(注意合写个数代表的意思)

1、padding 属性用于设置内边距，即边框与内容之间的距离。

2、语法：

**合写属性：**

值的个数	表达意思
padding: 5px;	1个值，代表上下左右都有5像素内边距；
padding: 5px 10px;	2个值，代表上下内边距是5像素 左右内边距是10像素；
padding: 5px 10px 20px;	3个值，代表上内边距5像素 左右内边距10像素 下内边距20像素；
padding: 5px 10px 20px 30px;	4个值， 上是5像素 右10像素 下20像素 左是30像素 顺时针

1:上下左右

2：上下/左右



eg

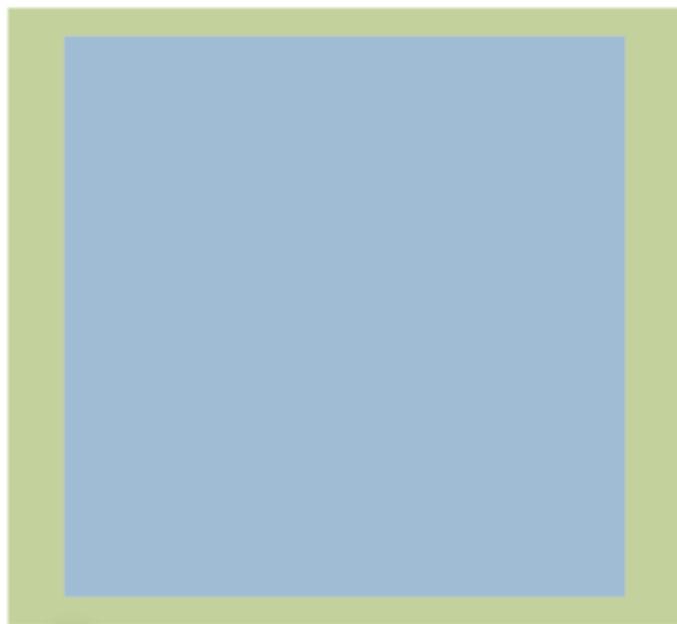
3: 上左右下

4: ☆☆**上右下左** 顺时针

分写属性:

属性	作用
padding-left	左内边距
padding-right	右内边距
padding-top	上内边距
padding-bottom	下内边距

tips: 可以使用f12来看盒子的像素!!!!



**div** 240 × 220

Background  #FFE4C4

Padding 10px 20px

ACCESSIBILITY

Name

Role generic

Keyboard-focusable

```
<style>
  div {
    padding: 10px 20px 10px;
    background-color: bisque;
    width: 200px;
    height: 200px;
  }
</style>
```

## 4.2、☆内边距会影响盒子实际大小

- 当我们给盒子指定 padding 值之后，发生了 2 件事情：

1. 内容和边框有了距离，添加了内边距。

2. padding影响了盒子实际大小。

2、内边距对盒子大小的影响：

- 如果盒子已经有了宽度和高度，此时再指定内边框，会撑大盒子。
- 如何盒子本身没有指定width/height属性，则此时padding不会撑开盒子大小。

3、解决方案：

如果保证盒子跟效果图大小保持一致，则让 width/height 减去多出来的内边距大小即可。

```
div {  
    width: 200px;  
    height: 200px;  
    background-color: pink;  
    padding: 20px;  
}
```

```
div {  
    width: 160px;  
    height: 160px;  
    background-color: pink;  
    padding: 20px;  
}
```

padding必须要有 所以只能改width和height

记得要减两边

- padding影响实际大小的好处



## 案例：新浪导航案例-padding影响盒子好处

padding内边距可以撑开盒子,我们可以做非常巧妙的运用.

因为每个导航栏里面的字数不一样多,我们可以不用给每个盒子宽度了,直接给padding最合适.



