HTML 4.01

——有用请投 44 号，谢谢（部分有错误，请辩证看待，自行更正）

1、前后状态的不同，叫做交互效果。有了人的操作，机器给了一个反馈。

2、save 或 ctrl+s，保存，存文件一定要加.html

3、Hypertext Markup Language 超文本标记语言

4、<html></html>根标签

<head></head>里面放的是思想，设置浏览器用的，是人看不到的

<body></body>是展示给用户看的

可以右键打开 Open in Browser

5、这就是一个属性：属性名 = “属性值”

属性名不用加双引号，属性值必须加双引号

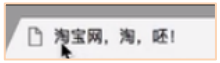
<meta charset = “utf-8”>一般都填这个

编码字符集：gb2312 国家编码字符集（简体，亚裔字符集）

gbk（gb2312+繁体）

unicode 是万国码，包括了所有国家的语言

uft-8 是 unicode



6、<title>淘宝网，淘</title>

7、<html lang = “en”>告诉搜索引擎爬虫，我们的网站是关于什么内容的

en 代表是英文，zh 是中文，德语是 de

竞价排名费 1-999 元/点击一次，但是有 IP 段锁定

例这样是告诉搜索引擎的

<meta content = “服饰” name = “keywords”>

<meta content = “时装” name = “description”>

8、 段落标签，让内容成段展示

9、标题标签，成段展示，独占一行，加粗字体，更改字体大小（从 1 到 6 依次减小）

<h1></h1>

<h2></h2>

<h3></h3>

<h4></h4>

<h5></h5>

<h6></h6

10、加粗

斜体

例又加粗又斜体，写成嵌套功能

又加粗又斜体

11、<del></del>这是一个中划线

12、<address></address>这是一个地址标签。可以用斜体+成段展示模拟

13、 独占一行

不独占一行

div 和 span 这两个标签是为了成块展示，规格化，这两个就是容器，功能如下：

功能 1：分块明确，让整个页面更加结构化；

功能 2：捆绑操作的作用（搬书架）

```
<strong style="color:#f40">a</strong>
<em style="color:#f40">b</em>
<del style="color:#f40">c</del>
```

```
<div style="color:#f40">
    <strong>a</strong>
    <em>b</em>
    <del>c</del>
</div>
```

14、如果是一个单词，溢出是不管的

空格的含义是英文单词分隔符，不代表文本的空格，作为分隔符，打多少个都只显示一个空格；回车也是文字分割符，也是打多少个都只显示一个空格

成哥 很帅

成哥 很帅

```
<div style="width:100px;height:100px;background-color:red;">
qweoirupqowiueriyqwepiruqpowieurpoqwiuer</div>
```


```
<div style="width:100px;height:100px;background-color:red;">
邓哥很磕碜邓哥很磕碜邓哥很磕碜邓哥很磕碜邓哥很磕碜邓哥很磕碜</div>
```

```
<div style="width:100px;height:100px;background-color:red;">qpo wie urp
oqi uwe rp oiu qwe pori uqwp oeiu rqpow eiur</div>
```

html 编码格式是&；常用就以下三个

1） 空格文本，写多少个就空几格

2）<左尖角号，小于的意思，less than，html 编码是<

3）>右尖角号，大于的意思，great than，html 编码是>

15、
换行符

大部分标签的作用是把他包裹的文本作用城他设置的样子，所以成对出现，有的标签自己就代表功能，就是单标签

16、<hr>水平线

17、有序列表

喜欢的电影

```
<ol>
<li>marvel</li>
<li>速 8</li>
<li>返老孩童</li>
<li>了不起的盖茨比</li>
</ol>
```

你们喜欢看的电影

1. marvel
2. 速8
3. 返老还童
4. 了不起盖茨比

如果写成：<ol type = “1”>　就以 ABC 排序，　改成 a，就以 abc 排序

此处的 type 值只有五个：数字，大写英文 A，小写英文 b，罗马数字大写 I，罗马数字小写 i

A 可以 27 进制

写成<ol type = “1” reversed = “reversed”>　就是倒序

如果想从第 2 个开始排序，就写<ol type = “1” start = “2”>

如果想从第 117 个开始排序，就写<ol type = “i” start = “117”>

想从第几个开始拍，start 里面写数字几

18、**无序列表 ul**，unorder list 只有 type = “”这一个属性可以改

```
<ul type = “disc”>
<li>草莓</li>
<li>苹果</li>
<li>橙子</li>
</ul>
```

你们喜欢看的电影

- 草莓
- 苹果
- 🍊

如 type =　“disc”　意思是 discircle，实心圆

如 type =　“square”　意思是 square，实心方块

如 type =　“circle”　意思是 circle，圈(空心圆)

ul 和 li 是一个很好的天生父子结构(柜子与抽屉)，可以做导航栏

可维护性好

```
*{
    margin:0;
    padding:0;
}
ul{
    list-style:none;
}
```

```
li{
    margin:0 10px;
    float:left;
    color:#f40;
    font-weight:bold;
    font-size:14px;
    height:25px;
    line-height: 25px;
    padding:0 5px;
}
```

```
li:hover{
    border-radius:15px;
    background-color:#f40;
    color:#fff;
}
```

```
<ul type="circle">
    <li>天猫</li>
    <li>聚划算</li>
    <li>天猫超市</li>
</ul>
```

天猫　聚划算　天猫超市

19、<img src = “sss.jpg style = “width:100px;”>

src 是 source 的缩写，img 的地址分：

1）网上 url

2）本地的绝对路径

3）本地的相对路径

如 html 和图片在同一文件下，是一种相对关系，相对路径，写法<img src = “123.jpg”>

D:/a/b/lesson2.html

D:/a/b/123.jpg

如 html 和图片不在同一文件下，是绝对路径，写法<img src = “D:/a/b/c/123.jpg”>

D:/a/b/lesson2.html

D:/a/b/c/123.jpg

例<img scr = “fy.jpg” style = “width:200px;” alt = “这是范爷”>

这是范爷

此处的 alt 是图片占位符。当地址出错时，图挂了，来展示这个信息；图片没问题，就不会展示这个信息

例<img scr = “fy.jpg” style = “width:200px;” alt = “这是范爷” title = “This is fy”>

此处的 title 是图片提示符。当鼠标放上去，就会显示这个信息

20、www.taobao.com

这个地址展示给用户是淘宝，实际给浏览器看的地址是百度

href 是 hyperText reference 超文本引用

<a>标签可以包裹图片

<a>是 anchor -->　锚，定在某个点（置顶）

<a>标签的功能

1）超链接

2）锚点

3）打电话，发邮件

4）协议限定符

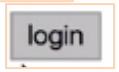
例<a href = “https://www.baidu.com” target = “_blank”>target = “_blank”意思是在新标签中打开这个地址

```
<div id="demo1" style="width:100px;height:100px;background-color:green;"
>demo1</div>
<br*100
<div id="demo2" style="width:100px;height:100px;background-color:red;">
demo2</div>
<a href="#demo1">find demo1</a>
<br*100
<a href="#demo2">find demo2</a>
```

```
<a style="
display:block;position:fixed;bottom:100px;right:100px;border:1px solid
black;height:50px;width:200px;background-color:#fcc;" href="#demo1">
find demo1</a>
<a style="
display:block;position:fixed;bottom:150px;right:100px;border:1px solid
black;height:50px;width:200px;background-color:#ffc;" href="#demo2">
find demo2</a>
```

例
```
<a href="tel:13836014778">给成哥打电话</a>
```
例
```
<a href="mailto:cheng.ji@alibaba-inc.com" >给成哥发邮件</a>
```
写一个你点我试试写了 JavaScript 就可以在里面写 js 了
```
<a href="javascript:while(1){alert('让你手欠')}">你点我试试呀!
</a>
```

21、**form 表单**，能发送数据，非常重要<form></form>
form method = “get/post”这是 form 发送数据的两种方式
action = “http://ssffg.php”这是发送给谁，就是 action 的位置
form 表单里面还需要配合 input 来写，**input 里面需要 type**
<input type = “text”> //这个是输入框的意思
<input type = “password”> //这个是密码框的意思，默认是暗文
<input type = “submit”> //这个是提交的组件，也就是登录
<input type = “submit” value = “login”> //这样就改变了提交框的值
<input type = “radio”> //是单选框
<input type = “checkbox”> //是复选框



要注意语义化，所以用 p 标签更好，p 标签天生的功能就是换行

```
<style type="text/css">
    input{
        border:1px solid #999;
    }

</style>
```

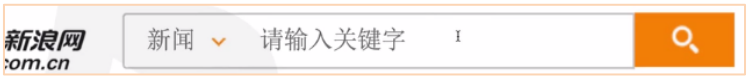
```
<form method="get" action="">

    <p>
        username:<input type="text">
    </p>
    <p>
        password:<input type="password">
    </p>
    <input type="submit">

</form>
```

这种方式发送不了数据
发送数据要注重数据的主题（数据名）和数据的内容（数据值），缺一不可，没有这个就发送不了数据
<input type = “text” name = “1234”> 此处 name 是数据名，1234 是数据值，此处 1234 可以随便填，最好填写接近意思的英文单词，方便使用
例
```
<form method="get" action="">

    <p>
        username:<input type="text" name="username">
    </p>
    <p>
        password:<input type="password" name="password">
    </p>
    <input type="submit">

</form>

pro/Desktop/lesson2.html?username=sunny&password=123123
```
html？后面有值，就是发送成功的体现

另外，暗文是仅对自己不可见，受网安局监管的公司，一般用 md5，不可逆的加密方式
<input type = “radio” name = “star”>是单选框，此处的 name 里面的值一样的，是告诉浏览器你们是一道题，那么这个时候就没有填数据值的地方了，可以写个 value = “” 来储存数据值
例
```
<form method="get" action="">

    <!-- <p>
        username:<input type="text" >
    </p>
    <p>
        password:<input type="password" >
    </p> -->
    你们所喜欢的男星
    1.贝克汉姆<input type="radio" name="star">
    2.莱昂纳多<input type="radio" name="star">
    3.姬成<input type="radio" name="star">
    你们所喜欢的男星
    1.贝克汉姆<input type="radio" name="star1">
    2.莱昂纳多<input type="radio" name="star1">
    3.姬成<input type="radio" name="star1">


    <input type="submit">
```
<input type = “radio” name = “star” value = “”>此处的 name 里面的值一样的，是告诉浏览器你们是一道题，那么这个时候就没有填数据值的地方了，可以写个 value = “” 来储存数据值，下面也是在 form 表单中填写
```
你们所喜欢的男星
1.贝克汉姆<input type="radio" name="star" value="xiaobei">
2.莱昂纳多<input type="radio" name="star" value="laiangnaduo">
3.姬成<input type="radio" name="star" value="handsome">

<input type="submit">
```
ro/Desktop/lesson2.html?star=xiaobei
在以下情况填写 value 数据值
```
<p>
    username:<input type="text" name="username" value="请输入用户名">
</p>
<p>
    password:<input type="password" name="username">
</p>
<input type="submit">
```
username: 请输入用户名
password:
提交

