

# GIT

git checkout (часть 1)

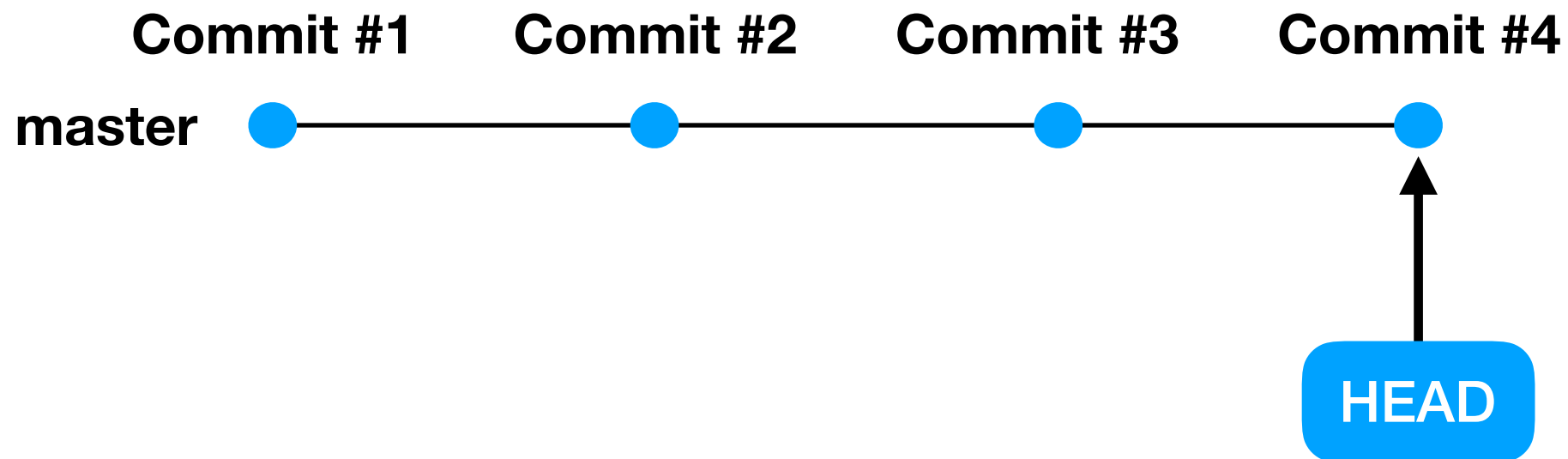
# git checkout

## (часть функционала)

- Команда очень мощная и многофункциональная (как и git reset)
- Используется для перемещения между **коммитами, версиями отдельных файлов и ветками**
- На этом уроке разберем только **часть функционала** этой команды. **Другую часть функционала** разберем после того, как разберемся с ветвлением (branching)

# git checkout

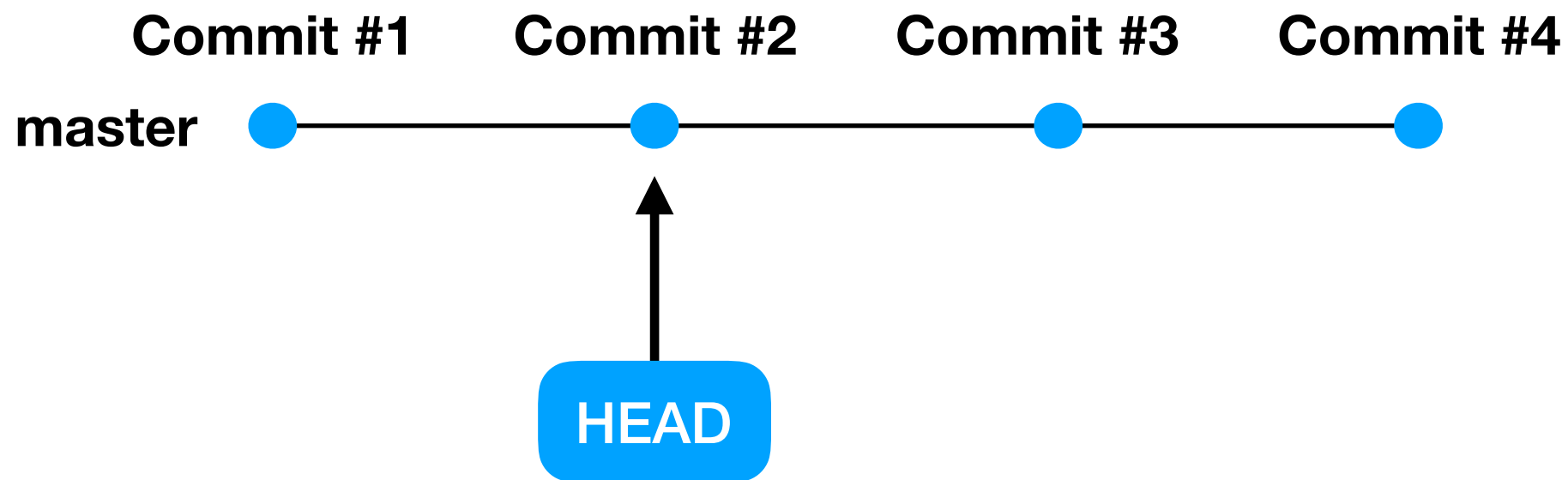
(перемещение между коммитами)



