1. 行、块元素

行元素

a，span, i, label, img, input...strong、del、em(呈现为被强调的文本)

块元素

div、table、h1-h6、p、ul、li...

表现形式

(1) 行元素不独占一行，不能设置宽高

(2) 块元素独占一行，能设置宽高

(3) 行内块元素不独占一行，但是可以设置宽高，如img

行、块转换

关键字:**display**

变行: display:inline;

```
  <style>
    div{
      width:200px;
      height:100px;
      background-color: red;
      display: inline;
    }
  </style>
</head>
<body>
  <div>div01</div>
  <div>div02</div>
</body>
```

变块: display:block;

```
 <style>
   span{
      display: block;
      width: 100px;
      height: 100px;
    }
  </style>
</head>
<body>
  <span>span1</span>
  <span>span2</span>

</body>
```

变行内块: display:inline-block;

```
  <style>
    span{
      display: inline-block;
      background-color: blue;
      width: 100px;
      height: 100px;
    }
  </style>
</head>
<body>
    <!--此时的span1和span2即在同一行，又可以设置宽度-->
  <span>span1</span>
  <span>span2</span>
</body>
```

表象：div之间、span之间有缝隙

原因：行级块元素和行级元素一样, 都会把回车变成文本分隔符

解决办法：元素之间不换行，写在同一行中

```
<body>
  <div>div01</div><div>div02</div>
  <span>span1</span><span>span2</span>
</body>
```

2. 定位

css定位有四种不同类型，position值分别为：static， relative，absolute，fixed

静态定位

关键字：static

默认值，元素框正常生成

相对定位

关键字：position:relative。

参照物：相对定位的偏移参考元素是元素本身，不会使元素脱离文档流。元素的初始位置占据的空间会被保留

```
  <style>
    .rel{
      width: 100px;
      height: 100px;
      background-color: red;
      position: relative;

      top: 200px;
```

```
      left: 50px;
    }
  </style>
</head>
<body>
  <div class="rel"></div>
  131省区市新增确诊124例<br/>
  本土117例热422岁失联女孩遗体已被打捞上岸<br/>
  2大连燃气管道爆炸已致3死8伤5<br/>
  拜登将恢复对多国旅行限制<br/>
  3墨西哥总统感染新冠病毒<br/>
  6文旅部原副部长李金早被双开<br/>
</body>
```

绝对定位

关键字：position:absolute。

参照物：相对于最近的已定位的父元素。如果元素没有已经定位的父元素，那么它的相对位置相对与html标签

常用案例：块元素居中

```
{
    position:absolute;
    width:300px;
    height:300px;
    left:50%;
    top:50%;
    margin-left:-150px;
    margin-top: -150px;
}
```

固定定位

关键字：position:fixed

参照物：相对于浏览器窗口。浏览器中存在滚动条且发生滚动时，它不会移动。

常用案例：头部定位导航条、左下角留言板。

```
  <style>
    #div1{
      height: 1000px;
    }
    .rel{
      width: 100px;
      height: 100px;
      background-color: red;
      position: fixed;/*可将其改为absolute,进行对比*/
      top: 200px;
      left: 50px;
    }

  </style>
```

```
</head>
<body>
<div id="div1">
  131省区市新增确诊124例<br/>
  本土117例热422岁失联女孩遗体已被打捞上岸<br/>
  2大连燃气管道爆炸已致3死8伤5<br/>
  拜登将恢复对多国旅行限制<br/>
  3墨西哥总统感染新冠病毒<br/>
  6文旅部原副部长李金早被双开<br/>
  <div class="rel"></div>
</div>
</body>
```

3. 浮动

浮动的意义

正常的文档流是从上到下，从左到右进行排列。元素浮动脱离正常的文档流,摆脱了块级元素和行内元素的限制。

关键字

float： left;/right;

浮动的特点

- 行级元素浮动后，会变成块级标签，我们可以设置他的宽和高（考虑到标签的语义，我们一般不这样使用）

```
<style>
    span{
      background-color: antiquewhite;
      width: 100px;
    }
    span:nth-of-type(1){
       float: left;
     }
    span:nth-of-type(2){
      float: left;
    }
  </style>
</head>
<body>
  <div>
    <span>我是span</span>
    <span>我是span</span>
  </div>
```

