



HTML и CSS. Адаптивная вёрстка и автоматизация

Уровень 2, с 21 ноября 2022 по 30 января 2023

Меню курса

[Главная](#) / [1. Старт](#) /

📖 1.20. Операции отмены

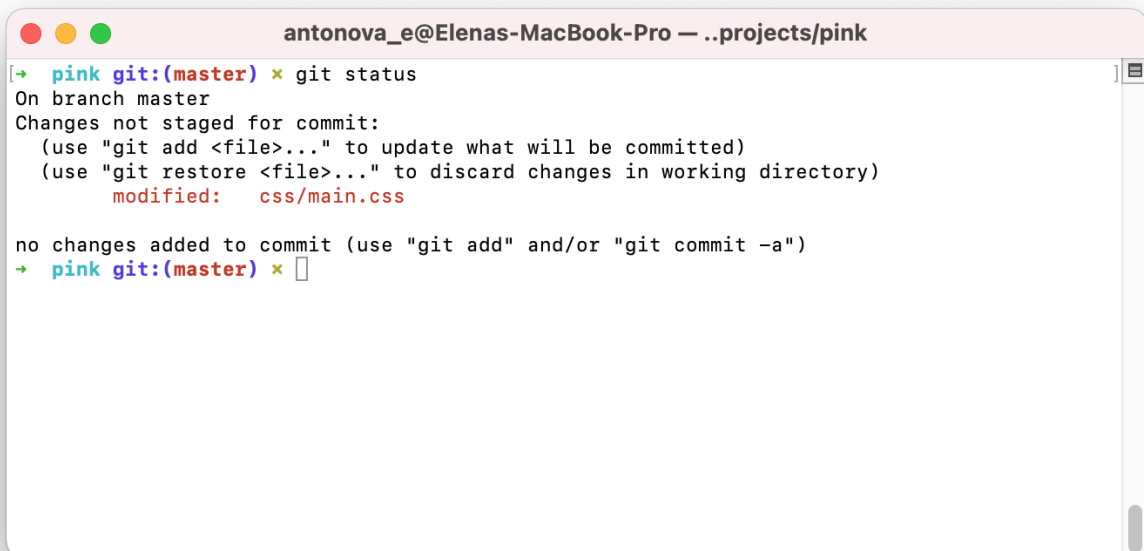
🕒 ~ 5 минут

В прошлых главах мы познакомились с основами контроля версий и разобрали базовый сценарий работы, когда вы просто фиксируете любое изменение. В этой главе мы рассмотрим более сложные сценарии.

Как откатить незакоммиченные изменения

Например, вы изменили файл, проверили какую-то идею и хотите вернуть файл в состояние, в котором он был в последнем коммите.

Узнаем текущее состояние командой `git status`. Видим, что файл `main.css` был изменён.

