

移动web-day03

移动适配 + rem + less



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌



目录

Contents

- ◆ 移动端特点
- ◆ Less使用
- ◆ Rem布局
- ◆ 实战演练



目录

Contents

- ◆ 移动端特点
- ◆ Less使用
- ◆ Rem布局
- ◆ 实战演练

一、移动端特点

目标：了解 移动端特点，为之后的移动端布局铺垫

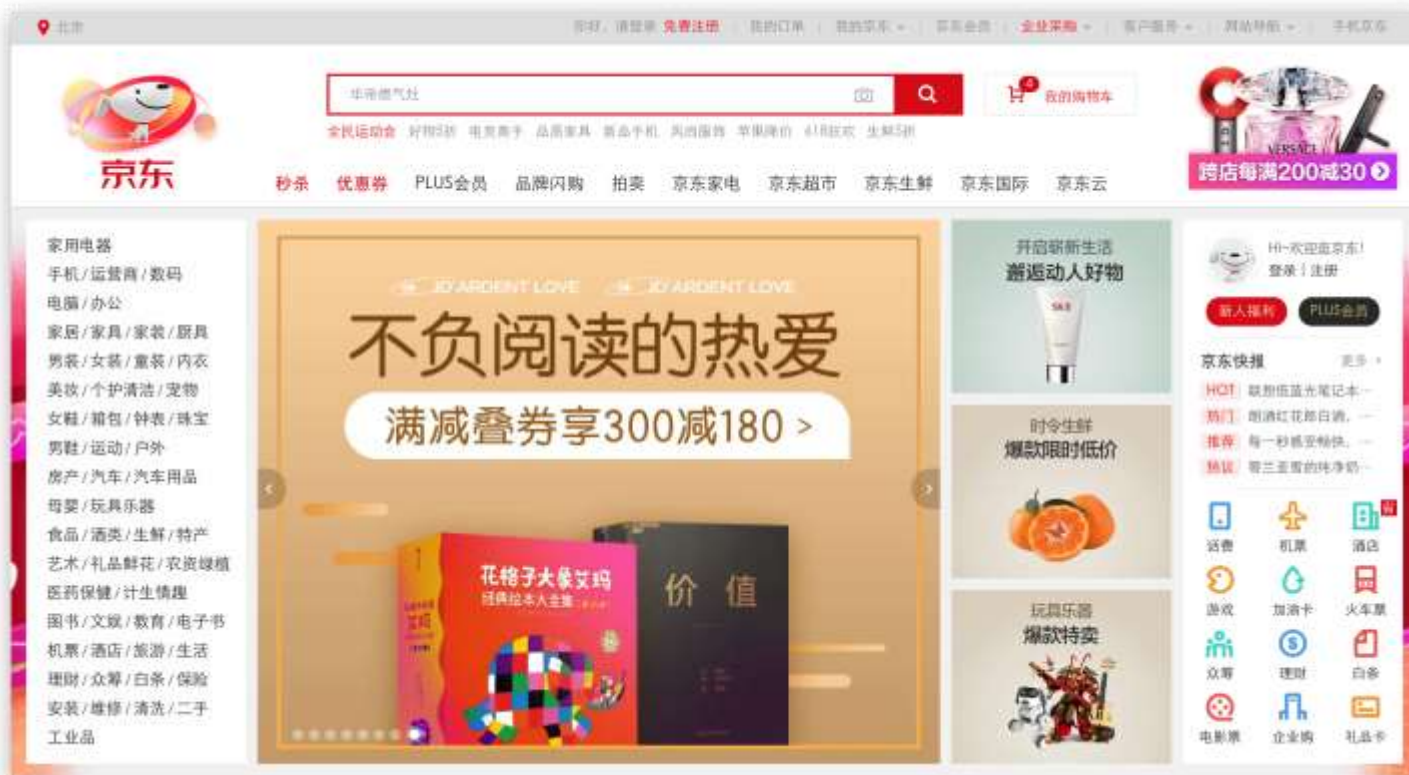
学习路径：

1. 移动端和PC端网页不同点
2. 谷歌浏览器-移动端调试工具
3. 视口
4. 物理像素和px的关系
5. 二倍图

一、移动端特点

1.1 移动端和PC端网页不同点

- PC端和移动端网页有什么不同呢？
 - PC端：屏幕较大，通过固定版心适配不同屏幕
 - 移动端：屏幕较小，网页宽度多数为100%，需要通过特殊布局方式



一、移动端特点

1.2 小结

- PC端网页和移动端网页有什么不同?
 - PC端：屏幕较大，通过固定版心适配不同屏幕
 - 移动端：屏幕较小，网页宽度多数为100%，需要通过特殊布局方式

一、移动端特点

目标：了解 移动端特点，为之后的移动端布局铺垫

学习路径：

1. 移动端和PC端网页不同点
2. 谷歌浏览器-移动端调试工具
3. 视口
4. 物理像素和px的关系
5. 二倍图

一、移动端特点

2.1 谷歌浏览器调试工具

- 在开发中，如果需要调试移动端网页怎么办？
 - 真机调试：使用真正的手机进行访问
 - 模拟器调试：Chrome DevTools（谷歌浏览器）的模拟手机调试
- 注意点：
 - 开发阶段一般使用模拟器调试，但是在实际上线之前，最终还是需要通过真机调试看在不同品牌型号中是否有特有的bug，一般会由测试部门进行测试。

一、移动端特点

目标：了解 移动端特点，为之后的移动端布局铺垫

学习路径：

1. 移动端和PC端网页不同点
2. 谷歌浏览器调试工具
3. 视口
4. 物理像素和px的关系
5. 二倍图

一、移动端特点

3.1 布局视口

- 问题：写一个div宽度为375px，使用手机端如：iPhone6/7/8打开，应该是占满的，但真的是这样吗？
 - 把vscode设置的默认视口去掉，发现网页的布局空间是：980px
- **布局视口**：用于页面布局的空间，默认是980px
 - 早期：只有pc端网页，早期页面的版心为980px，如果手机端直接看早期的pc端网页，效果不好，用户体验极差
 - 后来：为了让移动端看到完整的pc端网页，衍生出布局视口，让移动端查看pc端网页的问题

3.2 理想视口

- 理想视口其实就是布局视口的一个理想尺寸，也是实际开发移动web的常见设置
 - 早期：移动端只能访问pc端的网页，需要布局视口为980px让网页可以在手机上正常展示，没毛病
 - 现在：移动直接访问移动端网页即可，不需要访问980px的pc端网页了
- 理想视口：设置布局视口的尺寸等于当前设备屏幕的尺寸，就是理想视口

3.3 meta标签设置视口

```
<meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
```

属性	解释说明
width	设置视口的宽度，device-width表示当前设备的宽度
user-scalable	设置用户是否可以缩放，yes/no
initial-scale	初识缩放比
maximum-scale	最大缩放比
minimum-scale	最小缩放比

一、移动端特点

目标：了解 移动端特点，为之后的移动端布局铺垫

学习路径：

1. 移动端和PC端网页不同点
2. 谷歌浏览器-移动端调试工具
3. 视口
4. 物理像素和px的关系
5. 二倍图

一、移动端特点

3.4 物理像素和px的关系（了解）

- 在移动web开发的时候，会遇到如下情况：
 - 在移动端页面中需要 100*100px 的图片，然而UI给的图片分辨率是 200*200 的
 - 图片的宽高是实际设置的两倍，咱们把这两倍的图片我们称作：**二倍图**
- 二倍图的重点在于：
 - 告诉你是二倍图，css设置大小的时候，只需要除以2设置即可！
- 但是有同学好奇为什么不给一张一样大小的图片，非要给2倍图呢？
 - 如果想要知道原因，就需要了解物理像素和px的关系了~