把编程思想和编程工具结合在一起就是编程，把编程思想量化出来，用编程工具一步步写出来，修修补补，完成自己所需要的

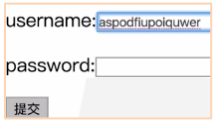
例下面做一个小项目，如图，聚焦，失去焦点，输入文字，发生改变



onfocus = “”是想鼠标聚焦时发生什么事

onblur = “”是在鼠标失去焦点时发生的状态

```
<p>
    username:<input type="text" name="username" style="color:#999"
    value="请输入用户名" onfocus="if(this.value=='请输入用户名'){this.
    value='';this.style.color='#424242'}" onblur="if(this.value=='')
    {this.value='请输入用户名';this.style.color='#999'}">
</p>
<p>
    password:<input type="password" name="username">
</p>
<input type="submit">
```



<input type = “checkbox”>　　//是复选框

```
1.apple<input type="checkbox" name="fruit" value="apple">
2.orange<input type="checkbox" name="fruit" value="orange">
3.banana<input type="checkbox" name="fruit" value="banana">


l?fruit=apple&fruit=orange
<input type="submit">
```

功能上要有培养用户懒习惯（吸引用户）

做一个产品需要 1 解决刚需，解决社交 2 用户体验感好（减少用户操作）3 用户粘性

默认选中 checked = “checked”

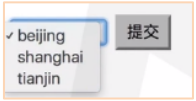
```
<h1>
    CHOOSE YOUR SEX!!!
</h1>
male:<input type="radio" name="sex" value="male" checked="
checked">
female:<input type="radio" name="sex" value="female">
<br>
```

form 表单里面的组件 select，是一个下拉菜单，可以选值

在这种情况，数据名写在 select 里面更好，因为是天生的父子结构，而 option 标签中间填的内容就相当于了一个数据值。如果写成<option value = “jc”>zjl</option>点提交，那么后台会使用 jc 这个属性

```
<select>
    <option></option>
    <option></option>
    <option></option>
</select>
```

```
<form method="get" action="">

    <h1>Province</h1>
    <select name="province">
        <option>beijing</option>
        <option>shanghai</option>
        <option>tianjin</option>
    </select>
```



浏览器

主流浏览器和内核，主流浏览器是有一定市场份额，并且有自己独立研发的内核

浏览器分 shell+内核

主流浏览器（必须有独立内核）市场份额大于 3%	**内核**
IE	trident
Friefox	Gecko
Google chrome	Webkit/blink
Safari	Webkit
Opera	presto

CSS 2.0

结构 html，样式 css，行为 js 相分离

css 相当于装修材料，cascading style sheet(层叠 样式 表)

注释掉的部分是浏览器不会看的，是备忘录的功能，快捷键是 ctrl+？

html 的注释是 <!—块注释 --> //行注释 快捷键 ctrl+?

css 注释方式仅有 /* 被注释的内容 */ 快捷键 ctrl+?

注释可以用来调节 bug，注释一行，调试一行

每天工作 50%在沟通，40-50%写代码，10%左右写报告

一个项目一个月的话，一般是一周做完，三周调试。编程为 10 分，7 分在调试，3 分在开发，时间也是这样分配的，一个项目开发时间*3 或者*3.5，调试时间很长

1、引入 css

1）行间样式

2）页面级 css

<head>

<style type = “text/css”>

</style>

</head>

这个 type = “text/css”的意思是告诉浏览器，我这里面是 css，可写可不写

3）外部 css 文件

打开一个文件以 leason.css 命名保存，一定要写.css

并且在头标签里面写

<head>

<link rel = “stylesheet” type = “text/css” href = “”>

</head>

rel = “stylesheet” type = “text/css” 是告诉浏览器我是 css，href 还是引入地址用的

border-radius: 50%方框变圆

代码快捷键：按个 div，或 link，再 tab
服务器要有一个地址，例如 192.168.000.001
而 www.baidu.com 是域名，是通过 dns 编译地址行程的
浏览器一般是下载一行执行一行，在下载到<link rel = “stylesheet” type = “text/css” href = “”>这个时，应该是同时下载 css 和 html，也就是异步加载

在计算机中：异步是指的同时进行，同步是指先一个，后一个（与生活相反）
异步 asynchronous;在计算机里面是同时执行的

2、选择器

下面介绍简单选择器

1）id 选择器

特点：一个元素只能有一个 id 值，一个 id 只对应一个元素，一一对应的
语法格式是 #（加上 id 后面的值是什么就填什么，如 only），就是选中这个 id 了
例 123
在 css 中写：

```
#only {
    background-color: red;
}
```

2）class 选择器（最常用的选择器）

语法格式 .class 就可以找到 class 选择器了
特点：一个 class 可以对应多个元素
例 123
 234

```
.demo{
    background-color: green;
}
```

在上面这种情况，123 和 234 都变成了绿色
例：如果想让第一个多一个值，多写一个 class 名
<div class = “demo demo1”> 123
 234

```
.demo{
    background-color: green;
}
.demo1{
    color: #f40;
}
```

3）标签选择器

语法格式 标签名{}
如果想选择 就写 div{}，如果想选择就写 span{}，不管被套多少层，都会被选择出来，而且是选择全部
例123

234

```
#only {
    background-color: red;
}

span{
    color:#F40;
    font-weight:bold;
}
```

123
234

4）通配符选择器

语法格式 *{}
*是任意的意思，此处是 all，所有的标签（包括<html>和<body>）

CSS 权重	
标签名	权重值
！important	infinity 正无穷
行间样式	1000
id 选择器	100
class 选择器、属性选择器、伪类选择器	10
标签选择器、伪元素选择器	1
通配符选择器	0
在计算机中，正无穷+1>正无穷	
如果权重值一样（优先级一样），会显示后面的 就是先来后到，谁在后面，谁（后面的）覆盖前面的——后面的会覆盖前面的	
在权重中，是 256 进制，是从 0 到 255 后变成 1 所以这里的 1000 不是一千，100 不是一百	

下面讲的是复杂选择器

5）父子选择器，派生选择器

语法格式 最外面的结构 外面的结构 里面的结构{} 就是一个父子结构
注意：父子选择器中，每一个层级，都不一样要是标签选择器，写 class 选择器也行，重要的是表达出来父子关系。而且这种父子关系有可能是间接地，也有可能是直接的

例只选出 div 里面的 span

```
<div>
    <span>123</span>
</div>
<span>234</span>
```

123
234

```
div span{
    background-color: red;
}
```

例选出里面的，右边两种方法都可以

```
<div class="wrapper">
    <strong class="box">
        <em>234</em>
    </strong>
</div>
<em>123</em>
```

```
.wrapper .box em{
    background-color: red;
}
```

```
div strong em{
    background-color: red;
}
```

234
123

例

```
<div>
    <em>1</em>
    <strong>
        <em>2</em>
    </strong>
</div>
```

```
div em{
    background-color: red;
}
```

12

```
div strong em{
    background-color: red;
}
```

12

第一种 css 写法理解是 div 下面所有的 em，包括直接的和间接的

6）直接子元素选择

如果写成 div > em{} 意思就是 div 下面直接一级的 em

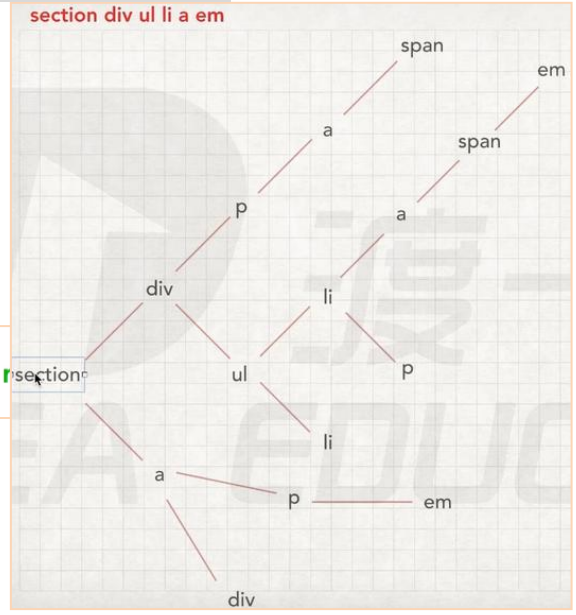
例

```
<div>
    <em>1</em>
    <strong>
        <em>2</em>
    </strong>
</div>
```

```
div > em{
    background-color: green;
}
```

看下面思考：浏览器是从右向左，还是从左向右找？

```
<section>
    <div>
        <p>
            <a href="">
                <span></span>
            </a>
        </p>
        <ul>
            <li>
                <a href="">
                    <span>
                        <em>1</em>
                    </span>
                </a>
                <p></p>
            </li>
            <li></li>
        </ul>
    </div>

    <a href="">
        <p>
            <em>2</em>
        </p>
        <div></div>
    </a>
</section>
```



答案：浏览器遍历的时候是从右向左找的(先看 em,再看 a,再看 li,再看 ul,再看 div)，更快更有效率

```
section div ul li a em{
    background-color: red;
}
```

7)并列选择器

格式 div.demo 在.之前不用写空格，这就是并列选择器

例在下面这种情况下，让 2 变红，如何写 css？

```
<div>1</div>
<div class="demo">2</div>
<p class="demo">3</p>
```

```
div.demo{
    background-color: red;
}
```

要不用 id 选择器，要不用并列选择器

例在下面这种情况下，有多少种写法能选出 1

```
<div class="wrapper">
    <div class="content">
        <em class="box" id="only
        ">1</em>
    </div>
</div>
```

```
#only{
    background-color: red;
}
```

```
.content em {
    background-color: green;
}
```

```
div.wrapper > div[class="content"] >
```

```
.wrapper > .content > .box{
    background-color: gray;
}
```

