Frontender[1.0] GIT Команды, консоль. Полный гайд, чистая практика. Блок 2

YouTube	https://youtu.be/UfUjlwWaUjU
Telegram	https://t.me/Dmitry_Kolotilshikov
# Номер урока	81

📌 **Git комманды (одни из популярных):**

git init (Создает новый репозиторий Git или инициализирует существующую папку как репозиторий Git)

git remote add origin <repository_name> (Добавляет удаленный репозиторий с указанным именем и URL)

git remote -v (Показывает список удаленных репозиториев, связанных с текущим репозиторием)

git remote show https://github.com/YOUR_ACCOUNT/YOUR_REP_NAME.git (Показывает информацию о указанном удаленном репозитории, включая URL и ветки)

git clone <URL> (Клонирует удаленный репозиторий на локальную машину)

git config --global user.email “user email” (Устанавливают адрес электронной почты, связанный с коммитами)
git config --global user.name “user name”
(Устанавливают имя пользователя, связанный с коммитами)

НА МАКЕ ЕСЛИ ХОТИМ ЧТО-ТО ПОСМОТРЕТЬ ГЛОБАЛЬНО ДОБАВЛЯЕМ ПЕРЕД КОМАНДОЙ СЛОВО **SUDO** (sudo git …)

ТАКЖЕ ЧТОБЫ УВИДЕТЬ СКРЫТЫЕ ФАЙЛЫ В ПАПКЕ НА МАКЕ, В САМОЙ ПАПКЕ ЖМЕМ **Command + Shift + .**

git config --list (Показывает список конфигураций)
git config --list | findstr user.name
(Показывает найденую строку в конфиге по запросу)

git branch (Показывает список локальных веток в репозитории)

git branch -a (Показывает список локальных и удаленных веток в репозитории)

git branch -M <branch_name> (изменят название текущей ветки локально)
git branch -D <branch_name>
(удаляет ветку локально)
git branch -m <old_branch_name> <new_branch_name>
(изменят название выбранной ветки на новое, локально)

git switch <branch_name>
(Переключает на ветку с указанным именем)
git switch -c <branch_name>
(Переключает на новую ветку с указанным именем)
git switch -
(Переключает обратно на предыдущую ветку)

git checkout <branch_name> (Переключает на ветку с указанным именем, тоже самое что и git switch <branch_name>)

git checkout -b <branch_name> (Переключает на новую ветку с указанным именем, тоже самое что и git switch -c <branch_name>)

git add <file name>
(Добавляет измененный/созданный файл в индекс (staging area) для последующего коммита)
git add .

(Добавляет все измененные/созданные файлы в индекс (staging area) для последующего коммита)

git commit -m “your commit name”
(Создает коммит с указанным сообщением, включая все файлы, добавленные в индекс)
git commit --amend -m “an updated commit message”
(Изменяет созданный коммит с указанным сообщением)
git commit --amend --no-edit
(Изменяет созданный коммит оставляя предыдущее сообщение)

git status (Показывает текущее состояние репозитория, включая неотслеживаемые файлы, измененные файлы и т.д.)

git push <branch_name>
(Отправляет изменения в удаленный репозиторий)
git push -u origin <branch_name>
(Отправляет изменения в удаленный репозиторий, это команда только когда локальная ветка отсутствует в удаленном репозитории)
git push origin --delete <branch_name>
(Удаляет ветку в удаленном репозитории)

git pull
(Получает последние изменения из удаленного репозитория и объединяет их с текущей веткой)

git log
(Показывает историю коммитов в репозитории)
git log --oneline
(Показывает историю коммитов в одну строку)
git log --all --decorate --oneline --graph
(Показывает историю коммитов красиво)

git cherry-pick <commit_hash>
(хэш коммита это просто уникальный id)

git cherry-pick --continue (Продолжает операцию cherry-pick)

git cherry-pick --abort (Отменяет операцию cherry-pick)

git merge <branch_name> (Объединяет указанную ветку с текущей веткой)
git merge --abort
(Отменяет объединение)

STASH:

git stash (Сохраняет текущие изменения в отдельной области (stash), чтобы временно переключиться на другую ветку без коммита)

git stash save “stash_message” (Сохраняет текущие изменения в отдельной области (stash), с установленным названием)

git stash save -u “stash_message” (флаг -u -include-untracked означает включать неотслеживаемые файлы в stash, команда также сохраняет текущие изменения в отдельной области (stash), с установленным названием)

git stash list (показывает список прятаний)

git stash apply <number> (применяет прятание по номеру или без номера, то последнее)

git stash pop <number> (применяет и удаляет прятание по номеру или без номера, то последнее)

git stash drop <number> (удаляет прятание по номеру или без номера, то последнее)

git stash show <number> (показывает какие изменения содержатся в прятание по номеру или последнем если без номера)

git stash clear (удаляет все прятания)

git stash branch <название> <stash_number> (создает новую ветку с последним прятаньем или по номеру прятания, и затем удаляет последнее или по номеру прятанье (как *git stash pop*))

RESET:

A - B - C - HEAD

git reset B(^ или ~)
git reset --soft B(^ или ~)
git reset --mixed B(^ или ~)
git reset --hard B(^ или ~)

если пушить изменения после git reset в репозиторий и если в этом репозитории присутствует отмененный коммит то нужно вызывать **git push -f (-f или --force).** Т.е пушим форсированно

`^` означает "(первый) родительский элемент". `~` аналогичен, но он принимает число как аргумент и в основном означает "предок". Итак, например:

```
HEAD            = latest commit
HEAD^  = HEAD~1 = parent of latest commit
HEAD^^ = HEAD~2 = grandparent of latest commit
HEAD~100        = 100th ancestor of latest commit
```

REVERT:
A - B - C - HEAD

git revert --no-commit A
git revert --no-commit A (можно A B C)

(Тоже само что и git restore --source A
(можно A B C))

git revert --abort

git restore --source A .

(Тоже само что и: git revert --no-commit A
(можно A B C))

git blame <file> (Показывает автора и последний коммит, внесшие изменения в каждую строку указанного файла)
git blame -L <start>,<end> <file>
(Показывает автора и последний коммит, внесшие изменения в указанный диапазон строк

файла)

DIFF (Показывает изменения в сравнении, эти команды часто заменяет ui интерфейс в IDE):

```
git diff
git diff HEAD
git diff <branch_name1> <branch_name2>
git diff <commit_hash> <commit_hash>
```

💡 **Основы работы с командной строкой (Терминалом) на Mac OS X:**

https://vertex-academy.com/tutorials/ru/komandy-komandnaya-stroka-terminal-v-macos/