

Git

Юрий Литвинов
yurii.litvinov@gmail.com

23.01.2019г

Комментарии по домашке

Концептуальные

- ▶ hashCode() может (и будет) возвращать отрицательные числа, программа должна быть к этому готова
- ▶ Надо решить, что делать с ключами и значениями null
 - ▶ Явно запретить и кидать IllegalArgumentException
 - ▶ Разрешить им быть null-ами и придумать другой способ сигнализировать об отсутствии значения
 - ▶ Вообще, null всегда допустим для всех ссылочных типов, поэтому о нём надо помнить
 - ▶ И иметь юнит-тесты на это!
- ▶ clear должен ресайзить таблицу обратно до исходного размера, иначе утечка памяти

Комментарии по домашке (2)

Технические

- ▶ Инициализация полей — за или против
- ▶ `@BeforeEach` для инициализации хеш-таблицы в тестах
- ▶ Если тип переменной очевиден из правой части — `var`
- ▶ Сдавать решение одним коммитом неумудро

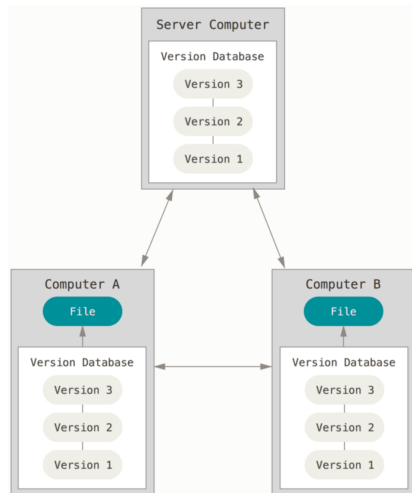
Комментарии по домашке (3)

По оформлению

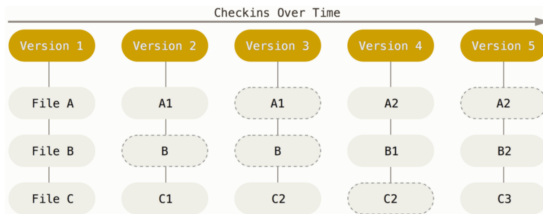
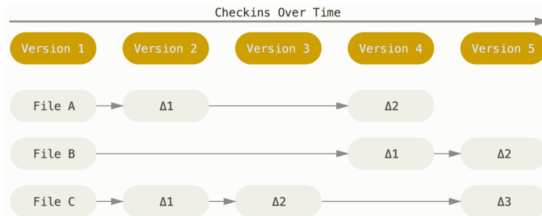
- ▶ Комментарии не должны начинаться с `@return`
- ▶ Не надо сокращать идентификаторы
- ▶ Стоит избегать побочных эффектов в выражениях
- ▶ Каждый класс (не вложенный) должен быть в своём файле

Git

- ▶ Не GitHub!
- ▶ Не файлообменник
- ▶ Распределённая система контроля версий
- ▶ Удобная поддержка веток



Управление версиями

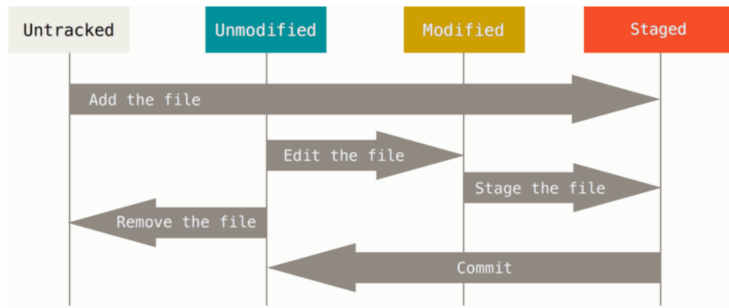


© <https://git-scm.com/book/ru>

Дельта

| | | |
|----|----|---|
| | 21 | <code>++include <QtScript/QScriptEngine></code> |
| | 22 | <code>++include <QtScript/QScriptValue></code> |
| 18 | 23 | |
| 19 | | <code>-#include <trikControl/brickInterface.h></code> |
| 20 | 24 | <code>#include <trikControl/brickFactory.h></code> |
| 21 | 25 | |
| | 26 | <code>++include <trikKernel/fileUtils.h></code> |
| | 27 | <code>+</code> |
| 22 | 28 | <code>using namespace tests;</code> |
| 23 | 29 | |