在并列选择器中，标签选择器和 id 选择器和 class 选择器在一起，标签选择器必须放前面

例

```
<div class="classDiv" id="idDiv"
>
    <p class="classP" id="idP">1
    </p>
</div>
```

```
#idDiv p{
    background-color: red;
}

.classDiv .classP{
    background-color: green;
}
```

上面这个显示红的，因为权重大只要写在同一行，权重值就会相加

例

```
<div class="classDiv" id="idDiv"
>
    <p class="classP" id="idP">1
    </p>
</div>
```

```
#idDiv p{
    background-color: red;
}

div .classP#idP{
    background-color: green;
}
```

上面显示绿色的

例

```
#idDiv p.classP{
    background-color: red;
}

div .classP#idP{
    background-color: green;
}
```

```
#idDiv p.classP{
    background-color: red!important;
}

div .classP#idP{
    background-color: green!importan
}
```

上面两种情况都会显示绿的，当权重一样时，后来先到，显示后面的

8）分组选择器，可以简化代码（常用功能）

例想把 1，2，3 都变红 ⟶

```
<em>1</em>
<strong>2</strong>
<span>3</span>
```

⟶

```
em{
    background-color: red;
}
strong{
    background-color: red;
}
span{
    background-color: red;
}
```

右侧这种写法耦合度太高了，很繁琐→

可以写成以下的分组选择器形式：

```
em, strong, span{
    background-color: red;
}
```

```
em,
strong,
span{
    background-color:
}
```

可以写成一行，但是开发中一般都竖着写，方便看

例

```
<div class="demo1"></div>
<div class="demo2"></div>
```

```
.demo1{
    width:100px;
    height:100px;
    background-color: red;
}

.demo2{
    width:100px;
    height:100px;
    background-color: green;
}
```

简化代码的写法

```
.demo1{
    background-color: red;
}

.demo2{
    background-color: green;
}

.demo1,
.demo2{
    width:100px;
    height:100px;
}
```

9）伪类选择器

格式 任意选择器:怎么动 后面有很多种写法

任意选择器:hover 是一种比较常见的写法

例 www.baidu.com

```
a:hover{            //a:hover 是当你鼠标移入到控制领域内，发生什么变化
    background-color:orange;
}          //也可以写成[href]:hover{}
```

例

```
<a href="www.baidu.com">www.baidu.com</a>
<a href="www.taobao.com">www.taobao.com</a>
<a href="www.jd.com">www.jd.com</a>
```

www.baidu.com www.taobao.com www.jd.com

```
a{
    text-decoration: none;
}
a:hover{
    text-decoration: underline;
    background-color: #f40;
    color:#fff;
    font-size:16px;
    font-weight:bold;
    font-family:arial;
    border-radius:10px;
}
```

例这个也是有权重的，因为行间样式权重大，所以 baidu 不显示下划线

```
<a style="text-decoration: none;" href="www.baidu.com">www.baidu.com</a>
<a href="www.taobao.com">www.taobao.com</a>
<a href="www.jd.com">www.jd.com</a>
```

www.baidu.com www.taobao.com

```
a:hover{
    text-decoration: underline;
    background-color: #f40;
    color:#fff;
    font-size:16px;
    font-weight:bold;
    font-family:arial;
    border-radius:10px;
}
```

border-radius:10px;是加一个圆角

3、CSS 代码块

以 为例，css 引入部分写成，用花括号包裹起来的是 css 代码区，在括号里面写 属性名：属性值； 每一个属性与属性之间用分号;隔开

```
div{
    font-size:12px;       //属性名：属性值；
    font-weight:bold;
}
```

下面来介绍一些属性，在 www. css88.com，相当于字典可以查

1）设置字体

属性名	属性值（举例）	效果/功能	
font-size:	12px	设置字体大小，默认是 16px，但是互联网字体一般用 12px 或 14px。这个设置的是字体的高度	
font-weight:	lighter	细体	字体的粗细是跟字体包有关
	normal	正常（默认值）	
	bold	加粗，跟效果是一样的	
	bolder	加更粗	
	100,200 到 900	从细到粗，都是整百来表达	
font-style:	italic;	斜体，em 这个标签本身就带有 italic 的样式	
font-family:	arial;	设置字体包的样式，这个是乔布斯发明的字体	
color:	见下面	注意字体颜色直接写 color，不写 font-color:red;	

字体颜色的 color 三种写法：
1、纯英文单词：red，green 等（一般不用）
2、颜色代码，如#ff4400; 其中每两位都是从 00 到 ff，分别对应红 r（从 00 到 ff），绿 g（从 00 到 ff），蓝 b（从 00 到 ff），这是一个十六进制的数，代表的是饱和度，如果没两位都可以重复，那么简写成#f40 淘宝红，#ffffff 白色，#000000 黑色。
3、颜色函数，如 color:rgb(255,255,255); //这是白色
rgb（0-255，0-255，0-255）里面的 0-255 是十进制数
transparent 是透明色			

border:	1px solid black;	这是一个复合属性。可以给容器加外边框。 第一位是 border-width 代表粗细； 第二位是 border-style 设置实心 solid，虚线； 第三位是 border-color 设置颜色 还可以单独写成 border-width:1px; border-style:dashed;代表虚线 dotted 是点	

思考：如何写出一个三角形？

```
<div><div>
div{
    width: 0px;
    height: 0px;
    border: 100px solid black;
    border-left-color:red;      //此处可以改为 transparent 透明色
    border-top-color:green;     //此处可以改为 transparent 透明色
    border-right-color:blue;    //此处可以改为 transparent 透明色
}
```

text-align:	left;	对齐方式：左对齐
	conter;	对齐方式：一行居中
	right;	对齐方式：右对齐
line-height:	16px;	单行文本所在的高度。 当 line-height=height （文本所占高度=容器高度）时，单行文本水平垂直居中
	1.2em；	单行文本所在的高度。意思是文字的行高是 1.2 倍行高
text-indent:	2em;	意思是首行缩进 2em（2 个文本单位）
单位的衡量标准：绝对单位（m，cm 等），相对单位（px，em 等） px 是像素的意思，一个像素只能显示一个颜色； 屏幕的分辨率，就是说的像素，国际标准是每英寸所能容纳的垂直像素点数。 em 是文本单位，1em = 1* font-size　　该文本的字体大小		
text-decoration:	line-through;	中划线
	none;	没有线
	underline;	下划线
	overline;	上划线（基本没用）
cursor:	pointer;	光标定位值（cursor: pointer;鼠标变成一个小手）

<del></del>基本不用（不符合行为、样式、结构相分离）

例

```
<del>原价50元</del>
<span>原价50元</span>
```

```
span{
    text-decoration: line-through;
}
```

例

```
<a href="www.baidu.com">www.baidu.com</a>

<span>www.baidu.com</span>
```

```
span{
    color:rgb(0, 0, 238);
    text-decoration: underline;
}
```

www.baidu.com

4、标签的分类（归类）基本的

1）行级元素，也叫内联元素 inline

特点 A 内容决定元素所占位置

B 不可以通过 CSS 改变宽高

例 span，strong，em，a，del

行级元素自带 CSS 属性，可更改自带属性

span 自带隐藏属性 display:inline;　可以通过改成 block 变成块级元素

2）块级元素，block

特点 A 独占一行

B 可以通过 CSS 改变宽高

例 div，p，ul，li，ol，form，address

3）行级块元素 inline-block

特点 A 内容决定大小

B 可以改变宽高

我们可以通过 display 更改元素属性，如果写成 display:none;元素就没有了

<img src = “”>　　//一般只设置宽或高，另外一个就等比例缩放了

例：解决图片之间有缝隙的 bug

```
<img src="fy.jpg">
<img src="fy.jpg">
<img src="fy.jpg">
<img src="fy.jpg">
```

```
img{
    border:0;
    width:100px;
    height:200px;
    margin-left:-6px;
}
```

上面这种解决方法不好，在上传到线上的时候就不准了。

凡是自带 inline 特效的元素，都有文字特性，有文字特性就会被分割

写成<img scr = “”><img scr = “”><img scr = “”>这样就可以解决了（在一行写，并且不写空格），写成一行，图片间的空格就没有了

原理 :在代码上传服务器时会进行两种压缩方法 ,A 字符压缩(如把 img 用 A 代替）；B 去空格，去回车

5、编程思路：

1 小白式一：先写 html，再写 css

2 小白式二：一边写 html，一边写 css

3 最好的编程思想：先写 css 定义颜色尺寸等，再写 html

也就是先定义功能，后选配功能（方便团队合作）

一个 html 可以引入多个 css，一个 css 可以对应很多 html

例

```
<div class="red size1"></div>
<div class="green size2"></
div>

<div class="gray size3"></div
>
```

→

```
.green{
    background-color: green;
}

.gray{
    background-color: gray;
}

.size1{
    width:100px;
    height:100px;
}

.size2{
    width:200px;
    height:200px;
}
.size3{
    width:300px;
    height:300px;
}
```

6、初始化元素（改变 html 自带的系统属性，变成自定义标签）

天生自带的	去掉自己带的

本身自定义是斜体	以下写法 em 就变成了红字标签
	em{
	font-style:normal;
	color:#c00;
	}

  1
  2
  3
 	
	ul{
	list-style:none; //去掉圆点
	padding:0;
	margin:0;
	}

123	a{
	text-decoration:none;
	color:#424242;
	}

*通配符选择器能**初始化所有的标签**	*{
	padding:0;
	margin:0;
	text-decoration:none;
	list-style:none;
	}