方式1： 给盒子宽度不合理

设为首页	新浪微博	微博	关注我
+			

方式2： 给盒子padding 合理

设为首页	新浪微博	微博	关注我
+			

```

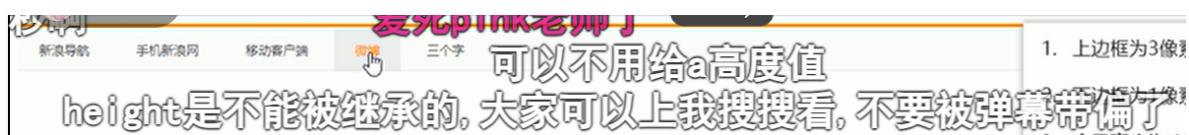
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .nav {
            height: 41px;
            background-color: #fcfcfc;
            /* 合写方便点 */
            border-top: 3px solid #ff8500;
            border-bottom: 1px solid #edeff0;
            /* 行高可以继承 很方便！ */
            line-height: 41px;
            /* 在没有定位到情况下可以使文字居中 */
        }
        /* 大小可以用ps测量 */
        /* a { */
        /* 注意写nav里的a！！复合的一定要习惯用从属关系 */
        .nav a {
            /* 如果要w300这样 只能首字母才能这么带值的缩写 */
            font-size: 12px;
            color: #4c4c4c;
            text-decoration: none;
            /* 需要控制a有一个大小 但是因为宽度是padding给的 所以就改个高度 */
            height: 41px; /* 和nav一个高度 */
            /* 如果只写height高度不生效 因为是行内元素 */
            /* 检查高度是否生效可以用f12 */
            /* 但是又需要往一行 所以改成行内块 */
            display: inline-block;
            /* 设置padding */
            padding: 0 20px;
            /* 0不能漏写 一个数就不是那个意思了 */
        }
        /* 接下来处理的是hover变底色 */
        .nav a:hover {
            background-color: #eee;
            color: #ff8500;
        }
    </style>
</head>
<body>
    <!-- 输入div.nav即可 -->
    <!-- 先写最外边div 写这个div的时候就给明确class -->
    <!-- 这些元素其实本质上就是盒子 -->
    <div class="nav">
        <!-- 行内的这种还是用复制粘贴快捷键比a*4好点 不然写到一排了 -->
        <a href="#">新浪</a>
        <a href="#">手机新浪网</a>
        <a href="#">移动客户端</a>
        <a href="#">微博</a>
    </div>
</body>
</html>

```

- 上面有一个小间隔？



- 不用给a高度值



把div里的line-height: 41px; 放到.nav a{}里面去也行 应该是因为内部盒子和外部盒子

清楚内外边距



实际开发



## 案例：小米导航案例修改-padding影响盒子大小计算

padding内边距可以撑开盒子,有时候,也会让我们去修改宽度  
所以小米侧边栏这个案例,文字距离左侧的距离不应该用 text-indent 这样不精确.  
实际开发的做法是给 padding值,这样更加精确.



但是2em不精确 真正开发要按照样板精确到几像素

(开发网站是先进行UI设计)



但是这样直接给一个padding 会把盒子撑大 但是大小不能随便变换 就把width改一下

**没有指定width和height时**

不会撑开盒子的大小

```
h1 {  
    width: 100%;  
    height: 200px;  
    background-color: pink;  
    padding: 30px;  
}
```

没有必要写这个width100%了 只要有了width这个属性值就会撑大 在父级的基础上又加上了 padding就会有水平拖动条了

```
        }
    div {
        width: 300px;
        height: 100px;
        background-color: purple;
    }
    div p {
        padding: 30px;
    }

```

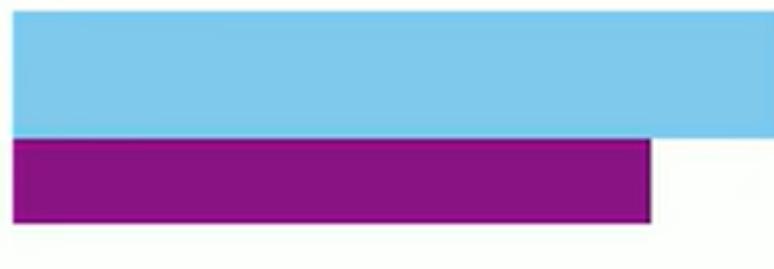
</style>

```
</head>
<body>
    <h1></h1>
    <div>
        <p></p>
    </div>
</body>
```



