[目录 2](#_Toc467740633)

[第1章 屏幕适配 3](#_Toc467740634)

[1.1 rem适配 3](#_Toc467740635)

[第2章 调试工具 4](#_Toc467740636)

[2.1 模拟调试 4](#_Toc467740637)

[2.2 远程调试 4](#_Toc467740638)

[第3章 媒体查询 4](#_Toc467740639)

[3.1 媒体类型 5](#_Toc467740640)

[3.2 媒体特性 5](#_Toc467740641)

[3.3 关键字 5](#_Toc467740642)

[3.4 引入方式 6](#_Toc467740643)

[3.5 常用特性 6](#_Toc467740644)

[第4章 CSS预处理器 7](#_Toc467740645)

[4.1 LESS 7](#_Toc467740646)

[6.1.1. 安装 7](#_Toc467740654)

[6.1.2. 编译 7](#_Toc467740655)

[6.1.3. 语法 8](#_Toc467740656)

[6.1.4. 浏览器中使用 9](#_Toc467740657)

# 屏幕适配

## rem适配

1、设置<meta name="viewport" content="width=device-width, initial-scale=1">

2. 设置页面元素宽度单位为 rem 或 em

注：此方案比较灵活，我们的案例将采用这种方案

**关于em和rem**

em 相对长度单位，其参照当前元素字号大小，如果当前元素未设置字号则会继承其祖先元素字号大小 例如 .box {font-size: 16px;} 则 1em = 16px .box {font-size: 32px;} 则 1em = 32px，0.5em = 16px

rem 相对长度单位，其参照根元素(html)字号大小 例如 html {font-size: 16px;} 则 1rem = 16px html {font-size: 32px;} 则 1rem = 32px，0.5rem = 16px; 注：所有浏览器默认字号都是16px（某些安卓手机可以调置系统字号后，浏览器默认字号会受影响）

移动端知识点，用rem实现屏幕比例适配

说这么复杂，核心一样

= HEIGHT() WEIGHT()

不过，他们是通过一个rem的单位来实现的



rem就是html的font-size

使用的时候： width: 28 \* 32 rem

实际长度 = 标注长度 \* 标注图的rem长度(比率与后面这个一致) / 上面声明写的各种设备宽度对应的rem长度(一般是屏幕宽度/20)

ios: 实际长度 = 标注长度 \* 标注图的宽度/ 各种设备的宽度

一方面rem做了屏幕适配，另一方面，这个单位也可以用来规定字体

我感觉前端这样，是因为css里没有宏定义这个做法，然后因为ios那样写，数字太大，所以找了个公因数，除了一下

# 调试工具

## 模拟调试

现代主流浏览器均支持移动开发模拟调试，通常按F12可以调起，其使用也比较简单，可以帮我们方便快捷定位问题。

## 远程调试

模拟调试可以满足大部分的开发调试任务，但是由于移动设备种类繁多，环境也十分复杂，模拟调试容易出现差错，所以真机调试变的非常必要。

有两种方法可以实现真机调试：

1、将做好的网页上传至服务器或者本地搭建服务器，然后移动设备通过网络来访问。

2、借助第三方的调试工具，如weinre、debuggap、ghostlab等

真机调试必须保证移动设备同服务器间的网络是相通的。

# 媒体查询

设备终端的多样化，直接导致了网页的运行环境变的越来越复杂，为了能够保证我们的网页可以适应多个终端，不得不专门为某些特定的设备设计不同的展示风格，要实现这个目标的前提是必须有能力了解当前网页是运行什么终端设备，通过媒体查询可以做到这一点。