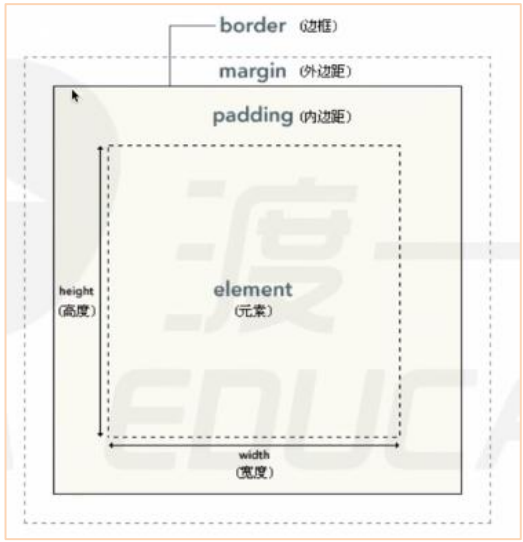
7、盒子模型

1）盒子的组成三大部分：

A 盒子壁 border

B 盒子内边距 padding

C 盒子内容 width+height



2）盒子模型（四部分组成）

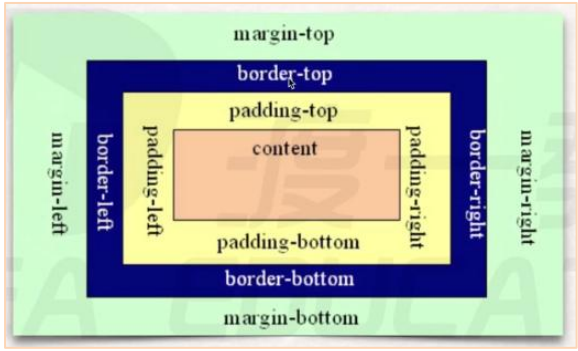
A 盒子壁 border

B 盒子内边距 padding

C 盒子内容 content=width+height

D 盒子外边距 margin

谷歌浏览器控制台最好



padding:上 右 下 左；（顺时针）	padding:上下 左右；（两个值）
padding:上 左右 下；（三个值，中间代表左右），一个值的时候设置四个方向	
padding-left:10px; 代表单独设置左侧，也可写成 padding:0 0 0 10px;	
margin 和 border-width 的设置方法和 padding 是一样的	
如 border-width:100px 10px 30px 50px; //这样盒子壁四边都不一样了	

例

```
<div>
韭菜秋葵香菜韭菜秋葵香菜韭菜秋葵香菜
</div>
```

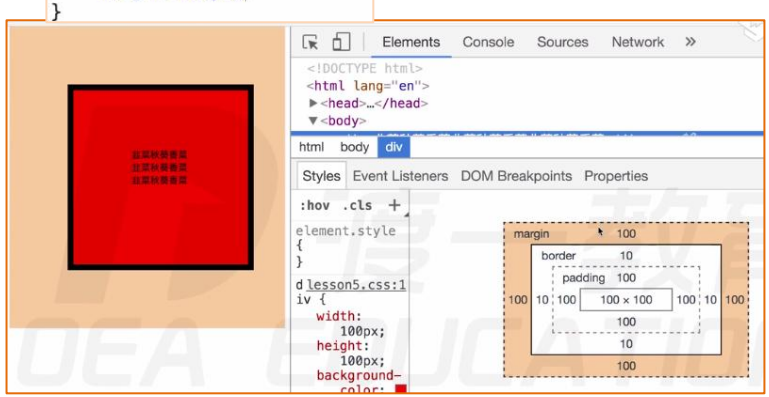
```
div{
    width:100px;
    height:100px;
    background-color: red;
    border:10px solid black;
    padding:10px;
}
```

上面这种，文字和盒子之间隔出了距离，就是 padding，padding 内边距是不能放内容的。内容是放到内容区 content 的。

浏览器是以左边和上边当做墙的，设置了 margin 就会推动盒子移动

如

```
div{
    width:100px;
    height:100px;
    background-color: red;
    border:10px solid black;
    padding:100px;
    margin:100px;
}
```



例现在这两盒子重叠在一起了

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

```
.wrapper{
    width:100px;
    height:100px;
    background-color: red;
}

.content{
    width:100px;
    height:100px;
    background-color: black;
}
```

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

```
.wrapper{
    border:10px solid green;
    width:100px;
    height:100px;
    background-color: red;
}

.content{
    width:100px;
    height:100px;
    background-color: black;
}
```

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

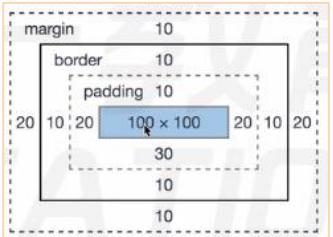
```
.wrapper{
    border:10px solid green;
    width:100px;
    height:100px;
    background-color: red;
    padding:100px;
}

.content{
    width:100px;
    height:100px;
    background-color: black;
}
```

3）盒模型的计算（可视区域的宽高，真实的大小）

例求 realWidth，realHeight

```
<div></div>
```

```
div{
    width:100px;
    height:100px;
    background-color: red;
    border:10px solid black;
    padding:10px 20px 30px;
    margin:10px 20px;
}
```



realWidth：100+10+10+20+20px=160px

realHeight：100+10+10+10+30px=160px

margin 不算盒子，所以计算的时候不能算上 margin

例

```
<div id="my-defined"></div>
```

```
#my-defined{
    width:100px;
    height:100px;
    padding:0 100px;
    margin:10px 20px 30px 40px;
    border:1px solid orange;
    background-color:orange;}
```

realWidth：100+100+100+1+1px=302px

realHeight：100+0+0+1+1px=102px

例拓展：远视图

```
<div class="wrapper">
    <div class="box">
        <div class="content">
            <div class="
            content1"></div>
        </div>
    </div>
</div>
```

```
.wrapper{
    width:50px;
    height:50px;
    background-color: #000;
    padding:10px;
}
```

```
.content1{
    height:10px;
    width:10px;
    background-color: #0f0;
}

.content{
    width:10px;
    height:10px;
    padding:10px;
    background-color: #000;
}

.box{
    width:30px;
    height:30px;
    background-color: #0f0;
    padding:10px;
}
```

例通过调整 margin，调整距离

```
<span class="demo">123</span>

<span class="demo2">123</span
>
```

```
.demo{
    background-color: red;
}
```

123 123

```
.demo{
    background-color: red;
    margin-right:10px;
}

.demo2{
    margin-left:40px;
}
```

123 123

8、定位 position

1）position:absolute;绝对定位定位：脱离原来位置定位

absolute 和 relative 都是层模型

例

```
<div></div>
```

```
div{
    width:100px;
    height:100px;
    background-color: red;
}
```

例：写成下面这样，看起来并没有什么变化，实际上已经改变了性质

下面的这个红框离浏览器是有一个边界的，这个是 body 自带的属性，是 margin:8px;

一般我们需要把 body 自带的 margin 去掉

```
<div></div>
```

```
div{
    position:absolute;
    width:100px;
    height:100px;
    background-color: red;
}
```

定位元素需要配合 right，left，top，bottom 使用（浏览器 y 轴是向下的），但是一般是不设置 bottom 的，因为浏览器一般没有底部

例：设置成 right:100px;是右边线距离 100px，left 是左边线距离 100px

```
<div></div>
```

```
div{
    position:absolute;
    left:100px;
    top:100px;
    width:100px;
    height:100px;
    background-color: red;
}
```

例：最初的形式（下面讲的是层模型部分）

```
<div class="demo"></div>
<div class="box"></div>
```

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

.demo{
    width:100px;
    height:100px;
    background-color: red;
    opacity: 0.5;
}
.box{
    width:150px;
    height:150px;
    background-color: green;
}
```

这就是层模型的特点，当一个元素 absolute 以后，就脱离了原来的层，跑到上一层面去了，所以在同一点出现了

例

```
<div class="demo"></div>
<div class="box"></div>
```

```
.box{
    width:150px;
    height:150px;
    background-color: green;
}
```

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

.demo{
    position:absolute;
    width:100px;
    height:100px;
    background-color: red;
    opacity: 0.5;
}
```

例 **absolute 脱离原来位置定位**

```
<div class="demo"></div>
<div class="box"></div>
```

```
.box{
    width:150px;
    height:150px;
    background-color: green;
}
```

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

.demo{
    position:absolute;
    left:100px;
    top:100px;
    width:100px;
    height:100px;
    background-color: red;
    opacity: 0.5;
}
```

2）position:relative;定位。relative 是保留原来位置进行定位

这种情况下 relative 和 absolute 是一样的

例

```
<div class="demo"></div>
```

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

```
.demo{
    position: relative;
    left:100px;
    top:100px;
    width:100px;
    height:100px;
    background-color: red;
}
```

例

```
<div class="demo"></div>
<div class="box"></div>
```

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

```
.demo{
    position:relative;
    width:100px;
    height:100px;
    background-color: red;
    opacity: 0.5;
}
.box{
    width:150px;
    height:150px;
    background-color: green;
}
```

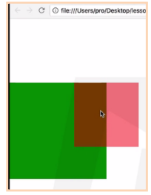
上下两种情况说明了，relative 是保留原来位置进行定位类似于灵魂出窍

例

```
<div class="demo"></div>
<div class="box"></div>
```

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

```
.demo{
    position:relative;
    left:100px;
    top:100px;
    width:100px;
    height:100px;
    background-color: red;
    opacity: 0.5;
}
.box{
    width:150px;
    height:150px;
    background-color: green;
}
```



例**这是最初的样式**

```
<div class="wrapper">
    <div class="content">
        <div class="box"></
        div>
    </div>
</div>
```

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

```
.wrapper{
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    width:100px;
    height:100px;
    background-color: black;
}

.box{
    width:50px;
    height:50px;
    background-color: yellow;
}
```

例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    width:50px;
    height:50px;
    background-color: yellow;
}
```

例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    width:50px;
    height:50px;
    background-color: yellow;
}
```

例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    position: absolute;
    left:50px;
    width:50px;
    height:50px;
    background-color: yellow;
}
```

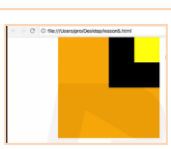
例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    position: relative;
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    position: absolute;
    left:50px;
    width:50px;
    height:50px;
    background-color: yellow;
}
```



例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    position: relative;
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    position: absolute;
    left:50px;
    width:50px;
    height:50px;
    background-color: yellow;
}
```


例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    position: relative;
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    position: absolute;
    left:50px;
    width:50px;
    height:50px;
    background-color: yellow
}
```

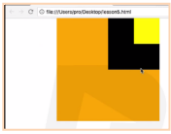
例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    position: relative;
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
position: relative;
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    position: absolute;
    left:50px;
    width:50px;
    height:50px;
    background-color: yellow;
}
```



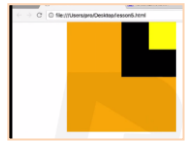
例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    position: relative;
    left:50px;
    width:50px;
    height:50px;
    background-color: yellow;
}
```



例

```html
<div class="wrapper">
    <div class="content">
        <div class="box"></div>
    </div>
</div>
```

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

```css
.wrapper{
    position: relative;
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange;
}

.content{
    position: relative;
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```css
.box{
    position: relative;
    left:50px;
    width:50px;
    height:50px;
    background-color: yellow;
}
```

3）absolute 和 relative 比较

absolute 定位：脱离原来位置定位。是相对于最近的有定位的父级进行定位；如果没有有定位的父级元素，就相对于文档进行定位

relative 定位：保留原来的位置进行定位，相对于自己原来（出生的）的位置进行定位

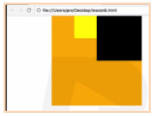
例