Жизненный цикл файла



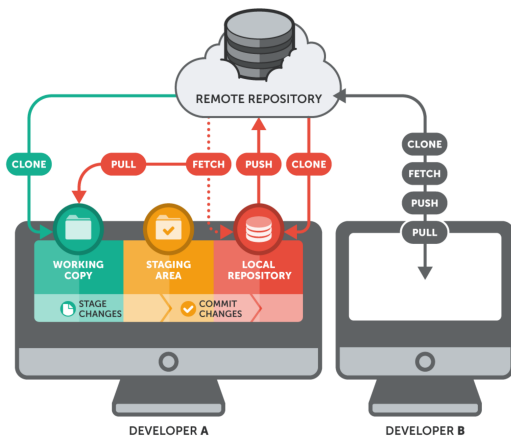
© <https://git-scm.com/book/ru>

Основные команды

- ▶ `git add` — добавить новый файл под управление git или добавить изменение к коммиту
- ▶ `git status` — показать список изменённых/добавленных/удалённых файлов
- ▶ `git diff` — показать изменения по каждому файлу
- ▶ `git commit` — зафиксировать изменения, создав новый коммит
- ▶ `git rm` — удалить файл и удалить его из репозиторий
- ▶ `git log` — просмотреть список коммитов
- ▶ `git checkout` — откатить изменения в файле или перейти на другую ветку

Удалённые репозитории

- ▶ git clone
- ▶ git remote
- ▶ git push
- ▶ git fetch
- ▶ git pull

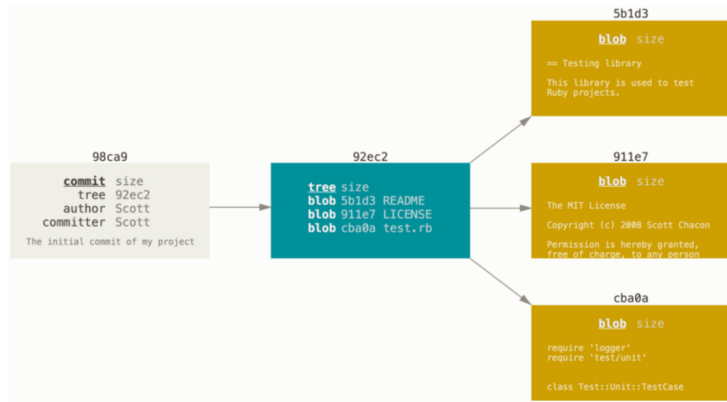


© <https://www.git-tower.com/learn/git/ebook/en>

Как всё устроено

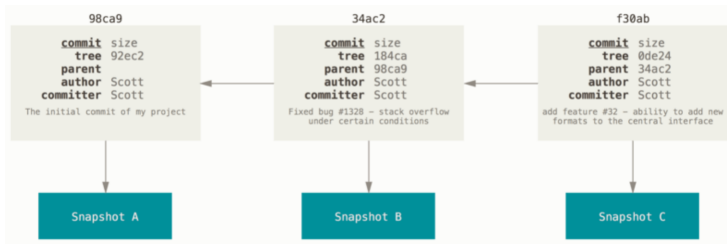
```
$ git add README test.rb LICENSE
```

```
$ git commit -m 'initial commit of my project'
```