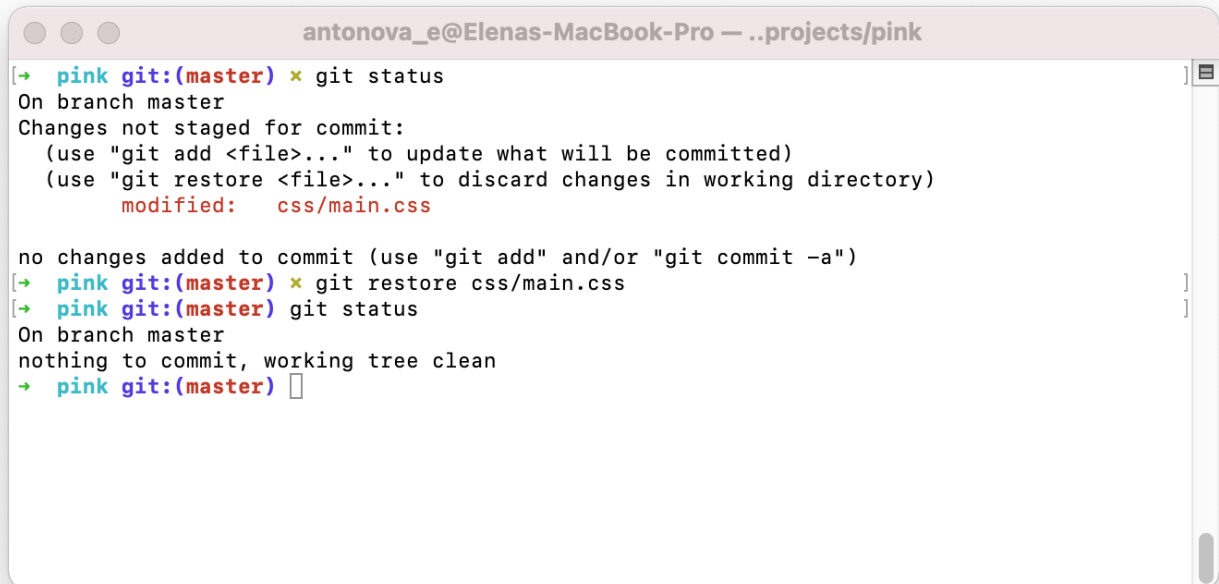


```
antonova_e@Elenas-MacBook-Pro — ..projects/pink
[→ pink git:(master) ✕ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   css/main.css

no changes added to commit (use "git add" and/or "git commit -a")
→ pink git:(master) ✕
```

Проверка состояния файлов

Git подсказывает, что отменить изменения (discard changes in working directory) можно командой `git restore <file>`. Давайте введём эту команду и снова проверим статус.



```
antonova_e@Elenas-MacBook-Pro — ..projects/pink
[→ pink git:(master) ✕ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   css/main.css

no changes added to commit (use "git add" and/or "git commit -a")
[→ pink git:(master) ✕ git restore css/main.css
[→ pink git:(master) git status
On branch master
nothing to commit, working tree clean
→ pink git:(master) ]
```

Изменения успешно отменены

Видим, что состояние чистое, файл не изменён.

Резюме

Командой `restore` можно вернуть файл в состояние последнего коммита, но вернуть состояние до команды `restore` уже не получится — всё, что вы не закоммитили, потеряется. Поэтому сбрасывайте только те изменения, которые вам не нужны.

Как откатить файл к состоянию в определённом коммите

Это можно сделать с помощью команды `git checkout`, указав ей коммит, из которого мы хотим взять состояние файла.

Вводим команду `git log` и ищем нужный коммит. Копируем хэш коммита, пишем команду `git checkout`, вставляем хэш коммита и пишем название файла, который нужно откатить. Проверим статус.

```
antonova_e@Elenas-MacBook-Pro — ..projects/pink
commit 364f5f95be6877c1fcdb686bc485f76f21aadd33 (HEAD -> master)
Author: Elena <academy@htmlacademy.ru>
Date: Tue Jun 8 00:11:50 2021 +0300

    Добавит стили для ссылок

commit 9b0c42c7e3130289fd26104c767d9b1da8ab18b1
Author: Elena <academy@htmlacademy.ru>
Date: Tue Jun 8 00:07:10 2021 +0300

    Начали вести историю
[→ pink git:(master) git checkout 9b0c42c7e3130289fd26104c767d9b1da8ab18b1 css/main.css ]
Updated 1 path from d8abe95
[→ pink git:(master) ✖ git status ]
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   css/main.css
[→ pink git:(master) ✖ ]
```

Файл с изменениями проиндексирован

Видим, что файл изменён и проиндексирован. Посмотрим, какие в нём изменения. Чтобы увидеть изменения в проиндексированном файле, вводим уже знакомую, но модифицированную команду `git diff`.

```
git diff --staged — git
diff --git a/css/main.css b/css/main.css
index 06cb603..9ec671b 100644
--- a/css/main.css
+++ b/css/main.css
@@ -1,7 +1,3 @@
 p {
   color: #ccc;
 }
-
-a {
-  color: red;
-}
(END)
```

Список изменений

Резюме

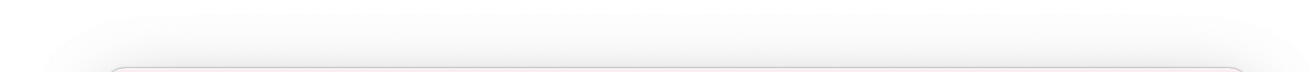
Командой `checkout` можно вернуть файл в состояние не только последнего коммита, но и любого коммита, если вы знаете его хэш.