```
<div class="wrapper">
    <div class="content">
        <div class="box"></
        div>
    </div>
</div>
```

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

```
.wrapper{
    margin-left:100px;
    width:200px;
    height:200px;
    background-color: orange
}

.content{
    margin-left:100px;
    width:100px;
    height:100px;
    background-color: black;
}
```

```
.box{
    position: relative;
    right:50px;
    width:50px;
    height:50px;
    background-color: yellow;
}
```



经验：什么时候用 relative，什么时候用 absolute？

如果想定位，最好找参照物。

absolute 定位：脱离原来位置定位。是相对于最近的有定位的父级进行定位；如果没有有定位的父级元素，就相对于文档进行定位

relative 定位：保留原来的位置进行定位，相对于自己原来（出生的）的位置进行定位

我们一般用 relative 作为参照物（不用 relative 进行定位），用 absolute 进行定位

给一个元素只设置一个 relative，不设置 left，right，top，bottom，看起来是对这个元素没什么影响的，但是它保留了原来的位置，就对后续元素没有什么影响

absolute 可以任意的调整自己的参照物，更加灵活，所以用于定位

想让谁成为基地，就给谁 relative 定位，并且不设置方向

4）position:fixed;固定定位

可以用作小广告，不管滚动条怎么动，它都在一个固定的位置上面

需要搭配 right，left，top，bottom 使用

例

```
<div>增强老邓身体机能的神奇药物...
</div>

br*100
```

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

div{
    width:50px;
    height:200px;
    background-color: red;
    color:#fff;
}
```

例

```
<div>增强老邓身体机能的神奇药物...
</div>

br*100
```

加上一个 fixed 定位，

配合 left，top。

就固定不动了

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

div{
    position: fixed;
    left:0;
    top:300px;
    width:50px;
    height:200px;
    background-color: red;
    color:#fff;
}
```

问题：居中怎么做？无论浏览器怎么伸缩，都是居中不变。

例像下面这种方法，我们定位的 50%是定位的左上的顶点，所以从元素的正中间到左上顶点还差半个身位

```
<div></div>
```

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

```
div{
    position: absolute;
    left:50%;
    top:50%;
    width:100px;
    height:100px;
    background-color: red;
}
```

例那么 div 离浏览器左边就有 10px

```
<div></div>
```

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

```
div{
    width:100px;
    height:100px;
    background-color: red;
    margin-left:10px;
}
```

例

```
<div></div>
```

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

```
div{
    width:100px;
    height:100px;
    background-color: red;
    margin-top:50px;
}
```

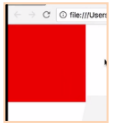
那么 div 离浏览器善变边就有 50px

例

```
<div></div>
```

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

```
div{
    width:100px;
    height:100px;
    background-color: red;
    margin-right:10px;
}
```



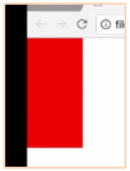
右侧有个外边距 10px，其他东西不能占这个位置，但是不会推动 div 移动

例

```
<div></div>
```

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

```
div{
    width:100px;
    height:100px;
    background-color: red;
    margin-left:-50px;
}
```



margin-left 能推动元素，margin-left:-就是一个反向运动

margin-left:10px;是向右推动 10px，　　margin-left:-10px; 是向左推动 10px

例下面这种写法无论浏览器怎么伸缩，div 都是居中不变。

```
<div></div>
```

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

```
div{
    position: absolute;
    left:50%;
    top:50%;
    width:100px;
    height:100px;
    background-color: red;
    margin-left:-50px;
    margin-top:-50px;
}
```

```
div{
    position: fixed;
    left:50%;
    top:50%;
    width:100px;
    height:100px;
    background-color: red;
    margin-left:-50px;
    margin-top:-50px;
}
```

因为 width:100px，所以 margin-left:-50px;　　是正好一半的值。

而 left:50%;和 top:50%;是相对浏览器的定位用的，

所以居中是正中，跟随页面改变而居中

而把 absolute 改成 fixed 就是一个随着滚动条移动但是永远居中的小广告

作业：写出五环，让五环在屏幕中永远居中

z-index 是表示元素在第几层，默认是 0，值越大，越靠近你。但是只在 position 中可以使用

border:10px solid black;

boder-radius:50%;　　//这样设置就变成了圆

答案

```
<div class="plat">
    <div class="circle1"></div>
    <div class="circle2"></div>
    <div class="circle3"></div>
    <div class="circle4"></div>
    <div class="circle5"></div>
</div>
```

```
.circle1{
    border-color:red;
    left:0;
    top:0;
}
```

```
.circle2{
    border-color:green;
    left:130px;
    top:0;
    z-index:3;
}
```

```
.circle3{
    border-color:yellow;
    left:260px;
    top:0;
}

.circle4{
    border-color:blue;
    left:65px;
    top:70px;
}

.circle5{
    border-color:purple;
    left:195px;
    top:70px;
}
```

```
*{
    margin:0;
    padding:0;
}
.plat{
    position: absolute;
    left:50%;
    top:50%;
    margin-left:-190px;
    margin-top:-93px;
    height:186px;
    width:380px;
}

.circle1,
.circle2,
.circle3,
.circle4,
.circle5{
    position: absolute;
    width:100px;
    height:100px;
    border:10px solid black;
    border-radius:50%;
}
```

9、两栏布局

例粉色先出生，黑色后出生

```
<div class="right"></div>
<div class="left"></div>
```

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

```
.right{
    width:100px;
    height:100px;
    background-color: #fcc;
}

.left{
    height:100px;
    background-color: #123;
}
```

例给粉色加了个 absolute，

黑色就上去了，粉色压住黑色了

```
<div class="right"></div>
<div class="left"></div>
```

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

```
.right{
    position: absolute;
    width:100px;
    height:100px;
    background-color: #fcc;
}

.left{
    height:100px;
    background-color: #123;
}
```

例粉色压住黑色

```
<div class="right"></div>
<div class="left"></div>
```

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

```
.right{
    position: absolute;
    right:0;
    width:100px;
    height:100px;
    background-color: #fcc;
}

.left{
    height:100px;
    background-color: #123;
}
```

例黑的是自适应元素，随着浏览器自适应，这是块级元素的特点

```
<div class="right"></div>
<div class="left"></div>
```

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

```
.right{
    position: absolute;
    right:0;
    width:100px;
    height:100px;
    background-color: #fcc;
}
```

让出了一个粉色的距离

```
.left{
    margin-right:100px;
    height:100px;
    background-color: #123;
}
```

这样就实现了一个两栏布局

opacity:0.5;是设置透明色

两栏布局的特点

1）必须先要定位过去

2）要让在下面被压着的元素，把位置让出来（上面元素的位置）

例这种是把 div 换了个位置的情况

```
<div class="left"></div>

<div class="right"></div>
```

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

```
.right{
    position: absolute;
    right:0;
    width:100px;
    height:100px;
    background-color: #fcc;
    opacity: 0.5;
}

.left{
    margin-right:100px;
    height:100px;
    background-color: #123;
}
```

是黑色先出生，粉色后出生

粉色调完了 position 以后，但是他出生在第二行，不会到第一行去

所以像这样反着写不能形成两栏布局

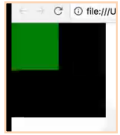
10、解决两个 bug：margin 塌陷和 margin 合并

1）margin 塌陷

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    width:100px;
    height:100px;
    background-color: black;
}

.content{
    width:50px;
    height:50px;
    background-color: green;
}
```



例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    margin-left:100px;
    margin-top:100px;
    width:100px;
    height:100px;
    background-color: black;
}

.content{
    width:50px;
    height:50px;
    background-color: green;
}
```

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    margin-left:100px;
    margin-top:100px;
    width:100px;
    height:100px;
    background-color: black;
}

.content{
    margin-left:50px;
    width:50px;
    height:50px;
    background-color: green;
}
```

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    margin-left:100px;
    margin-top:100px;
    width:100px;
    height:100px;
    background-color: black;
}

.content{
    margin-left:50px;
    margin-top:100px;
    width:50px;
    height:50px;
    background-color: green;
}
```

设置 margin-top:100px;时失效了绿色的方块并没有跟着动

而当 margin-top:150px;时（大于父级高），那么绿色的就会带着父级黑色的一起动

margin 塌陷：

父子嵌套元素垂直方向的 margin，父子元素是结合在一起的，他们两个会取其中最大的值

正常情况应该是父级相对于浏览器进行定位，子级应该相对于父级定位的

但是 margin 塌陷是在父级相对于浏览器进行定位时，子级没有相对于父级定位，就像父级的棚子没有了一样

子级相对于父级，就像塌陷了一样

margin 塌陷解决方案：不能用旁边的土法子

应该用 bfc，改变父级的渲染规则

block format context

块级格式化上下文

触发 bfc，特定的盒子会遵循另一套规则

margin 塌陷的解决方法只改了 css

```
.wrapper{
    margin-left:100px;
    margin-top:100px;
    width:100px;
    height:100px;
    background-color: black;
    border-top:1px solid red;
}
        土法子
.content{
    margin-left:50px;
    margin-top:150px;
    width:50px;
    height:50px;
    background-color: green;
}
```

弥补 margin 塌陷，可以使用 bfc，如何触发一个盒子的 bfc？

1）position:absolute;

2）display:inline-block;

3）float:left/right;　　//浮动

4）overflow:hidden;　　//溢出盒子的部分要隐藏展示

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    margin-left:100px;
    margin-top:100px;
    width:100px;
    height:100px;
    background-color: black;
    /*overflow:hidden;*/
}

.content{
    margin-left:75px;
    /*margin-top:150px;*/
    width:50px;
    height:50px;
    background-color: green;
}
```

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    margin-left:100px;
    margin-top:100px;
    width:100px;
    height:100px;
    background-color: black;
    overflow:hidden;
}

.content{
    margin-left:75px;
    /*margin-top:150px;*/
    width:50px;
    height:50px;
    background-color: green;
}
```

加了 overflow:hidden;后，溢出部分隐藏

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    margin-left:100px;
    margin-top:100px;
    width:100px;
    height:100px;
    background-color: black;
    overflow:hidden;
}

.content{
    margin-left:50px;
    margin-top:50px;
    width:50px;
    height:50px;
    background-color: green;
}
```

