Введение в контроль версий

Лекция 1

Контроль версий - практика, позволяющая отслеживать изм-я исходн кода и управлять ими.

- хранит разн версии проекта (сайта)
- возвращ-ся к разным версиям.

Git хранит не файлы, а отличия м/у ними

readme.md

VS Code

- Проводник - открыть папку -
- Вид - Терминал

git config --global user.name "имя"

git config --global user.email "@"

git --version (проверка установки git)

git init (инициализ-я) → созд-ся скрытый файл в папке .git
⇒ (в этой папке нужно следить за контролем версий)

git status

- создать новый файл hello world.md (mark down)
- напиши текст, сохр-и (Ctrl+S)

  git status → появились неотслеж файлы

  git add hello world.md.

  git commit -m "Создали нов.файл" (изм-я сохр-ны)

- допишем текст в файл, Ctrl+S (Файл - Автосохр-е в VSC)
- git status → появились изм-я в файле.

  git add

  git status

  git commit -m "Добавили нов строку"

- git log (журнал изм-й) - для просмотра всех записей ↓, выйти: q
- если нужно вернуться к одной из версий:

  git checkout номер коммита (первые 5 символов) → возврат к др версии

  git checkout номер др коммита.

- после перекл-я м/у версиями нужно вернуться в ветку master

  git checkout master

- при перекл-и р-ла он удаляется и созд-ся новый с нов именем
- git diff (показ-т разницу) первые символы одного коммита — перв симв второго коммита

Язык маркдаун

- *Курсив* ~зачеркнутый текст~

Полужирный

Список:

* эл-т 1
* эл-т 2

1. Первый эл-т
2. Второй эл-т

Заголовки:

Заг. 1-го уровня

Линия под заголовком:

Заголовок

Заголовок
===

Windows:

dir (аналог ls - показать список файлов/папок)
cd (change directory) : cd.. - возврат в предыд. директорию
pwd (print working directory)

cd папка/ - переход в папку

- для

Урок 2

- git branch text_formatting - созд-е новой ветки
- git branch - просмотр всех веток (* показ-т тек. ветку)
- git checkout text_formatting - переключ-е м/у ветками → master
  -то, что делается в новой ветке, не влияет на предыдущ. * text_form...
  версию
- из ветки master созд-ся 3-я ветка lists
- для слияния ветки с изм-ми с веткой master нужно перейти в ветку master и набрать git merge text_formatting
  ↓
  git checkout master
- если ветка text_formatting больше не нужна, ее можно удалить:
  git branch -d text_formatting.
- для вставки изображ-я нужно файл с картинкой сохр-ть в папке.
  ![Это котик](kotik.jpg)
  будет выведен, если ошибка в имени файла.
- добавлять к индексу файл с картинкой не нужно. Для этого созд-ся файл в этой же папке .gitignore, где пишется имя файла с картинкой.
  git add .gitignore далее git commit -m "Добавили gitignore файл"

- если в ветке master и другой ветке отличаются данные, происходит конфликт версий. После их слияния нужно решить, какую версию принять (первую, вторую, обе) и после этого сделать еще один коммит.
- для визуализации веток и коммитов используется
  git log --graph.
- просмотр коммитов более компактно:
  git log --oneline
- чек-боксы:
  - [ ] эл-т → ☐ эл-т
  - [x] эл-т → ☑ эл-т
- цитата:
  > это цитата
- код на python

```
''' python
a=5
b=6
...
'''
```

- вставить картинку
  ![GB](/название файла)
  / подтягивает изображение из папки
  файл лучше хранить в папке проекта.
- Markdown Preview Enhanced - расширение
  Git Graph
  Html CSS Support
- таблицы

Name	Value	Amount
Comp	1600	3
Phone	42	2
Pipe	900	1

← выровнивание (по левому кр., центру, прав.)

Лекция 3

Удаленный репозиторий - как взять чужой реп-й

На GitHub найти реп-й, Code - Clone - Https - copy

В VSC открыть папку, в терминале - git clone https://... (ссылка на GitHub)

Как загрузить свой реп-й

- git init в созданной папке
  добавляем файл → git add → git commit
- в GitHub создать нов. реп-й
- push an existing repo
  git remote
  git add
  git push
- после внесения изм-й в лок реп-и
  git add , git commit , git push
- после внесения изм-й на GitHub, чтобы загрузить в лок-й реп-й
  - git pull
- чтобы перенести реп-й с чужого аккаунта - fork в GitHub
  git clone на лок. реп-й
  создаем ветку с изм-ми
  git push в свой реп-й
  pull request