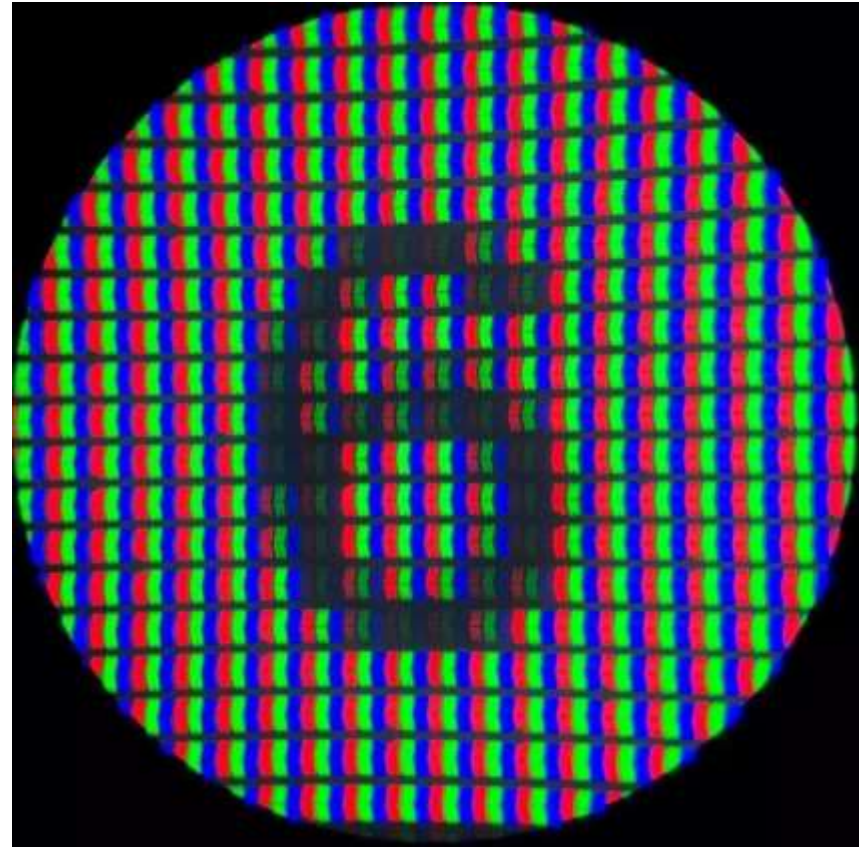
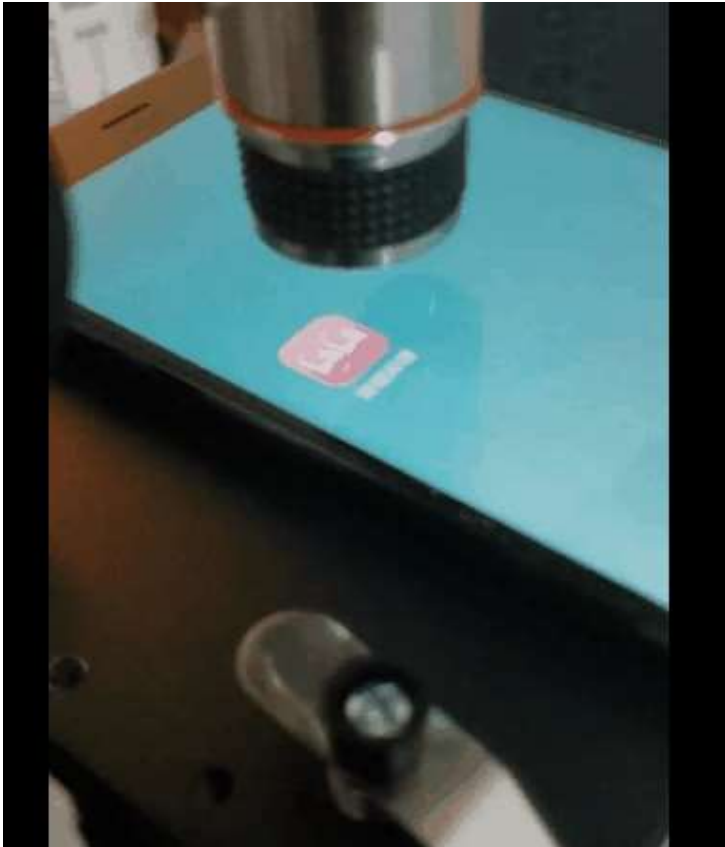
一、移动端特点

3.4 物理像素和px的关系（了解）

- 买电子设备的时候，会有很多的参数，比如：iphone4手机的分辨率是 640*960
 - 其实参数中的分辨率，指的就是：**物理像素**
- 物理分辨率：也称之为物理像素，是生产屏幕时是固定的，是屏幕中最小显示单位，是物理现实中存在的 **发光点**
 - 比如：同学们拿显微镜看屏幕，就能拿到一个一个的发光点

一、移动端特点

3.4 物理像素和px的关系（了解）



一、移动端特点

3.4 物理像素和px的关系（了解）

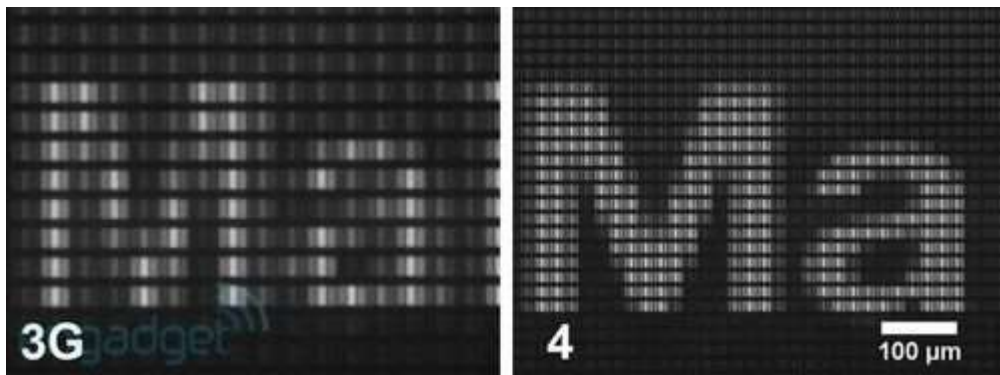
- 思考：px也叫做css中的像素，那么物理像素和px像素之间有没有什么关系呢？
- 其实物理像素和px像素之间存在对应关系：
 - 早期pc端：1个px控制1个物理像素 → 1个px对应于1个屏幕发光点
 - 现在移动端：1个px控制多个物理像素 → 1个px对应于多个屏幕发光点
- 比如：
 - iPhone4分辨率（物理像素）：640 * 960
 - iPhone4的px大小：320*480
- 设备像素比（dpr）= 物理像素：px像素 = 640：320 = 2：1