改变了大盒子的渲染规则

这四种方法都能触发 bfc，但是使用的时候都带来了新的麻烦

在具体情况中，针对需求，哪个触发方式没有影响，就用哪个解决塌陷问题

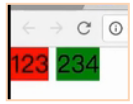
2）margin 合并

例

```
<span class="box1">123</span>
<span class="box2">234</span>
```

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

```
.box1{
    background-color: red;
}

.box2{
    background-color: green;
}
```



例

```
<span class="box1">123</span>
<span class="box2">234</span>
```

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

```
.box1{
    background-color: red;
    margin-right:100px;
}

.box2{
    background-color: green;
    margin-left:100px;
}
```

123 和 234 之间有 200px 的 margin。正常情况，123 和 234 的区域不能共用

例初始情况：

```
<span class="box1">123</span>
<span class="box2">234</span>
<div class="demo1">1</div>
<div class="demo2">2</div>
```

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

```
.box1{
    background-color: red;
    margin-right:100px;
}

.box2{
    background-color: green;
    margin-left:100px;
}
```

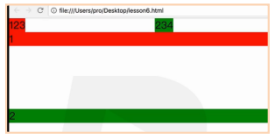
```
.demo1{
    background-color: red;
}

.demo2{
    background-color: green;
}
```

例将初始情况改成这样：

```
.demo1{
    background-color: red;
    margin-bottom:100px;
}
```

例将初始情况改成这样：

```
.demo1{
    background-color: red;
    margin-bottom:100px;
}

.demo2{
    background-color: green;
    margin-top:100px;
}
```



结果 demo1 和 demo2 之间的 margin 还是 100px;

两个兄弟结构的元素，他俩垂直方向的 margin 是合并的

margin 合并依然使用 bfc 解决

例

```
<span class="box1">123</span>
<span class="box2">234</span>
<div class="demo1">1</div>
<div class="wrapper">
    <div class="demo2">2</div>
</div>
```

```
.demo1{
    background-color: red;
    margin-bottom:200px;
}

.demo2{
    background-color: green;
    margin-top:100px;
}

.wrapper{
    overflow:hidden;
}
```

给 demo2 加上一层并加了 overflow

两个之间的 margin 变 200px

或者把这两个 div 都加一层 bfc

例

```
<span class="box1">123</span>
<span class="box2">234</span>
<div class="wrapper">
    <div class="demo1">1</div>
</div>
<div class="wrapper">
    <div class="demo2">2</div>
</div>
```

```
.box1{
    background-color: red;
    margin-right:100px;
}

.box2{
    background-color: green;
    margin-left:100px;
}
```

```
.demo1{
    background-color: red;
    margin-bottom:200px;
}

.demo2{
    background-color: green;
    margin-top:100px;
}

.wrapper{
    overflow:hidden;
}
```

两个 div 之间的 margin 变 200px

但是上面这两种解决 margin 合并的方法，改变了 HTML 的结构，这在开发中是不允许的

实际开发中，在 margin 合并这个 bug 上，我们不用 bfc（不能改变 html 的结构）

假如我们需要两个 div 直接有 300px ,那就设置上面的 margin-bottom:300px;来解决距离的问题

三大模型是盒模型，层模型，浮动模型

11、浮动模型 float:left/right;

例

```
<div class="wrapper">
    <div class="content"></div>
    <div class="content"></div>
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    width:300px;
    height:300px;
    border:1px solid black;
}
```

例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
</div>
```

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

```
.wrapper{
    width:300px;
    height:300px;
    border:1px solid black;
}

.content{
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

例 float 浮动用法：让元素排队

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
</div>
```

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

```
.wrapper{
    width:300px;
    height:300px;
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

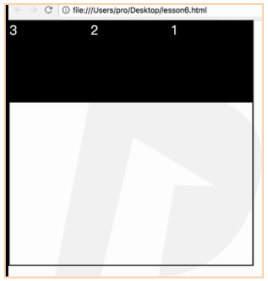
float：left 使方块按 123 排序

例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
</div>
```

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

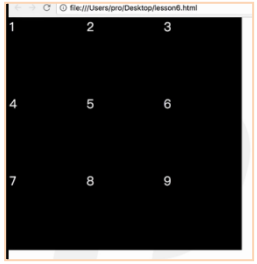
```
.wrapper{
    width:300px;
    height:300px;
    border:1px solid black;
}

.content{
    float:right;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```



float：right 使方块按 321 排序

例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
</div>
```




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

```
.wrapper{
    width:300px;
    height:300px;
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

float：left 使方块按下面方法排序：

123

456

789

例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
</div>
```


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

```
.wrapper{
    width:300px;
    height:300px;
    border:1px solid black;
}

.content{
    float:right;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

float：right 使方块按下面方法排序：

321

654

987

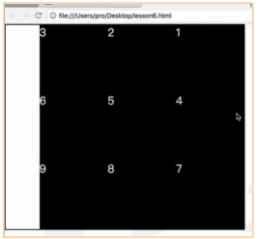
float:left 是让元素自左向右排序，float:right 是让元素自右向左排序

例扩宽了大容器

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
</div>
```

```
.wrapper{
    width:350px;
    height:300px;
    border:1px solid black;
}

.content{
    float:right;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

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

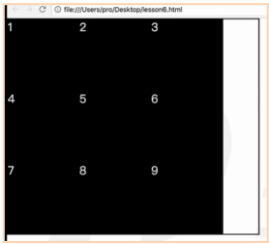


例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
</div>
```

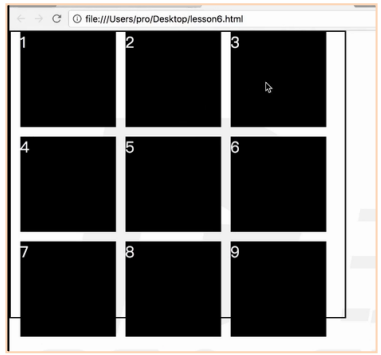
```
.wrapper{
    width:350px;
    height:300px;
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

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



例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
</div>
```




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

```
.wrapper{
    width:350px;
    height:300px;
    border:1px solid black;
}

.content{
    float:left;
    margin-left:10px;
    margin-bottom:10px;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

例浮动元素上可以加 margin-left 或 margin-top，这下面几个图片就是用浮动做的

例

```html
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
</div>
```

```css
.wrapper{
    width:400px;
    height:300px;
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

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

上面是按 1234,5678,9 这样站队的
说明 float 站队时，够站一个就站一个，不够站一个就换行
float 站队的边界就是父级元素的边界

例

```html
<div class="box"></div>
<div class="demo"></div>
```

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

```css
.box{
    width:100px;
    height:100px;
    background-color: black;
}
.demo{
    width:150px;
    height:150px;
    background-color: green;
}
```

例

```html
<div class="box"></div>
<div class="demo"></div>
```

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

```css
.box{
    float:left;
    width:100px;
    height:100px;
    background-color: black;
    opacity: 0.5;
}
.demo{
    width:150px;
    height:150px;
    background-color: green;
}
```

float 元素也分层了。跟 absolute 效果是类似的

浮动元素：

1）浮动元素产生了浮动流

（浮动流产生的效果）所有产生了浮动流的元素，块级元素看不到他们。

产生了 bfc 的元素和文本类属性（带有 inline 属性就是文本类属性）的元素以及文本都能看到浮动元素。

例

```html
<div class="box"></div>123
```

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

```css
.box{
    float:left;
    width:100px;
    height:100px;
    background-color: black;
    opacity: 0.5;
}
```

文本能看到浮动元素。浮动元素不意味着分层，只能说明他产生了浮动流。

例

```html
<div class="box"></div>
<img src="fy.jpg">
```

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

```css
.box{
    float:left;
    width:100px;
    height:100px;
    background-color: black;
    opacity: 0.5;
}
```

例

```html
<div class="box"></div>
<div class="demo"></div>
```

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

```css
.box{
    float:left;
    width:100px;
    height:100px;
    background-color: black;
    opacity: 0.5;
}
.demo{
    float:left;
    width:150px;
    height:150px;
    background-color: green;
}
```

例

```html
<div class="box"></div>
<div class="demo"></div>
```

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

```css
.box{
    float:left;
    width:100px;
    height:100px;
    background-color: black;
    opacity: 0.5;
}
.demo{
    display: inline-block;
    width:150px;
    height:150px;
    background-color: green;
}
```

例

```html
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
</div>
```

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

```css
.wrapper{
    border:1px solid black;
}

.content{
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

此时变成了自适应的容器了，靠内容撑起容器

例

```html
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
</div>
```

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

```css
.wrapper{
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
```

父级包不住了，因为父级是块级元素，看不到浮动元素

想父级包住子集，方法一是加 height（只能解决一小部分）；方法二是清除浮动流

用 clear 清除周边的浮动流

clear：left；或 clear：right；或 **clear：both；**　　//一般直接写成 both

例

```html
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <p></p>
</div>
```

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

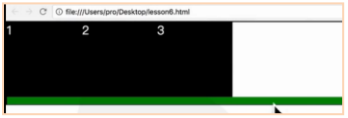
```css
.wrapper{
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
p{
    border-top:10px solid green;
}
```

例

```html
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <p></p>
</div>
```

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

```css
.wrapper{
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
p{
    border-top:10px solid green;
    clear:both;
}
```



例

```html
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <p></p>
</div>
```

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

```css
.wrapper{
    border:1px solid black;
}

.content{
    float:left;
    color:#fff;
    background-color: black;
    width:100px;
    height:100px;
}
p{
    border-top:0 solid green;
    clear:both;
}
```

父级包住浮动元素的解决方法：

p 元素只要逻辑上在这里出生，并且清除浮动流，父级自然而然被他撑开

不管有多少个 div，大盒子都会被撑开

正常来说，在清除浮动的时候，是不能像上面那样通过增加<p>来改变 html 结构，从而清除浮动，这种方式在开发中是不能使用的

伪元素选择器

伪元素天生就存在于任意一个元素（标签 tag element）里面。天生是行级元素。

为什么叫伪元素？因为他的元素结构是存在的，但是他没写到 HTML 里面，他可以被 css 选中，并正常操作，但是他没有 html 结构，所以叫伪元素。

他有两个特殊的伪元素 before 和 after

伪元素可以跟正常元素差不多可以一起来使用，但是他没有正常的元素结构

一个标签诞生的时候，他就有了伪元素，伪元素存在于逻辑的最前（after）和逻辑的最后（before），只是一般看不见，除非我们通过 css 选中，并操作他

写成 span::before{}就是选择了 span 的 before 伪元素，里面有一个必须要有的值是 content: “” 。而 content 也只能用在伪元素中

