sass和git

1.SASS简介

CSS不是一个编程语言，可以用它来开发网页样式，但是没有办法用它进行编程。SASS的出现，让CSS实现了通过代码编程来实现的方式。

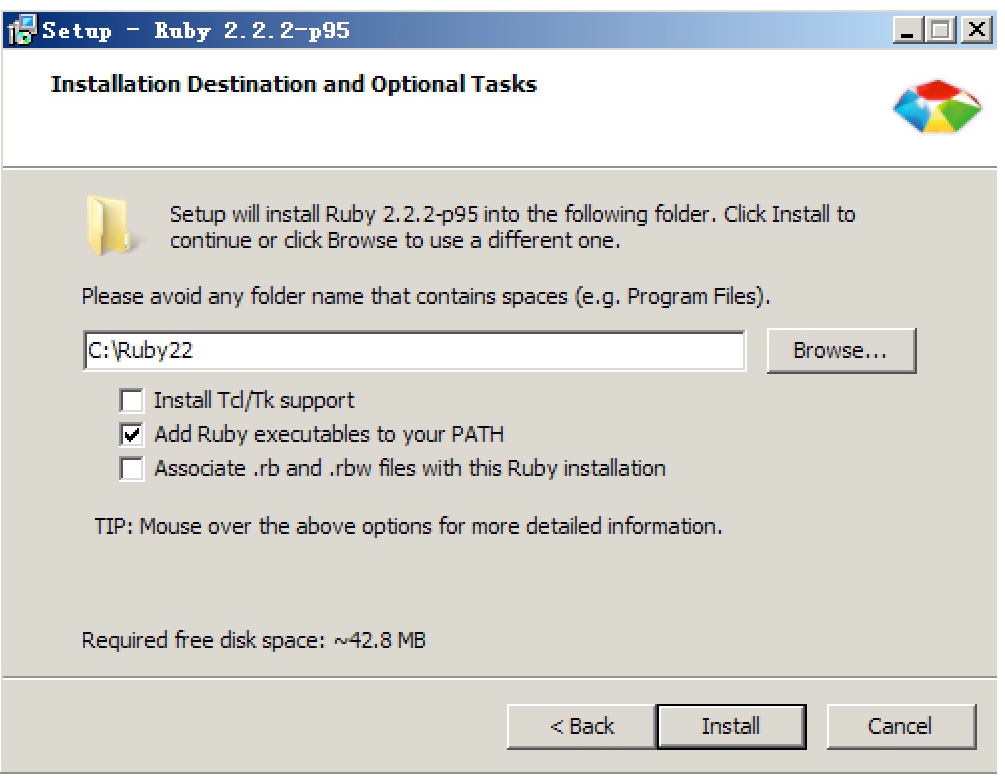
SO...SASS是一种CSS开发工具，提供了许多便利的写法，让CSS的处理实现了可编程处理。

SASS扩展了CSS3，增加了规则、变量、混入、选择器、继承等等特性，可以生成风格良好的CSS样式表文件，易于组织和维护。

2.SASS安装

sass基于Ruby语言开发而成，因此安装sass前需要安装Ruby。（注:mac下自带Ruby无需在安装Ruby!）

window下安装SASS首先需要安装Ruby，先从官网下载Ruby并安装。安装过程中请注意勾选Add Ruby executables to your PATH添加到系统环境变量



安装完成后需测试安装有没有成功,运行CMD输入以下命令：

ruby -v

//如安装成功会打印

ruby 2.2.2p95 (2015-04-13 revision 50295) [i386-mingw32]

//1.删除原gem源

gem sources --remove https://rubygems.org/

//2.添加国内源

gem sources -a https://gems.ruby-china.com/

//3.打印是否替换成功

gem sources -l

//4.更换成功后打印如下

\*\*\* CURRENT SOURCES \*\*\*

https://gems.ruby-china.com/

gem install sass

安装完成之后，你应该通过运行下面的命令来确认应用已经正确地安装到了电脑中：

sass -v

Sass 3.x.x (Selective Steve)

