Front-end

JavaScript

JS. GIT

- GIT

JS. GIT

Git - это консольная утилита, для отслеживания и ведения истории изменения файлов, в вашем проекте. Чаще всего его используют для кода, но можно и для других файлов. Например, для картинок - полезно для дизайнеров.

С помощью Git-а вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий.

Репозиторием называют хранилище вашего кода и историю его изменений. Git работает локально и все ваши репозитории хранятся в определенных папках на жестком диске.

JS. GIT

Так же ваши репозитории можно хранить и в интернете. Обычно для этого используют три сервиса:

- GitHub
- Bitbucket
- GitLab

Каждая точка сохранения вашего проекта носит название коммит (commit). У каждого commit-а есть hash (уникальный id) и комментарий. Из таких commit-ов собирается ветка. Ветка - это история изменений. У каждой ветки есть свое название. Репозиторий может содержать в себе несколько веток, которые создаются из других веток или вливаются в них.

JS. GIT

Создание Git-репозитория

Обычно вы получаете репозиторий Git одним из двух способов:

Вы можете взять локальный каталог, который в настоящее время не находится под версионным контролем, и превратить его в репозиторий Git, либо

Вы можете клонировать существующий репозиторий Git из любого места.

В обоих случаях вы получите готовый к работе Git репозиторий на вашем компьютере.

JS. GIT

Имя пользователя

Первое, что вам следует сделать после установки Git — указать ваше имя и адрес электронной почты. Это важно, потому что каждый коммит в Git содержит эту информацию, и она включена в коммиты, передаваемые вами, и не может быть далее изменена:

```
git config --global user.name "John Doe"

git config --global user.email johndoe@example.com
```

JS. GIT

git init

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git, содержащий все необходимые файлы репозитория — структуру Git репозитория. На этом этапе ваш проект ещё не находится под версионным контролем.

"init" - это сокращение от "initialize" (т.е. "инициализироать" или "включать", "запускать").

Это значит, что когда Вы пишете "git init", Git "включается" или "запускается" для данного репозитория (т.е. папки).

JS. GIT

git init

Что происходит в момент "включения"?

Git создает в указанной папке скрытую папку ".git". В этой папке хранятся служебные файлы Git. Кстати, если её удалить, то Git уже не будет "видеть" эту папку (т.е. если мы напишем "git status", нам ответят что это "не Git репозиторий").

Зачем нужна эта команда? Это - одна из нескольких команд, с которой можно начать работу с Git. Например, если Вы создаете проект у себя на компьютере, а потом уже хотите его отправить куда-нибудь "в интернеты" на удаленный репозиторий, то Вам пригодится эта команда.

JS. GIT

git status

Чтобы освоить Git и пользоваться им через командную строку, Вам необходимо освоить несколько команд. С их помощью Вы будете добавлять файлы, удалить файлы, изменять файлы и т.д. Но иногда Вам нужно будет проверить, сработала ли та или иная команда, или в общем понять, что Вы там "наделали ". В этом Вам поможет команда git status.

С англ. status в данном случае так и переводится - "статус". То есть Вы можете посмотреть текущее состояние Вашего репозитория.

JS. GIT

git status

Итак, что можно проверить:

Является ли папка Git репозиторием ("есть" ли в ней Git) или нет

Мы можем просматривать, какие файлы были добавлены или изменены.

JS. GIT

git log

Команда git log используется для просмотра истории коммитов, начиная с самого свежего и уходя к истокам проекта. По умолчанию, она показывает лишь историю текущей ветки

JS. GIT

git add

Команда git add добавляет изменение из рабочего каталога в раздел проиндексированных файлов. Она сообщает Git, что вы хотите включить изменения в конкретном файле в следующий коммит.

JS. GIT

git commit

Коммит в Git является эквивалентом сохранения. Традиционное сохранение — это операция файловой системы, которая используется для перезаписи существующего файла или записи нового. В отличие от нее, коммит Git выполняется над набором файлов и каталогов.

Git-коммит — это операция, которая берет все подготовленные изменения и отправляет их в репозиторий как единое целое.