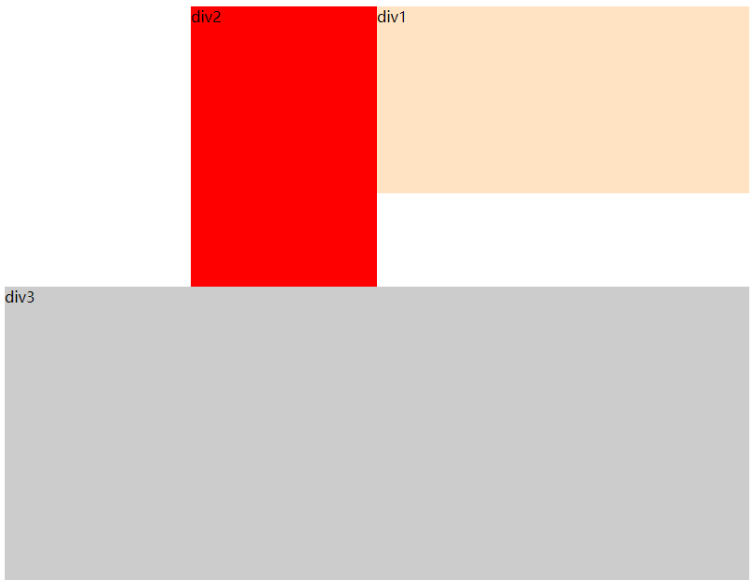
- 紧贴上一个浮动元素（同方向）或父级元素的边框，如宽度不够将换行显示

```
  <style>
    #d1{
      width:400px;

      height:200px;
```

```
      background: bisque;
      float: right;
    }
    #d2{
      width:200px;
      height:300px;
      background: red;
      float: right;
    }
    #d3{
      width:800px;
      height:400px;
      background: #ccc;
      float: right;
    }
  </style>
</head>
<body>
<div id="d1">div1</div>
<div id="d2">div2</div>
<div id="d3">div3</div>
</body>
```

效果如图：



- 占据行内元素的空间，导致行内元素围绕显示

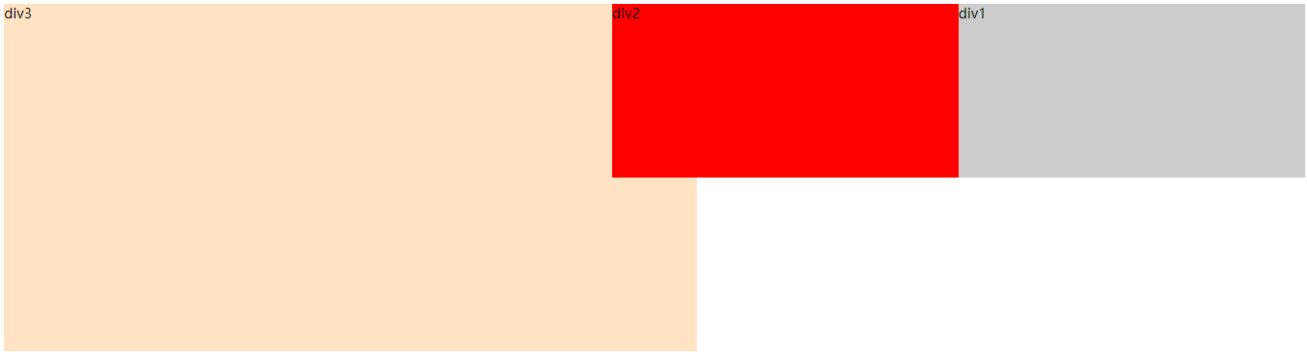
```
  <style>
    #d1{
      width:400px;
      height:200px;
      background: bisque;
      float: right;
    }
    #d2{

      width:200px;
```

```
      height:300px;
      background: red;
      float: right;
    }
    #d3{
      width:400px;
      height:400px;
      background: #ccc;
      float: right;
    }
  </style>
</head>
<body>
<div id="d1"></div>
<div id="d2"></div>
<span>
    div覆盖div,出现div与div盒子之间产生重叠覆盖现象,
   而内容没有出现覆盖重叠现象原因与解决方法。
  div覆盖div,出现div与div盒子之间产生重叠覆盖现象,
   而内容没有出现覆盖重叠现象原因与解决方法。
  div覆盖div,出现div与div盒子之间产生重叠覆盖现象,
   而内容没有出现覆盖重叠现象原因与解决方法。
   而内容没有出现覆盖重叠现象原因与解决方法。
    .....................
</span>
<div id="d3"></div>
</body>
```