小结

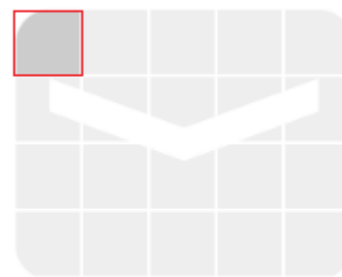
- 设备的分辨率指的是什么像素?
 - 物理像素
- 早期pc端1px控制几个物理像素?
 - 1:1
- 现在移动端，比如iPhone4中，1px控制几个物理像素?
 - 1:2

3.5 移动端屏幕的特点（了解）

- 有同学可能会好奇：为啥移动端屏幕和pc端屏幕的设备像素比不能一样都是1:1呢？为啥非要是1:多呢？
- 这其实和屏幕的发展有关：
- 早期：
 - pc端和移动端屏幕，设备像素比确实是 1:1，即：1px = 1个物理像素（发光点）
- 后来：随着技术的发展，出现了一系列的高清显示屏技术，比如：Retina（视网膜屏幕）
 - 屏幕可以将把更多的物理像素发光点压缩原本相同大小的屏幕，即：1px = 多个物理像素（发光点）
 - 这样：相同大小屏幕中，发光点越多，则发光点越细密，显示的效果也就越细腻

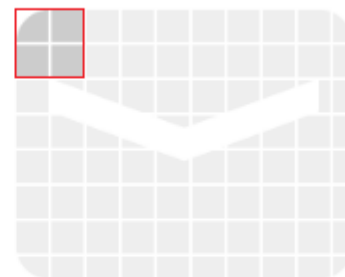


宽度比较：1px ⇔ 1个物理像素（发光点）



iPhone 3gs

1px ⇔ 2个物理像素（发光点）



iPhone 4s

小结

- 相同大小屏幕中，物理像素发光点越多，屏幕显示效果是越清晰还是越不清晰？
 - 越清晰

一、移动端特点

目标：了解 移动端特点，为之后的移动端布局铺垫

学习路径：

1. 移动端和PC端网页不同点
2. 谷歌浏览器-移动端调试工具
3. 视口
4. 物理像素和px的关系
5. 二倍图

3.6 二倍图的介绍（了解）

- 我们已经知道在移动端中，1px = 多个物理像素，那么对于咱们开发有什么影响？
- 问题：图片也有分辨率的概念，图片的分辨率指的是：物理像素 还是 px像素？
- 图片分辨率：指的是图片在水平垂直方向需要显示多少个物理像素（小光点）
 - 图片的分辨率是：200*200，表示图片在屏幕中要显示完成需要用 200*200个发光点
- 提问：以 200*200分辨率的图片显示为例
- 如果图片需要显示完全，需要？*？的发光点？
 - 200*200
- 在早期pc端中，需要写？*？px？
 - 200 * 200 px
- 现在移动端中，如iPhone4中需要写？*？Px？
 - 100 * 100 px



拍摄日期:
标记:
分级:
分辨率:

指定拍摄日期
添加标记
☆☆☆☆☆
200 x 200

一、移动端特点

3.6 二倍图的介绍（了解）

- 为了同学们有直观的感觉，下面是移动端的图片效果：



一、移动端特点

3.6 二倍图的介绍（了解）

➤ 结论：

- 在现在移动端中，如果：iPhone4中，需要显示多少px的图片，为了清晰不被强行放大，其实需要使用宽高为2分辨率的图片才是最好的效果
- 而这使用的宽高为2倍分辨率大小的图片，称之为2倍图
- 实际开发中还存在2倍图、3倍图、4倍图之类的，但具体使用哪一种看公司的需要或者要求

➤ 命名特点：

- [xxxxxx@2x.png](#)：二倍图
- [xxxxxx@3x.png](#)：三倍图

一、移动端特点

4.3 二倍图的使用

- 现阶段设计稿参考 iPhone6/7/8, 设备宽度 375px 产出设计稿
- 二倍图设计稿尺寸: 750px



一、移动端特点

小结

- 开发中遇到二倍图的情况，应该如何使用？
 - 把大小除以2使用

一、移动端特点

移动端适配方案

- 宽度适配：宽度自适应
 - 百分比布局
 - Flex布局
- 等比适配：宽高、字体大小等一起等比例缩放
 - rem
 - vw



目录

Contents

- ◆ 移动端特点
- ◆ Less使用
- ◆ Rem布局
- ◆ 实战演练

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

1. Less的基本介绍
2. Less的编译插件及配置
3. Less语法学习

1.1 维护css的弊端

- CSS是一门非编程语言，没有变量、函数、作用域等概念
 - CSS需要书写大量没有逻辑的代码，冗余度较高
 - 不方便修改维护和拓展，不利于复用
 - CSS没有很好的计算能力
- Less比CSS功能更加强大，可以解决以上css中的弊端

1.2 Less的介绍

- Less是一门CSS预处理语言，也叫做CSS预处理器。它拓展了CSS的写法，增加了变量、函数等特性。
 - Less字如其名：更少（less）的代码，做更多的事情
 - 常见的CSS预处理器还有：Less、Sass、Stylus
- 注意点：
 1. 在less中，完全兼容css的语法，可以直接在less文件中写css！
 2. 浏览器不能直接认识less文件，需要先把less文件编译成.css文件，再使用！

