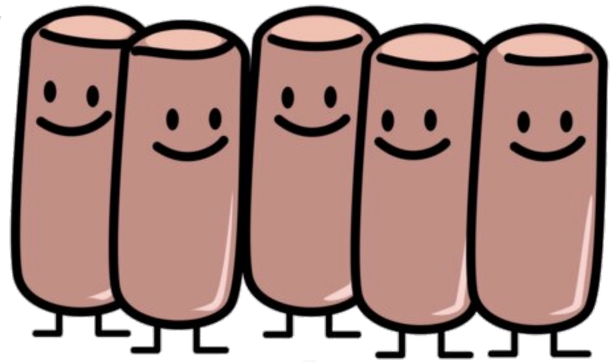
Исправляем ошибки прошлого

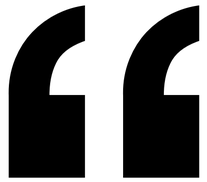
Seitmagambet Olzhas

winexy

ДИСКЛЕЙМЕР

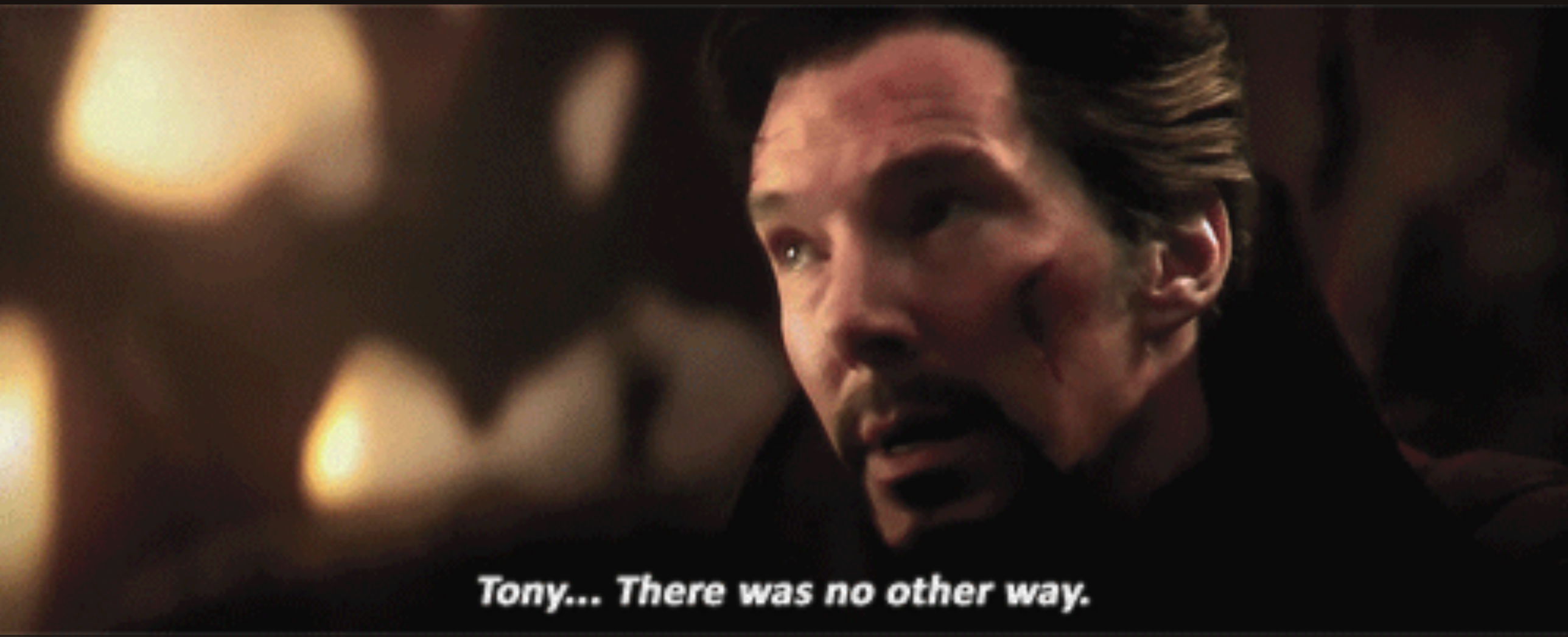
ДОКЛАД НЕСЕТ ЮМОРИСТИЧЕСКИЙ ХАРАКТЕР.
АВТОР МАТЕРИАЛА ДРАМАТИЗИРУЕТ.
ВСЕ СОВПАДЕНИЯ СЛУЧАЙНЫ.





The process of developing software, similar to the process of making sausage, is a messy messy business; all sorts of stuff happens in the process of developing software. Bugs are inserted into the code, uncovered, patched over. The end result may be a tasty program, but anyone looking at the process of how it was created (through inspection of the commits) may end up with a sour taste in their mouth. If you hide the sausage making, you can create a beautiful looking history where each step looks as delicious as the end-product.

http://sethrobertson.github.io/GitBestPractices/#sausage


Tony... There was no other way.