The screenshot shows a browser window with the developer tools open. The 'Elements' tab is selected. A blue rectangular box with a red border is highlighted. Inside the box, there is a red plus sign (+) located in the center of the padding area. To the left of the box, there is a vertical pink bar.

p同理 不写默认是父级容器的宽度 写了100% 就会额外加上padding了



注意元素继承只有一部分哈

宽高不是被继承哈 和继承不一样

这个时候是和父级容器的大小 实际开发里面也会用

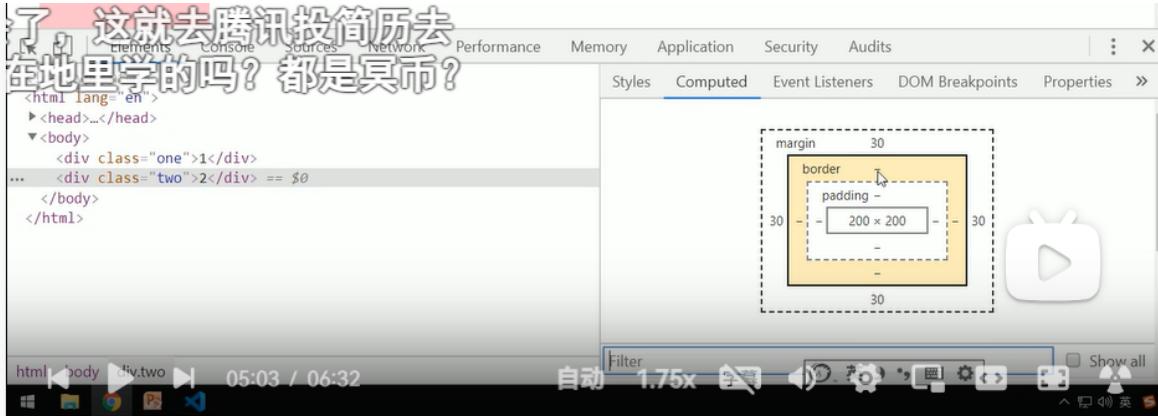
## 5、外边距 (margin)

### 5.1、外边距的使用方式

margin 属性用于设置外边距，即控制盒子和盒子之间的距离。

属性	作用
margin-left	左外边距
margin-right	右外边距
margin-top	上外边距
margin-bottom	下外边距

## margin的简写同padding



可以这样看

## 5.2、☆外边距典型应用-块级盒子水平居中(常见)(内含其他元素居中方法)

外边距可以让块级盒子水平居中的两个条件：

- 盒子必须指定了宽度 (width) 。
- 盒子左右的外边距都设置为 auto 。

常见的写法，以下三种都可以：

```
margin-left: auto; margin-right: auto;/左右auto
margin: auto;/上下左右auto
margin: 0 auto;/上下没有外边距 左右auto
```