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

1. Less的基本介绍
2. Less的编译插件及配置
3. Less语法学习

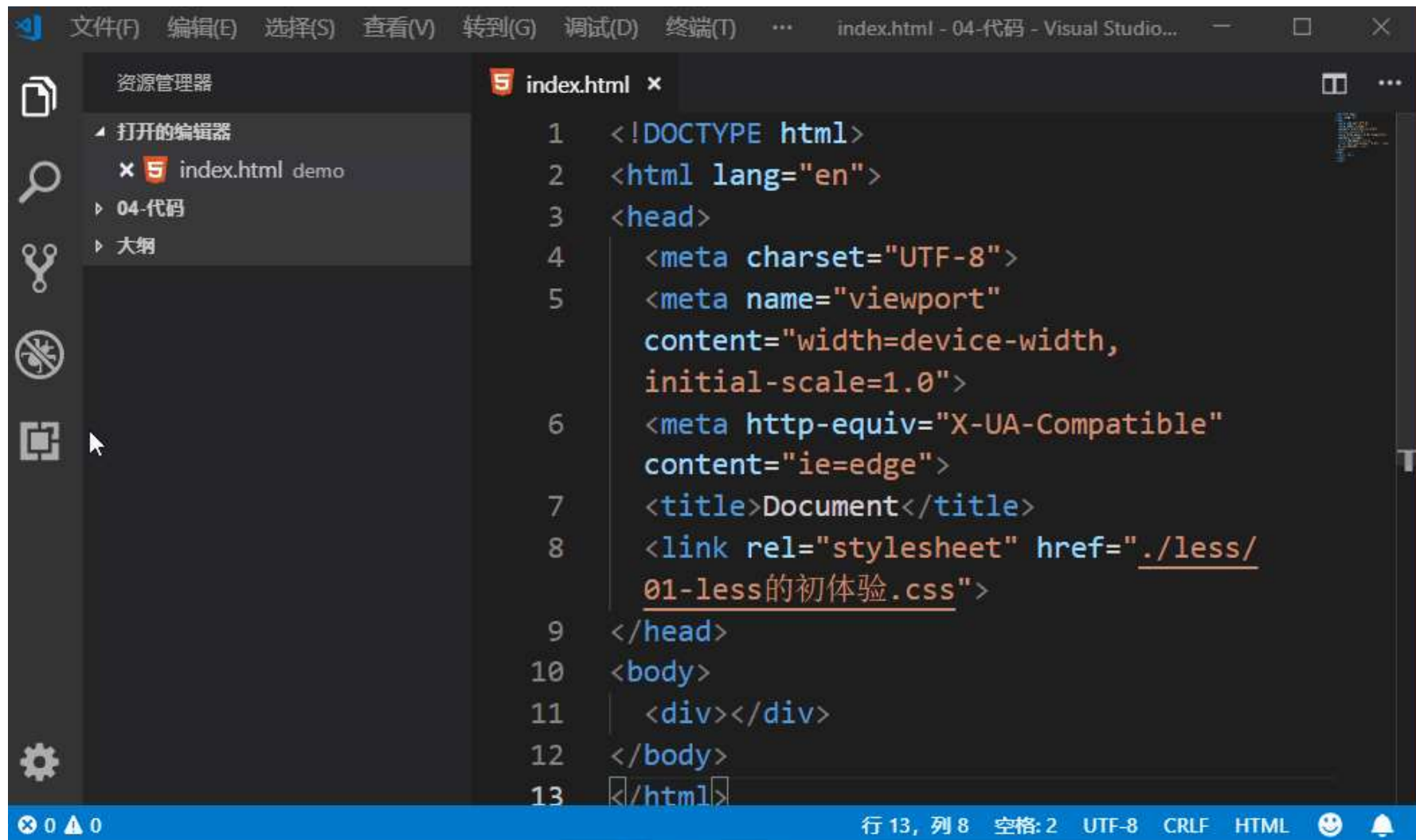
2.1 less的编译插件及配置

- Less写完之后需要编译成.css文件之后才能使用，可以使用VsCode中的插件完成
- 步骤步骤：
 1. 选择左侧第五个拓展按钮，下载插件 easy less，点击安装，再点击重新加载
 2. 安装好之后，重新加载或者关闭vscode重新打开
 3. 在VSCode的设置文件中进行配置

```
"less.compile": {  
  "out": "../css/"  
},
```