例

```html
<span>很帅</span>
```


```css
span::before{
    content:"成哥";
}

span::after{
    content:",是的，的确是这样!";
}
```

伪元素可以跟正常元素一样操作

```css
span::before{
    content:"";
    display: inline-block;
    width:100px;
    height:100px;
    background-color: red;
}
```

而为了清除 float 的浮动流，就可以用到伪元素，也是最好的选择

clear 很特殊，想让他生效，必须是块级元素才可以，而::after 是行级元素

例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
</div>
```

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

```
.wrapper::after{
    content:"";
    clear:both;
    display: block;
}

.wrapper{
    border:1px solid black;
}

.content{
    float:left;
    width:100px;
    height:100px;
    background-color: black;
    color:#fff;
}
```

浮动元素用完，尽量都清除浮动流

除了 float 以外，还有其他方式将他包裹住——能触发 bfc 的元素

例

```
<div class="wrapper">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
</div>
```


```
.wrapper{
    display: inline-block;
    border:2px solid red;
}

.content{
    float:left;
    width:100px;
    height:100px;
    background-color: black;
    color:#fff;
}
```

```
.wrapper{
    display: inline-block;
    border:2px solid red;
}
```

```
.wrapper{
    position:absolute;
    border:2px solid red;
}
```

上面都是紧紧的包住，而不是把他撑开了

凡是设置了 position:absolute； 或 float：left/right；他会打系统内部，把元素转换成 inline-block，所以由内容来决定宽高，改成 block 才能独占一行

例

```
<span>123</span>
```


```
span{
    position: absolute;
    width:100px;
    height:100px;
    background-color: red;
}
```

```
span{
    float: left;
    width:100px;
    height:100px;
    background-color: red;
}
```

浮动最开始是实现文字环绕的

例

```
<img class="img1" src="fy.jpg">
范冰冰，1981年9月16日出生于山东青岛，
中国影视女演员、制片人、流行乐女歌手，
```

```
img{
    width:100px;
}
```

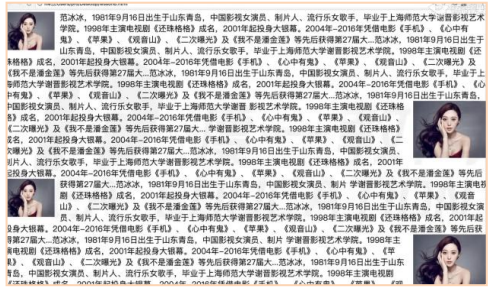
例

```
<img class="img1" src="fy.jpg">
范冰冰，1981年9月16日出生于山东青岛，
中国影视女演员、制片人、流行乐女歌手，
```

```
img{
    float:left;
    width:100px;
}
```

例 多放几个图片

```
.img1{
    margin-right:10px;
    float:left;
    width:100px;
}

.img2{
    float:right;
    width:150px;
    margin-left:10px;
}
```



11、一些项目及细节

项目一：


```
<ul class="nav">
    <li class="list-item">
        <a href="#">天猫</a>
    </li>
    <li class="list-item">
        <a href="#">聚划算</a>
    </li>
    <li class="list-item">
        <a href="#">天猫超市</a>
    </li>
</ul>
```

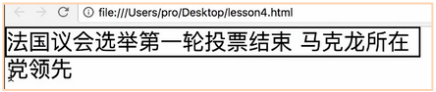
```
*{
    margin:0;
    padding:0;
    color:#424242;
    font-family:arial;
}
a{
    text-decoration: none;
}
.nav::after{
    content:"";
    display: block;
    clear:both;
}
.nav{
    list-style:none;
}
```

```
.nav .list-item{
    float:left;
    margin:0 10px;
    height:30px;
    line-height: 30px;
}
.nav .list-item a{
    padding:0 5px;
    color:#f40;
    font-weight:bold;
    height:30px;
    display: inline-block;
    border-radius:15px;
}

.nav .list-item a:hover{
    background-color: #f40;
    color:#fff;
}
```

项目二：当文字溢出容器，要打点展示

1）单行文本（唯一的技术）

```
<p>法国议会选举第一轮投票结束
马克龙所在党领先</p>
```




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

p{
    width:300px;
    height:20px;
    line-height: 20px;
    border:1px solid black;
}
```

想让单行文本溢出文字，打点，步骤：

首先先让文本失去换行功能，white-space:nowrap;

再溢出部分不展示隐藏，overflow:hidden;

最后把溢出部分打点，text-overflow:ellipsis;

法国议会选举第一轮投票结束 马克龙所…

```
p{
    width:300px;
    height:20px;
    line-height: 20px;
    /*border:1px solid black;*/

    white-space: nowrap;
    overflow: hidden;
    text-overflow:ellipsis;
}
```

2）多行文本（做截断）

用手写上去的，估算出来的，老版本的浏览器支持不太好，所以不用这种方法

移动端上可以实现

height 是 line-height 的两倍，就展示两行，内容区域高是单行文本的高度整数倍。算好了高度就可以做截断了

例

```
<p>法国议会选举第一轮投票结束  马克龙所在党领先法国议会选举第一轮投票结束 马克龙所在党领先</p>
```

法国议会选举第一轮投票结束 马克龙所在党领先法国议会选举第一轮投票结束 马克龙所在党领先

```
p{
    width:300px;
    height:40px;
    line-height: 20px;
    border:1px solid black;
    overflow: hidden;
}
```

background-image:url(这里放地址);	图片当做背景图片来展示
background-size:200px 200px;	设置背景图片的大小，第一个是高，第二个是宽
background-repeat:repeat;	默认值是 repeat，重复出现。 设置成 no-repeat 就不重复出现，就只出现一张；（一般都用 no-repeat） 设置成 repeat-x 就是 x 轴重复； 设置成 repeat-y 就是 y 轴重复。
background-position:100px 100px;	设置图片在容器中的位置，图片的左上定点离容器的左上定点的距离。
background-position:left bottom;	左下。填 center center 就居正中了
background-position:50% 50%;	就是居正中，和 center center 一样，不是按照左上定点来居中的

例当一张图片不能铺满容器时，就会重复出现

```
<div></div>
```

```
div{
    width:200px;
    height:200px;
    border:1px solid black;
    background-image:url(fy.jpg);
    background-size:100px 100px;
    background-repeat:no-repeat;
    background-position:50% 50%;
}
```

例：淘宝网图片背景处理

```
<a href="http://www.taobao.com" target="_blank"></a>
```


```
a{
    display: inline-block;
    text-decoration: none;
    color:#424242;
    width:190px;
    height:90px;
    border:1px solid black;
    background-image:url(https://img
    background-size:190px 90px;
}
```

上面这种情况，一旦采取默认加载策略，就是一片空白

例改成下面这样，在正常情况下就会出现问题

```
<a href="http://www.taobao.com" target="_blank">淘宝网</a>
```

上面这种情况，正常加载，图片与文字重合

一旦浏览器采取了默认的加载策略（只加载 html，不加载 css 和 js），网页也要好用

想让 css 不能使用时，文字出来；css 能用的时候，文字隐藏。有两种方法。

方法一：

用 text-indent:190px;首行缩进，

缩进距离大于内容区的宽度

用 white-space:nowrap;强制文本不换行

用 overflow:hidden;溢出部分隐藏

```
a{
    display: inline-block;
    text-decoration: none;
    color:#424242;
    width:190px;
    height:90px;
    border:1px solid black;
    background-image:url(https://i
    background-size:190px 90px;

    text-indent: 200px;
    white-space: nowrap;
    overflow: hidden;
}
```

方法二：

背景颜色，图片会作用在 padding 上，

但是内容（文字）不能作用在 padding 上

让 height=0；

容器就会变成一条线，并且文字在图片外面

然后再加 padding-top:90px；

那么 padding 里面就会显示图片

最后用 overflow:hidden;溢出部分隐

```
a{
    display: inline-block;
    text-decoration: none;
    color:#424242;
    width:190px;
    height:0px;
    padding-top:90px;
    overflow: hidden;
    border:1px solid black;
    background-image:url(https://im
    background-size:190px 90px;
}
```

特殊的点：

我们在元素嵌套时，行级元素只能嵌套行级元素，块级元素可以嵌套任何元素。

div 里面可以加任何东西，span 里面只能加行级元素

例外，块级元素 p 标签里面不能嵌套 div

例外二，<a>标签里面不能嵌套<a>标签

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

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

作业：做淘宝首屏，发邮箱 79368009@qq.com

12、css 补充

1）淘宝网左右的留白，随着浏览器变化，就没有了

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    height:30px;
    background-color: #123;
}

.content{
    width:800px;
}
```

这个长条就是一个宽度自适应的黑长条

黑长条是背景

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

```
.wrapper{
    height:30px;
    background-color: #123;
}

.content{
    width:1200px;
    height:30px;
    background-color: #0f0;
}
```

现在想让绿色的部分居中，随着浏览器宽的改变而改变

只用加一个 margin:0 auto;　　auto 是一个自适应的意思

例

```
<div class="wrapper">
    <div class="content"></div>
</div>
```

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

```
.wrapper{
    height:30px;
    background-color: #123;
}

.content{
    margin:0 auto;
    width:1200px;
    height:30px;
    background-color: #0f0;
}
```

2）inline 和 inline-block 都是文本类元素

凡是带有 inline 属性的元素，都有文本类特点，所以叫文本类元素

而 img 是 inline-block

例：解决图片之间有缝隙的 bug

写成<img scr = “”><img scr = “”><img scr = “”>这样就可以解决了（在一行写，并且不写空格），写成一行，图片间的空格就没有了

原理 :在代码上传服务器时会进行两种压缩方法 ,A 字符压缩(如把 img 用 A 代替）；B 去空格，去回车

图片地址其实就是一个二进制，可以直接复制二进制编码，那么就不用解析了

3）给行级元素加上 position:absolute；和 float:left/right 会内部隐式的把属性改成 display:inline-block;就可以加宽高了

文本和文本之间都是默认的底对齐

文本和图片之间也是底对齐。但是把 span 变成 inline-block ,一旦 span 里面放了文字，那么外面的文字就和和他里面的文字进行底对齐

例

```
<span></span>　　呵呵
```

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

```
span{
    display: inline-block;
    width:100px;
    height:100px;
    background-color: red;
}
```

例

```
<span>123</span>　　呵呵
```

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

```
span{
    display: inline-block;
    width:100px;
    height:100px;
    background-color: red;
}
```

例

