第一章 响应式布局

第一节 响应式布局简介

- 什么是响应式布局 （https://www.microsoftstore.com.cn/）

  一套html布局去适配多个终端;

- 需兼容的尺寸
  - pc 普屏 1000px
  - pc宽屏 1200 / 1366 / 1440 / 1600 /1680 / 1920
  - pad 768 （横屏|竖屏 1024*768 | 768*1024）
  - phone 手机

第二节 媒体查询语法

- 媒体查询语法
  - @media screen and
  - min-width /max-width
  - orientation : landscape 横屏
  - orientation : portrait 竖屏
- 自定义container盒子的尺寸，由此引入媒体查询

```
【范例】 通过媒体查询实现栅格container和的尺寸

@media screen and (min-width:768px){

        .container{
                width:750px;
        }

}

@media screen and (min-width:992px){

        .container{
                width:980px;
        }

}

@media screen and (min-width:1200px){

        .container{
                width:1170px;
        }

}

// 原则 phone> pad > pc > pc宽屏

// 永远从最小的写起;
```

- 根据屏幕大小引入不同的样式文件

```
<link rel="stylesheet" type="text/css" src="pad.css" media="(min-width:768px)"/>
<link rel="stylesheet" type="text/css" src="pc.css" media="(min-width:992px)"/>
```

第三节 响应式页面实例

- 范例：利用媒体查询实现tech页面的响应式布局

> 页面结构的搭建
>
> 语义化标签
>
> header footer nav article aside audio video
>
> 页面样式
>
> 响应式布局的实现

第四节 响应式布局的实现思想

响应式布局，主要是在不同屏幕下，元素样式不同，

一般会显示或隐藏元素，或者是改变元素样式，设计师会给定不同尺寸的设计稿，

前端工程师只要实现不同屏幕尺寸下对应的样式效果就可以。

总结为: 搭建不同屏幕尺寸下的静态页面;

第二章 自定义ui组件库

第一节 样式框架的思想

> 通过控制外层容器类名，来批量修改子元素的样式

- 范例演示-- 通过简单的范例演示样式框架思想

第二节 自定义组件库

- 什么是ui组件库

  范例：http://www.zcool.com.cn/work/ZMTcyMTM4NTI=/1.html

- 定义一组大小和颜色不同的按钮与文本框

```
// 按钮和文本框会用到相同的颜色，由此可引入sass变量

 // 定义按钮的颜色

.btn-success{
     background-color:green;
 }

 .btn-alert{
     background-color:red;
 }


 // 定义文本框的颜色
 .has_success{
      border:1px green solid;
 }

 .has_error{
      border:1px red solid;
 }
```

第三章 sass简介

第一节 sass简介

> sass 和 less 都属于预编译css，为提高css应用的灵活性和效率，sass是基于ruby，而less是基于javascript，
>
> 在大文件编译方面，sass略胜一筹；sass功能会比less更强大，而less 属于极简。
>
> bootstrap框架最新版底层就是基于sass。

第二节 sass的安装与编译

- sass的安装
  - 下载和安装ruby
  - 下载和安装sass
- sass的编译
  - sass的文件格式

    > sass的编译 ： sass写好之后，在html页面中引入的仍然是css文件，只不过sass需要编译为对应的css再使用
    >
    > Sass 和 SCSS 其实是同一种东西，我们平时都称之为 Sass，两者之间不同之处有以下两点：
    >
    > (1) 文件扩展名不同，Sass 是以“.sass”后缀为扩展名，而 SCSS 是以“.scss”后缀为扩展名
    >
    > (2) 语法书写方式不同，Sass 是以严格的缩进式语法规则来书写，不带大括号({})和分号(;)，
    >
    > 而 SCSS 的语法书写和我们的 CSS 语法书写方式非常类似。

- 命令行方式

  > sass test.scss css/test.css
  >
  > sass --watch test.scss:css/test.css // 自动编译
  >
  > sass --watch app/sass:public/stylesheets // 监测目录

- GUI工具编译（koala软件）
  - 自动化编译（ide工具自动编译）

    > 注意事项： (1) 使用utf-8编码 (2) 使用英文或拼音的路径;
    >
    > GUI工具编译 推荐工具
    >
    > Koala (http://www.w3cplus.com/preprocessor/sass-gui-tool-koala.html)