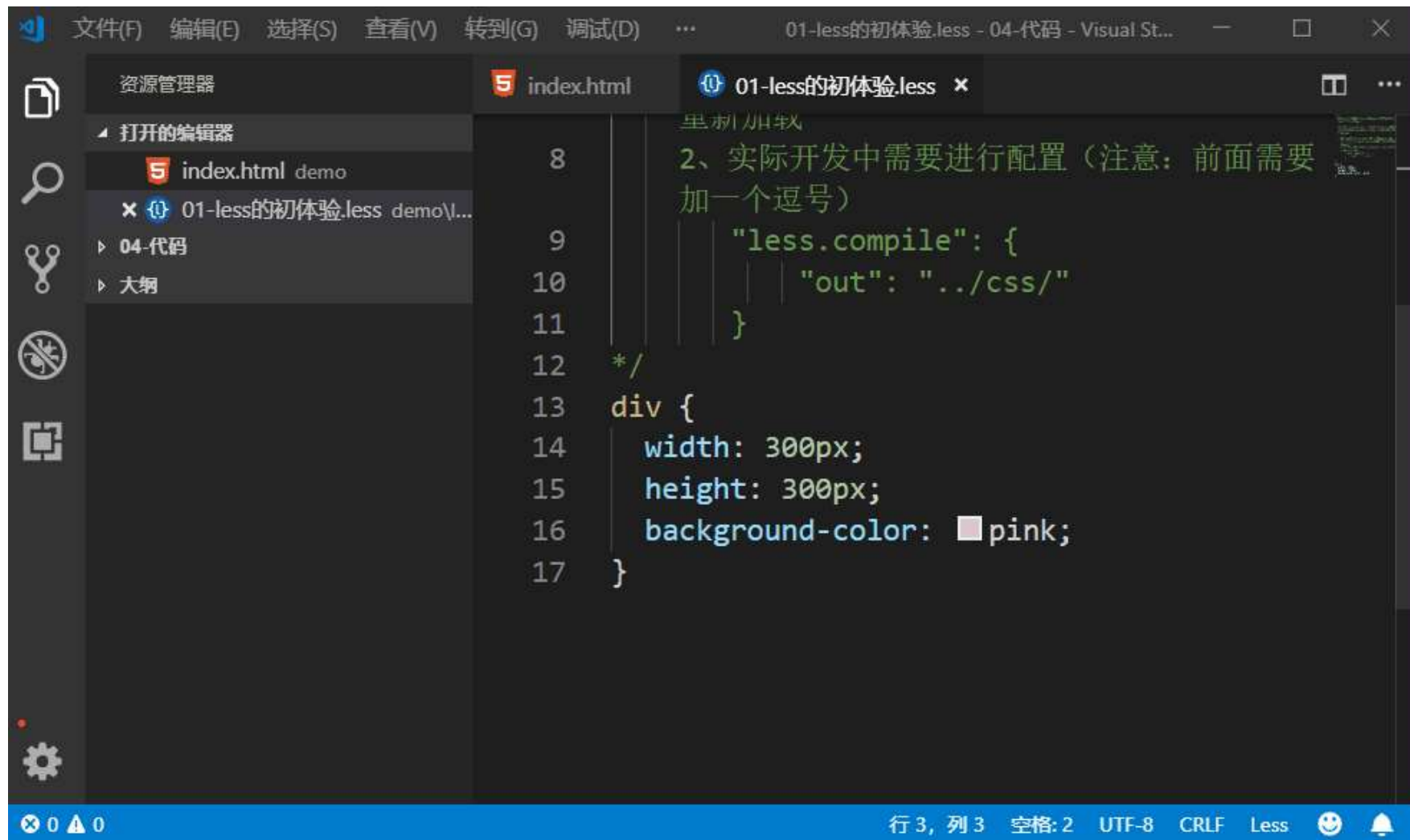
二、Less使用

2.1 less的编译插件及配置



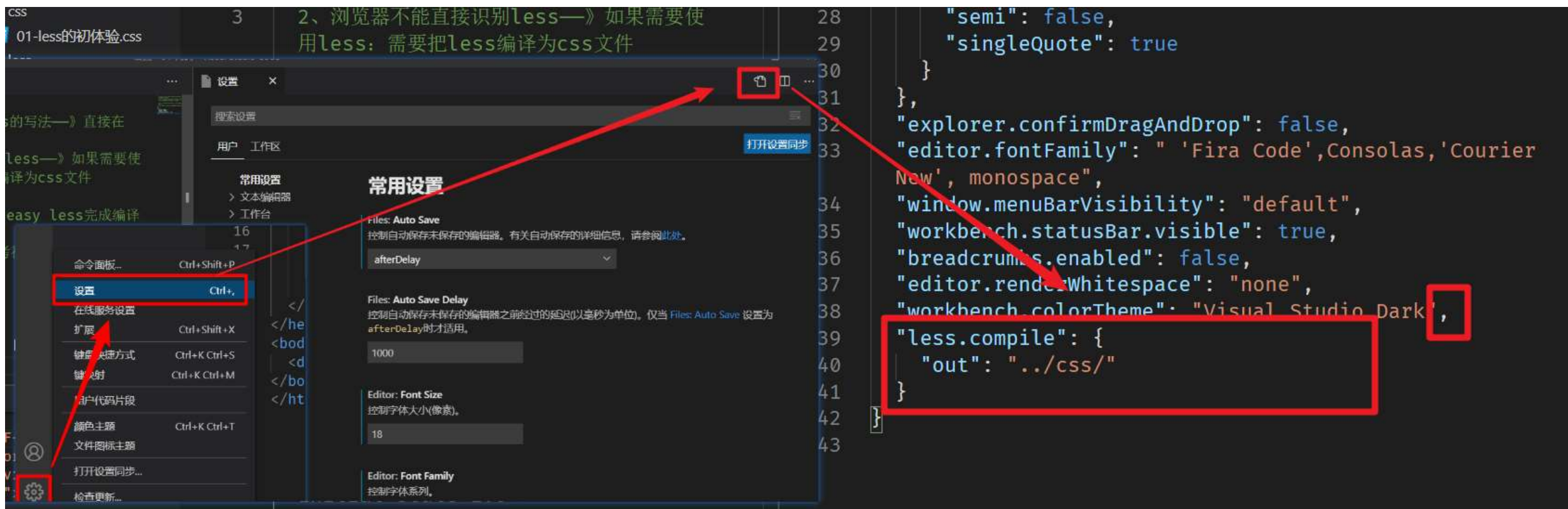
二、Less使用

2.1 less的编译插件及配置



二、Less使用

2.1 less的编译插件及配置



小结

- 在使用less写样式时，写的是什么文件？页面中引入的是什么文件？
 - 写的是：less文件
 - 引入的是：CSS文件

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

1. Less的基本介绍
2. Less的编译插件及配置
3. Less语法学习

3.1 less的注释

- 在less中支持两种注释的写法
- 写法一：/* 注释的内容 alt + shift + a */，最后会编译展示在css文件中
- 写法二（推荐）：// 注释的内容 ctrl + /，只在less中使用，不会编译展示在css文件中

小结

- 在less中推荐使用什么注释?
 - Less中的注释: //

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

1. Less的基本介绍
2. Less的编译插件及配置
3. Less语法学习

3.2 less的变量

- 需求：网页中存在用的较多的颜色（主题色），如果此时需要快速更换网页主题色，如何解决？
- 方法一：一个一个的改，可以但是麻烦
- 方法二：用less的变量可以迅速的完成
- 例子：

```
.box1 {  
    width: 100px;  
    height: 100px;  
    background-color: pink;  
}  
.box2 {  
    width: 200px;  
    height: 200px;  
    background-color: pink;  
}  
.box3 {  
    width: 300px;  
    height: 300px;  
    background-color: pink;  
}
```

3.2 less的变量

- 变量：可以变化的量
- 语法：@变量名:变量值;
- 作用：编译时会把less中所有变量名替换成变量值，这样可以统一修改某一个值

```
@mainColor:#e92322;
```

```
.box1 {  
  width: 100px;  
  height: 100px;  
  background-color: @mainColor;  
}  
.box2 {  
  width: 200px;  
  height: 200px;  
  background-color: @mainColor;  
}  
.box3 {  
  width: 300px;  
  height: 300px;  
  background-color: @mainColor;  
}
```

小结

- 针对于样式中不断变化的值，less中可以使用什么表示？
 - 变量
- Less中变量的语法是什么样？
 - @变量名:变量值;

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

1. Less的基本介绍
2. Less的编译插件及配置
3. **Less语法学习**

3.3 less的嵌套

- 在less中，选择器的关系可以通过嵌套来表示
- CSS写法：

```
// 后代选择器
.father .son {
    width: 200px;
    height: 200px;
    background-color: red;
}
// 子代选择器
.father > .son {
    width: 100px;
    height: 100px;
    background-color: blue;
}
// 并集选择器
.father .one,
.father .two {
    background-color: red;
}
// 交集选择器
.father.blue {
    background-color: blue;
}
// 链接伪类选择器
.father:hover {
    background-color: red;
}
// 伪元素
.father::after {
    content: "";
}
```

3.3 less的嵌套

- 在less中，选择器的关系可以通过嵌套来表示
- Less写法：

```
// less中的嵌套：less中选择器可以嵌套
// 1、后代选择器，选择器嵌套即可
// 2、子代选择器，前面使用>
// 3、并集选择器，前面直接写，
// 4、交集选择器，前面使用&（&表示上一级选择器）
// 5、伪元素，前面使用&（&表示上一级选择器）
.father {
  width: 300px;
  height: 300px;
  background-color: red;
  // 后代选择器：通过嵌套表示后代关系
  .son {
    width: 100px;
    height: 100px;
    background-color: blue;
  }
  // 子代选择器：通过嵌套 + > 表示
  >.son {
    background-color: red;
  }
  // 并集选择器：通过嵌套可以省略父级选择器
  .one,
  .two {
    background-color: yellow;
  }
  // 交集选择器：&表示上一级选择器
  &.red {
    background-color: red;
  }
  // 结构伪类选择器：&表示上一级选择器
  &:nth-child(1) {
    background-color: red;
  }
  // 链接伪类选择器：&表示上一级选择器
  &:hover {
    background-color: green;
  }
  // 伪元素：&表示上一级选择器
  &::after { content: "";
  }
}
```

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

1. Less的基本介绍
2. Less的编译插件及配置
3. **Less语法学习**

3.4 less的运算

- 在less代码中，可以直接进行加减乘除进行计算

// 在less中可以直接书写+ - * / 的式子

```
.father {  
  width: 800px+100;  
  width: 800px-100;  
  width: 800px*100;  
  width: (800px/100);  
  height: 800px;  
  background-color: pink;  
}
```

- 注意点：在less4.0版本之后，除法运算/如果在括号()外，不会执行除法，推荐使用()包裹即可

小结

- 如果需要在less使用除法，直接写 $200\text{px}/2$ ，可以编译成功吗？
 - 不能，需要用 $()$ 包裹起来

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

1. Less的基本介绍
2. Less的编译插件及配置
3. Less语法学习

3.5 less的函数（了解）

- 在样式中会经常遇到重复的代码，此时可以使用节省代码的方式计算

```
.red {  
    width: 300px;  
    height: 300px;  
    background-color: red;  
}  
.blue {  
    width: 300px;  
    height: 300px;  
    background-color: blue;  
}  
.green {  
    width: 300px;  
    height: 300px;  
    background-color: green;  
}
```

3.5 less的函数（了解）

- 方法一：提取公共类，然后给标签设置公共类
- 方法二：提取less函数，在选择器中调用函数即可
- 函数语法：.函数名() { 重复的样式 }

```
.common() {  
  width: 300px;  
  height: 300px;  
}  
.red {  
  .common();  
  background-color: red;  
}  
.blue {  
  .common();  
  background-color: blue;  
}  
.green {  
  .common();  
  background-color: green;  
}
```