Типичная ситуация

```
+ import { fun } from './fun';
+
+ function main() {
+   fun();
+ }
+
+ main();
```

```
+ function fun() {
+   console.log('fun');
+ }
```

`git commit -m “feat: add main”`

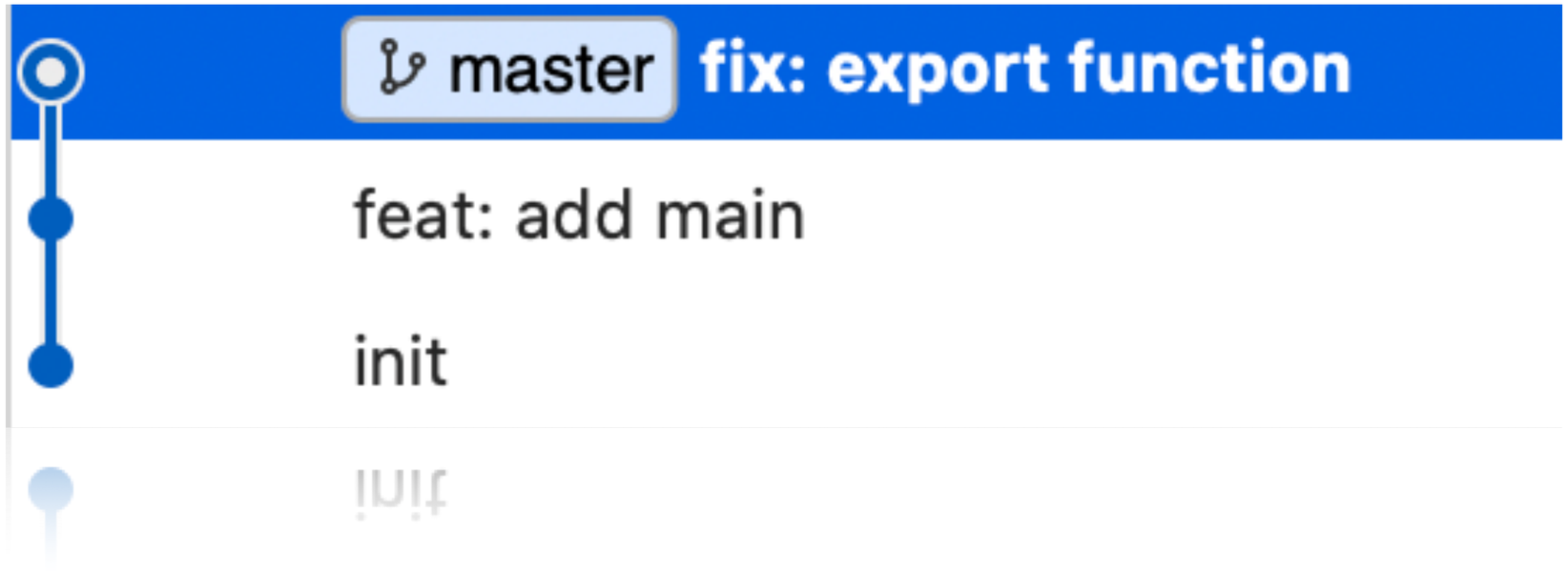
Типичная ситуация

Понимаем что допустили ошибку

```
git commit -m “fix: export function”
```

```
- function fun() {
+ export function fun() {
    console.log('fun');
  }
```

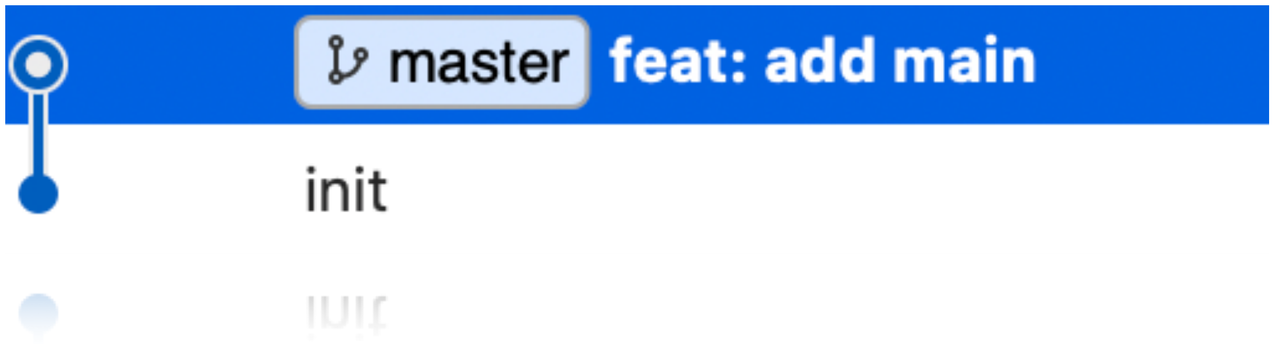
Но ведь мы можем лучше?



#1 Reset

Откатываем предыдущее изменение

```
git reset –soft HEAD~1
git commit -m “feat: add main”
```



#2 Amend

Идем дальше

`git commit –amend`

```
feat: add main

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Thu Oct 29 01:00:22 2020 +0600
#
# On branch master
# Changes to be committed:
#       new file:   fun.js
#       new file:   index.js
#
```