效果如图:

清除浮动

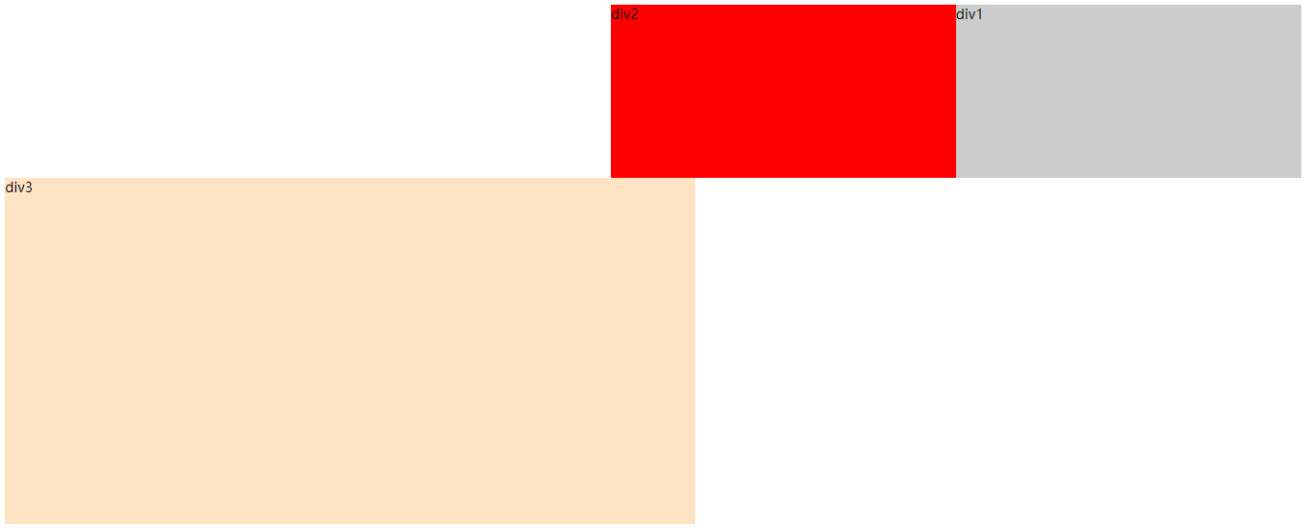


div1、div2右浮动，从而脱离文档流，div3受前两个div的影响，会从文档流的起始部分开始排列。

如果想让div3不受前面div浮动的影响，另起一行排列。那么div3需要使用"清除浮动"的属性。如：

```
  <style>
    #d1{
      width:400px;
      height:200px;
      background: #ccc;
      float: right;
    }
    #d2{
      width:400px;
      height:200px;
      background: red;
      float: right;
    }
    #d3{
      width:800px;
      height:400px;
      background: bisque;
      /*clear:right*/
    }
  </style>
</head>
<body>
<main>
  <div id="d1">div1</div>
  <div id="d2">div2</div>
  <div id="d3">div3</div>
</main>
</body>
```

效果图：



- clear作用

  如果前一个元素存在左浮动或右浮动，则换行以区隔 只对块级元素有效
- clear属性的取值

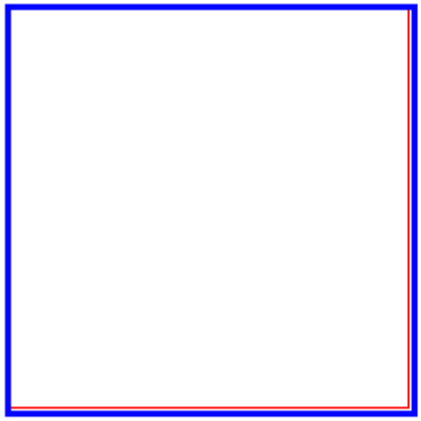
  right；left；both；none
- 不论clear取什么值，换行之后的元素依然居左显示