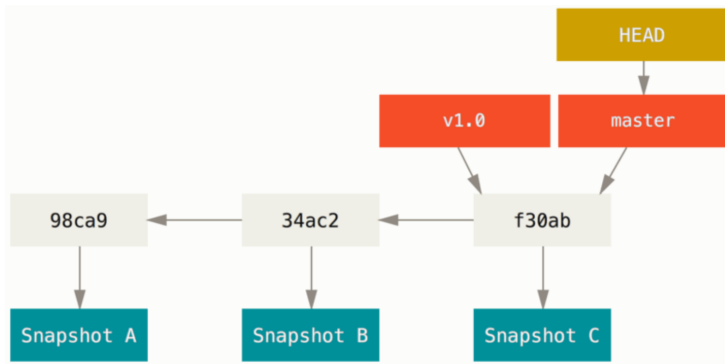
© <https://git-scm.com/book/ru>

Коммит и его родители



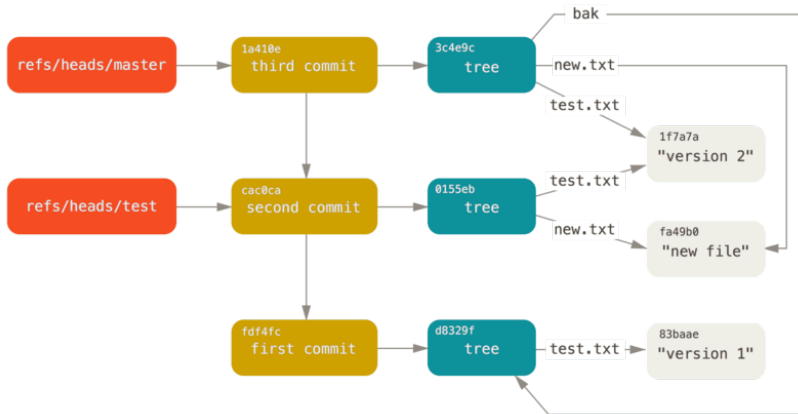
© <https://git-scm.com/book/ru>

Ветки



© <https://git-scm.com/book/ru>

Ссылки, как это выглядит



HEAD

Теперь не надо помнить хеши, но как переключаться между ветками?

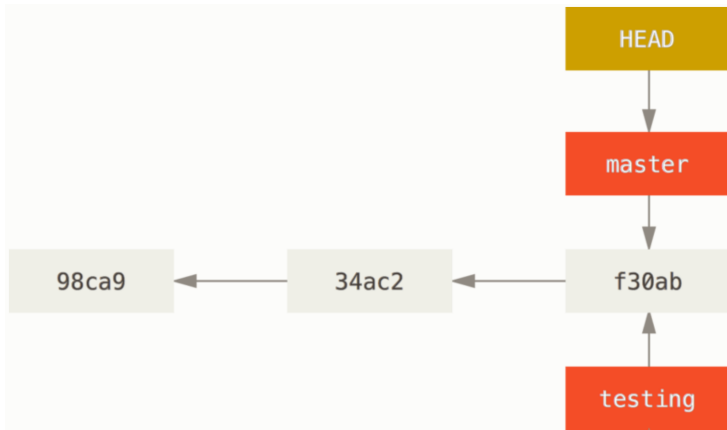
Текущая ветка хранится в HEAD. HEAD — символическая ссылка, то есть ссылка на другую ссылку.

```
$ cat .git/HEAD
```

```
ref: refs/heads/master
```