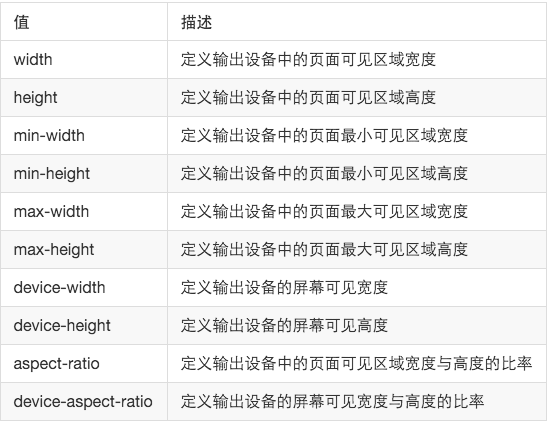
## 媒体类型

可以通过媒体类型对不同的设备指定不同的样式，一般我们使用screen。



## 媒体特性

判断条件或者查询依据

****

## 关键字

1、and 可以将多个媒体特性连接到一起，相当于“且”的意思。

2、not 排除某个媒体类型，相当于“非”的意思，可以省略。

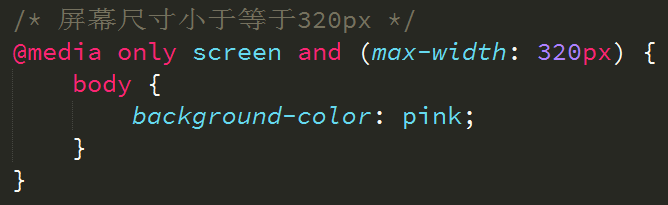
3、only指定某个特定的媒体类型，可以省略。

## 引入方式

1、link方法，见代码示例3-1.html

<link href="./3-1.css" media="only screen and (max-width: 320px)">

2、@media方法（写在CSS里），见代码示例3-2.html



## 常用特性

1、width / height 完全等于layout viewport，见代码示例3-3.html

2、max-width / max-height 小于等于layout viewport，见代码示例3-4.html

3、min-width / min-height 大于等于layout viewport，见代码示例3-5.html

4、device-width / device-height 完全等于ideal viewport，见代码示例3-6.html

5、orientation: portrait | landscape 肖像/全景模式，见代码示例3-7.html

[.....还有其它](https://developer.mozilla.org/zh-CN/docs/Web/Guide/CSS/Media_queries)

# CSS预处理器

CSS 预处理器是一种语言，用来为 CSS 增加一些编程的的特性，无需考虑浏览器的兼容性问题，并且你可以在 CSS 中使用变量、简单的程序逻辑、函数等等在编程语言中的一些基本技巧，可以让你的 CSS 更简洁，适应性更强，代码更直观等诸多好处。

常见的CSS预处理器有：LESS、SASS、Stylus等

其使用方法大致相同，我们在这里以LESS为例进行学习

## LESS



### 安装

1、安装Nodejs环境 Node Package Manager (验证 node -v npm -v)

2、打开控制台（cmd），执行npm install -g less (验证 lessc -v)

3、执行 npm install -g less-plugin-clean-css

4、命令行编译 lessc path/xxx.less path/xxx.css

### 编译

一般都会借助于编辑器插件来完成编译任务，以sublime\_text为例

1、ctrl+shit+p打开命令面板

2、输入install package然后回车

3、安装 LESS、lessc、Less2Css三个插件

4、alt+s快捷键即可实现编译

### 语法

1、变量

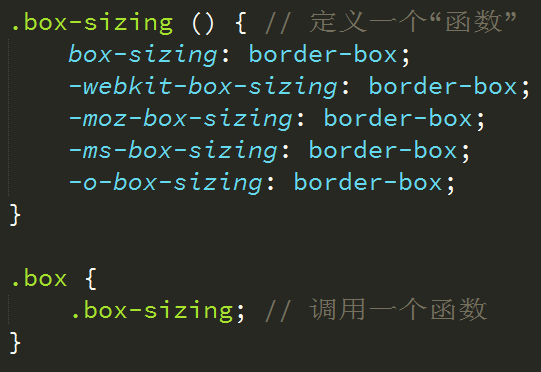
格式：@变量名: 值，定义完成后可以重复使用



见代码示例4-1.less

2、混合

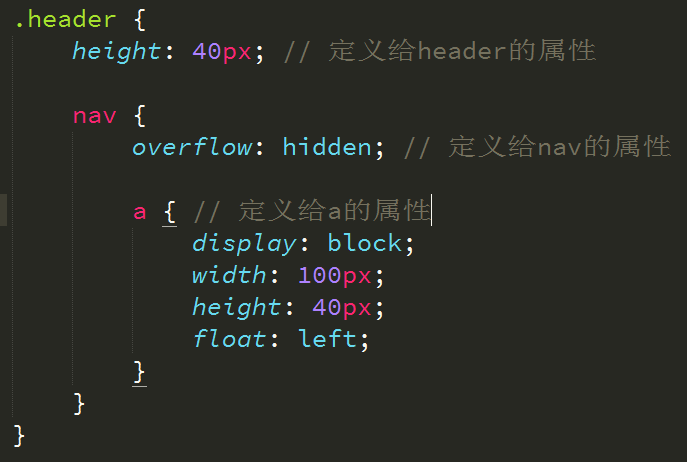
我们可以像使用函数一样来使用CSS



见代码示例4-2.less

3、嵌套

嵌套可以非常方便的管理我们的CSS层级关系



见代码示例4-3.html

**@import “路径” 我们可以引入其他的less文件**

**:not(.class): 过滤选择器，排除某一个，其他的都选**

### 浏览器中使用

我们可以引入一个less.js文件，实现实时的解析，而不必每次修改都要编译，最后完成所有开发任务后，再通过编辑器编译成css文件。

1、[下载](https://raw.github.com/less/less.js/v2.5.3/dist/less.min.js)然后引入less.js

2、引入xx.less文件，如：

<link rel="stylesheet/less" type="text/css" href="文件名.less" />

<script src = ‘less.js’>

注意：rel属性必须指定成stylesheet/less，并且demo.less要先于less.js引入。

必须以服务器方式访问，可以放到study目录下，或者webstrom自带服务器功能，因为script标签是没有访问本地文件的能力的，必须要通过浏览器来请求访问加载

这两行代码的本质是：

* + - 1. 先将内容替换，然后创建一个style标签
      2. 然后赋值，style.innerHTML = 替换的结果