3.SASS的编译

SASS文件就是普通的文本文件，里面可以直接使用CSS语法。文件后缀名是.scss，意思为Sassy CSS。

下面的命令，可以在屏幕上显示.scss文件转化的css代码。（假设文件名为test。）

这个可以不用管：

　　\* nested：嵌套缩进的css代码，它是默认值。

　　\* expanded：没有缩进的、扩展的css代码。

　　\* compact：简洁格式的css代码。

　　\* compressed：压缩后的css代码。

生产环境当中，一般使用最后一个选项。

　　sass --style compressed test.sass test.css

简单使用案例：

test.sass

#box{

    width:100px;

    height:100px;

    background: myColor;

}

通过sass test.scss:test.css 命令

则可生成test.css文件

在html文件中直接引入新生成的test.css文件即可使用

除此之外，我们还可以监听scss文件

你也可以让SASS监听某个文件或目录，一旦源文件有变动，就自动生成编译后的版本。

　　// watch a file

　　sass --watch test.scss:test.css

　　// watch a directory 监听./mycss目录下的scss文件，将生成的css文件同样放入/mycss目录下

　　sass --watch ./mycss:mycss

   sass --style compact --watch ./mycss:mycss

4.基本语法

1）变量

sass的变量名必须是一个$符号开头，后面紧跟变量名

@charset "utf-8";

//注意加单位

$myHeight:200px;

$myColor:yellow;

$myFontSize:22px;

#box1 {

    background:$myColor;

    height: $myHeight;

    font-size: $myFontSize;

    width: 100px;

}

特殊变量：如果变量嵌套在字符串中，则需要写在 #{} 符号里面，如：

$myStringSize:size;

#box2 {

    background:blue;

    height: $myHeight;

    font-#{$myStringSize}: $myFontSize;

    width: 200px;

}

默认变量：仅需在值后面加入 !default即可， 默认变量一般用来设置默认值，当该变量出现另外一个值时，无论定义先后，都会使用另外一个值，覆盖默认值

$color: skyblue;

$color: blue !default;

#box3 {

    background:$color;

    height: $myHeight;

    font-size: $myFontSize;

    width: 100px;

}

2）计算功能

sass允许使用算式。

#box4 {

    background:$color;

    height: $myHeight;

    font-size: $myFontSize;

    width: 2 \* 30px;

}

3）嵌套

标签嵌套

sass文件

#box5 {

    width: 100px;

    height: 100px;

    background: yellow;

    color: #333;

    a {

       font-size:14px;

       &:hover {

          text-decoration:underline;

       }

    }

}

css文件

#box5 { width: 100px; height: 100px; background: yellow; color: #333; }

#box5 a { font-size: 14px; }

#box5 a:hover { text-decoration: underline; }

5、高级用法

1）函数 function

sass允许用户编写自己的函数，以@function开始

@charset "utf-8";

$fontSize: 10px;

@function pxTorem($px) {

    @return $px \* $fontSize;

}

#box6{

    width: pxTorem(10);

    height: pxTorem(20);

    background: green;

}

// css编译后样式

#box6 { width: 100px; height: 200px; background: green; }

2）if条件语句

  @if语句可以用来判断

// sass样式

$type: monster;

#box7 {

    width: 100px;

    height: 100px;

    @if $type == ocean {

        background: blue;

    } @else if $type == matador {

        background: red;

    } @else if $type == monster {

        background: green;

    } @else {

        background: black;

    }

}

// css编译后样式

#box7 { width: 100px; height: 100px; background: green; }

/\*# sourceMappingURL=test5.css.map \*/

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <link href="mycss/test5.css" rel="stylesheet" type="text/css" />

    </head>

    <body>

        <div id="box7">

            123

        </div>

    </body>

</html>

3）循环语句

for循环有两种形式，分别为：@for $var from <start> through <end> 和 @for $var from <start> to <end>。 $var 表示变量，start表示开始值，end表示结束值，两种形式的区别在于 through 包括 end 的值，to 不包括 end 值。