Создание ветки

\$ git branch testing



© <https://git-scm.com/book/ru>

Переключение ветки

\$ git checkout testing



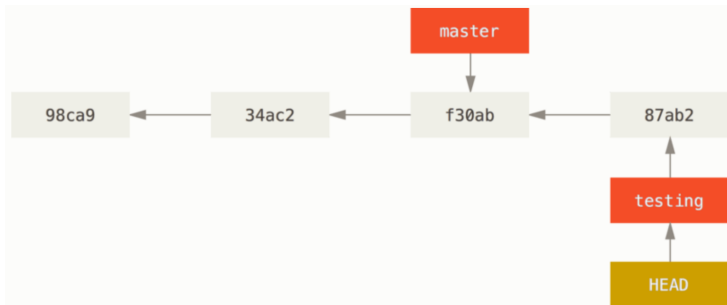
© <https://git-scm.com/book/ru>

Новый коммит

<Что-то поделали с файлами в рабочей копии>

\$ git add <изменения, которые хотим коммитить>

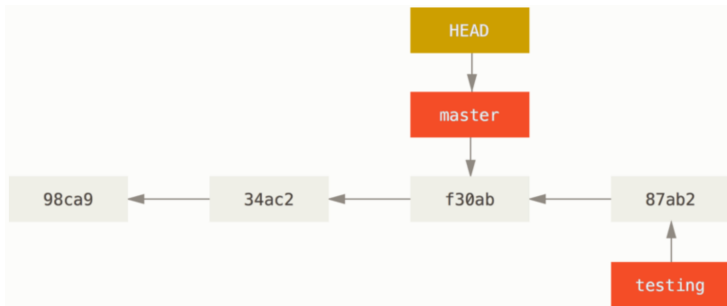
\$ git commit -m 'made a change'



© <https://git-scm.com/book/ru>

Переключимся на master

\$ git checkout master



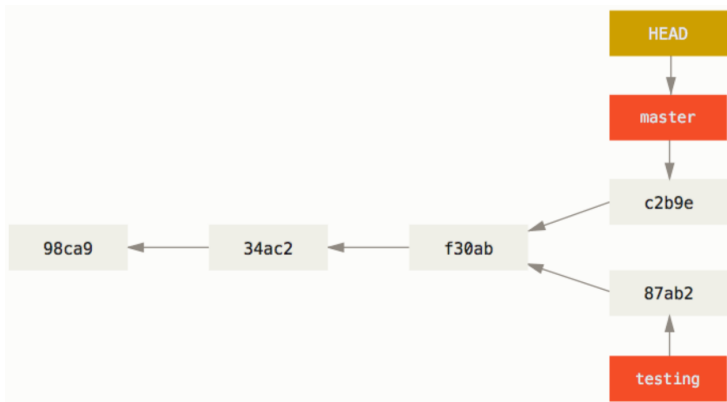
© <https://git-scm.com/book/ru>

Сделаем новый КОММИТ там

<Что-то поделали с файлами в рабочей копии>

\$ git add <изменения, которые хотим коммитить>

\$ git commit -m 'made other changes'



© <https://git-scm.com/book/ru>

Слияние веток

\$ git checkout master

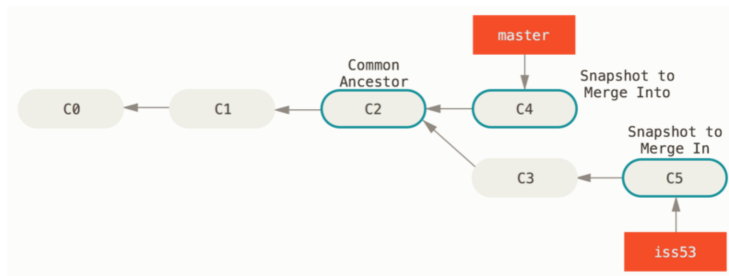
Switched to branch 'master'

\$ git merge testing

Merge made by the 'recursive' strategy.

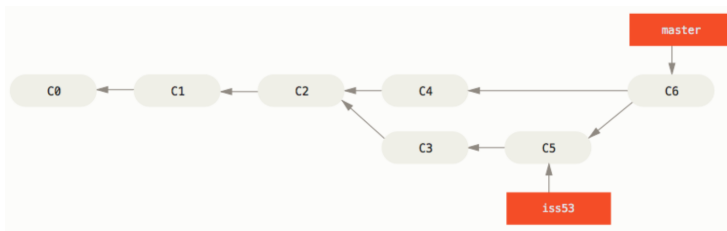
index.html | 1 +

1 file changed, 1 insertion(+)



