

移动端rem布局适配方案

知识点:

1.明白em单位与rem的区别: em是相对父级的字体大小的单位,而rem是相对于html元素的字体大小的单位,rem的优势就是通过修改html中文字大小来改变页面中元素的大小

2.媒体查询(Media Query) 是CSS3的新语法


语法: @media mediatype and|not|only (media feature) {}

mediatype取值: all 所有设备 print:用于打印和打印预览 screen: 用于电脑屏幕,平板,手机

关键字: and并且 not不 only只

媒体特性: media feature width:定义输出设备可见区域宽度

min-width: 定义输出设备中页面可见的最小宽度 max-width:可见区域最大宽度

: @media screen and (min-width:320px) and (max-width:640px) {}

屏幕宽度大于等于320px 小于等于640px

3.rem + 媒体查询 例如定义屏幕宽度大于320px, 1rem = 50px

@media screen and (min-width:320px) {

html {

font-size: 50px;

}

}

4.less 简介

css是一门非程序式语言,没有变量,函数,以及作用域等概念,不方便维护,不利于复用,而且没有计算能力,因而衍生出了less.

less (leaner style sheets) 是一门css的扩展语言,也称为css预处理器,扩展了css的动态性,它没有减少css的功能,而是在css语法上,为css加入了程序式语言特性,引入了变量,运算以及函数功能,简化了css的编写

5.要使用less,先安装less `npm install -g less`

1).less 变量 @变量名: 值; 注意less 大小写敏感

2).less的编译,html中不可以直接使用less,需要把less编译为css文件,借助vscode插件:easy less

3).less 嵌套

针对后代选择器写法: 如果.header 中包含a标签,less中的写法如下

```
.header {  
  a {  
  }  
}
```

针对交集选择器 | 伪类选择器 | 伪元素选择器 如果内层选择器的前面没有&符号,则它被解析为父选择器的后代, 如果有&,他就被解析为父元素自身或父元素的伪类

```
.header {  
  a{  
    &:hover{}  
  }  
}
```

编译为css文件后:

```
.header a:hover {  
  
}
```

4).less的运算: 注意点:一 运算符中间一定要加空格隔开 二: 对于两个不同单位的值之间的运算,运算结果取第一个值的单位,如果两个值只有一个有单位,则结果取该单位

5).less文件中导入其他less文件 @import "common"

6.rem适配的两种解决方案:

1).less + rem + 响应式

利用媒体查询来设置不同屏幕尺寸下的html字体大小,利用less好用的扩展语法来编写css样式文件,利用rem/fontsize 的方式来得到最终的rem fontsize 一般是每份的宽度

2).flexible.js + rem (vscode插件 cssrem做辅助工具)

相对于第一种方案,flexible.js + rem 再也不用写不同屏幕下的媒体查询,只要屏幕宽度变了,布局就会发生变化,比媒体查询还要敏感

原理: 他把当前设备的屏幕尺寸平分为十等分,每一份的值作为html的字体大小.

cssrem作为rem的辅助计算工具,他默认html的字体大小为16px,在vscode中搜索cssroot 设置为设计稿宽度的1/10.

7.个人认为移动端页面搭建的最优方案

首先一定是单独制作移动端的页面

然后选择flex弹性布局来布局子元素的位置以及排列方式 (主要布局方式)

最后用flexible.js + rem (cssrem做辅助工具) 来单独做字体的适配,我认为flexible.js + rem 只用来做字体适配就可以,至于子元素的大小,没必要. (辅助布局方式)