// sass样式

@for $i from 1 to 4 {

    .item-#{$i} {width: 2em \* $i;}

}

// css编译后样式

.item-1 {

    width: 2em;

}

.item-2 {

    width: 4em;

}

.item-3 {

    width: 6em;

}

while循环

// sass样式

$i: 2;

@while $i > 0 {

    .item-#{$i} {width: 2em \* $i;}

    $i: $i - 1;

}

// css编译后样式

.item-2 {

  width: 4em;

}

.item-1 {

  width: 2em;

}

4）混合(mixin)

sass中使用@mixin声明混合，可以传递参数，参数名以$符号开始，多个参数以逗号分开

sass文件:

@charset "utf-8";

@mixin opacity($opacity:50) {

  opacity: $opacity / 100;

  filter: alpha(opacity=$opacity);

}

//使用include老调用opacity

#opacity{

  @include opacity;      //参数使用默认值  50/100 = 0.5

}

#opacity-80{

  @include opacity(80); //传递参数  80/100 = 0.8

}

css文件:

#opacity { opacity: 0.5; filter: alpha(opacity=50); }

#opacity-80 { opacity: 0.8; filter: alpha(opacity=80); }

/\*# sourceMappingURL=test7.css.map \*/

6.git简介

**什么是Git**

Git是一个开源的分布式版本控制系统，用于敏捷高效的处理任何项目的版本问题。

Git 是Linus Torvalds 为了帮助管理Linux 内核开发而开发的一个开放源码的版本控制软件。

Git 与常用的版本控制工具CVS,SVN等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

**关于项目版本管理**

项目在开发的过程中，经常会出现多人分工协作进行项目分发并开发整合的过程，所以项目在刚开始流行的时候经常会出现一些协作开发的同步的问题，同时存在项目整体进度的控制和管理的问题，所以在程序开发行业衍生出来了版本管理工具

版本管理工具，首先是一个内容管理工具，可以将项目的内容信息存放在版本管理服务器上方便项目组人员进行访问和查询修改。

版本管理具有里程碑意义的主要有三个阶段

CVS阶段SVN阶段Git

 CVS阶段

 项目搭建开发过程中，每次提交项目都会将整个项目提交到服务器进行保存，服务器存储着项目的N个备份，开发过程中的协作效率较低，同时也出现了各种传输的问题，所以慢慢淡出了行业。

 SVN阶段

 考虑到CVS的缺陷，开发人员根据项目的实际情况，研发出专门针对项目版本控制的软件Subversion（简称SVN），SVN同样也是搭建服务器，让项目组成员将数据存储在服务器上，但是每次改动并提交的时候，SVN服务器并不重新保存整个项目的完整信息，而是和原来的项目进行对比，只保存改动的信息。这样就在很大的程度上对于项目版本服务器、项目协作效率有了显著的提升。所以至今为止，有很多公司依然选用SVN作为公司内部项目协作的版本控制软件

 Git阶段

 前面的CVS和SVN都是基于一个服务器的，如果脱离服务器，项目的版本保存就没有了任何意义，Git恰恰处理了这样的问题，Git是一个分布式的版本控制系统，在Git中即使用户离线，也能进行项目的提交和更新操作，等到下次连线服务器时进行整体的同步操作。

1. **安装**

Git（git\_for\_[windows.xp510.com.rar](http://windows.xp510.com.rar/)）

1. **使用**
   1. 进入到 git bash（命令行工具）
   2. 初始化user.name,user.email

 git config  --global user.name  "Your Name"

 git config  --global user.email  "[email@example.com](mailto:email@example.com)"

1. **开发项目**
   1. 进入HBulid新建项目（如：e:\web20170210），把git bash工具的当前路径转入到你的项目目录

   命令：

         cd  e:\web20170210

1. **给项目配置git仓库(建立仓库)**
   1. 初始化仓库：git init
      1. 在git bash里，转换路径： cd e:/ web20170210
      2. 输入 git init。就会在e:/ web20170210下产生目录 .git(隐藏目录)，表示e:\ web20170210里的项目代码（即所有的文件）会使用git进行版本管理。
2. **把开发的代码放在本地版本库**
   1. Git add