注意：以上方法是让块级元素水平居中，行内元素或者行内块元素水平居中给其父元素添加 text-align:center 即可。

父元素！！！！！

对行内元素用margin: 0 auto没有用噢

## 5.3、☆☆外边距合并(相邻/嵌套塌陷)

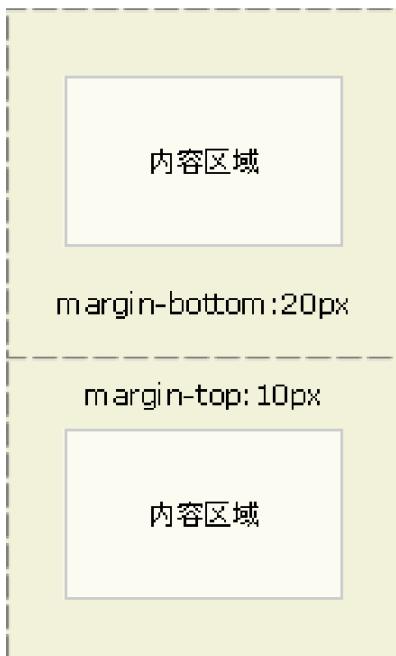
使用 margin 定义块元素的垂直外边距时，可能会出现外边距的合并。

主要有两种情况：

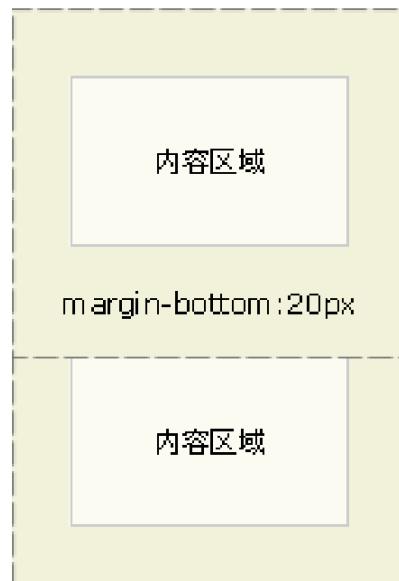
### 1、相邻块元素垂直外边距的合并

当上下相邻的两个块元素☆（兄弟关系）相遇时，如果上面的元素有下外边距 margin-bottom，下面的元素有上外边距 margin-top，则他们之间的垂直间距不是 margin-bottom 与 margin-top 之和。☆取两个值中的较大者这种现象被称为相邻块元素垂直外边距的合并。