**Хотим посмотреть, как выглядел проект в каком-то снимке в прошлом**

# git checkout

(перемещение между коммитами)



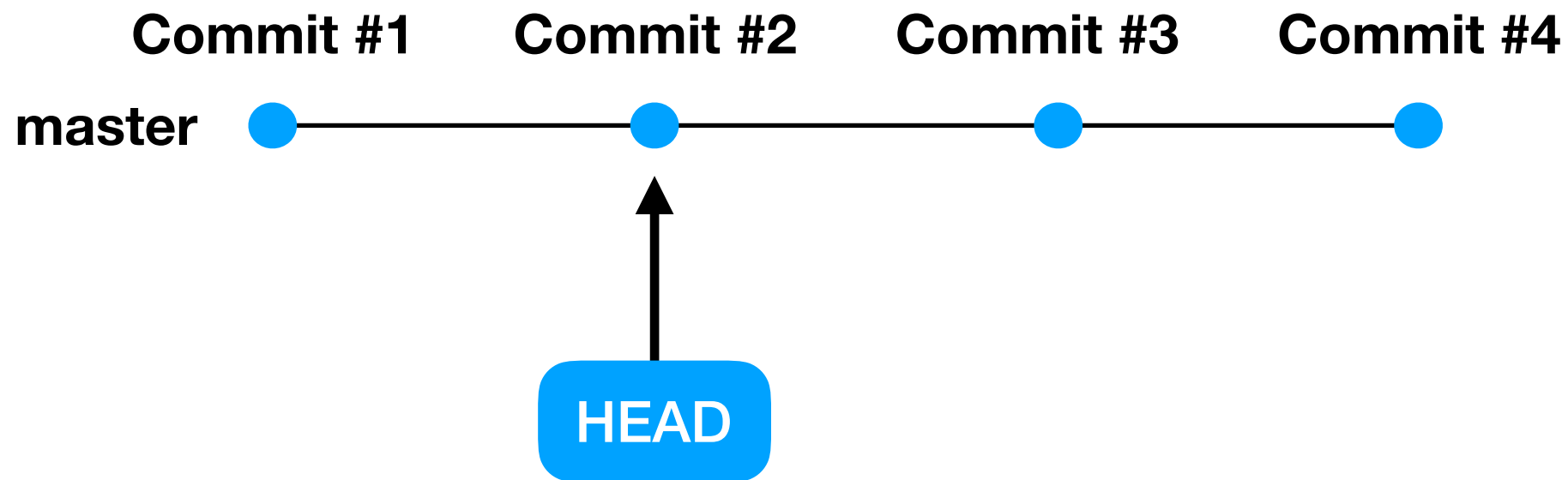
**git checkout <хэш commit #2>**

**git checkout HEAD^^**

**git checkout HEAD~2**

# git checkout

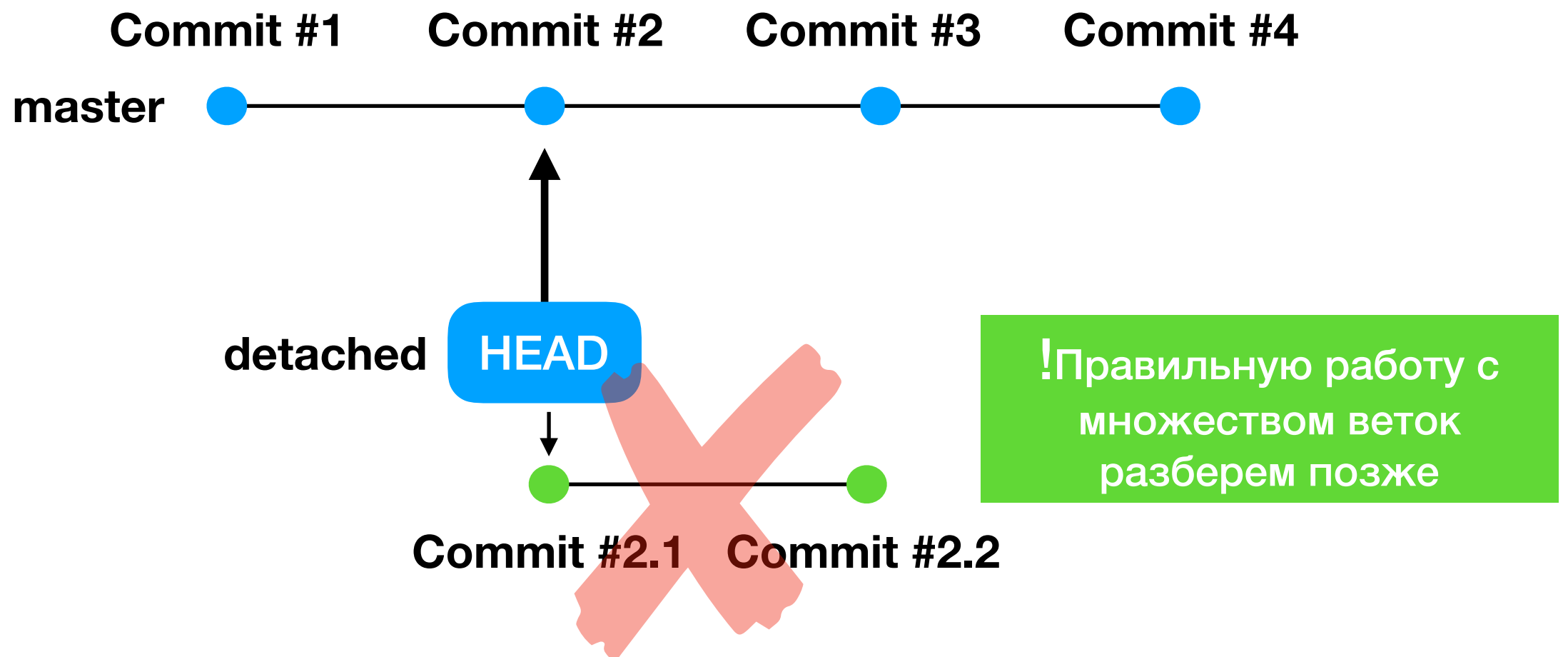
## (перемещение между коммитами)



- Состояние проекта полностью вернулось к указанному снимку. При этом никакие коммиты не удалились. Мы в любой момент можем перенестись обратно в актуальную версию.
- Указатель HEAD находится в состоянии DETACHED (рус. отделенный). Он отделен от актуальной версии проекта. Любые изменения или коммиты сделанные в этом состоянии удаляются сборщиком мусора при переходе к другому коммиту.