Это все конечно хорошо...
И так можно

Но об этом чуть позже

Еще одна причина, почему хорошо иметь атомарные коммиты

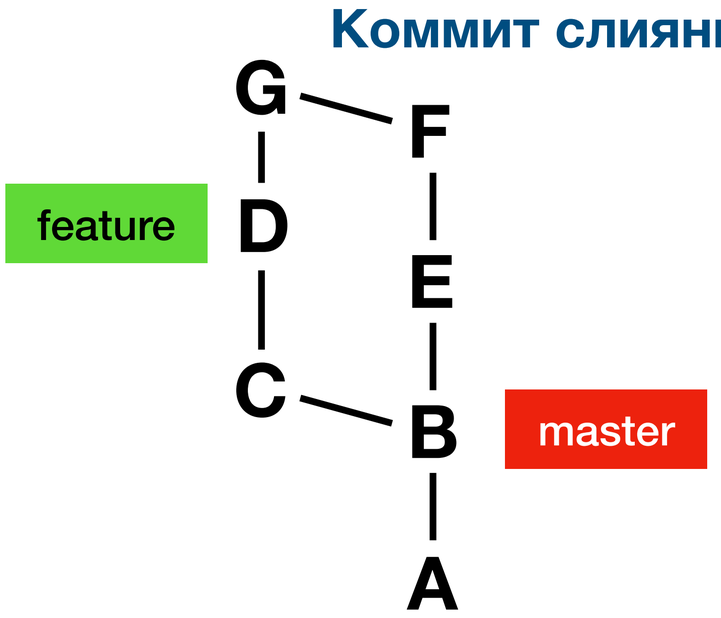
#3 Cherry pick 🍒

```
git commit -m “add component”
```

```
git cherry-pick 22b8bb039
```

звук барабанной дроби

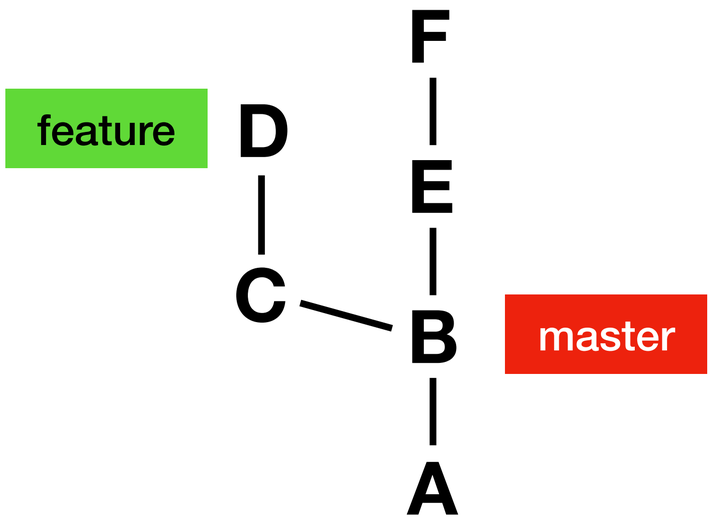
#4 Rebase



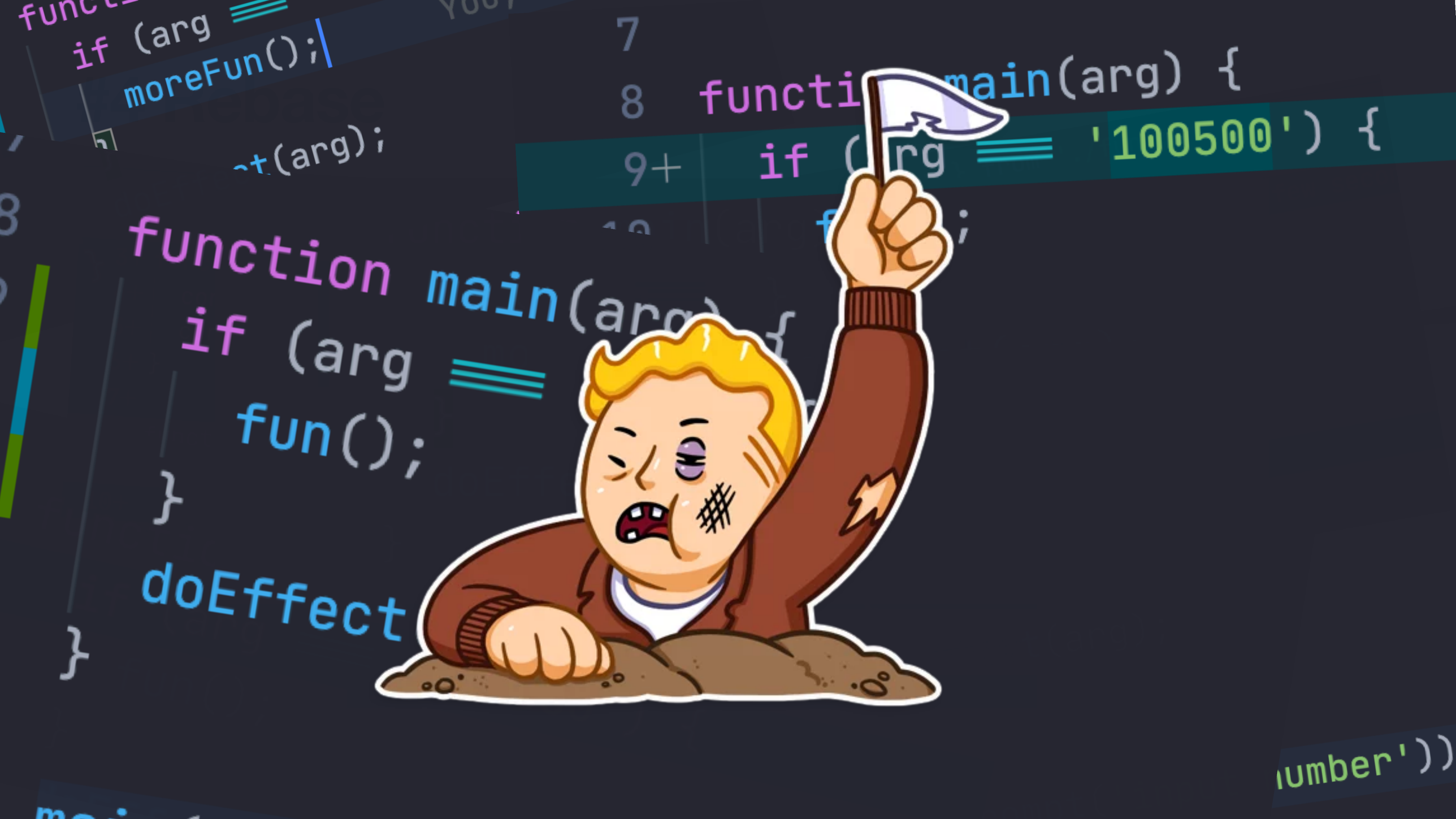
Press F to Pay Respects

```
git merge master
```