## 合并之前



## 合并之后



解决方案：

尽量只给一个盒子添加 margin 值。

## ☆☆2、嵌套块元素垂直外边距的塌陷

对于两个嵌套关系☆（父子关系）的块元素，父元素有上外边距同时子元素也有上外边距，此时父元素会塌陷较大的外边距值。

## 合并之前



## 合并之后



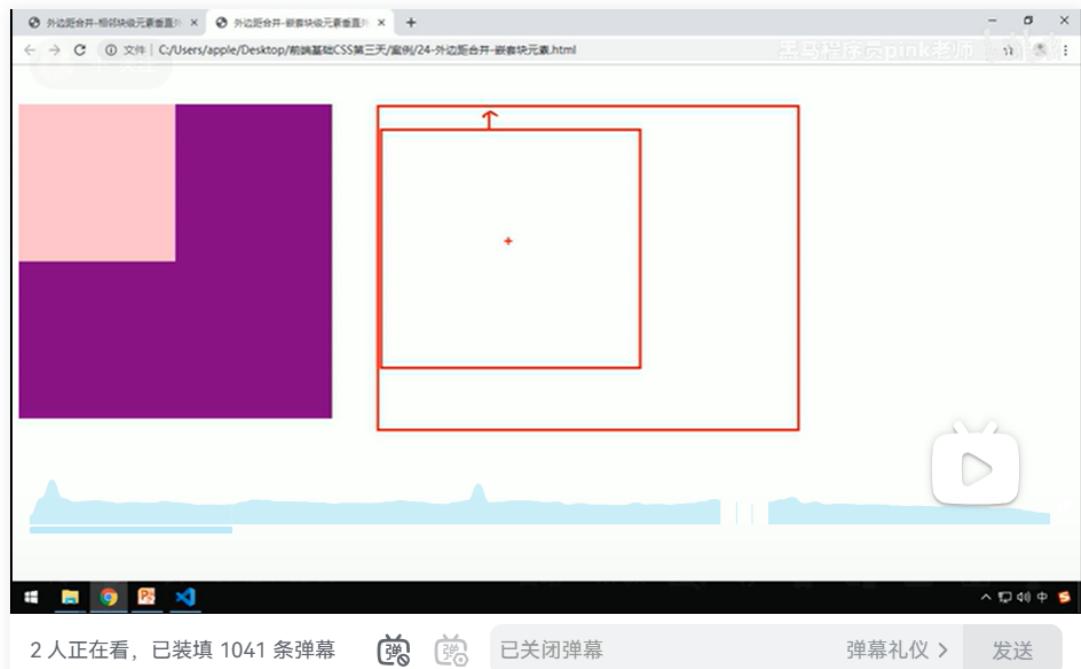
解决方案：

- 可以为父元素定义上边框。
- 可以为父元素定义上内边距。
- 可以为父元素添加 overflow:hidden。

例子：

step1

```
<style>
    .father {
        width: 400px;
        height: 400px;
        background-color: #purple;
        margin-top: 50px;
    }
    .son {
        width: 200px;
        height: 200px;
        background-color: #pink;
    }
</style>
</head>
<body>
    <div class="father">
        <div class="son"></div>
    </div>
```

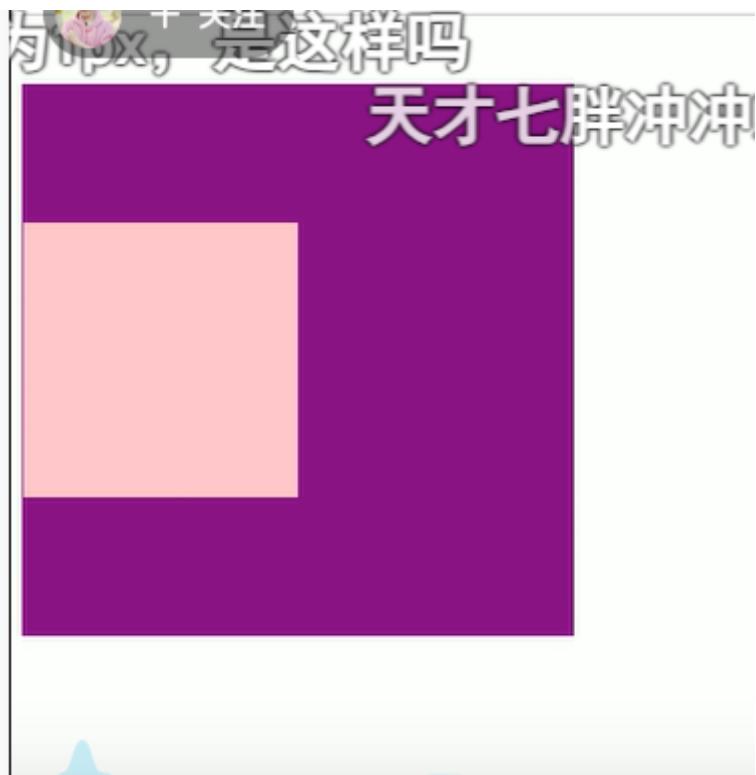


step2



发现塌陷了 是因为父亲和孩子挨得近

1. 定义父元素上边距
2. 定义父元素内边距
3. 使用overflow: hidden



## 5.4、清除内外间距

网页元素很多都带有默认的内外间距，而且不同浏览器默认的也不一致。因此我们在布局前，首先要清除下网页元素的内外间距。

相当于把先天自带的特性清除 然后全部重新设置

- 这里要使用\*通配符全选

```
* {  
    padding:0; /* 清除内间距 */  
    margin:0; /* 清除外间距 */  
}
```

