1. Git

分布式版本控制系统

没有中央服务器，每个人的电脑都是一个完整的版本库。

2. Git 设置

① Git -> Git Bash

② git init

③ git config --global user.email "2205579492@qq.com"

④ git config --global user.name "wap233"

⑤ ssh-keygen 按三次enter

⑥ 进入磁盘C:\Users\hp\.ssh

⑦ 记事本格式打开"id\_rsa.pub", 并copy

⑧ 登陆Github, https://github.com/

⑨ Settings -> SSH and GPG keys -> New SSH key -> Title: 随便输, Key：copy ->Add SSH key

⑩ ssh -T git@github.com (yes)

3. 创建版本库

一：cd D：

二：mkdir 1www1

三：cd 1www1

四：mkdir testgit

五：cd testgit

六：pwd

七：git init

4. 版本添加和修改文件

一：新建readme.txt内容：11111111

二：git add readme.txt

三：git commit -m "readme.txt"

四：git status

五：修改readme.txt内容，添加一行：22222222

六：git status

七：git diff readme.txt

八：git add readme.txt

九：git commit -m "add 22222222"

十：git status

5. 查看历史记录

一：修改readme.txt内容，添加一行：33333333

二：git add readme.txt

三：git commit -m "add 33333333"

四：git log

五：git log --pretty=oneline

6. 版本回退

回退上个版本：git reset --hard HEAD^

回退上上个版本：git reset --hard HEAD^^

回退前n个版本：git reset --hard HEAD~n

查看readme文件：cat readme.txt

恢复回退：

获取恢复版本号：git reflog

恢复本版：git reset --hard 版本号

7. 工作区和暂存区

工作区：电脑上看到的目录(.git隐藏目录版本库除外)

版本库(Repository)：工作区中的隐藏目录.git不属于工作区，是为版本库。

版本库里面存了很多东西，其中就包括暂存区(stage)。

版本库Git，自动创建了第一个分支master，以及一个指向master的指针HEAD。

8. 使用Git提交文件到版本库的两步骤

第一步：是使用 git add 把文件添加进去，实际上就是把文件添加到暂存区。

第二步：使用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支上。

9. Git撤销修改操作

一：修改readme.txt内容，添加一行：44444444

二：git status

三：git checkout -- readme.txt

10. Git删除文件操作

一：新建a.txt文件

二：git add a.txt

三：git commit -m "a.txt"

四：rm a.txt

五：git rm a.txt

六：git commit -m "delete a.txt"

11. 直接进入仓库目录

cd d:/1www1/testgit

12. 添加远程仓库

一：登陆Github, https://github.com/

二：New repository

三：Repository name：testgit

四：Create repository

五：git remote add origin https://github.com/wap233/testgit.git

六：git push -u origin master

13. 同步本地仓库与远程仓库

本地仓库git commit后，还需要git push origin master

14. 克隆远程仓库到本地

git clone https://github.com/wap233/testgit.git

15. 退回上一级文件目录

cd ..

16. 创建分支

方法一：

1. git checkout -b branch\_wap
2. git branch

方法二：

1. git branch branch\_wap
2. git checkout branch\_wap
3. git branch

17. 删除分支

git branch --d branch\_wap

18. 分支合并操作

一：cat readme.txt

二：修改readme.txt内容，添加一行：77777777

三：cat readme.txt

四：git add readme.txt

五：git commit -m "分支wap增加内容77777777"

六：git checkout master

七：cat readme.txt

八：git merge branch\_wap

九：cat readme.txt

19. 解决冲突

一：git checkout branch\_wap

二：cat readme.txt

三：修改readme.txt内容，添加一行：88888888

四：cat readme.txt

五：git add readme.txt

六：git commit -m "分支wap增加内容88888888"

七：git checkout master

八：cat readme.txt

九：修改readme.txt内容，添加一行：99999999

十：cat readme.txt

十一：git add readme.txt

十二：git commit -m "分支master增加内容99999999"

十三：git merge branch\_wap

十四：出现冲突

十五：git status

十六：cat readme.txt

十七：修改readme.txt内容，手动解决冲突

十八：cat readme.txt

十九：git add readme.txt

二十：git commit -m "conflict fixed"

