目录

- `scss`
- `git`

1.sass简介

- `CSS` 预处理器 -> 看 `html` 第三部分笔记2
- `sass` 和 `scss` 其实都是一种东西, 我们平常都称之为 `sass` ,不同之处有以下两点:
  1. 文件扩展名不同 `.sass` `.scss`
  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(;) scss 语法和书写 css`语法方式非常类似
- `bootstrap` 有 `sass` 和 `less` 编写的源码
- `sass` 中文网: https://sass-lang.cn/documentation
- `sass` 环境的安装:
  1. vscode
  2. ruby
  3. `gulp` (node 11.4.0)
     1. `cnpm init` 初始化
     2. 安装 `gulp cnpm i gulp@3.9.1 -D`
     3. 安装 `gulp-sass` 和 `sass`
        `cnpm i sass -D`
     4. 引入文件

```
var gulp = require('gulp')
var sass = require('gulp-sass')(require('sass'));
```

     5. 编写scss任务

```
gulp.task('scss',function(){
  console.log('完毕')
  return gulp.src("*.{sass,scss}")
  .pipe(sass())
  .pipe(gulp.dest('dist/css/'))
})
```

     6. 启动监听

2.scss的语法(变量、选择器嵌套)

- sass是古老的版本，现在使用的是scss
- 变量: `$width: 300px;`

- 默认变量: `!default`
  - `$width: 200px !default;`
    - 只要有新值把原来的值覆盖, 就使用新值

```
$width: 50px;
$width: 200px !default;
.box{
  width: $width; // 50px
  //当没有 default时,就是200px
}
```

- 全局变量: `$color:red;`
  - 局部变量:只在大括号内生效
    - `!global` 将局部变量提升为全局变量

```
.box{
    $color:blue !global;
}
```

- 特殊变量(插值) `#{$color}` 进行字符串拼接

```
$top:top;
.box3{
  border-#{$top}:red solid 2px;
  // border-top: red solid 2px;
}
```

- 选择器嵌套

```
ul{
  li{
    color:red
  }

  &:hover{}
  // &代表父级元素选择器
}
```

3.sass语法(混合)

- `@mixin 混合名字{}`
- 调用:
  `@include 混合名字;`
  - 可以选择有参数或没参数
- 无参数:

```
@mixin xxx{
  border: 1px solid black;
}

.box{
  @include xxx;
}
```

- 有参数:

```
@mixin xxx($width:1px, $color:black, $style: solid){
  border: $width $color $style;
}
// <1> 不传参, 使用默认值
{@include xxx;}
// <2> 按顺序传参
{@include xxx(2px,red,dashed);}
// <3> 指定参数传参
{@include xxx($color:red, $width:2px,$style:dotted);}
```

4.sass注释

```
// 普通注释      不会显示在.css文件中
/*
    多行注释     会出现在.css文件中, 不会出现在压缩版本中
*/
/*!
    强制注释     任何版本下都会保留
*/
```

5.scss数据类型(7种)

数据类型	示例
数字	`1.2`, `13`, `10px`
字符串	`"foo"`, `'bar'`, `baz`
颜色	`blue`, `#04a3f9`, `rgba(0,0,0,1)`
布尔值	`true`, `false`
空值	`null`
列表 `list`	用空格或逗号隔开 `c 1.5em 1em 0 2em Arial`
映射	`key1:value1, key2:value2` `map` 映射 键->值

- `scss` 数字函数
  - scss不同数据类型不能进行运算

    - `abs()`、`round()`、`ceil()`、`floor()`、`percentage()`:计算百分比、`min()`、`max()`
- `scss`列表函数 -> 下标从1开始

函数	描述
`length(list列表)`	获取列表长度
`nth(5px 10px,1)` -> `5px`	获取指定位置的列表项
`index(1px red,red)` -> `1`	查询指定元素的位置, 没查询到返回`Null`
`append(10 20, 5)`	添加一个元素
`join($lst1, $lst2)`	合并列表

- `scss`布尔值
    - 比较运算符: `> >= < <= != ==`
    - 逻辑运算符: `and` `or` `not`
- `scss`映射函数 `map`

```
$map: (key1:value1, key2:value2);
```

    - `length($map)` : 获取`$map`长度
    - `map-get($map, key)` : 获取`$map`中名称为`key`的值
    - `map-keys($map)` : 获取`$map`中的所有`key`,返回值: `list`
    - `map-values($map)` : 获取`$map`中的所有`value`
    - `map-has-key($map, key)` : 判断`$map`中是否包含`key`
    - `map-merge($map1, $map2)` : 合并`$map1`和`$map2`
    - `map-remove($map, key)` : 将制定名称的`key`从`$map`中移出

6.scss控制指令 (@if、@for、@each、@while)

```
@if($count > 4){
  width: 100px + $count;
}@else if($count <=3 ){
  width: $count + 20px;
}@else{
  width: $count + 10px;
}
```

```
// for循环
  @for $var from <开始值> through <结束值>
  @for $var from <start> to <end>
```

