这些开发中很常见的问题,所以git的取消提交,回退甚至返回上一版本都是特别重要的.  
大致分为下面2种情况:  
  
1.没有push  
这种情况发生在你的本地代码仓库,可能你add ,commit 以后发现代码有点问题,准备取消提交,用到下面命令  
reset  
git reset [--soft | --mixed | --hard  
  
上面常见三种类型  
  
--mixed  
会保留源码,只是将git commit和index 信息回退到了某个版本.  
git reset 默认是 --mixed 模式   
git reset --mixed 等价于 git reset  
  
--soft  
保留源码,只回退到commit 信息到某个版本.不涉及index的回退,如果还需要提交,直接commit即可.  
  
--hard  
源码也会回退到某个版本,commit和index 都回回退到某个版本.(注意,这种方式是改变本地代码仓库源码)  
当然有人在push代码以后,也使用 reset --hard <commit...> 回退代码到某个版本之前,但是这样会有一个问题,你线上的代码没有变,线上commit,index都没有变,当你把本地代码修改完提交的时候你会发现权是冲突.....  
所以,这种情况你要使用下面的方式  
  
2.已经push  
对于已经把代码push到线上仓库,你回退本地代码其实也想同时回退线上代码,回滚到某个指定的版本,线上,线下代码保持一致.你要用到下面的命令  
  
revert  
git revert用于反转提交,执行evert命令时要求工作树必须是干净的.  
git revert用一个新提交来消除一个历史提交所做的任何修改.  
revert 之后你的本地代码会回滚到指定的历史版本,这时你再 git push 既可以把线上的代码更新.(这里不会像reset造成冲突的问题)  
  
revert 使用,需要先找到你想回滚版本唯一的commit标识代码,可以用 git log 或者在adgit搭建的web环境历史提交记录里查看.  
git revert c011eb3c20ba6fb38cc94fe5a8dda366a3990c61  
  
通常,前几位即可  
git revert c011eb3  
  
git revert是用一次新的commit来回滚之前的commit，git reset是直接删除指定的commit  
看似达到的效果是一样的,其实完全不同.  
第一:  
上面我们说的如果你已经push到线上代码库, reset 删除指定commit以后,你git push可能导致一大堆冲突.但是revert 并不会.  
第二:  
如果在日后现有分支和历史分支需要合并的时候,reset 恢复部分的代码依然会出现在历史分支里.但是revert 方向提交的commit 并不会出现在历史分支里.  
第三:  
reset 是在正常的commit历史中,删除了指定的commit,这时 HEAD 是向后移动了,而 revert 是在正常的commit历史中再commit一次,只不过是反向提交,他的 HEAD 是一直向前的.