#4 Rebase




```
git rebase master
```

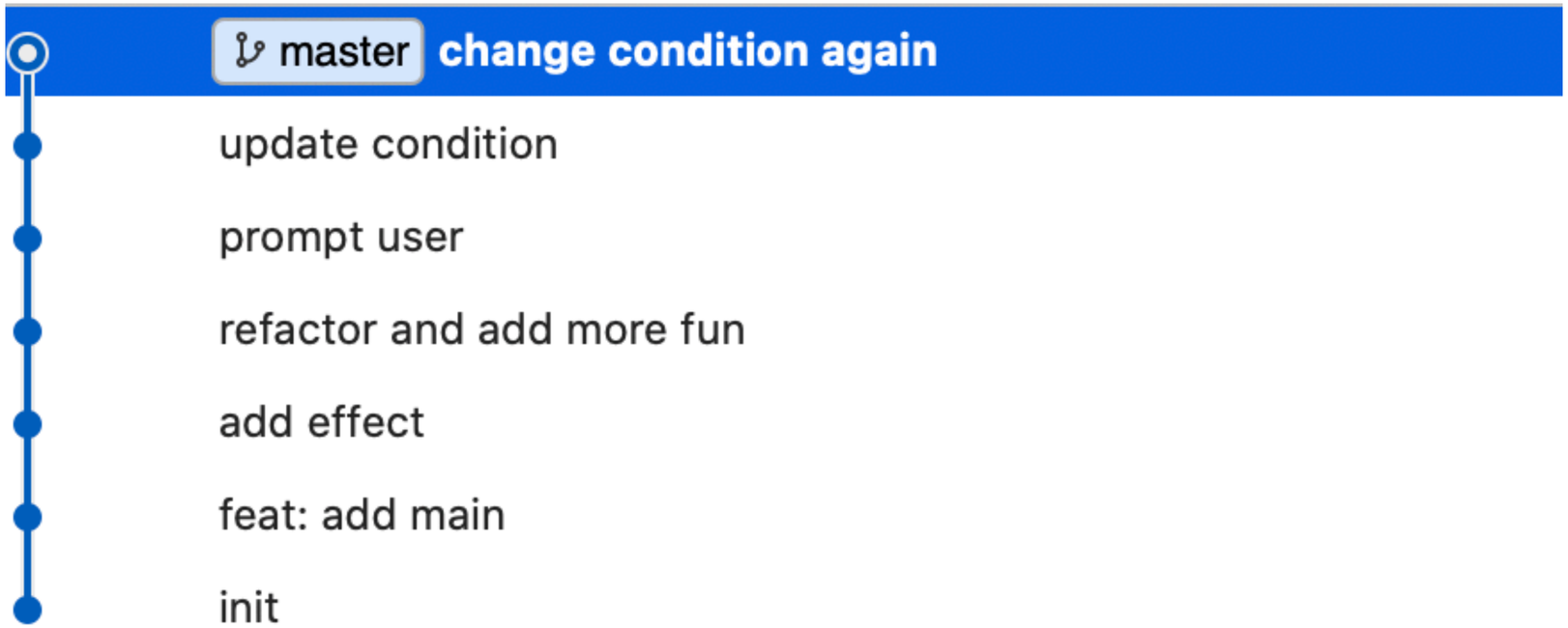

if (arg
moreFun();
function main(arg) {
if (arg
fun();
}
doEffect
}
'100500') {

#4 Rebase

Interactive Rebase

- Изменение комментария
- Изменения порядка коммитов в истории
- Изменение коммитов
- Разбивка коммитов
- Слияние коммитов воедино
- Удаление коммитов
- И Так Далее

#4 Rebase


master
change condition again
update condition
prompt user
refactor and add more fun
add effect
feat: add main
init

#4 Rebase

`git log --oneline`

```
5ebc427 (HEAD → master) change condition again
17e58c1 update condition
8c0a55d prompt user
0ff9ac7 refactor and add more fun
dcdbbaf add effect
5017f30 feat: add main
3a68c4e init
(END)
```

hash предыдущего коммита

#4 Rebase

`git rebase -i 5017f30`

```
pick dcdbbaf add effect
pick 0ff9ac7 refactor and add more fun
pick 8c0a55d prompt user
pick 17e58c1 update condition
pick 5ebc427 change condition again

# Rebase 5017f30..5ebc427 onto 17e58c1 (5 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
```