浮动的影响

- 浮动的元素会影响其后面的元素，即如果一个元素浮动，那么其后的元素感知不到他的存在。

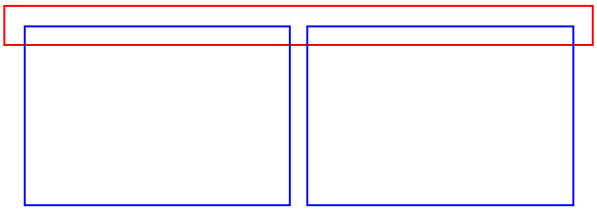
```
  <style>
    div{
      width: 200px;
      height: 200px;
      border: solid 1px red;
    }
    div:nth-of-type(1){
      border:  solid 3px blue;
      float: left;
    }
      /*第二个div会从头排列，因为第一个div浮动了，脱离了文档流，所以第二个div感知不到他的存在*/
    div:nth-of-type(2){
      border:  solid 1px red;
    }
  </style>
</head>
<body>
  <div></div>
  <div></div>
</body>
```

效果如图：



- 高度塌陷

当有浮动的子元素存在高度时，如果父元素没有设置浮动，此时子元素不会撑起父元素，父元素的高度消失。如果不清除浮动，往后的dom元素也会存在高度上的影响。在这种情况下需要清除浮动。



解决高度塌陷1——使用伪类清除浮动

:after伪类的使用

- :after 选择器在被选元素的内容中插入内容。
- 请使用 content 属性来指定要插入的内容。

```
<style>
  main{
    width: 630px;
    /*height: 220px;*/
    margin: 0 auto; /*外边距*/
    border: solid 3px red;
    padding: 20px; /*内边距*/
  }
    /*添加伪类，即在main标签内插入一个div*/
  main:after{
    content: '';
    clear: both;
    display: block;
  }
  div{
    width: 300px;
    height: 200px;
```

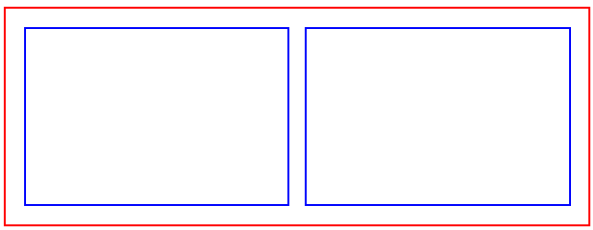
```
      border: solid 3px blue;
    }
    div:nth-of-type(1){
      float: left;
    }
    div:nth-of-type(2){
      float: right;
    }
  </style>
</head>
<body>
  <main>
    <div></div>
    <div></div>
  </main>
</body>
```

2. 解决高度塌陷2——overflow:hidden

为**父元素**添加overflow: hidden;

```
main{
  width: 630px;
  margin: 0 auto; /*外边距*/
  border: solid 3px red;
  padding: 20px; /*内边距*/
  /* 为父元素添加overflow: hidden可清除其内部标签的浮动，从而能够感知子元素的存在*/
  overflow: hidden;
}
```

效果图:



4. 浏览器页面前端自适应方案

常见概念

px：像素，屏幕上显示数据的最基本的点。width:100px；

屏幕分辨率: 1920*1080 -》宽度上1920个像素点，高度上1080个像素点。

rem：相对于根元素font-size的长度单位。 font-size:0.9rem;

em：相对于父元素font-size的长度单位。font-size:0.9em; 几乎不用。

vw:视窗宽度，移动端。1wm = 视窗宽度的1%;

vh：视窗高度，移动端。 1vh = 视窗高度的1%；

解决方案

大布局用百分比，小调整用媒体查询, 字体控制用rem。

大布局用百分比

- **对于较大的定位区域，比如整个页面、左侧菜单区域、右侧内容区域，使用百分比。如：**

calc()

- 什么是calc()

  calc是英文单词calculate(计算)的缩写，是css3的一个新增的功能，用来指定元素的长度

- calc()能做什么?