Как убрать файл из индекса

Мы подготовили изменения к коммиту, добавили их в индекс и вдруг заметили, что какой-то файл индексировать не нужно.

Например, мы внесли изменения в несколько файлов и эти изменения никак между собой не связаны. И мы, возможно, потом захотим как-то независимо с этими изменениями работать, поэтому мы хотим сохранить эти изменения отдельно друг от друга.

Можно отменить индексацию изменений при помощи команды `git restore --staged` и имя файла, который мы не хотим включать в коммит. Git подсказывает нам эту команду, когда мы проверяем статус:



Поиск изменённых файлов

Давайте уберём один из файлов из индекса и снова проверим статус.

Только один файл попадёт в коммит

Видим, что изменения в файле `main.css` проиндексированы, а изменения в файле `index.html` не проиндексированы. Когда мы сделаем коммит, в него попадут из только `main.css`.

Как исправить сообщение коммита

Вы написали сообщение коммита (commit message) с ошибкой. Как её исправить? У команды `commit` есть параметр `amend`, который позволяет изменить последний коммит.

```
git commit --amend -m "Правильное сообщение коммита"
```

Обратите внимание, что при изменении сообщения коммита меняется и его хэш. То есть, происходит не просто переименование коммита, а создаётся новый коммит с теми же изменениями, но с новым текстом.

Как удалить лишний файл из коммита

Вы закоммитили лишний файл и поняли, что он не нужен. Можно удалить его из коммита при помощи команды `git rm` и имя файла.

Давайте удалим ненужный файл и проверим статус.

Статус файла изменился

Видим, что файл стал помечен как удалённый.

Осталось изменить коммит:

```
git commit --amend --no-edit
```

Напоминаем, что при этом поменяется хэш коммита, потому что когда мы делаем `amend`, Git не изменяет старый коммит, а создаёт новый и заменяет старый на новый.

Обратите внимание, что при этом способе файл удаляется не только из коммита, но и из папки проекта. Что делать, если нужно исключить файл из коммита, но при этом не удалять его?

Это можно сделать при помощи той же команды и флага `--cached`:

```
git rm --cached путь-к-файлу
```

Посмотрим статус: файл одновременно и удалён (изменения, которые будут закоммичены), и `untracked`, то есть не отслеживается.

Файл исключен из поиска

Осталось изменить прошлый коммит командой `git commit --amend --no-edit`.

Похожим образом мы можем и добавить файл в коммит, если поняли, что он должен быть в этом коммите.

Прочитали главу?

Нажмите кнопку «Готово», чтобы сохранить прогресс.

Готово

⚠ Если вы обнаружили ошибку или неработающую ссылку, выделите ее и нажмите Ctrl + Enter

Поиск по материалам

Git

[Все материалы](#)

В самом начале



- ☐ Пройдите опрос
- ☐ Укажите персональные данные
- ☐ Изучите регламент
- ☐ Прочитайте FAQ
- ☐ Добавьте свой Гитхаб
- ☐ Выберите наставника
- ☐ Создайте проект

Мой наставник



[Выбрать наставника](#)

Работа с наставником

У вас осталось **10** из 10 консультаций.



Практикум

Тренажёры

Подписка

Для команд и компаний

Учебник по PHP

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Информация

Об Академии

О центре карьеры

Услуги

Работа наставником

Для учителей

Стать автором

Остальное

Написать нам

Мероприятия

Форум

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696



Участник

© ООО «Интерактивные обучающие технологии», 2013–2023