#4 Rebase

До

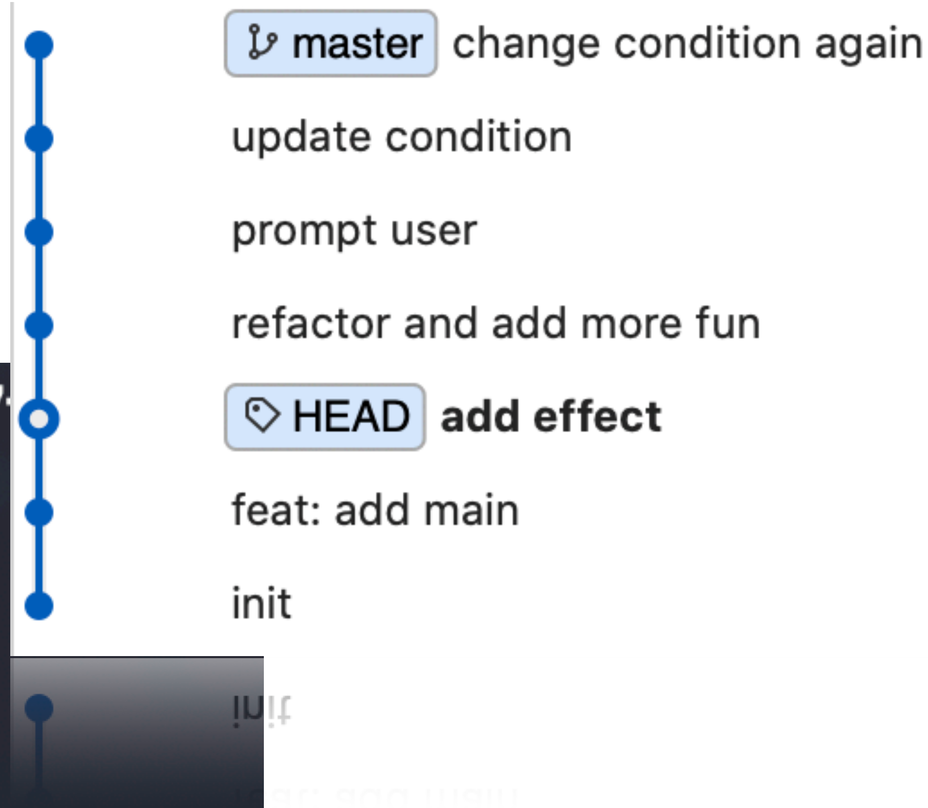
```
1 pick dcdbbaf add effect
2 pick 0ff9ac7 refactor a
3 pick 8c0a55d prompt use
4 pick 17e58c1 update con
5 pick 5ebc427 change con
```

После

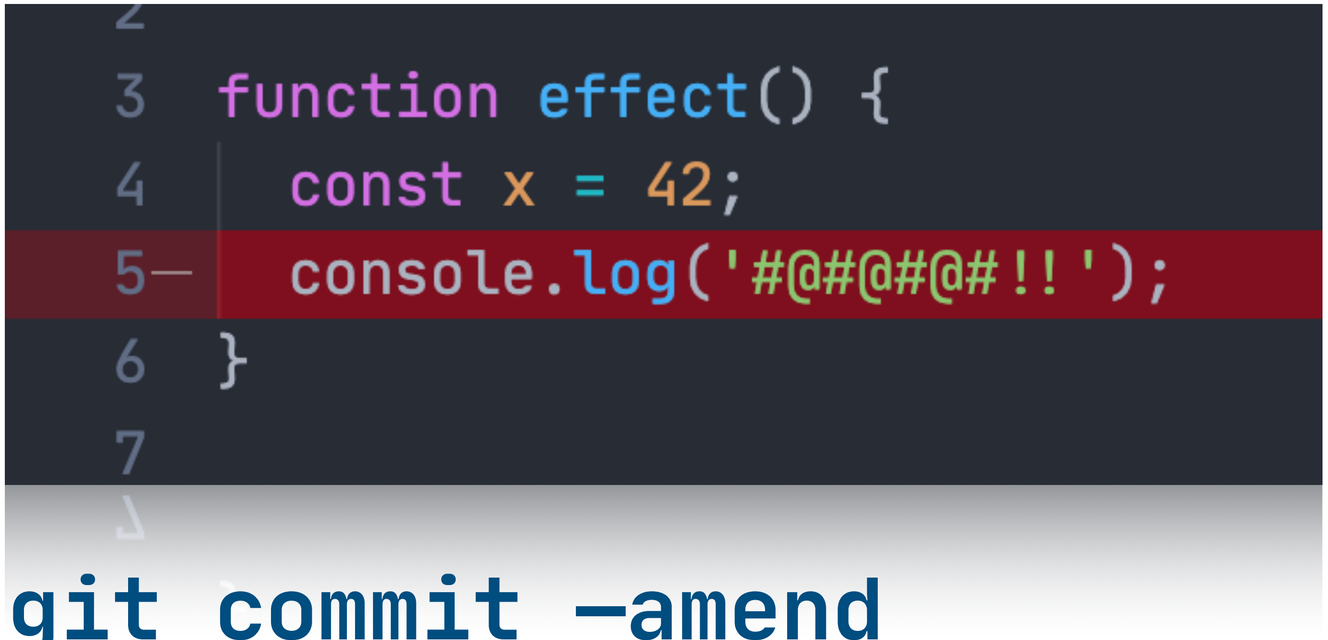
```
1 edit dcdbbaf add effect
2 edit 0ff9ac7 refactor and add more fun
3 reword 8c0a55d prompt user
4 fixup 17e58c1 update condition
5 fixup 5ebc427 change condition again
```

