Пн Вт Ср Чт Пт Сб Вс notes

Урок 1

Введение в контроль версий

Для чего?

1. Возможность хранить версии проекта → Возм-ть возвращаться к различным версиям

git

um

git - это программа, к-ая берет на себя вопрос контроля версий над проектом. Сохраняет в памяти не файлы целиком, а разницу м/у файлами.

Старый принцип	git
[1 стр] + [2 версия]	[стр ya] + 2 слова

Что нужно:

Сохранять версии — перемещаться м/у ними

Установить git и visual (VS cod)

ommits - фиксация, сохранение

ello world.md

ntracked file - неотслеживаемый

ежедневник

notes

Пн Вт Ср Чт Пт Сб Вс

git log	- команда, журнал изменений в хронологич. порядке
git checkout	- перейти к определенному сохранению, к разным версиям
Markdown маркдаун	- язык разметки
>	- комментарий
# _	Заголовок
## _	выделение текста
** * ** * *	жирный текст (полужирный) курсив
= или -	показать уровень заголовка
цифры 1,2,3	- нумерованные списки
* в начале строки	- ненумерованные списки
отступ	- вложенные списки
git init	- сделать репозиторий
git commit -m "..."	- фиксирует изменения сообщает о появлении новой версии файла
git diff	- отличия показывает
git log	- журнал всех изменений в хронологич-ом порядке
git add	- добавляет файлу версионность в локальном репозитории

ежедневник

Пн - тишина

Семинар 1.

Пн Вт Ср Чт Пт Сб Вс

notes

Quiz - викторина (квиз)

1. Запуск git init
2. Приглашение, начать отслеживание: git add # tab (название файла) с расширением!
3. Проверить статус (status) - д.б. зеленое: new file: ...
4. git commit -m "..."

Чтобы посмотреть имя пользователя - напрать [git config], чтобы войти - → клавиша Q

git config --list

Введение в контроль версий. Работа с Git. Составление инструкции по работе с Git.

Синтаксис языка Markdown

Справочник по Markdown от Microsoft:
https://docs.microsoft.com/ru-ru/contribute/markdown-reference

- # Заголовок – выделение заголовков. Количество символов "#" задаёт уровень заголовка (поддерживается 6 уровней).
- = или - – подчёркиванием этими символами (не менее 3 подряд) выделяют заголовки первого ("=") и второго ("-") уровней.
- ** Полужирное начертание** или __ Полужирное начертание__
- *Курсивное начертание* или _Курсивное начертание_
- ***Полужирное курсивное начертание***
- ~~Зачёркнутый текст~~
- * Строка – ненумерованные списки, символ "*" в начале строки
- 1, 2, 3 ... – нумерованные списки

Лекция 2. Введение в контроль версий. Ветвление.

branch - ветка

~~git~~ clear - очистить терминал

Создаем ветку (git branch название ветки) →

→ ~~git log~~ Проверяем (git branch) → переключ на новую ветку (git checkout ~~branch~~ название ветки) → проверяем что мы на ней (git branch)

git merge название ветки - слияние этой ветки с master

git branch -d ~~-D~~ название ветки - удаление ветки после её слияния с master

git log --graph - гр. отображение лог коммитов с визуализацией ветвлений

git merge --abort - отменить слияние

Лекция 3. Введение в контроль версий. GitHub.

Git - программа
- одна из систем контроля версий
- способ организации и поддержания версионности
- самая популярная система контроля версий

GitHub - сервис
- предоставляет хостинг
- самый популярный сервис Git
- огромный архив различного кода

remote - удалённый, внешний

git push - отправить

git pull - стянуть из сети, с др. компьютера

fork - вилка, делаем ответвление ← кнопка в GitHub

git clone - команда, позволяющая склонировать внешний репозиторий на наш ПК.

git pull - команда, позволяющая скачать всё из текущего репозитория и авт-ки слить ветви с нашей версией

git push - команда, поз-ющая отправить наш репозиторий на внешний репозиторий

1. Создаем аккаунт на Github.com
2. Создать локальный репозиторий
3. Подружить ваш локальный и удаленный репозиторий Github (при создании нового репозитория подскажет как это можно сделать).
4. Отправить (push) ваш локальный репозиторий в удаленный (на Github), при этом вам, возможно, нужно будет авторизоваться на удаленном репозитории.
5. Провести изменения с другого компьютера
6. Вытягивать (pull) актуальное состояние из удаленного репозитория

Проект на гитхаб аккаунте:

1. Делаем форк (fork) интересующего нас репозитория
2. Мы делаем git clone для нашей версии этого репозитория
3. Мы создаем ветку с предлагаемыми изменениями
4. Производим все изменения только в этой ветке
5. Отправляем эти изменения на свой аккаунт (push)
6. В форке на Github появляется возможность отправить pull request.

Как сделать pull request

- делаем fork репозитория
- делаем clone своей версии репозитория
- создаем новую ветку и в НЕЕ вносим свои изменения
- фиксируем изменения (делаем commit)
- отправляем свою версию в свой Github
- на сайте Github нажимаем кнопку pull request

Основные команды Git

- **git init** – инициализация локального репозитория
- **git status** – получить информацию от git о его текущем состоянии
- **git add** – добавить файл или файлы к следующему коммиту
- **git commit -m "message"** – создание коммита.
- **git log** – вывод на экран истории всех коммитов с их хеш-кодами
- **git checkout** – переход от одного коммита к другому
- **git checkout master** – вернуться к актуальному состоянию и продолжить работу
- **git diff** – увидеть разницу между текущим файлом и закоммиченным файлом

Введение в контроль версий. Работа с Git. Составление инструкции по работе с Git.

Основные команды Git

- **git init** – инициализация локального репозитория
- **git status** – получить информацию от git о его текущем состоянии
- **git add** – добавить файл или файлы к следующему коммиту
- **git commit -m "message"** – создание коммита.
- **git log** – вывод на экран истории всех коммитов с их хеш-кодами
- **git clone <url-адрес репозитория>** – клонирование внешнего репозитория на локальный ПК
- **git pull** – получение изменений и слияние с локальной версией
- **git push** – отправляет локальную версию репозитория на внешний