  给元素做计算。比如：你可以给一个div元素，使用百分比、em、px和rem单位值计算出其宽度或者高度，比如说“width:calc(50% + 2px)”，这样一来你就不用考虑元素DIV的宽度值到底是多少，而把这个烦人的任务交由浏览器去计算。

- calc()的好处

  计算时，可以组合不同的单位；

  浏览器中的值可以更加动态，并且可以随着视图的变化而改变；

- calc()语法

```
/*expression是一个用来计算长度的表达式*/
.elm {
    width: calc(expression);
}
```

- 表达式expression规则

  一个数学表达式，结果将采用运算后的返回值。

  calc()使用通用的数学运算规则，但是也提供更智能的功能,：

1. 使用“+”、“-”、“*” 和 “/”四则运算；
2. 可以使用百分比、px、em、rem等单位；
3. 可以混合使用各种单位进行计算；
4. **表达式中有“+”和“-”时，其前后必须要有空格**，如"widht: calc(12%+5em)"这种没有空格的写法是错误的；
5. 表达式中有“*”和“/”时，其前后可以没有空格，但建议留有空格。

html:

```
<body>
<div class="header">顶部</div>
<div class="content_body">
  <div class="left">左侧</div>
  <div class="right">右侧</div>
</div>
<div class="footer">底部</div>
</body>
```

css:

```
  <style>
    body,div{
      margin: 0;
      padding: 0;
    }
    .header{
      width: 100%;
      height: 80px;
      background-color: lightblue;
    }
    .content_body .left{
      width: 220px;
      /*calc(), */
      height: calc(100vh - 120px);
      background-color: darkgray;
      float: left;
    }
    /*.content_body父元素没有设置高度，它的高度根据内部元素的高度来设定，
    但内部元素设置为了浮动，会脱离文档流，父元素检测不到，高度就是0，所以底部div会排在上面*/
    .content_body{
      overflow: hidden;
    }
```

```
    .content_body .right{
      width: calc(100% - 220px);
      height: calc(100vh - 120px);
      background-color: red;
      float: left;
    }
    .footer{
      width: 100%;
      height: 40px;
      background-color: gray;
    }
  </style>
```

字体控制用rem

什么是rem单位

rem（root em）是一个相对单位，类似于em，em是相对于**父元素*字体大小***。

不同的是rem的基准是相对于html元素的字体大小

比如：根元素（html）设置font-size=12px;非根元素设置width:2rem；则换成px就是24px

em的应用案例：

em相对于父元素的字体大小来说的

```
  <style>
    div{
      font-size: 12px;
    }
    p{
        /*em的参考是父元素div的字体大小*/
      width: 10em;//10*12px
      height: 10em;//10*12px
      background-color: #6600ff;
    }
  </style>
</head>
<body>
  <div>
    <p></p>
  </div>
</body>
```

rem相对于html元素字体大小来说的，案例：

```
  <style>
    html{
      font-size: 14px;
    }
    p{

      width: 10rem;//rem的参考是root（html）的字体大小
```

```
      height: 10rem;
      background-color: #6600ff;
    }
  </style>
</head>
<body>
  <div>
    <p></p>
  </div>
</body>
```

结论：

rem的优点就是可以通过修改html里面的字体大小来改变页面中元素的大小。可以整体控制。

当页面调整时只需根据当前显示区域大小修改html的font-size。

局部调整使用媒体查询

- **什么是媒体查询**
  - 媒体查询是CSS3新语法
  - 使用@media查询，可针对不同的媒体类型定义不同的样式
  - **@media可以针对不同的屏幕尺寸设置不同的样式**
  - 当你重置浏览器大小的过程中，页面也会根据浏览器的宽度和高度重新渲染页面
  - 目前针对很多的苹果手机，Android手机，平板等设备都用得到多媒体查询

  语法

  @media mediatype and|not|only(media feature){

  css-code;