#4 Rebase

```
[→  whyyyy git:(master) git rebase -i 5017
Stopped at dcdbbaf...  add effect
You can amend the commit now, with

  git commit --amend

Once you are satisfied with your changes, run

  git rebase --continue
```



#4 Rebase

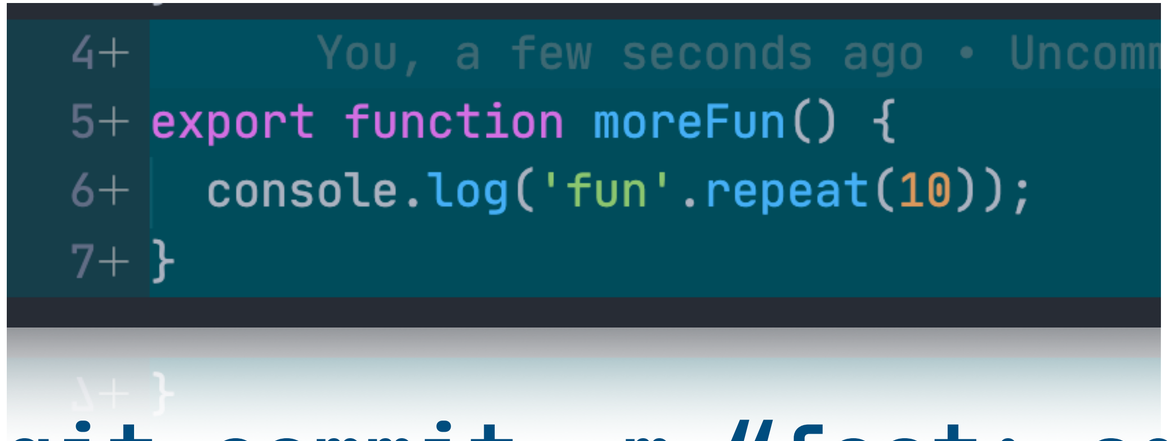



```
git commit –amend
git rebase –continue
```

#4 Rebase


```
git commit –amend –m “feat: add doEffect”
```

#4 Rebase



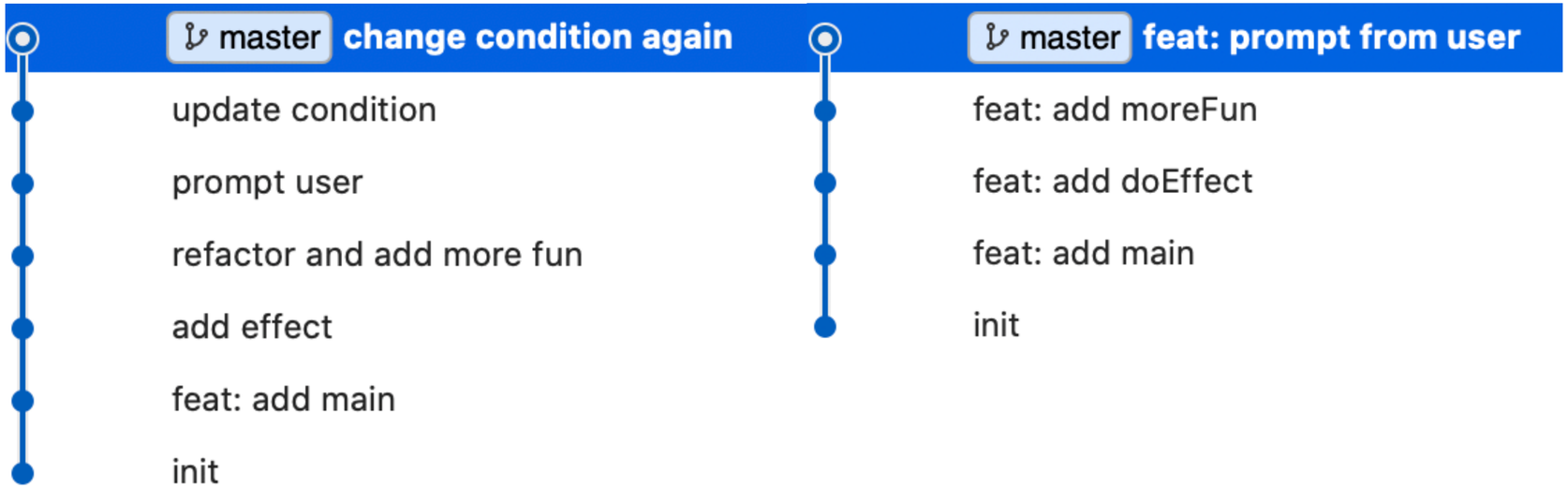

```
git commit -m “feat: add moreFun”
git rebase –continue
```

#4 Rebase

```
1 feat: prompt from user
2
3 # Please enter the commit message for yo
4 # with '#' will be ignored, and an empty
5 #
6 # Date:      Thu Oct 29 23:26:05 2020 +0
```

