git init

git add ./

git status

git diff

git commit -am “Commit name”

git commit --amend

git log

git log -2

git log -p -2

git diff 4da05f..88351e

git show 88351e

git checkout 4da05f

git checkout 88351e ./css/style.css

git reset --hard 4da05f

**Команды командной строки:**

ls – позволяет получить список файлов и папок данной директории (*в CMD команда dir*).

cd (change dir) – смена директории.

Пример: cd~ - тильда приведёт нас обратно в наш каталог.

cd/ - слэш приведёт прямо на диск C: , с которого открыт терминал.

cd c: - обращение к диску C:\ .

cd .. – возвращает на каталог выше.

cd ../.. – возвращает на два каталога выше.

cd ../../.. – возвращает на три каталога выше.

cd . – текущий каталог (например, нужно перейти в какую-то папку относительно текущей, тогда пишем cd ./Desktop/школа ).

clear - очищает консоль (когда много чего понаписывал) (в CMD команда CLS , как Clear Screen).

В Git все команды начинаются со слова git. Например, git init, git commit, git status.

Дальше может идти опция, которая относится к конкретной команде (git config --local). У одной команды может быть несколько опций.

У Git есть встроенный справочник команд, который доступен из консоли. Данный справочник вызывается git help -a . Для пролистывания справочника нажимаем Пробел. Когда дошли до конца и хотим выйти из справочника, жмём клавишу Q (в английской раскладке).

Если нужна справка по какой-либо команде, пишем эту команду и в конце добавляем опцию

--help.

Например, git status --help.

**Команды Git**

git init – инициализация системы контроля версий Git в текущем проекте, для того чтобы можно было начать делать снимки своего проекта в Git (делаем один раз, если уже работали с данным проектом и применяли эту команду, то второй раз делать не нужно).

**Репозиторий** – специальный каталог, в котором хранятся все изменения, которые когда либо были сделаны во время всей жизни вашего проекта, с момента инициализации git-репозитория и до полного завершения проекта.

После инициализации появляется скрытая папка .git , которую нельзя изменять, удалять, переносить и т.д.

git commit – специальная команда для сохранения файла в .git-репозиторий (файл копируется в .git-репозиторий и получает специальный идентификатор хэш).

**Хэш** – специальный набор символов (40 16-тиричных символов). Он хранит в себе инфо о том кто сделал изменения, дату изменения и ссылку на предыдущую версию проекта, если она была. Если ничего не менялось, то будет хранить ссылку на предыдущую версию (коммит).

Все изменения попадают в .git-репозиторий навсегда и хранятся там.

Последняя версия проекта получает специальный указатель HEAD и весь локальный репозиторий ссылается на HEAD.

HEAD – указатель на последнее изменение.

Конфигурационный файл:

* системный
* глобальный
* локальный

git config --system --list - отображает параметры системного конфига (сюда не лезем).

git config --global --list - отображает параметры глобального конфига.

\* значения user.name и user.email .

Глобальный конфиг – настраивается на уровне вашего компьютера. Ты можешь работать над своим проектом за чужим компьютером, за своим.

git config --global user.name “Alex” – добавление для user.name значения “Alex”

git config --local –list

git status – даёт инфо в каком состоянии находятся файлы текущего проекта.

git add – добавление файлов в staging area (отслеживаемые) для отслеживания.

git add index.html - добавление в отслеживаемые файла index.html

git add ./ - добавляет всю текущую директорию и все файлы в ней в отслеживаемые.

git diff - даёт возможность посмотреть разницу между версиями до изменения в отслеживаемых файлах и текущей версией (рассматривает разницу между рабочим каталогом и индексом, т.е. staging area и именно до сохранения).

git commit – сохранение изменений

git commit -am “First commit” - сохранение изменений с добавлением комментария

Для изменения последнего коммита (только последнего):

git commit --amend - (откроется окно редактора, вверху название последнего коммита).

Для изменения имени коммита нажимаем клавишу Insert (Ins). После изменения жмём Esc, потом Ctrl + C.

Если хотим отменить внесённые изменения в имени коммита, жмём :qa!

Если хотим сохранить изменения в названии коммита, жмём :wq .

git log – просмотр истории всех коммитов

git log -2 - показывает 2 последних коммита

git commit -p -2 - показывает разницу между двумя последними изменениями, на уровне строк.

У коммитов есть хэши и если указать интервал между хэшами, то можно увидеть разницу между ними (записывают первых 6 символов хэша):

git diff 4da05f..88351e - покажет разницу между коммитами с хэшами с указанными первыми 6 символами хэшей (двоеточие это как интервал). Сначала указываем старый хэш, потом новый, но можно и наоборот, тогда будет помечено минусами, потому что этого ещё не было.

git show - показывает, что изменилось в текущем проекте по сравнению с каким-то коммитом и указать его хэш (git show 4da05f).

git checkout – переключение между коммитами (или ветками)

git checkout 4da05f - переключаемся на состояние коммита 4da05f (после переключения нельзя ничего менять в других коммитах, иначе оторвём голову HEAD). Если хотим делать изменения, то нужно вернуться в master или в коммите сделать новую ветку).

Если хотим посмотреть коммит в определённом файле, то указываем:

git checkout 4da05f и пишем путь относительно текущей папки проекта, например ./css/style.css

Полностью: git checkout 4da05f ./css/style.css (при этом голова не отрывается, мы всё равно находимся в каталоге master).

Откатывать файлы по одиночке не опасно, а вот все вместе опасно! (есть возможность откатывать конкретные файлы, но если делаем git checkout всего проекта, то не отрываем голову). Если оторвать голову, вернуться будет сложнее.

git reset – возврат всех изменений всего проекта на какой-то конкретный коммит.

git reset --hard 4da05f - (отменяет навсегда изменения абсолютно всех файлов и всего проекта), и указываем коммит до которого мы хотим откатиться (идёт затирание и всей истории).

Работая только в одной ветке *master,* мы рискуем запутаться, в том числе в истории. Рискуем наделать чего-то большого, что не сможем принять решение по откату. И в master есть возможность переключиться на другую ветку (не будем отрывать головы и сможем продолжить работать над проектом от какой-то версии).

git branch – ветка (показывает открытые ветки)

git branch catalog – создание новой ветки с названием catalog.

После создания ветки, для перехода в неё и работы в ней нужно набрать git checkout catalog