git操作规范

创建与合并分支

1、 从master分支创建dev分支并切换到dev分支

```
git checkout master

git checkout -b dev
```

其中，git checkout -b dev 等价于:

```
git branch dev

git checkout dev
```

(1)

```
git branch
```

查看本地当前的分支，分支前面带“*”表示当前分支，剩下的分支表示本地有的分支。

(2)

```
git  branch  -a
```

查看远程全部的分支，白色的表示本地有的，红色的表示本地没有，仅在远程存在。

2、修改代码、提交代码（当前的操作是在dev分支上进行）

```
git add a.html

git commit -m "提交文件a.html"
```

3、分支合并(将dev合并到master)

```
git checkout master

git merge dev
```

4、合并完成后，删除dev分支.(删除dev分支时，注意我们当前所在的分支不能是dev分支)

```
git branch -d dev
```

5、删除后，查看分支(此时看不到dev分支了)

```
git branch
```

6、总结： 工作中经常从master创建新的分支，具体操作如下

```
master创建新分支:

git checkout master

git checkout -b  issues1234

git push origin issues1234

git add ..

git commit -m "***"

git push origin issues1234
```

7、删除分支:

```
git branch -D   issues1234  //本地强制删除分支issues1234

git push origin  :issues1234  //推到远程
```

解决冲突

1、发生冲突的文件

```
<<<<<<< HEAD
Creating a new branch is quick & simple.
=======
Creating a new branch is quick AND simple.
>>>>>>> feature1
```

其中，git使用<<<<<<<，=======，>>>>>>>标记文件中自己和别人产生冲突的部分。

在<<<<<<<，=======之间为自己的代码； =======，>>>>>>>之间为别人的代码。

如果保留自己的代码，将别人的代码删掉即可。

2、冲突解决后提交

```
git status

git add ***

git commit -m "fix conflict"

git push origin 分支名
```

Bug分支

1、储藏更改:将当前更改的代码储藏起来，等以后恢复使用

```
git stash
```

2、恢复储藏的代码

```
git stash pop //恢复的同时把stash内容删掉
```

```
或者

git stash apply  //恢复stash，但是stash内容并不删除

git stash drop //在上面操作的基础上，以此来删除stash


注：  git stash list //查看全部的stash列表
```

版本回退

1、回退至上一个版本

```
git reset --hard HEAD
```

2、回退至指定版本

```
git reset --hard  版本号
```

3、查看以往版本号(本地的commit)

```
git reflog
```

4、查看各版本号及信息(所有的commit：本地commit + 其他同事的commit)

```
git log
```

撤销修改

1、撤销修改

```
git  checkout -- a.html
```

分两种情况分析：

①： 还没有执行 `git add` 操作，执行上面的操作后，会恢复到和版本库中一模一样的版本状态。

②： 执行了`git add` ，还没执行 `git commit` ,再执行上面的操作后，会恢复到`git add` 结束后的状态

注：一旦执行了git commit -m "***"，就不能再使用上面的命令回退。

对于已经push的版本，进行回退

1、第一步：

```
git reset --hard 版本号 //本地回退到指定的版本
```

2、第二步：

```
git push  -f origin dev    //将远程的也回退到指定版本
```