  }

  - 用@media开头，注意@符号
  - mediatype 媒体类型
    - 可取值【all：用于所有设备；print：用于打印机和打印机预览；screen：用于电 脑，平板电脑，智能手机等】
  - 关键字 and|not| only，
    - 关键字将媒体类型或多个媒体特性连接到一起做为媒体查询的条件
    - and：可以将多个媒体特性连接到一起，且
    - not：排除某个媒体类型，非
    - only：指定某个特定的媒体类型，可省略
  - media featrue 媒体特性 必须有小括号包含
    - 每种媒体类型都具备各自不同的特性，根据不同媒体类型的媒体特性设置不同的展示风格，我们这里了解三个

      width：定义输出设备中页面可见区域的宽度

      min-width：定义输出设备中页面最小可见区域宽度

      max-width：定义输出设备中页面最大可见区域宽度

- 注意他们要用小括号包含

案例1 @media的使用：

```
<style>
  /*这句话的意思就是：在我们屏幕上 且 最大的宽度是 800px 设置我们想要的样式
  max-width <=800px
  媒体查询可以根据不同的屏幕尺寸，更改不用的样式
  */
  @media screen and (max-width: 800px){
    body{
      background-color: palegoldenrod;
    }
  }
  @media screen and (max-width: 500px){
    body{
      background-color: #ff6600;
    }
  }
</style>
```

- **媒体查询+rem实现元素动态大小变化**

  rem单位是根据html来走的，有了rem页面元素可以设置不同大小尺寸

  媒体查询可以根据不同设备宽度来修改样式

```
<style>
    /*媒体查询一般是按照从大到小，或 从小到大的顺序写*/
    /*屏幕宽度在768px和1090px之间时*/
   @media screen and (min-width:768px) and (max-width:1090px){
      html{
        font-size: 100px;
      }
    }
    /*屏幕宽度在1091px和1366px之间时*/
    @media screen and (min-width:1091px) and (max-width:1366px){
      html{
        font-size: 200px;
      }
    }
    @media screen and (min-width:1600px) and (max-width:1920px){

    }
    @media screen and (min-width:1367px) and (max-width:1599px){

    }
    .top{
      height: 1rem;
      font-size: 0.5rem;
      background-color: green;
      color: white;
      text-align: center;/*设置文字内容水平居中*/
```

```
      line-height: 1rem;/*设置文字垂直居中*/
    }
  </style>
</head>
<body>
  <div class="top">金桥工程</div>
</body>
```

5. CSS三大特性

1.层叠性

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。

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

案例：

```
  <style>
    div{
      color:red;/*样式采用“就近原则”，会被覆盖掉*/
      font-size: 12px;/*文字大小的样式不冲突，所以不会被覆盖*/
    }
    div{
      color:pink;
    }
  </style>
</head>
<body>
  <div>好好学习 天天向上</div>
</body>
```

2.继承性

子标签会继承父标签的某些样式，如文本颜色和字号。

恰当的使用继承可以简化代码，降低CSS样式的复杂性。

子元素可以继承父元素的样式（text-、font-、line-这些元素开头的可继承，以及color属性）

```
 <style>
    div{
      color:pink;
      font-size: 20px;
    }
  </style>
</head>
<body>
  <div>
    <p>龙生龙，凤生凤，老鼠的孩子会打洞</p>
  </div>
</body>
```

3.优先级

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
- 选择器不同，则根据选择器权重执行

选择器权重如下：

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

```
  <style>
    div{
      color: red!important;/*优先级最高*/
    }
    .test{
      color: #ff6600;
    }
    #demo{
      color: green;
    }
  </style>
</head>
<body>
  <div class="test" id="demo" style="color: #6600ff">
    你笑起来真好看
  </div>
</body>
```

样式优先级：内联样式 > ID 选择器 > 类选择器 = 属性选择器 = 伪类选择器 > 标签选择器 = 伪元素选择器

属性选择器：属性选择器属于类选择器

```
input[type = "text"]
{
    width:200px;
    height:30px;
}
```