20. 禁用"Fask forward"模式

git merge --no-ff -m "merge with no-ff" branch\_wap

21. bug分支

一：git checkout branch\_wap

二：修改readme.txt内容，删除最后一行

三：git status

四：git stash

五：git status

六：git checkout -b issue-404

七：修改readme.txt内容，删除最后一行后新增一行：99999999

八：cat readme.txt

九：git add readme.txt

十：git commit -m "fix bug 404"

十一：git checkout branch\_wap

十二：git status

十三：git stash list

十四：git stash apply / git stash pop

十四：git stash drop

十五：git stash list

22. 推送分支：

一：git remote

二：git remote -v

三：git push origin master

四：git checkout -b branch\_wap origin/branch\_wap

五：git push origin branch\_wap

23. 推送冲突

一：git push origin branch\_wap

二：git branch --set-upsttream branch\_wap origin/branch\_wap

三：git pull

四：cat readme.txt

五：若还有冲突，手动修改后commit

六：git push origin branch\_wap

24. 常用命令

mkdir： XX (创建一个空目录 XX指目录名)

pwd： 显示当前目录的路径。

git init 把当前的目录变成可以管理的git仓库，生成隐藏.git文件。

git add XX 把xx文件添加到暂存区去。

git commit –m “XX” 提交文件 –m 后面的是注释。

git status 查看仓库状态

git diff XX 查看XX文件修改了那些内容

git log 查看历史记录

git reset –hard HEAD^ 或者 git reset –hard HEAD~ 回退到上一个版本

(如果想回退到100个版本，使用git reset –hard HEAD~100 )

cat XX 查看XX文件内容

git reflog 查看历史记录的版本号id

git checkout — XX 把XX文件在工作区的修改全部撤销。

git rm XX 删除XX文件

git remote add origin https://github.com/ev-power/XiaoYong 关联一个远程库

git push –u(第一次要用-u 以后不需要) origin master 把当前master分支推送到远程库

git clone https://github.com/ev-power/XiaoYong 从远程库中克隆

git checkout –b dev 创建dev分支 并切换到dev分支上

git branch 查看当前所有的分支

git checkout master 切换回master分支

git merge dev 在当前的分支上合并dev分支

git branch –d dev 删除dev分支

git branch name 创建分支

git stash 把当前的工作隐藏起来 等以后恢复现场后继续工作

git stash list 查看所有被隐藏的文件列表

git stash apply 恢复被隐藏的文件，但是内容不删除

git stash drop 删除文件

git stash pop 恢复文件的同时 也删除文件

git remote 查看远程库的信息

git remote –v 查看远程库的详细信息

git push origin master Git会把master分支推送到远程库对应的远程分支上

25. html链接css

<link rel="stylesheet" href="{{url\_for('static',filename='welcomepage/background.css')}}">

"内部再用引号，必须使用'单引号'。"

26. HTML

Hyper Text Markup Language 超文本标记语言

27. <!DOCTYPE>

<!DOCTYPE> 声明位于文档中的最前面的位置，处于 <html> 标签之前。

<!DOCTYPE> 声明不是一个 HTML 标签；它是用来告知 Web 浏览器页面使用了哪种 HTML 版本

<!DOCTYPE html>

HTML5标准网页声明

28. 简单的HTML5文档

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>文档标题</title>

</head>

<body>

文档内容......

</body>

</html>

29. <meta>

<meta charset="UTF-8">

对于中文网页需要使用 <meta charset="utf-8"> 声明编码

<meta charset="gbk">

有些浏览器会设置GBK为默认编码

30. <meta name="viewport" content="width=device-width,initial-scale=1">

name:

viewport (显示窗口)

content:

width:可视区域的宽度，值可为数字或关键词device-width

height:同width

intial-scale:页面首次被显示是可视区域的缩放级别，取值1.0则页面按实际尺寸显示，无任何缩放

maximum-scale=1.0, minimum-scale=1.0;可视区域的缩放级别，

maximum-scale用户可将页面放大的程序，1.0将禁止用户放大到实际尺寸之上。

user-scalable:是否可对页面进行缩放，no 禁止缩放

31. <title>

<title> 标签定义文档的标题，在所有 HTML 文档中是必需的。

32. <link rel="stylesheet" href="{{url\_for('static',filename='XX.css')}}">

rel是relationship的英文缩写

stylesheet中style是样式的意思,sheet是表格之意,总起来是样式表的意思

rel="stylesheet" 描述了当前页面与href所指定文档的关系.即说明的是,href连接的文档是一个新式表