Зачем нужен коммит, если Git и так следит за всеми изменениями? Коммиты разбивают процесс разработки, состоящий из большого количества правок, на отдельные шаги. То есть коммит — это некое логически завершенное изменение внутри проекта и понятная (в том числе и другим разработчикам) точка, к которой можно вернуться, если возникнут какие-то проблемы.

JS. GIT

git commit

При создании коммита в репозитории можно добавить однострочное сообщение с помощью параметра commit с флагом -m. Само сообщение вводится непосредственно после флага, в кавычках.

```
git commit -m "Your short summary about the commit"
```

JS. GIT

git branch

Создать новую ветку можно с помощью параметра branch, указав имя ветки.

```
git branch new_branch_name
```

Но Git не переключится на неё автоматически. Для автоматического перехода нужно добавить флаг -b и параметр checkout.

```
git checkout -b new_branch_name
```

JS. GIT

git branch

Можно просматривать полный список веток, используя параметр branch. Команда отобразит все ветки, отметит текущую звёздочкой (*) и выделит её цветом.

```
git branch
```

Также можно вывести список удалённых веток с помощью флага -a.

```
git branch -a
```

JS. GIT

git branch

Удалить ветку можно параметром branch с добавлением флага -d и указанием имени ветки. Если вы завершили работу над веткой и объединили её с основной, можно её удалить без потери истории. Однако, если выполнить команду удаления до слияния — в результате появится сообщение об ошибке. Этот защитный механизм предотвращает потерю доступа к файлам.

```
git branch -d existing_branch_name
```

JS. GIT

git checkout

Команда git checkout позволяет перемещаться между ветками, созданными командой git branch . При переключении ветки происходит обновление файлов в рабочем каталоге в соответствии с версией, хранящейся в этой ветке, а Git начинает записывать все новые коммиты в этой ветке.

git checkout -b <имяветки> является шорткатом для git branch <имяветки> за которым идет git checkout <имяветки>

JS. GIT

git clone

Слово "clone" в команде git clone, конечно, расшифровывается как "клонировать". Сама команда имеет такую форму записи:

git clone <ссылка на репозиторий> <название папки>

То есть мы просим Git создать копию репозитория, который находится по ссылке (<ссылка на репозиторий>), и можем указать название новой папки, в которую Git скопирует репозиторий (<название папки>). Если его не указать, папка будет называться так же, как и сам репозиторий.

JS. GIT

git push

Отправлять изменения в удалённый репозиторий можно параметром push с указанием имени репозитория и ветки.

git push origin main

Эта команда передаёт локальные изменения в центральный репозиторий, где с ними могут ознакомиться другие участники проекта.

JS. GIT

git push

Передать новую ветку в удалённый репозиторий можно параметром push с флагом -u, указав имя репозитория и имя ветки.

```
git push -u origin new_branch
```

JS. GIT

git pull

Для загрузки изменений из удалённого репозитория используется параметр pull. Он скачивает копию текущей ветки с указанного удалённого репозитория и объединяет её с локальной копией.

JS. GIT

git fetch

Git fetch — команда в распределенной системе контроля версий Git, которая используется для скачивания изменений из удаленного или локального репозитория в свой локальный репозиторий. Польза git fetch в том, что эта команда привносит актуальность локальному репозиторию без изменения его дерева или текущей ветки.

JS. GIT

git merge

Объединить две ветки можно параметром merge с указанием имени ветки. Команда объединит указанную ветку с основной.

```
git merge existing_branch_name
```

JS. GIT

git remote

Команда git remote позволяет создавать, просматривать и удалять подключения к другим репозиториям.

Получить подробные сведения об удалённом репозитории можно с помощью параметра remote show с указанием имени репозитория — например, origin.

git remote show origin

JS. GIT

git remote

Добавить удалённый репозиторий можно параметром remote add, указав shortname и url требуемого репозитория.

git remote add awesomeapp https://github.com/someurl..

JS. GIT

HEAD

HEAD — это указатель на текущую ветку, которая, в свою очередь, является указателем на последний коммит, сделанный в этой ветке. Это значит, что HEAD будет родителем следующего созданного коммита. Как правило, самое простое считать HEAD снимком вашего последнего коммита.