小结

- 针对于less中重复的代码，可以通过什么进行优化？
 - Less中的函数
- Less中的函数语法是什么样？
 - .函数名() { 重复的代码 }

二、Less使用

目标：了解 Less的使用，为之后在项目中使用

学习路径：

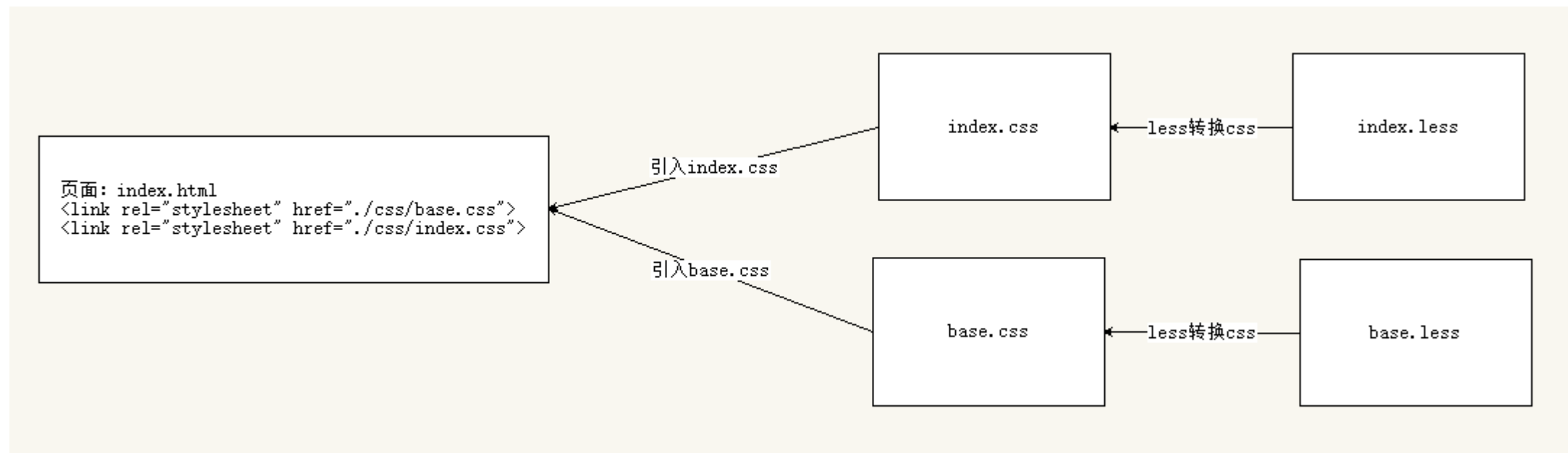
1. Less的基本介绍
2. Less的编译插件及配置
3. Less语法学习

3.6 less的引入

- 之前：一个页面需要同时写很多link标签，引入很多css文件
- 使用less之后：可以把很多less文件都引入到一个less文件中，最后页面只需要link引入一个css文件即可
- 语法：@import '需要引入的less文件的路径'

3.6 less的引入

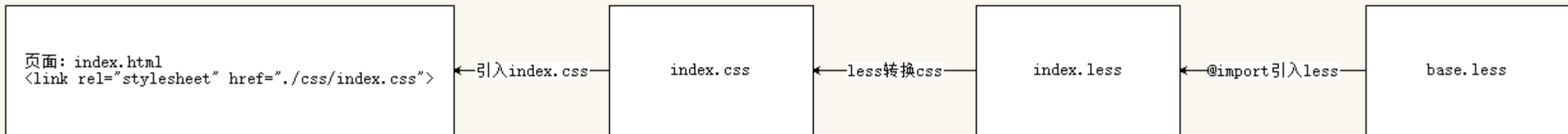
➤ 之前CSS的引入方式:



二、Less使用

3.6 less的引入

- 现在@import的引入方式:



小结

- 在一个less文件中引入另一个less文件的语法是什么？
 - `@import '需要引入的less文件的路径' ;`



目录

Contents

- ◆ 移动端特点
- ◆ Less使用
- ◆ Rem布局
- ◆ 实战演练

三、rem布局

目标：熟练使用rem布局，能够后续项目

学习路径：

1. Rem布局的介绍
2. Rem和em的区别
3. Rem布局的实现原理
4. 媒体查询
5. Rem布局的适配方案

1.1 rem布局的介绍

- 之前学习的：宽度百分比/flex布局.....都是针对于盒子宽度的适配变化，但是盒子的高度和文字大小等大小是无法进行适配变化的
 - 缺点：在较大屏幕下，盒子高度和文字大小不会变化，用户观感较差：盒子拉成了、盒子太扁了、文字太小了、显示效果不好
- 现在学习的：rem布局/vw布局，可以做到页面中所有盒子的宽高、文字大小等，都能跟随屏幕进行等比例缩放
 - 优点：无论是大屏还是小屏，因为是等比例缩放，盒子都不会拉长、文字都不会太小，视觉效果较好
- 总结：如果需要让页面中所有元素大小一起等比例缩放，可以使用rem布局/vw布局

小结

- 如果需要对页面中所有元素大小一起等比例缩放，可以使用什么布局完成？
 - Rem布局/vw布局

三、rem布局

目标：熟练使用rem布局，能够后续项目

学习路径：

1. Rem布局的介绍
2. Rem和em的区别
3. Rem布局的实现原理
4. 媒体查询
5. Rem布局的适配方案

2.1 rem和em的区别

- Rem布局中的rem其实就是一个单位，如果需要使用rem布局，需要先认识rem单位
- em：相当于当前元素的字体大小 $\rightarrow 1em = \text{当前标签的font-size}$
- rem：相当于根元素（html）的字体大小 $\rightarrow 1rem = \text{html标签的font-size}$
- 注意点：浏览器默认的font-size大小为：16px

小结

- 1em的大小为多少?
 - 1em = 当前标签的font-size大小
- 1rem的大小为什么?
 - 1rem = 根标签 (html) 的font-size大小

三、rem布局

目标：熟练使用rem布局，能够后续项目

学习路径：

1. Rem布局的介绍
2. Rem和em的区别
3. Rem布局的实现原理
4. 媒体查询
5. Rem布局的适配方案

3.1 rem布局的基本原理

- Rem布局的效果：
 - 屏幕越大，元素越大
 - 屏幕越小，元素越小
- Rem布局的基本原理：
 - 当屏幕越大，让html标签的font-size越大
 - 当屏幕越小，让html标签的font-size越小
- 实现rem布局的操作步骤：
 - 把元素的单位改成 rem 单位
 - 当屏幕宽度改变时，自动改变html标签的font-size大小

小结

- Rem布局的使用步骤分别是什么?
 - 把单位转换成rem单位
 - 当屏幕屏幕变化时，自动改变html标签的font-size

三、rem布局

目标：熟练使用rem布局，能够后续项目

学习路径：

1. Rem布局的介绍
2. Rem和em的区别
3. Rem布局的实现原理
4. 媒体查询
5. Rem布局的适配方案

4.1 媒体查询

- 问题：实现rem的第二步需要当屏幕宽度变化后，自动改变html标签的font-size大小，但是如何实现呢？
- 方法一：css3中的媒体查询可以完成
- 方法二：通过js代码实现 (flexible.js)

4.1 媒体查询

- 媒体查询：可以动态查询当前屏幕宽度，自动控制样式是否生效
- 语法：@media screen and (条件) { 选择器 }
- 含义：
 - 判断当前条件是否满足，如果满足，此时让内部的选择器生效
 - 判断当前条件是否满足，如果不满足，此时让内部的选择器失效
- 条件：