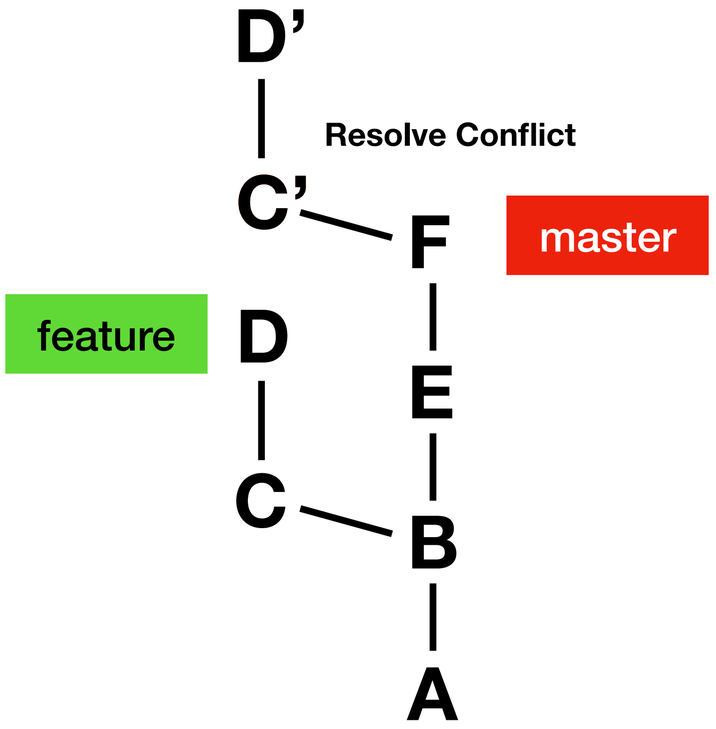
```
[detached HEAD 29c6e5a] feat: prompt from user
 Date: Thu Oct 29 23:26:05 2020 +0600
 1 file changed, 4 insertions(+), 2 deletions(-)
Successfully rebased and updated refs/heads/master.
```

#4 Rebase


master change condition again
update condition
prompt user
refactor and add more fun
add effect
feat: add main
init
master feat: prompt from user
feat: add moreFun
feat: add doEffect
feat: add main
init

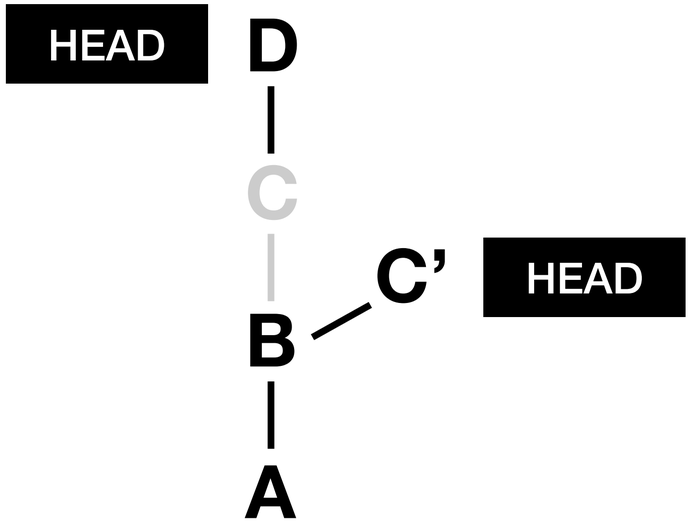

WHAT IF...?

Что если произойдет конфликт во время перебазирования?



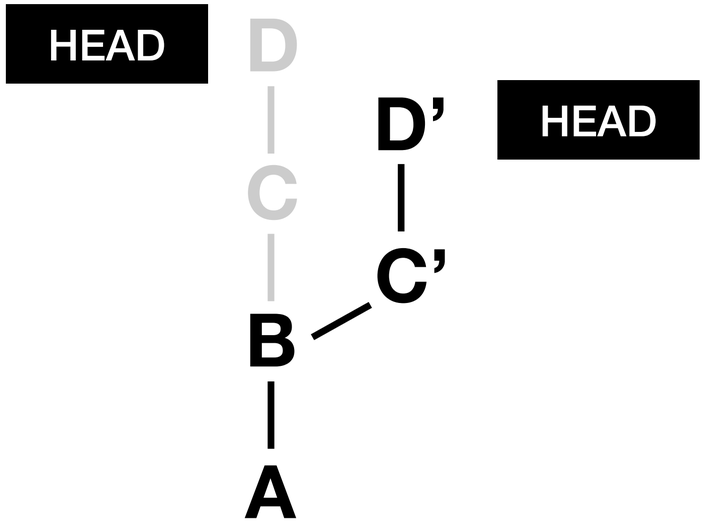

```
git rebase master
git rebase –continue
```

Что если мы допустили ошибку и хотим сбросить все на начало?



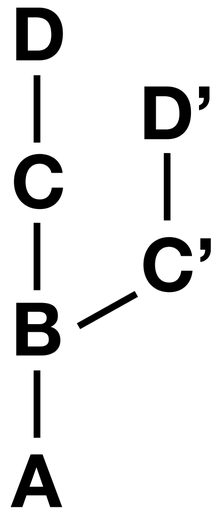

```
git rebase –abort
```

Что если мы передумали после завершения перебазирования?




```
git reset –hard D
```

Что если во время перебазирования появятся ошибки в коде?