- 编译的格式
  - **nested**： 嵌套缩进的css代码
  - **expanded**：展开的多行css代码（常规）
  - compact：简洁格式的css代码 （单行）
  - compressed：压缩后的css代码

    > 学习可以按默认或expanded
    >
    > 生产环境当中，一般使用最后一个选项。

- sass的导入
  - 为什么需要导入scss文件（模块化思想）
  - 导入scss文件的基本语法
  - 导入scss文件的优点

```
【范例】 导入其他scss文件，sass的import效率要高于css的 import

reset.scss

html,
body,
ul,
ol {
   margin: 0;
  padding: 0;
}

// base.scss

@import 'reset';

body {
  font-size: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

- sass的注释
  - // 单行注释　注意单行注释不会编译到css文件中;
  - /* 普通注释 */ 会出现在生成的css文件内;

第四章 sass的基本用法

第一节 sass变量的使用

- 变量的定义和赋值

  范例： 使用变量来定义媒体查询中设备尺寸;

  ```
  $screen-sm-min:768px;  // 平板
  $screen-md-min:992px;  // 普通桌面
  $screen-lg-min:1200px;  // 大屏
  ```

- 变量使用规则
  - sass的变量必须是以$开头;
  - 变量值和变量名之间冒号隔开（就像CSS属性设置一样）;

- 在属性中使用变量 #{$btn-color}
- 值后面加上 !default则表示默认值。
- 变量的默认值在组件化开发的时候非常有用

```
【范例1】 sass变量的使用

$fontStack:    Helvetica, sans-serif;
$primaryColor: #333;
$side = left;

body {
  font-family: $fontStack;
  color: $primaryColor;
  border-#{side}-radius:5px;
}
```

第二节 sass的嵌套

- 引入：用传统css方法做一个简单的导航
- sass的优点：简化css语法，层次结构清晰，可读性强
- 选择器嵌套和属性嵌套
- &父级选择器的使用（a的hover范例）

```
【范例2】 sass的嵌套使用

nav ul li;
nav li
nav a

// 选择器嵌套

nav {

  ul { list-style: none; }
  li {  display: inline-block; }
  a { text-decoration: none; }

  &:hover{ color: red; }

}

// 属性嵌套

.border{
     border{ width:1px; color:red; style:solid; }
}

相当于

.border{
    border-width:1px;
    border-color:red;
    border-style:solid;
}

// border:1px red solid;
```

第三节 sass的 mixin混合器的用法

> 如果你的整个网站中有几处小小的样式类似（例如一致的颜色和字体），那么使用变量来统一
>
> 当你的样式变得复杂，需要大段重用样式，可以通过`sass`的混合器实现大段样式的重用。

- 混合器的声明 @mixin 混合器名;
- 混合器的调用 @include 混合器名;
- 有参数的混合器声明与调用，参数使用$开头，多个参数使用逗号分隔 ;
- 多个参数的声明与调用

【范例1】混合器的基本用法 - 圆角的实现

```
@mixin rounded-corners {
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px;
  border-radius: 5px;
}

.notice {
  background-color: green;
  border: 2px solid #00aa00;
  @include rounded-corners;
}
```

【范例2】混合器传参-box-sizing的兼容写法;

```
@mixin box-sizing ($sizing) {
    -webkit-box-sizing:$sizing;
       -moz-box-sizing:$sizing;
            box-sizing:$sizing;
}

.box-border{
    border:1px solid #ccc;
    @include box-sizing(border-box);
}

//------------------------------
```

【范例3】 不透明度的兼容写法;

```
@mixin opacity($opacity:50) {
  opacity: $opacity / 100;
  filter: alpha(opacity=$opacity);
}