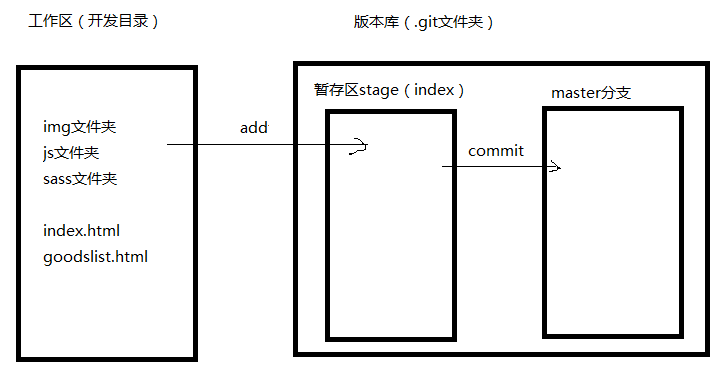
假设在项目下有个index.html文件。在 git Bash，输入：git   add   index.html

* 1. Git commit(提交)
     1. 在git Bash里输入： git commit -m "这是新建的index.html"。其中"这是新建的index.html"只是说明或者说解释。

理解：     add和commit；

         Add:添加；相当于打了标记（实际上是存储在了暂存区stage里），告诉git，下次提交时把该文件进行提交。

Commit：提交。把打过标记的（即用add进行添加的文件），一次性进行提交。即可以一次性把暂存区里的文件全部进行提交，提交了master分支（主分支）。提交完成后，暂存区里就没有文件了。



1. **恢复（跳转）到某个版本的文件**
2. 、查看所有的版本的命令

        git  log

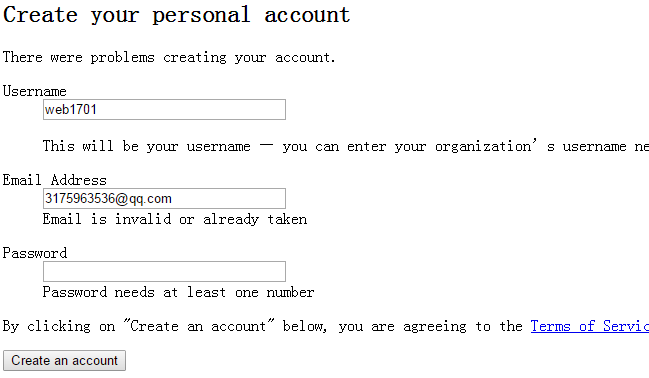
1. 、要恢复到上一个版本

        git reset  --hard   HEAD^

1. 、恢复到指定版本

       git  reset  --hard  版本序列号

1. **把本地版本库的代码放在远程库**
   * 1. **进入github官网（**[**https://github.com/**](https://github.com/)**）**
     2. **注册账号**



* + 1. **登录（Sign in）**

C:\Users\ADMINI~1.SKY\AppData\Local\Temp\enhtmlclip\Image(53).png

* + 1. **创建本地与远程的连接（用ssh）（搭桥）**
       - 1. **在本地创建ssh key（公钥）**。

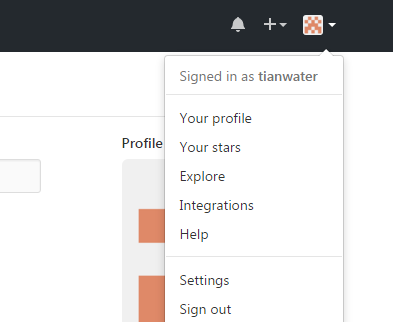
创建SSH Key：

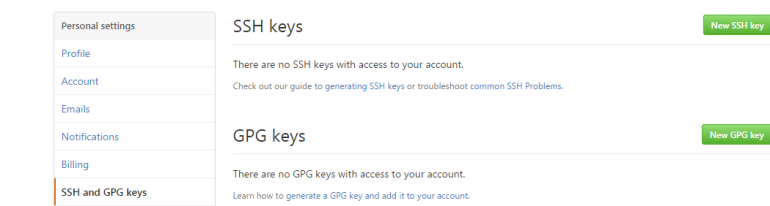
在git bash里输入： ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com)你需要把邮件地址换成你⾃⼰的邮件地址，然后⼀路回车，使⽤默认值即可.