伪类选择器：属于类选择器

```
input:first-child{
    width:200px;
    height:30px;
}
```

伪元素选择器 :before :after

权重叠加

1. 权重是有4组数字组成，但是不会有进位
2. 可以理解为：行内选择器永远>id选择器>类选择器>元素选择器，依次类推
3. 等级判断从左到右，如果某一位数值相同，则判断下一位数值
4. 可简单记忆为：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选择器为100，行内样式表为1000，!important无穷大
5. 继承的权重为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>
    /*li 的权重是0, 0, 0, 1*/
    li{
      color: red;
    }
    /*复合选择器会有权重叠加的问题, */
    /*ul li 权重0, 0, 0, 1+0, 0, 0, 1=0, 0, 0, 2*/
    ul li{
      color: green;
    }
    /*.nav li 权重0,0,1,0+0,0,0,1=0,0,1,1*/
    .nav li{
      color: pink;
    }
  </style>
</head>
<body>
  <ul class="nav">
    <li>三国演义</li>
    <li>西游记</li>
    <li>水浒传</li>
    <li>红楼梦</li>
  </ul>
</body>
```

可用于换肤，菜单样式切换。

6. 其他常用css

1.单行文本溢出添加省略号

三步走：

1.先强制一行内显示文字

white-space: nowrap;

2.超出部分隐藏

overflow: hidden;

3.文字用省略号部分隐藏

overflow: hidden;

```
 <style>
    div{
      width:150px;
      height: 80px;
      margin: 100px auto;
      background-color: blue;
    /*  1.如果文字在一行内显示不开，不运行换行，默认值是normal即换行*/
      white-space: nowrap;
    /*  2.溢出的部分隐藏起来*/
      overflow: hidden;
    /*  3.文字超出部分使用省略号代替*/
      text-overflow: ellipsis;
    }
  </style>
</head>
<body>
  <div>
    啥也不说，此处省略一万字
  </div>
</body>
```

2.多行文本溢出添加省略号

多行文本溢出显示省略号，有较大兼容性问题，适合于webKit浏览器或移动端（移动端大部分是webkit内核）

chrome实现方法：不兼容IE及其他浏览器

```
{
    /*弹性伸缩盒子模型显示*/
    display: -webkit-box;
    /*设置或检索伸缩盒对象的子元素的排列方式*/
    -webkit-box-orient: vertical;
    /*限制在一个块元素显示的文本行数*/
    -webkit-line-clamp: 3;
    overflow: hidden;
 }
```

案例：

```
 <style>
    div{
      width:150px;
      height: 80px;
      margin: 100px auto;
      background-color: blue;
     /*将上面的样式copy过来即可*/
      display: -webkit-box;
      -webkit-box-orient: vertical;
      -webkit-line-clamp: 2;
      overflow: hidden;
    }
  </style>
</head>
<body>
  <div>
    啥也不说，此处省略一万字，啥也不说，此处省略一万字。
  </div>
</body>
//可通过调整div的height属性来调整显示样式
```

IE及其他浏览器实现方法：

使用插件clamps .js

插件下载地址：https://github.com/josephschmitt/Clamp.js

使用：

```
    // 引入
    <script src="js/Clamp.js-master/clamp.js"></script>
    // 样式
    <style>
        .divarea {
            width: 300px;
            height: 84px;
        }

