1.直接进入沙盒

在 URL 后头加上 ?NODEMO 就可以了

2. 相关篇节

2.1 基础篇

1. Git Commeit
   - 提交记录: 保存的是你的目录下所有文件的快照,就是ctrl+c，但是更优雅、轻便
   - 可以做到的结果：可以快速地在这些提交记录之间切换
   - hide goal命令关闭窗口
2. Git Branch
   - 分支：早建分支！多用分支！
   - 与上面的提交记录结合起来
     - 一种结合方法：新创建的分支 newImage 指向的是提交记录
       - 切换新分支：`git checkout newImage;git commit`并且修改保存到新分支中
       - 简洁的切换方法(创建新分支并切换新分支)：`git checkout -b <your-branch-name>`
3. Git Merge
   - 用处：两个分支都是独立但是没有整体，如果需要整体的提交记录这时候就可以采用合并分支这一个命令
     - `git merge bugFix`
   - 合并分支的几个步骤

```
git branch bugFix
git commit
git checkout master
git merge bugFix
随时用objective打开对话框
```

4. Git Rebase（另外一种合并方案）
   - 本质：实际上就是取出一系列的提交记录，“复制”它们，然后在另外一个地方逐个的放下去
   - 代码实现：

```
git checkout -b bugFix  新建并切换回bugFix
git commit  提交一次
git checkout master  切换回master
git commit     再提交一次
git checkout bugFix  切换回bugFix
git rebase master    rebase合并到master
```

5. 树上进行移动
    1. HEAD
        - 一个对当前检出记录的符号引用 —— 也就是指向你正在其基础上进行工作的提交记录。
        - 总是指向当前分支上最近一次提交记录，修改提交树也是针对head开始的；
        - 对提交做的一些更改，可以通过其看到。
        - 可以通过 cat .git/HEAD 查看head指向；
        - 指向的是一个引用，还可以用 git symbolic-ref HEAD 查看它的指向

```
git checkout C1;
git checkout master;
git commit;
git checkout C2;
```

2. 分离的 HEAD
    - 让其指向了某个具体的提交记录而不是分支名
    - 待解决的问题：
        - 想完成此关，从 bugFix 分支中分离出 HEAD 并让其指向一个提交记录。
        - 通过哈希值指定提交记录。每个提交记录的哈希值显示在代表提交记录的圆圈中。
3. 相对引用：
    - 通过指定提交记录哈希值的方式在 Git 中移动不太方便，在git中只需要提供能够唯一标识提交记录的前几个字符即可。
    - 2个非常有用的操作：
        - a.使用 `^` 向上移动 1 个提交记录
        - b.使用 `~<num>` 向上移动多个提交记录，如 ~3
        - 代码构成：

```
git checkout C3;
git checkout HEAD^;
git checkout HEAD^;
git checkout HEAD^;
```

- 强制修改分支位置
    - 直接使用 -f 选项让分支指向另一个提交：`git branch -f master HEAD~3`

6. 撤销变更：
    - 组成部分：底层部分（暂存区的独立文件或者片段）和上层部分（变更到底是通过哪种方式被撤销的）

        - 我们关注的是后者：上层部分
    - 两种撤销更改指令：
        - `git reset`(本地)
        - `git revert`(远程)：用一个新提交来消除一个历史提交所做的任何修改
    - 完成的任务：
        - 分别撤销 local 分支和 pushed 分支上的最近一次提交。共需要撤销两个提交（每个分支一个）。
        - 记住 pushed 是远程分支，local 是本地分支 —— 这么说你应该知道用分别哪种方法了吧？

2.2 处理复杂问题

2.2.1 修改提交树

1. 自由修改提交树`git cherry-pick`
    - 清楚知道所需要提交的记录
2. 交互式的`rebase`
    - 从一系列的提交记录中找到想要的记录
    - rebase --interactive(-i)

- 当 rebase UI界面打开时, 你能做3件事:

    - 调整提交记录的顺序（通过鼠标拖放来完成）
    - 删除你不想要的提交（通过切换 pick 的状态来完成，关闭就意味着你不想要这个提交记录）
    - 合并提交。 遗憾的是由于某种逻辑的原因，我们的课程不支持此功能，因此我不会详细介绍这个操作。简而言之，它允许你把多个提交记录合并成一个。

2.3 杂项

1. 本地自由栈式提交
    - 针对各个不同分支有一些调试或者打印的提交记录：复制解决问题的那几个提交就是
    - 提交的技巧
        - 先用 `git rebase -i` 将提交重新排序，然后把我们想要修改的提交记录挪到最前
        - 然后用 `commit --amend` 来进行一些小修改
        - 接着再用 `git rebase -i` 来将他们调回原来的顺序
        - 最后我们把 `master` 移到修改的最前端（用你自己喜欢的方法），就大功告成啦！