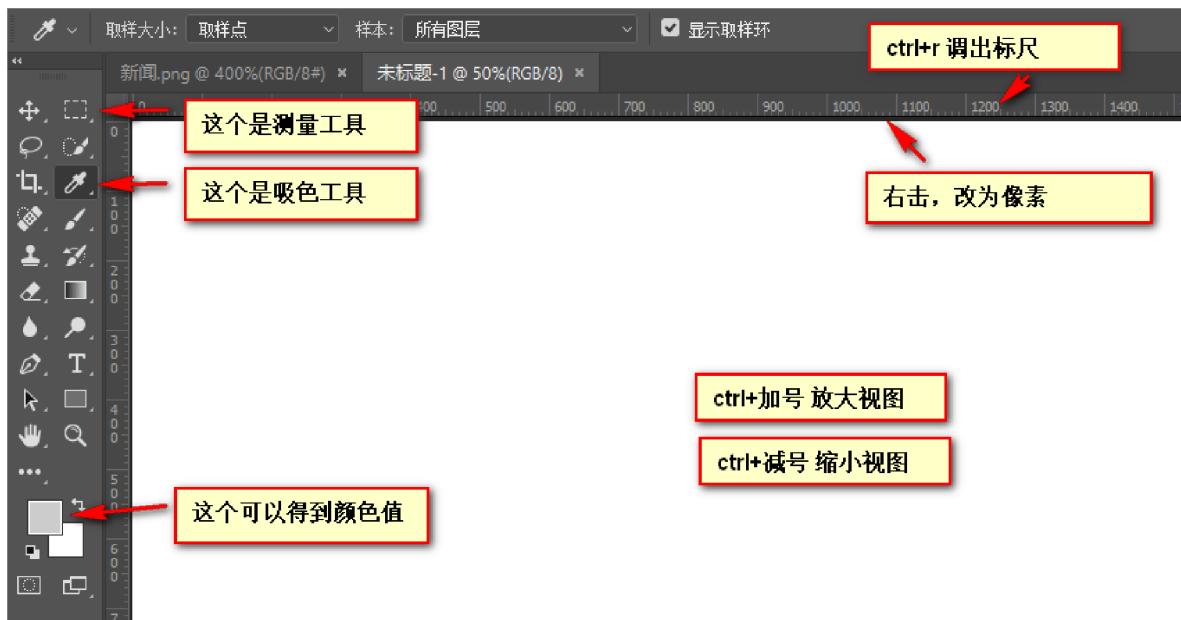
注意：行内元素为了照顾兼容性，尽量只设置左右内外间距，不要设置上下内外间距。但是转换为块级和行内块元素就可以了

## 三、PS 基本操作

因为网页美工大部分效果图都是利用 PS (Photoshop) 来做的，所以以后我们大部分切图工作都是在 PS 里面完成。

部分操作：

- 文件→打开：可以打开我们要测量的图片
- Ctrl+R：可以打开标尺，或者 视图→标尺
- 右击标尺，把里面的单位改为像素
- Ctrl+ 加号(+)可以放大视图，Ctrl+ 减号(-)可以缩小视图
- 按住空格键，鼠标可以变成小手，拖动 PS 视图
- 用选区拖动 可以测量大小
- Ctrl+ D 可以取消选区，或者在旁边空白处点击一下也可以取消选区



## 其他综合写法

时候用盒子  
一个超链接么  
最后一个盒子再划分

information

不是p里不能加div吗。。。

img ← div.box

p.review ←

div.appraise ←

div.info ←

布局分析很重要

## Pink 老师总结

### 1. 布局为啥用不同盒子,我只想用div ?

标签都是有语义的,合理的地方用合理的标签。比如产品标题就用 h, 大量文字段落就用p

### 2. 为啥用辣么多类名 ?

类名就是给每个盒子起了一个名字,可以更好的找到这个盒子,选取盒子更容易,后期维护也方便。

### 3. 到底用 margin 还是 padding ?

大部分情况两个可以混用,两者各有优缺点,但是根据实际情况,总是有更简单的方法实现。

### 4. 自己做没有思路 ?

布局有很多种实现方式,同学们可以开始先模仿我的写法,然后再做出自己的风格。



---

1. 语义化

2. 取类名 后期维护

3. margin/padding 注意是否撑开盒子(padding) 是否会塌陷