条件	含义	解释
min-width	样式生效的最小宽	当屏幕宽度大于等于该宽度时，选择器样式才生效
max-width	样式生效的最大宽	当屏幕宽度小于等于该宽度时，选择器样式才生效
width	样式生效的宽度	当屏幕宽度正好等于该宽度时，选择器样式才生效

4.1 媒体查询

➤ 注意点：

1. 媒体查询只能控制样式是否生效，不能提高选择器的优先级
2. 如果媒体查询中需要同时设置多个条件，需要以and连接
3. 媒体查询中的语法，空格不能省略

小结

- 媒体查询的语法结构是什么？
 - @media screen and (条件) { 选择器 }
- 媒体查询的条件分别是什么含义：

条件	含义	解释
min-width	样式生效的最小宽	当屏幕宽度大于等于该宽度时，选择器样式才生效
max-width	样式生效的最大宽	当屏幕宽度小于等于该宽度时，选择器样式才生效
width	样式生效的宽度	当屏幕宽度正好等于该宽度时，选择器样式才生效

- 媒体查询中如果需要同时满足多个条件，条件之间需要通过什么连接？
 - and

媒体查询的小练习

➤ 案例：

```
/* 1、屏幕宽度：400~600  盒子大小：200*200px 背景：绿色 */  
/* 2、屏幕宽度：600~800  盒子大小：300*300px 背景：蓝色 */  
/* 3、屏幕宽度：800~1000 盒子大小：400*400px 背景：黄色 */  
/* 4、屏幕宽度：1000~正无穷 盒子大小：500*500px 背景：紫色 */
```

➤ 注意点：为了保证大屏的媒体查询不被覆盖，媒体查询一般从小屏往大了写

三、rem布局

目标：熟练使用rem布局，能够后续项目

学习路径：

1. Rem布局的介绍
2. Rem和em的区别
3. Rem布局的实现原理
4. 媒体查询
5. **Rem布局的适配方案**

5.1 rem布局适配步骤-媒体查询（了解）

- 需求：当750屏幕的设计图中，量取某div的宽高为400*400，需要使用rem布局适配375、640、750的屏幕
- 适配步骤：
 1. 把px单位转换成rem单位
 1. $Px = rem * \text{html标签的font-size}$
 2. $Rem = px / \text{html标签的font-size}$ （习惯：设计图的10%，即75）
 2. 利用媒体查询，设置不同屏幕下的html标签的font-size的大小（等比例缩放）

$$\frac{\text{设计图屏幕宽度}}{\text{设计图font-size}} = \frac{\text{适配屏幕1宽度}}{\text{适配屏幕1font-size}} = \text{比例}$$
$$\frac{\text{适配屏幕1宽度}}{\text{适配屏幕1font-size}} = \text{比例} \Rightarrow \frac{\text{适配屏幕1宽度}}{\text{比例}} = \text{适配屏幕1font-size}$$

小结

- 如果设计图大小为750，在设计图中量取的尺寸px，一般如何转换成rem单位？
 - 把量取的值除以设计图的10%，即除以75即可

5.2 rem布局适配步骤-flexible.js

- 需求：当750屏幕的设计图中，量取某div的宽高为400*400，需要使用rem布局适配所有屏幕呢？
 - 解决：通过js动态检测当前屏幕宽度，即可完成所有屏幕适配
 - Flexible.js：是手淘开发出来适配移动端的js文件，可以根据不同视口宽度，自动设置html标签的font-size
- 适配步骤：
 1. 把px单位转换成rem单位（把量取的px/设计图的十分之一即可）
 2. 在项目中引入flexible.js文件即可

```
<body>
  <div class="box"></div>
  <script src="./js/flexible.js"></script>
</body>
```

小结

- 在项目中如何引入flexible.js文件?
 - 在项目body标签内容的最后，通过script标签引入即可

```
<body>  
  <div class="box"></div>  
  <script src="./js/flexible.js"></script>  
</body>
```




目录

Contents

- ◆ 移动端特点
- ◆ Less使用
- ◆ Rem布局
- ◆ 实战演练

使用rem布局，实现疾速问诊



移动端布局前注意点

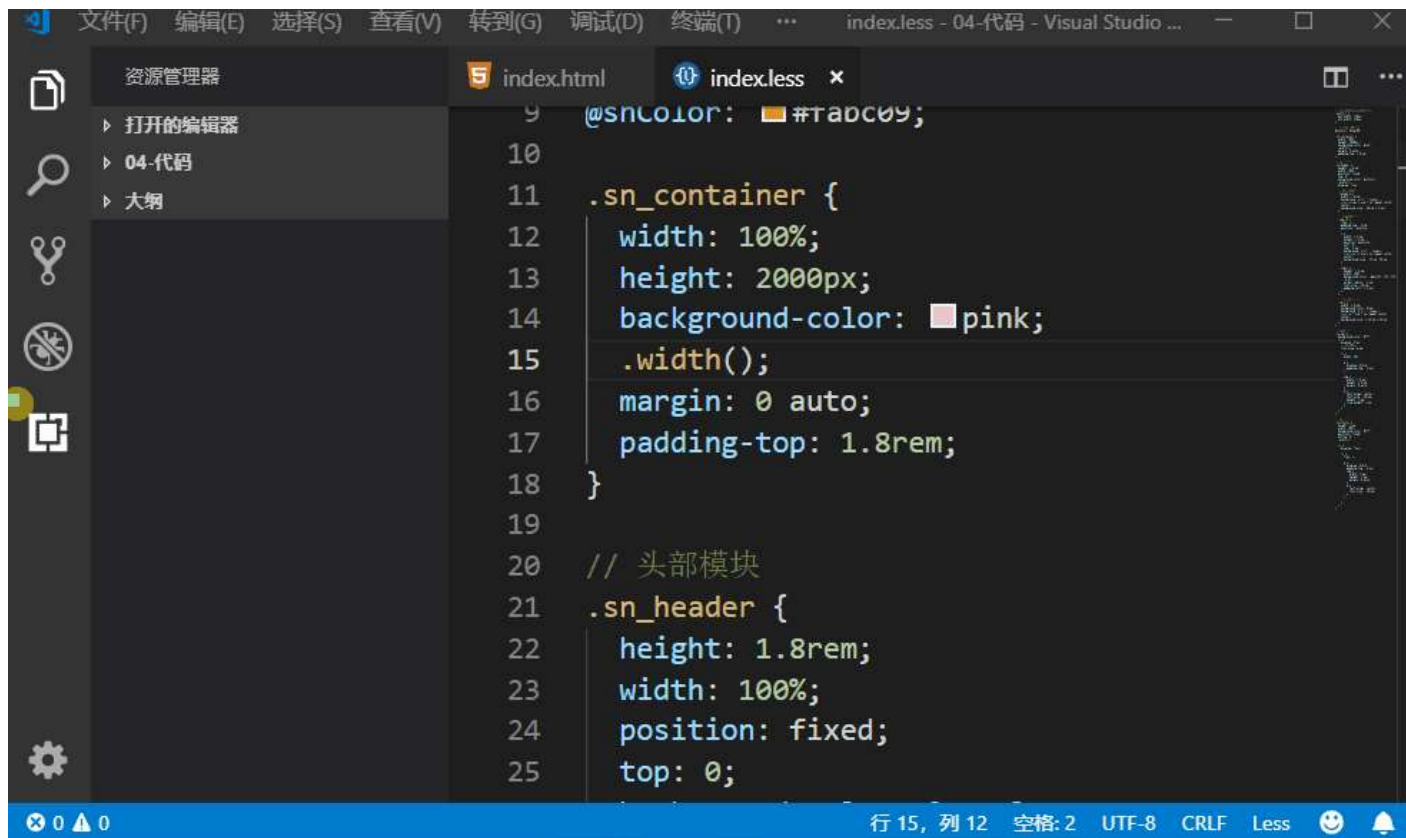
- 设置的完整视口如下：

```
<meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
```

px2rem插件使用

- 在书写rem布局项目时，如果手动把px单位转换成rem，可以，但是非常麻烦，因此可以借助VSCode的插件实现
- 可以实现的插件有很多种，这里带同学们使用px2rem插件：
- 使用步骤如下：