1. `to`和`through`都是表示一个区间
2. 唯一的区别就是停止循环的地方不一样

3. *var*可以是任何一个变量名称如i
4. `<start>`和`<end>`是`SASS`表达式 并且必须是整数

```
$count:5;
@for $i from 0 through $count{
  .box#{$i}{
    width: 10px / 12 * $i;
  }
}
```

```
// while循环
$num: 4;
@while $num >0 {
  .item#{$num}{
    width: 1em * $num;
  }
  $num: $num - 1;
}
```

7.scss函数

```
@function sum($i,$j){
  @return $i+$j
}

sum(1,2)
```

8.@warn @error报错

- `@warn "在这个映射里面没有#{$key}的值";`

```
$colors:(light:white,drak:black);

@function color($key){
  @if not map-has-key($colors, $key){
    @warn "在这个映射里面没有#{$key}的值";
    @error "在这个映射里面没有#{$key}的值";
  }
  @return map-get($colors, $key);
}
```

  - `@warn` 会在`gulp`监听控制台报错
  - `@error` 会在`gulp`监听控制台报错,并且中断监听

9.版本控制工具 git 准备工作

- `git`是多线程,云端代码编辑

1. 安装`git` https://github.com

2. 菜单中找到 `Git` 文件夹 有三个选项
    - `Git Bash`： 支持 `Linux` 命令的 `git` 控制台(常用)
    - `Git CMD`： 支持 `Windows` 命令的控制台
    - `Git GUI`： `git` 可视化界面
    - 苹果电脑自带 `git`
3. `VSCode` 安装 `git` 插件
    - `ctrl + ~` 打开终端
4. `github` 官方网站注册账号
    - https://github.com/
    - 账号: QQ邮箱
    - 密码: `wangbaole1024`

10.git命令操作

1. 创建文件夹 通过 `cd 路径` 进入文件夹
2. 配置一些 `git` 基本操作
    - `git config --global user.name "git上的用户名"`
    - `git config --global user.email "git上的邮箱"`
        - 没有消息就是好消息
    - `git` 概念
        - 本地磁盘目录: 工作区
        - 暂存区: 虚拟仓库(在本地磁盘)
        - `github` 仓库: 远程仓库
            - 工作区的文件先放到暂存区
3. 进行 `git` 操作， 仓库创建
    - `github` 仓库(远程仓库)的创建
        - 登录 `git` -> 右上方 `+` (加号) -> 第一个选项(新仓库) 进行新建仓库
    - 暂存区(虚拟仓库)的创建
        1. 先在文件夹内 进行 配置(上述 `步骤2`)
        2. `git init` 在本地进行初始化(建立暂存区)
            - 初始化完成后, 会在文件夹内生成 `.git` 隐藏文件夹,此时内存里的虚拟仓库就建立好了
            - `.git` 文件存储当前项目的所有版本信息
4. 此时就可以在 `init` 初始化的文件夹内 开始项目开发
    - `readme.md` -> 此文件是这个项目的说明书
5. 从工作区 -> 暂存区
    - `git add 文件名` 或 `git add *` (提交所有文件)
        - 每次提交后都要跟一次描述，
        - 提交描述的作用:提交描述后,就会有版本
    - `git commit -m` "这一次提交的表述"
6. 查看当前工作区的状态

    - `git status`
        - 英文提示: 没有什么可提交的，工作树干净;
7. 恢复文件 (暂存区文件 覆盖 本地磁盘)
    - `git checkout index.html` ： 恢复指定文件
    - `git checkout *` ： 恢复所有文件
8. 查看工作区和暂存区 版本的区别： `git diff`
9. `clear` 清屏操作 （`window`：`cls`）
10. `git log` 查看已经提交到暂存盘的历史版本
    - `commit 119f9a878d8e79534ba1bcb4d602f5f173b6986d`
        - `commit` 后面是版本号
11. 恢复文件到指定的某一个版本
    - `git reset --hard 版本号`
    - 如果从 第四个版本 恢复到 第一个 那么第 二、三版本都会消失

11.git命令操作2 将暂存区代码 放到远程仓库

1. 获取远程仓库链接：https://github.com/white1010293/wbltest1.0.git
2. 生成`ssh`密匙
    - `ssh-keygen -t rsa -C "github邮箱地址"`
        - 之后一直回车
3. `windows`电脑查找文件: 我的电脑 => 用户 => 用户名文件夹 => `.ssh`(隐藏文件) => `.pub`(存有密匙)
4. 去`githib`账户配置密匙
   头像 => `Setting` => 左侧`SSH and GPG` => `new SSH key` => 将密匙拷贝在`key`中,可以在`title`中给这个密匙取个名字
5. 将暂存区仓库 提交到 远程仓库
    1. `git remote add origin 仓库地址`
    2. `git push -u origin master`
    - 提交过程, 可能会输入用户名、密码
    - 第二次提交 只用 `git push`
6. `git clone 仓库地址`： 从远程仓库 克隆项目到本地
7. `git pull`： 从远程仓库 同步代码 到本地 (更新)