# git checkout

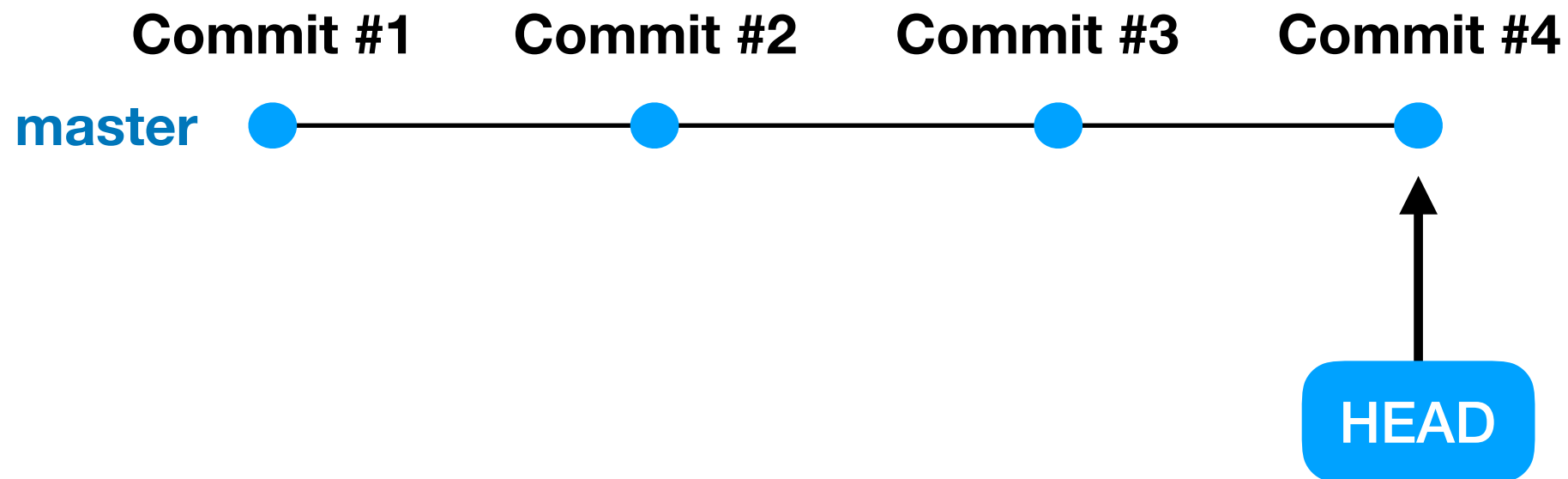
## (перемещение между коммитами)



- **Указатель HEAD находится в состоянии DETACHED (рус. отделенный). Он отделен от актуальной версии проекта. Любые изменения или коммиты сделанные в этом состоянии удаляются сборщиком мусора при переходе к другому коммиту.**

# git checkout

(перемещение между коммитами)



- **git checkout master** - переход обратно к актуальному коммиту
- **master** - название текущей ветки. О ветвлении позже.

# **Практика**

## **(перемещение между коммитами)**



# **git checkout**

**(перемещение между версиями файлов)**

**Хотим вернуть конкретный файл(-ы) к какой-то  
версии в прошлом**

# git checkout

## (перемещение между версиями файлов)

- `git checkout <указатель коммита> -- путь_до_файла_1 путь_до_файла_2`

Пример: `git checkout a0e33627548578d5b94c3b8f4f885303a2cd4eес -- file1 file2`

Возвращает два файла **file1** и **file2** к версии, которая была у  
НИХ в **указанном** КОММИТЕ

# git checkout

## (перемещение между версиями файлов)

- `git checkout -- путь_до_файла_1 путь_до_файла_2`

Пример: `git checkout -- file1`

Возвращает `file1` к версии, которая была у него в последнем коммите (`HEAD`). Работает только для **неотслеживаемых** (`untracked` или `modified`) изменений.

# git checkout

## (перемещение между версиями файлов)

- `git checkout -- .`

Возвращает **все файлы** в репозитории к версии, которая была у них в последнем коммите (**HEAD**). Работает только для **неотслеживаемых** (**untracked** или **modified**) изменений.

# git checkout

## (перемещение между версиями файлов)

Что если я хочу удалить **отслеживаемые** изменения?  
(уже сделал git add)

! Вспоминаем урок про git reset и комбинируем команды

1. **git reset** - делает то же самое, что **git reset --mixed HEAD**.  
Переводит все **отслеживаемые** изменения в  
**неотслеживаемые**.
2. **git checkout -- .** - удаляем все **неотслеживаемые**  
изменения

# Зачем использовать две черты (--)? Ведь можно и без них.

Можно делать:

```
git checkout a0e33627548578d5b94c3b8f4f885303a2cd4eec file1 file2
```

```
git checkout file1
```

```
git checkout .
```

Но не стоит. Почему?

Две черты указывают, что после них идет обычный текст (в нашем случае путь до файла), а не команда или параметр для команды. Где это бывает нужно?

# Зачем использовать две черты (--)? Ведь можно и без них.

## Пример:

У нас есть файл, который называется `master`. Этот файл мы изменили и теперь хотим вернуть его к версии из последнего коммита.

- `git checkout master` произведет переключение на ветку `master`.

VS

- `git checkout --master` произведет возвращение файла с названием `master` к версии из последнего коммита.

# **Практика**

## **(перемещение между версиями файлов)**