© <https://git-scm.com/book/ru>

Результат



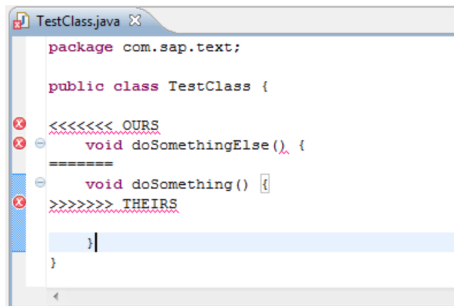
© <https://git-scm.com/book/ru>

Конфликты

| Theirs - REMOTE | Mine - LOCAL |
|---|--|
| 128 → /// - Enables or disables all editor actions.↵ | 138 → /// - Enables or disables all editor actions.↵ |
| 129 → void·setEnabled(bool·enabled);↵ | 139 → void·setEnabled(bool·enabled);↵ |
| 130 ↵ | 140 ↵ |
| 131 → void·checkConstraints(IdList·const·&elementsList);↵ | 141 → /// Handles deletion of the element from scene.↵ |
| 132 ↵ | 142 → void·onElementDeleted(Element·*element);↵ |
| | 143 ↵ |
| | 144 → /// Enable or Disable mousegestures↵ |
| | 145 → void·enableMouseGestures(bool·enabled);↵ |
| | 146 ↵ |
| 133 public·slots:↵ | 147 public·slots:↵ |
| 134 → qReal::Id·createElement(const·QString·&type);↵ | 148 → qReal::Id·createElement(const·QString·&type);↵ |
| 135 ↵ | 149 ↵ |
| 136 → void·cut();↵ | 150 → void·cut();↵ |
| 137 → void·copy();↵ | 151 → void·copy();↵ |
| 138 → void·paste(bool·logicalCopy);↵ | 152 → void·paste(bool·logicalCopy);↵ |
| 139 ↵ | 153 ↵ |

| Merged - editorViewScene.h |
|--|
| 138 → /// - Enables or disables all editor actions.↵ |
| 139 → void·setEnabled(bool·enabled);↵ |
| 140 ↵ |
| 141 ~~~~~ |
| 142 ~~~~~ |
| 143 ~~~~~ |
| 144 ~~~~~ |
| 145 ~~~~~ |
| 146 ~~~~~ |
| 147 public·slots:↵ |
| 148 → qReal::Id·createElement(const·QString·&type);↵ |
| 149 ↵ |
| 150 → void·cut();↵ |
| 151 → void·copy();↵ |
| 152 → void·paste(bool·logicalCopy);↵ |
| 153 ↵ |

Конфликты в коде



```
TestClass.java X
package com.sap.text;

public class TestClass {

<<<<<<< OURS
    void doSomethingElse() {
=====
    void doSomething() {
>>>>>>> THEIRS

    }
}
```

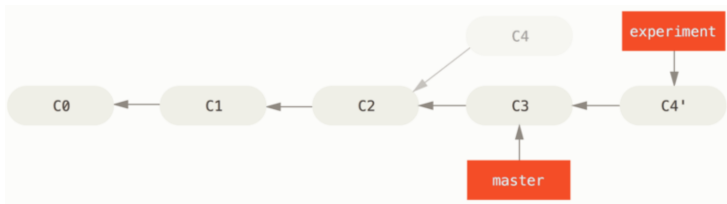

Rebase

\$ git checkout experiment

\$ git rebase master

First, rewinding head to replay your work on top of it...

Applying: added staged command



© <https://git-scm.com/book/ru>

Тэги

Последний из объектов в Git — tag. Это просто указатель на коммит.

- ▶ Легковесный тэг:

```
git update-ref refs/tags/v1.0 cac0cab538b970a37ea1e769cbbde608743bc96d
```

Или просто `git tag`

- ▶ Аннотированный тэг:

```
$ git tag -a v1.1 1a410efbd13591db07496601ebc7a059dd55cfe9 -m 'test tag'
```

```
$ git cat-file -p 9585191f37f7b0fb9444f35a9bf50de191beadc2
```

```
object 1a410efbd13591db07496601ebc7a059dd55cfe9
```

```
type commit
```

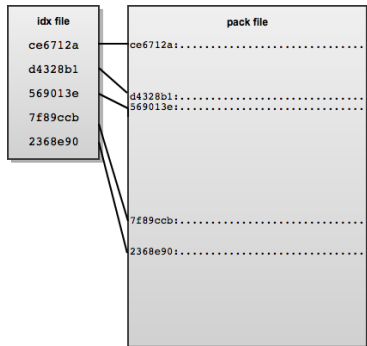
```
tag v1.1
```

```
tagger Scott Chacon <schacon@gmail.com> Sat May 23 16:48:58 2009 -0700
```

```
test tag
```