        p {
            width: 100%;
```

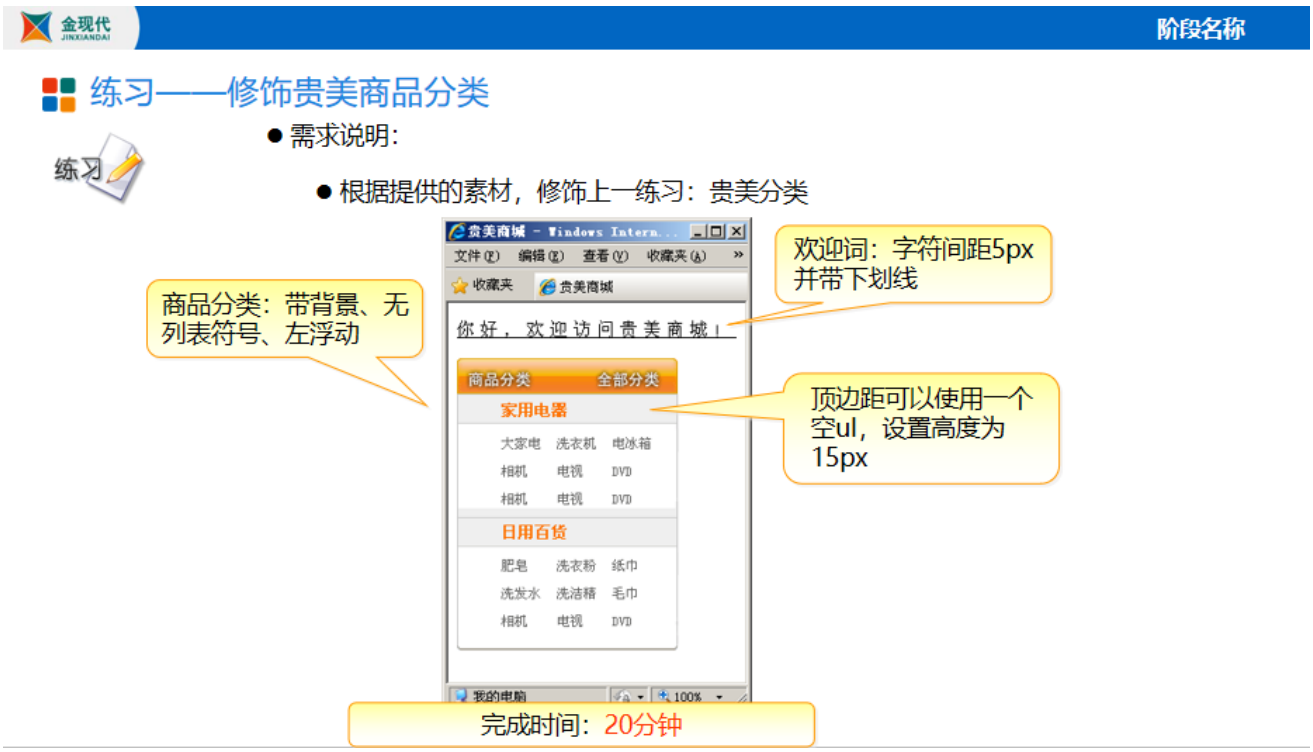
```
        }
    </style>
    // html
    <div class="divarea">
        <p>这两个选择元素的API,ualCrew小组耗时两年翻译，保持与D3 V3最后一版（3.5.17）一致.D3 V4最新版API请参考d3.v4-API翻译</p>
    </div>
    // js
    <script>
    $(document).ready(function () {
        $clamp($(".divarea p")[0], {
            clamp: '3'
        })
    })
</script>
```

3.三角形绘制

```
{
    width: 0;
    height: 0;
    border-left: 50px solid transparent;
    border-right: 50px solid transparent;
    border-bottom: 100px solid red;
}
```

7. 作业练习

练习一：

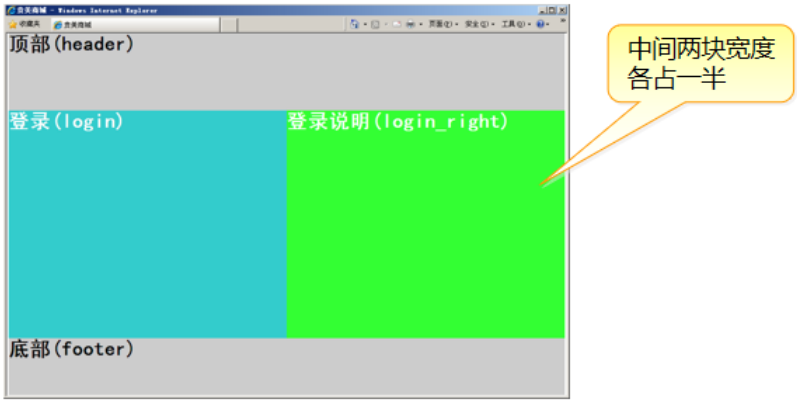


练习二：

练习——实现中间布局

练习

- 需求说明：
  - 实现页面中间布局



完成时间：10分钟