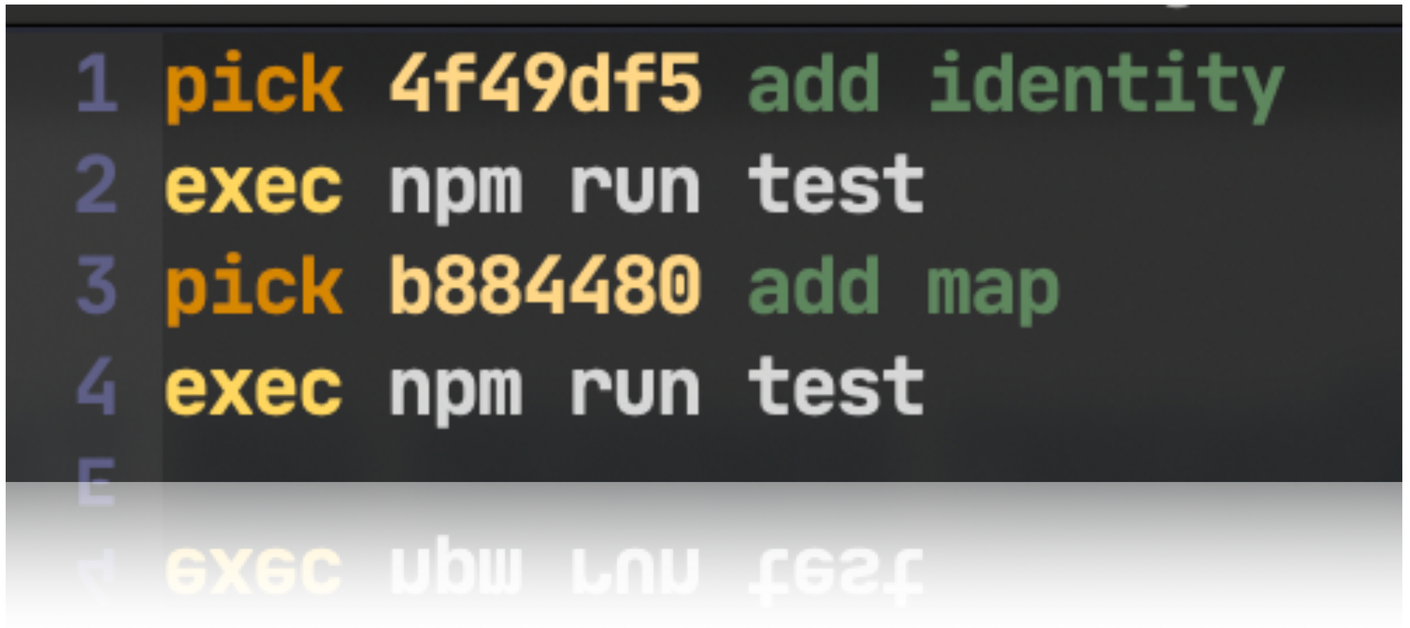



```
git rebase -i B
```

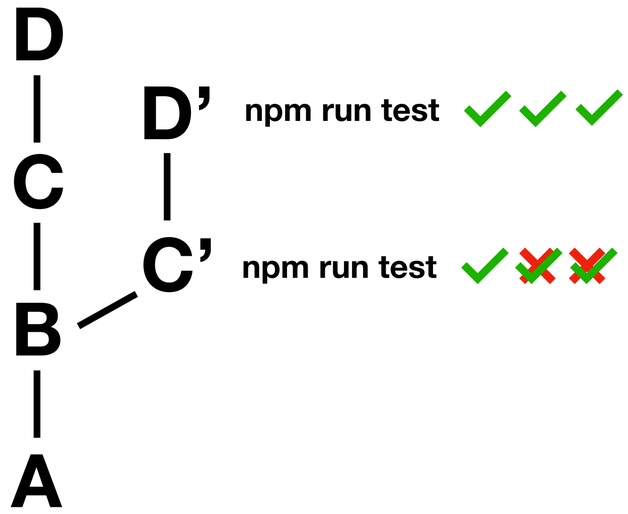

UNIT
TESTS

Что если во время перебазирования появятся ошибки в коде?

```
git rebase -i hash -x “npm run test”
```



Что если во время перебазирования появятся ошибки в коде?




```
git rebase -i B -x “npm run test”
git rebase –continue
```

#4 Rebase

Не изменяйте историю коммитов которые уже отправлены в общий репозиторий.

Так как при этом изменяется SHA-1 коммита.

Это приведет к конфликтам и к проблемам у ваших коллег если их изменения основываются на ваших


ПОСПЕШИЛ
ЛЮДЕЙ НАСМЕШИЛ


НОРМАЛЬНО ДЕЛАЙ
it-people.ru
НОРМАЛЬНО БУДЕТ

Итого

#1 `git reset —soft HEAD~1` - Откатываем локальные изменения

#2 `git commit —amend` - Дополняем последний коммит

#3 `git cherry-pick […commit-hash]` - Копируем коммиты

#4 `git rebase -i commit-hash` - Крутим вертим коммиты как пожелаем

Ссылки

- Скринкаст по Git - https://learn.javascript.ru/screencast/git
- Patching - https://git-scm.com/book/en/v2/Appendix-C%3A-Git-Commands-Patching
- Rewriting History - https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History
- Post-Production editing using Git - http://sethrobertson.github.io/GitPostProduction/gpp.html