Git方面

1.进入项目目录

cd 项目目录的路径 #进入项目目录

2.初始化本地git仓库

git init #初始化一个git仓库(repository)，初始化完成之后会在

项目目录中自动创建一个.git隐藏文件夹，此文件夹是

git的版本记录，用来跟踪或者管理git仓库

3.查看版本库的状态

git status #查看当前git仓库的状态

git status -s #简单方式查看git仓库的状态

git diff #查看修改和新增内容

4.把工作区的文件添加到暂存区

git add 文件名 #把没有添加到暂存区的文件添加到暂存区(文件名可写多个)

git add . #把工作区没有添加到暂存区文件全部添加到暂存区(一次添加多个)

5.把缓存区的文件提交到本地仓库

git commit -m '提交的备注信息' #把加入暂存区的文件提交到本地仓库

commit 提交，-m表示给提交的内容添加描述信息，用来备注本次

提交的信息。

注意：提交的备注信息不能省略，写的要有意义

（有的公司会作为工作内容的考核）

6.查看版本库的历史操作记录

git log #查看git操作的详细历史记录

比如：

commit b979bc1782e8f56750554323474da72ebca03553 #本次操作的版本号(commit id 版本号是唯一的)

Author: \*\*\* <\*\*\*@qq.com> #用户信息

Date: Thu Jan 19 11:40:55 2017 +0800 #提交的日期

初始化了git仓库并创建index.txt #提交的备注信息

git log --oneline #把git操作日志简化为一行显示

git log --pretty=oneline #把git操作日志格式化成为一行

git log –pretty=oneline –abrev-commit #格式化日志使用缩略id的形式

git reflog #查看操作命令及日志

二、版本回退

1.修改的内容还没有添加到暂存区

git checkout -- 文件名 #撤销工作区最后一次修改的内容，

前提是修改的内容还没有添加到暂存区

2.修改的内容已经添加到暂存区，但是还没有提交到本地仓库

先使用 git reset HEAD 文件名 #撤销添加到暂存区的内容

再使用 git checkout -- 文件名 回退到上一个状态

3.修改的内容已经提交到了本地仓库

git reset --hard 版本号 #回退到指定的版本号的状态

git reset logID –hard #代码回退到指定的位置

三、本地仓库和远程仓库的连接

1.从远程仓库克隆到本地

git clone https://git.oschina.net/jxz0522/H21weather.git #克隆项目，把远程库创建的好的(已经有的项目)项目同步到本地

git branch 分支名 #创建新分支

git checkout 分支名 #迁出(切换到)分支

git merge 需要合并的分支名 #把指定分支的代码合并到当前分支（合并在某个分支前需要先切换到某个分支）

git log --graph #图形化的方式 查看历史记录

git branch –d 分支名 #删除分支

git pull #同步远程库的文件到本地git

git push #同步本地的文件到远程库

svn 集中式源代码版本管理工具

版本历史都集中在在中央服务器上(仅此一份)，本地只有最新版本。

集中式对中心服务器的依赖性很强，工作时需要保持网络连接；

没有中心服务器的时候无法提交，也无法查看历史版本或不能

进行其他大部分操作

git 分布式源代码版本管理工具

本地有完整的历史版本，可以随时查看历史，进行提交、分支合并等操作，

不存在中心服务器，不需要网络连接就可以完成几乎所有的版本管理工作；

在有网络的情况下文件可以复制很多版本(成千上万的分支)，并分散到

很多电脑上

使用git可以在没有网络的情况下，无论是提交代码还是查看历史，还是

分支、合并都可以秒完!

git允许你很频繁的提交代码，更加详尽的记录开发过程！

Git 提交不不要的文件或文件夹配置

1.在仓库中创建隐藏文件“.gitgnore”，在gitbash中输入命令touch.gitigonre

2.用文本编辑器输入需要忽略的文件名或文件夹名：

##ignore this file##

写在这里

/文件夹/

\*.后缀名

##filter databfile、sln file##

\*.mdb

\*.ldb

\*.sln

##class file##

\*.com

\*.class

\*.dll

\*.exe

\*.o

\*.so

# compression file

\*.7z

\*.dmg

\*.gz

\*.iso

\*.jar

\*.rar

\*.tar

\*.zip

\*.via

\*.tmp

\*.err

# OS generated files #

.DS\_Store

.DS\_Store?