Packfiles

Пока что получалось, что все версии всех файлов в Git хранятся целиком, как они есть. Все они сжимаются zlib, но в целом, если создать репозиторий, добавлять туда файлы, коммитить и т.д., все версии всех файлов будут в нём целиком. На помощь приходят .pack-файлы:



Хорошие практики

- ▶ Коммитим только исходные тексты, конфиги, картинки и т.п.
- ▶ Если что-то может быть автоматически сгенерировано по тому, что уже есть в репозитории, это НЕ коммитим
- ▶ Всегда пишем адекватные комментарии к коммитам
 - ▶ Одно-два осмысленных предложения
 - ▶ За комментарии типа “fix” придётся освоить искусство отката веток (или хотя бы amend)
- ▶ Коммитим как можно чаще
 - ▶ Сделали что-то осмысленное — коммит

Хорошие практики (2)

- ▶ Коммит не должен содержать в себе файлы, не относящиеся к изменениям
 - ▶ .gitignore
 - ▶ Обязательно надо проверять, что коммитим
- ▶ Коммит не должен добавлять/убирать пустые строки, менять пробелы на табы и т.д., если это не суть коммита
- ▶ Стиль исходного кода и отступов должен совпадать с текстом вокруг
- ▶ Делаете пуллреквест — посмотрите diff

In case of fire



1. git commit

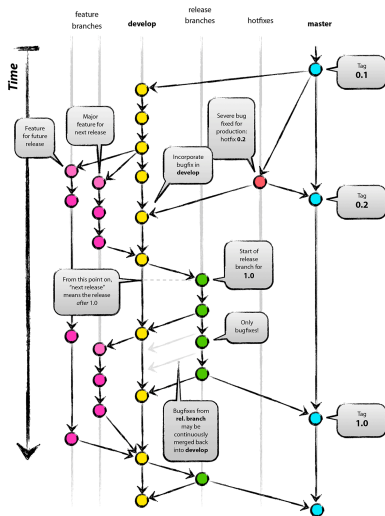


2. git push



3. leave building

Git Flow



© <https://nvie.com/posts/a-successful-git-branching-model/>

Ещё полезные команды

- ▶ `git add -p` — интерактивное добавление изменений к коммиту, позволяет коммитить только часть файла
- ▶ `git commit --amend` — исправить последний коммит
 - ▶ `git commit --amend -m "an updated commit message"`
 - ▶ Применять **только до** `git push`
- ▶ `git reset --hard` — откатить все изменения в рабочей копии до последнего коммита
 - ▶ Обязательно проверить `git status`, что не откатите лишнего
- ▶ `git reset --hard <хеш коммита>` — откатить все изменения в текущей ветке до указанного коммита, забыть все коммиты, что были после
 - ▶ И случайно грохнуть всю домашку перед зачётом

.gitignore

- ▶ Локальный для репозитория и глобальный
- ▶ Glob-шаблоны
- ▶ # — комментарий
- ▶ Можно заканчивать шаблон символом / для указания каталога
- ▶ можно инвертировать шаблон, используя ! в качестве первого символа
- ▶ Glob
 - ▶ * — 0 или более символов
 - ▶ ? — один символ
 - ▶ [abc] — любой символ из указанных в скобках
 - ▶ [0-9] — любой символ из интервала

Пример

не обрабатывать файлы, имя которых заканчивается на .a

**.a*

НО отслеживать файл lib.a, несмотря на то, что мы игнорируем все .a

!lib.a

игнорировать только файл TODO, находящийся в корневом каталоге

/TODO

игнорировать все файлы в каталоге build/

build/

игнорировать doc/notes.txt, но не doc/server/arch.txt

doc/.txt*

игнорировать все .txt файлы в каталоге doc/

*doc/**/*.txt*

► <https://github.com/github/gitignore>