```
<span>123</span>　　呵呵
```

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

```
span{
    display: inline-block;
    width:100px;
    height:100px;
    background-color: red;
    font-size:50px;
}
```

例 vertical-align:10px;文字就会往下调，如果是-10px，就会往上调 middle 是中对齐

例淘宝的导航员是用 span 或者 ul，li 做的

是两个小容器，一个往左浮动，一个往右浮动

```
<div>
    <ul style="float:left"></ul>
    <ul style="float:right"></ul
    >
</div>
```

13、css 升华篇-项目演练

让字体居中，可以 line-height=height，然后用图片部分用 padding 展示出来

例

```
<div>姬教授贴吧</div>
```

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

```
div{
    width:200px;
    line-height: 20px;
    height:20px;
    font-size:12px;
    background:linear-gradient(to
    color:rgba(255,255,255,0.8);
}
```

```
div::before{
    content:"";
    display: inline-block;
    width:12px;
    height:11px;
    background-image:url(data:imag
    background-size:100% 100%;
    margin-right:5px;
    vertical-align: -1px;
}
div::after{
    content:"";
    display: inline-block;
    background-size:100% 100%;
    width:6.5px;
    height:11.5px;
    float:right;
    margin-top:3px;
    background-image:url(data:imag
}
```

例上面那个例子更好的写法看下面

```
<div>姬教授贴吧</div>
```

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

```
div{
    padding:10px 10px;
    width:200px;
    line-height: 12px;
    height:12px;
    font-size:12px;
    background:linear-gradient(to
    color:rgba(255,255,255,0.8);
}
```

```
div::before{
    content:"";
    display: inline-block;
    width:12px;
    height:11px;
    background-image:url(data:imag
    background-size:100% 100%;
    margin-right:5px;
    /*vertical-align: -1px;*/
}
```

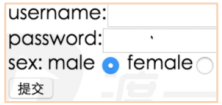
```
div::after{
    content:"";
    display: inline-block;
    background-size:100% 100%;
    width:6.5px;
    height:11.5px;
    float:right;
    background-image:url(data:imag
}
```

15、一些题目

1）请写出 HTML 基本的结构　　html，body，head

2）请写出常见的行级元素和块级元素（每样不少于 4 个）

3）请设计一个以用户名密码提交表单



4）请详细说明 position 定位的值有什么区别

5）font-size:20px;设置的是字体的宽还是高

6）2015 阿里笔试：编写一个 css 让一个已知宽高的 div，在 pc/手机端水平垂直居中

7）2015 阿里笔试：内核为 webkit 的浏览器包括：

A.IE　　B.Firefox　　C.Chrome　　D.Safari　　E.opera7（较老版本）

//新版本的 opera 也用的 webkit

8）2015 阿里笔试：通常 html 标签都是需要特别的书写来闭合，例如<a>标签的闭合就是，name 下列哪些标签不需要类似的闭合？

A.
　　B.<hr>　　C.<command>　　D.<meta>

10)2015 阿里笔试:请使用至少两种方案，实现左中右三栏布局，左栏定宽 100px；右栏定宽 100px，中栏宽度不固定，三栏宽度加起来正好是 100%

答案：解决 ul 包住 li 的问题，用三件套。父级的 after 和 content “” ，display：block，clear：both；　然后再父级里面加一个新的 zan mu 一

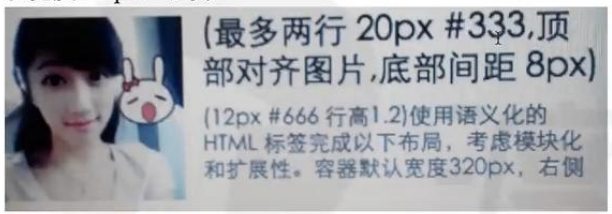
11）

```
-11.有如下html和css代码
<style type="text/css" >

ul{
    list-style-type:none;
    padding:0px;
}

ul>li{
    float:left;
}

</style>
 <ul>
     <li></li>
     <li></li>
</ul>

请在添加css代码，使ul在外观上可以包住li
```

- 9.（2015.3.31阿里笔试题)使用语义化的html标签及css完成以下布局：

{最多俩行20px #333，顶部对齐图片，底部间距8px}

{12px #666 行高1.2}使用语义化的html标签完成以下布局，考虑模块化和扩展性。容器默认宽度320px，右侧。



容器默认宽度320px,图片100*100

hover时容器宽度变成400px

答案

```
<div class="wrapper">

    <img src="fy.jpg" class="
    img">
    <p class="content1">
    {最多俩行20px #333，顶部对齐
    图片，底部间距8px}</p>
    <p class="content2">
    {12px #666 行高1.2}使用语义
    化的html标签完成以下布局，考虑
    模块化和扩展性。容器默认宽度32
    0px，右侧。</p>


</div>
```

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

.wrapper{
    width:320px;
}
.wrapper:hover{
    width:400px;
}
```

```
.wrapper .img{
    width:100px;
    height:100px;
    float:left;
}

.content1{
    font-size:20px;
    line-height: 20px;
    height:40px;
    overflow: hidden;
    color:#333;
    margin-bottom:8px;
}
.content2{
    font-size:12px;
    color:#666;
    line-height: 1.2em;
}
```

15、百度项目——首页

先从分析结构（分区）开始

li 是块级标签，一行行的，先 float:right;

mark man 取色器，选取颜色

```
<html>
    <head>
        <meta charset='utf-8'/>
        <title>百度一下，你就知道</title>
        <link rel="stylesheet" href="./src/css/index.css"></link>
    </head>
```

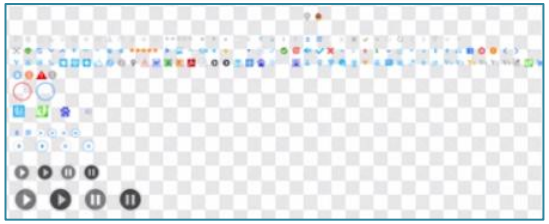
```
<body>
    <div class='header'>
        <ul class='nav'>
            li>a*
        </ul>
    </div>

    <div class='search'>

    </div>

    <div class='info'>

    </div>
</body>
```

```
<div class='header'>
    <ul class='nav'>
        <li class='more'>
            <a href="#" >更多产品</a>
        </li>
        <li class='gray'>
            <a href="#">设置</a>
        </li>
        <li  class='gray'>
            <a href="#">登录</a>
        </li>
            <a href="#">地图</a>
        </li>
        <li>
            <a href="#">hao123</a>
        </li>
        <li>
            <a href="#">新闻</a>
        </li>
    </ul>
</div>
```

```
<div class='search'>
    <div class='logo'>
        <img src="./src/img/bd.png"/>
    </div>
    <div class='keyWord'>
        <form action="./" method='GET' class='fm'
            <div class='inpWrapper'>
                <span class='camera'></span>
                <input type="text" name='kw'></ir
            </div>
            <div class='btnWrapper'>
                <input type="submit" value='百度一下'></
            </div>
        </form>
</div>
v>
```

```
<div class='info'>
    <div class='weixin'>
        <img src="./src/img/weixin.png" alt="二维码">
        <p>手机百度</p>
    </div>
```

```
<div class='message'>
    <p class='lineOne'>
        <a href="#">把百度设为主页</a>
        <a href="#">关于百度</a>
        <a href="#">About Baidu</a>
        <a href="#">百度推广</a>
    </p>
    <p class='lineTwo'>
        <span>©2017 Baidu</span>
        <a href="#">使用百度前必读</a>
        <a href="#">意见反馈</a>
        <span>京ICP证030173号</span>
        <span class='icon icon1'></span>
        <a href="#">京公网安备11000002000001号</a>
        <span class='icon icon2'></span>
    </p>
</div>
iv>
```



```
* {
    padding: 0px;
    margin: 0px;
}
/* header style */
.header {
    height: 60px;
    line-height: 60px;
}

.header .nav {
    list-style: none;
}

.header .nav::after {
    content: '';
    clear: both;
    display: block;
}

.header .nav li {
    float: right;
    margin-right: 20px;
}

.header .nav .more {
    margin-right: 12px;
}

.header .nav li a {
    color: #333;
    font-weight: 700;
    font-size: 13px;
}

.header .nav .gray a{
    font-weight: 100;
}

.header .nav li a:hover {
    color: #00c;
}

.header .nav .more a {
    background-color: #38f;
    text-decoration: none;
    font-weight: 400;
    color: #fff;
    padding: 4px;
}
```

```
/* search style */
.search {
    margin-top: 40px;
}
.search .logo {
    text-align: center;
}

.search .logo img{
    width: 270px;
    height: 129px;
}

.search .keyWord {
    margin-top: 20px;
    text-align: center;
}

.search .keyWord form div{
    display: inline-block;
}

.search .keyWord form .inpWrapper {
    position: relative;
}

.search .keyWord form .inpWrapper input{
    width: 539px;
    height: 34px;
    border: 1px solid #b6b6b6;
}

.search .keyWord form .inpWrapper span {
    position: absolute;
    right: 10px;
    top: 50%;
    margin-top: -8px;
    width: 18px;
    height: 16px;
    background-image: url('../img/camera.png');
    background-position: 0px 0px;
}

.search .keyWord form .inpWrapper span:hover {
    background-position: 0px 18px;
    cursor: pointer;
}

.search .keyWord form .inpWrapper {
    vertical-align: top;
}

.search .keyWord form .btnWrapper {
    margin-left: -5px;
}

.search .keyWord form .btnWrapper input {
    width: 100px;
    height: 34px;
    background: #3385ff;
    cursor: pointer;
    color: #fff;
    font-size: 15px;
    border: none;
}
```

```
/* info style */

.info {
    margin-top: 270px;
    text-align: center;
}

.info .weixin img {
    width: 60px;
    height: 60px;
}

.info .weixin p {
    margin-top: 4px;
    font-weight: 600;
    font-size: 12px;
}

.info .message {
    margin-top: 15px;
}

.info .message p {
    padding: 5px 0px;
}

.info .message .lineOne a {
    color: #999;
    font-size: 12px;
    margin-left: 10px;
}

.info .message .lineTwo a, span{
    color: #999;
    font-size: 12px;
}

.info .message .lineTwo .icon {
    display: inline-block;
    width: 14px;
    height: 17px;
    background-image: url('../img/icons.png');
}

.info .message .lineTwo .icon1 {
    margin-right: 15px;
    background-position: -600px -96px;
}

.info .message .lineTwo .icon2 {
    background-position: -623px -96px;
}
```