.opacity{
  @include opacity; //参数使用默认值
}

.opacity-80{
  @include opacity(80); //传递参数
}
```

【范例4】多参数的调用

```
@mixin horizontal-line($border:1px dashed #ccc, $padding:10px){
    border-bottom:$border;
    padding:$padding 0;
}
.text1 li{
    @include horizontal-line(1px solid #ccc);
}
.text2
    @include horizontal-line($padding:15px);
}
```

第四节 sass继承的用法

继承与混合器的区别

> 继承也是减少代码重用，不过与混合器不同，混合器主要用于展示性样式重用，而继承是基于类的，所以继承是建立在语义化关系基础之上，例如： .seriousError 和 .error ,因前者是基于后者，所以可以使用继承 extend ;

```
// 基于类的语义化关系

.error{
     color:red;
}

.seriousError{
    extend .error;
    border:1px red solid;
}
```

```
// 基于类的语义化关系

.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}

.error {
  @extend .message;
  border-color: red;
}

.warning {
  @extend .message;
  border-color: yellow;
}
```

第五节 sass的运算

```
.container { width: 100%; }

article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}

aside[role="complimentary"] {
  float: right;
  width: 300px / 960px * 100%;
}
```

第六节 Compass简介

```
//Compass 是一个非常丰富的样式框架，是sass核心团队开发出来的大量已经定义好的 mixin函数库

// 常用的函数

// 自定义部分写一个 Compass中的混合器

【范例】鼠标悬停颜色加深;

$linkColor: #08c;

a {
    text-decoration:none;
    color:$linkColor;
    &:hover{
      color:darken($linkColor,10%);
    }
}

// 此外还有 lighten($linkColor,10%);  颜色减轻10%;
```

第五章 使用sass搭建ui库

第一节 sass搭建ui组件库(按钮)

- 通过混合器传参来定义不同大小按钮;
- 通过混合器传参来定义不同颜色按钮;
- 定义按钮大小和颜色的混合器;
- 声明变量，对于按钮的大小和颜色进行定义;
- 实现混合器的声明，变量的声明模块化;

> 当sass文件名称以 “_” 开头，并且被用来引入到其他sass文件，这样的文件不会单独被编译，称为独立的模块，模块化的方式可更有利于sass文件的维护；

```
// 变量的定义

$padding-small-vertical:    5px !default;
$padding-small-horizontal:  10px !default;
$font-size-small:ceil(($font-size-base * 0.85)) !default;
$line-height-small:         1.5 !default;
$btn-border-radius-small:  $border-radius-small !default;
$border-radius-small:       3px !default;


// 混合器的声明

@mixin button-size($padding-vertical, $padding-horizontal, $font-size, $line-height, $border-
radius) {

  padding: $padding-vertical $padding-horizontal;
  font-size: $font-size;
  line-height: $line-height;
  border-radius: $border-radius;

}

// 使用混合器定义按钮
.btn-sm{
  @include button-size($padding-small-vertical, $padding-small-horizontal, $font-size-small,
                       $line-height-small, $btn-border-radius-small);
}
```

第二节 栅格布局的sass实现

- 栅格布局的概念
- 栅格布局的实现--百分比宽度
- 栅格布局的普通css定义实现
- 栅格布局使用sass来实现
    - sass循环的语法实现

```
// 通过sass循环可以批量定义
@mixin loop-grid-columns($class,$i:1){

    @for $i from 1 through 12{
        .col-#{$class}-#{$i}{
            width:$i/12*100%;
            float:left;
        }
    }

}

// 调用混合器
@include loop-grid-columns("sm");
@include loop-grid-columns("md");
@include loop-grid-columns("lg");
```

- 栅格布局偏移的sass实现

```
// 使用sass的条件判断来实现批量定义

@for $i from 1 through 12{

        @if ($type==width){
            .col-#{$class}-#{$i}{
                width:$i/12*100%;
                float:left;
            }
        }

        @if ($type==offset){
            .col-#{$class}-offset-#{$i}{
                left:$i/12*100%;
                position: relative;
            }
        }

 }
```

第六章 bootstrap 基础知识

第一节 bootstrap准备

- 下载bootstrap
- 目录结构的介绍
- 基本模板介绍

第二节 栅格系统

- 使用sass实现栅格布局

- 栅格布局的演示
- 栅格布局的组合使用
- 响应式列重置
- 列偏移，列嵌套，列排序

第三节 排版系统

第四节 bootstrap组件

第五节 bootstrap 插件

第六节 bootstrap实例项目