.\_\*

.Spotlight-V100

.Trashes

Icon?

ehthumbs.db

Thumbs.db

Vm编辑器：

插入命令（i）、保存(w)、退出(q)

nodejs方面

node –v #查看node的版本

npm –v #查看npm的版本

npm i npm –g #npm全局安装

npm i 模块名 #使用npm命令安装模块

npm uninstall 模块名 #卸载指定的模块

npm install 模块名 –g #全局安装模块

npm i 模块名@版本号 --save #安装指定版本的模块

npm i 模块名 --save/-save-dev #安装模块名并自动更新package.json

npm uninstall 模块名 --save/-save-dev #安装模块名并自动更新package.json

npm list –g #查看所有全局安装的模块

npm list 模块名 #查看指定模块的版本号

npm update 模块名 #更新模块

npm search 模块名 #搜索模块

npm cache clear #清除npm缓存

npm init #初始化一个空白的nodejs项目

npm i anywhere –g #安装anywhere临时服务器演示

node 文件名/文件名.js #直接运行指定js文件

**eslint部分**

npm i eslint –g #全局安装eslint

eslint –init #初始化一个配置文件

**vue部分**

npm i vue-cli –g #全局安装vue-cli

vue init webpack 项目名 #创建一个基于webpack模板的项目，使用webpack进行打包处理

npm i #安装依赖项

npm run dev #运行项目

**element部分**

npm i element-ui –s #安装element-ui组件库

**gulp部分**

npm i gulp –g #全局安装gulp

npm i –save-dev gulp #项目的开发依赖项(devdependencied)安装

**antd部分**

npm i antd –save #使用npm命令安装antd

yarn add antd #使用yarn命令安装antd

npm i antd-init –g #安装脚手架工具

npm start #开发调试

**mint –ui部分**

npm i mint-ui #安装组件库，重新启动项目

npm run build #把vue开发的项目进行打包放在dist文件夹下

**iView部分**

npm i iview --save #安装iview组件

**art-template部分**

npm i art-template –save #安装art-template模板引擎

**Vuex部分**

npm i vuex --save #使用npm命令安装vuex

yarn add vuex #使用yarn命令安装vuex

**vux部分**

npm i vux --save #使用npm命令安装vux

yarn add vux #使用yarn命令安装

yarn upgrade vux #更新yarn

npm run build –report #生成并查看捆绑信息

**redux方面**

npm i --save redux #安装redux

npm i --save react-redux #React捆绑库

npm i –save-dev redux-devtools#开发者工具

**dva框架搭建react项目**

npm i dva-cli -g #全局安装dva命令终端

dva new 项目名 #初始化dva项目

dva g route 路由组件名 #生成一个路由文件(routes文件夹下)

dva g model 模型名 #生成模型文件(models文件夹下)

dva g component 组件名 #生成一个组件文件(components文件夹下)

npm i –g create-react-app #

create-react-app 项目名

cd 项目名

npm start

**angular.js方面**

npm i angular-cli –g #全局安装angular

ng new 项目名 #创建新项目

ng g component 组件名 #创建新组件

ng server #启动server

ng build –prod #默认生成dist目录，并将打包内容放置在dist里面

ng –v #查看angular版本

npm i –g @angular/cli@lates#全局安装angular命令行工具

npm remove –g angular-cli #卸载angular

**Cordova方面**

npm i –g cordova #安装cordova

cordova create 项目名 #创建一个新项目

cordova platform add 平台 #添加一个平台

cordova run 项目名 #运行app

cordova build 平台 #运行编译