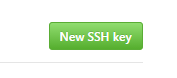
在⽤户主目录下(/c/Users/Administrator/.ssh/id\_rsa))，看看有没有.ssh目录，如果有，再看看这个目录下 有没有id\_rsa和id\_rsa.pub这两个⽂件，如果已经有了，可直接 跳到下⼀步。

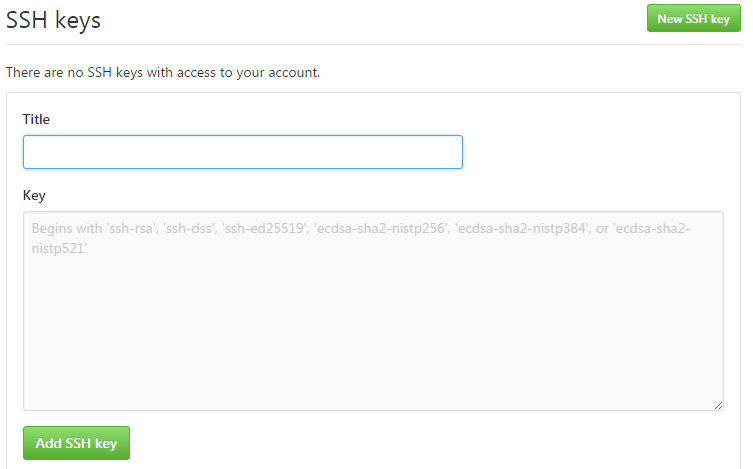
* + - * 1. **在远程创建ssh key**：

登陆GitHub，打开“Account settings”，“SSH Keys”⾯： 然后，点“Add SSH Key”，填上任意Title，在Key⽂本框⾥粘贴id\_rsa.pub⽂件的内容：





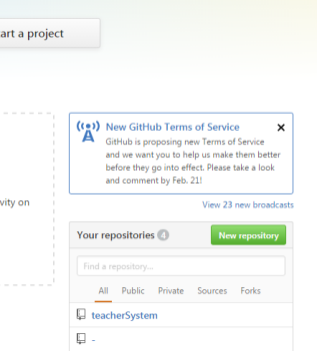




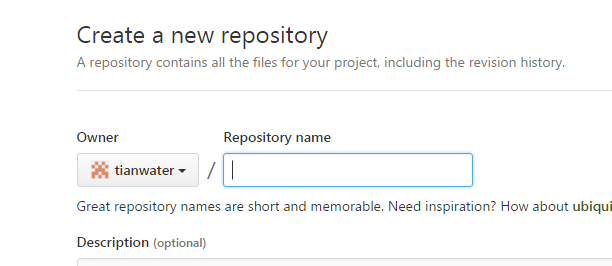
点击“add SSH Key”

* + 1. **创建远程仓库：**

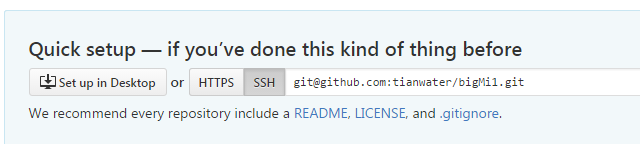
1. 在页面中找 “new repository”



1. 输入仓库名



点击“create repository”



* + 1. **把本地库的内容推送到远程库(确保本地执行过git add和git commit)**

git push -u origin master

注意：origin是远程库名，根据实际情况进行更改（或者写远程库的地址）

注意：确保本地执行过git add和git commit，即，本地版本库要有的版本。