1. 安装px2rem

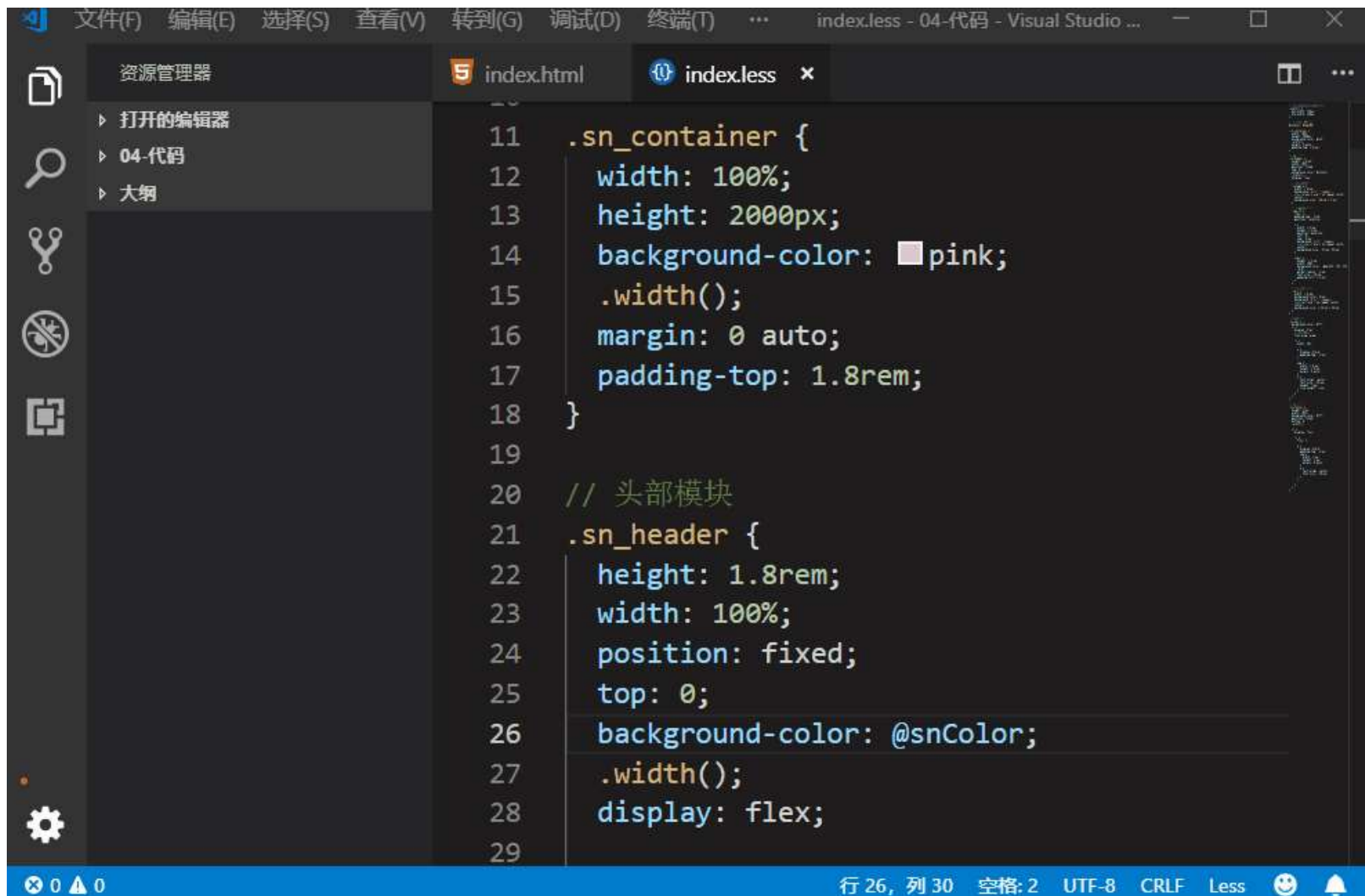


```
9 @snColor: #fabc09;
10
11 .sn_container {
12     width: 100%;
13     height: 2000px;
14     background-color: pink;
15     .width();
16     margin: 0 auto;
17     padding-top: 1.8rem;
18 }
19
20 // 头部模块
21 .sn_header {
22     height: 1.8rem;
23     width: 100%;
24     position: fixed;
25     top: 0;
```

px2rem插件使用

➤ 使用步骤如下:

1. 安装px2rem
2. 设置px2rem
3. 每次输入px, 选择rem单位



The screenshot shows the Visual Studio Code editor interface. The left sidebar contains the 'Resource Explorer' (资源管理器) with a tree view showing '打开的编辑器' (Opened Editors), '04-代码' (04-Code), and '大纲' (Outline). The main editor area displays two files: 'index.html' and 'index.less'. The 'index.less' file is open, showing the following code:

```
11 .sn_container {
12   width: 100%;
13   height: 2000px;
14   background-color: pink;
15   .width();
16   margin: 0 auto;
17   padding-top: 1.8rem;
18 }
19
20 // 头部模块
21 .sn_header {
22   height: 1.8rem;
23   width: 100%;
24   position: fixed;
25   top: 0;
26   background-color: @snColor;
27   .width();
28   display: flex;
29 }
```

The status bar at the bottom indicates '行 26, 列 30' (Line 26, Column 30), '空格: 2' (Spaces: 2), 'UTF-8', 'CRLF', 'Less', and a smiley face icon.

晚自习安排

1. 参考老师的PPT内容，**梳理今日上课的xmind**（可以跟着老师写好的xmind写一遍，加深印象）
2. 在梳理每日内容时，如果发现模糊的地方，可以在单独快速的看一遍本节视频（切记：只看遗漏的，不要全都看）
3. 把**上课的案例多敲几遍**，直到能不看老师代码和视频，**能独立把案例敲出来为止**（忘记了也有上课录制的视频兜底，不怕做不出，就怕懒的做）
4. 把**每日综合案例独立做完**，第一次可以参考老师的视频/代码，但是要多做几次，直到同学们能做到不看视频和代码能独立做出了，才算是学扎实了。
5. 学习过程中遇到问题先独立思考5~10min左右，能自己解决的bug印象最深。如果超出时间可以求助同学或者助教或者百度或者先记在本子上，第二天直接问题即可。（切记：讨论声音不要影响其他同学）
6. 到了晚上 **9:00 记得在博学谷填写反馈** 网址为：
 - 教室内局域网：<http://ntlias-stu.boxuegu.com/>
 - 外网：<https://tlias-stu.boxuegu.com/login>



传智教育旗下高端IT教育品牌