git

TODAY

1. Left-overs from yesterday
2. git, what is it
3. Prerequisites
4. A short interactive tour
5. Let's memorize some words
6. Setting up a project
7. Adding & commiting
8. Pushing
9. Checkout
10. Resetting
11. ~~Branches~~
12. ~~Merging~~
13. ~~Checkout~~
14. ~~Dictionary~~

LEFT-OVERS

1. The assignment
2. Introduction to VS Code
3. Anything else?

GIT, WHAT IS IT

Git is as "version control system"

That means we can store our "project" in different versions

And we can go back to previous versions


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.


ONE DOES NOT SIMPLY
UNDERSTAND GIT


ONE DOES NOT SIMPLY
UNDERSTAND GIT


ONE DOES NOT SIMPLY
UNDERSTAND GIT


ONE DOES NOT SIMPLY
UNDERSTAND GIT

BUT FIRST

Is git really worth all of this trouble?

BUT FIRST

Is git really worth all of this trouble?

1. YES, and it's what the cool kids use

BUT FIRST

Is git really worth all of this trouble?

1. YES, and it's what the cool kids use
2. YES, it allows us to try something out without risking anything

BUT FIRST

Is git really worth all of this trouble?

1. YES, and it's what the cool kids use
2. YES, it allows us to try something out without risking anything
3. YES, and it's not that hard, it's just confusing

BUT FIRST

Is git really worth all of this trouble?

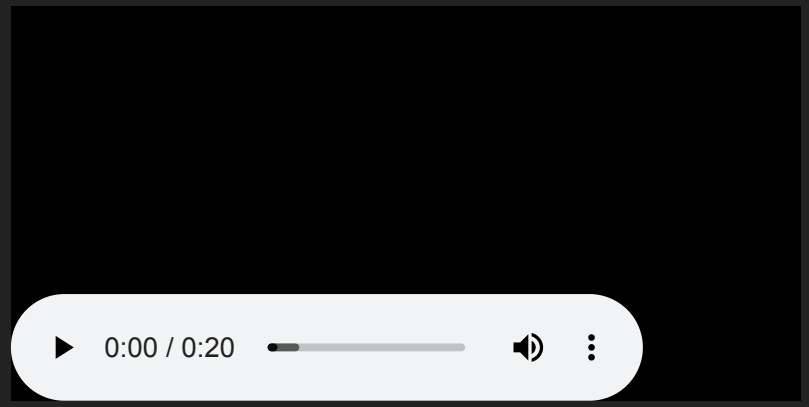
1. YES, and it's what the cool kids use
2. YES, it allows us to try something out without risking anything
3. YES, and it's not that hard, it's just confusing
4. YES, In teams, it's a life saver

INSTALL TIME

PREREQUISITES

Instructions on fronter (Install Day)

1. VSCode
2. Git
3. GitHub account
4. Live Server
5. GitLens
6. axe Accessibility Linter
7. Error Lens



A SHORT INTERACTIVE TOUR

EXERCISE #1

GITFOGRAPHIC<3

https://jonasholbech.github.io/gitfographic3/

Complete the following levels

1. Introduction
2. Overview
3. .gitignore
4. Commits

30 minutes

Try to define the following keywords

- add
- commit
- git
- staging area
- push
- init
- repository
- pull
- remote
- local

SETTING UP A PROJECT

SETTING UP A PROJECT

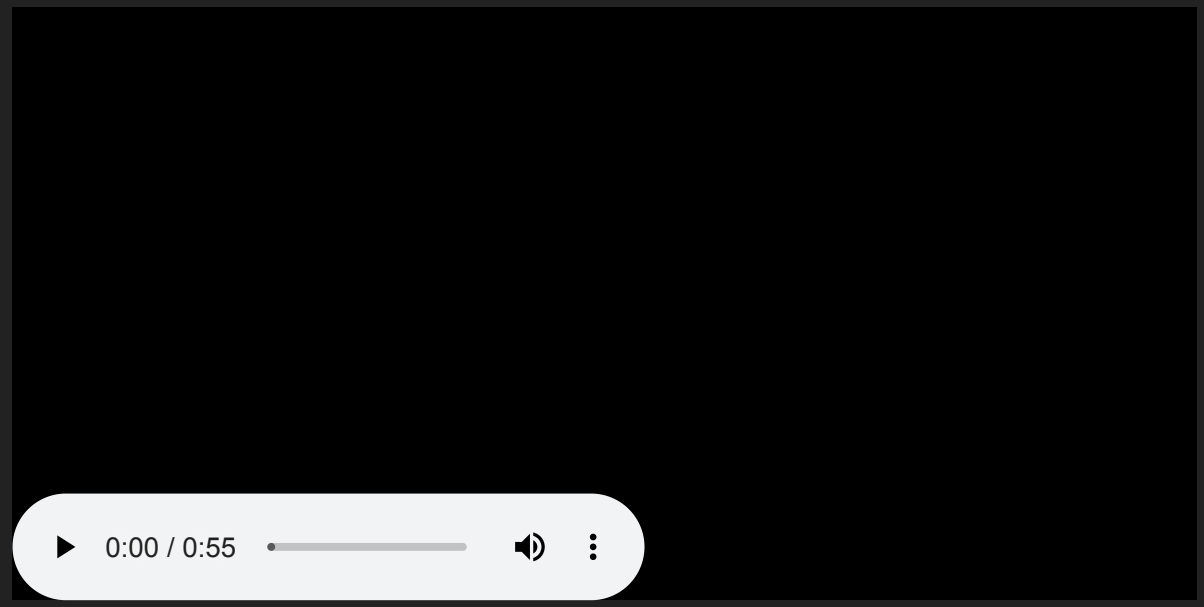
Locally

SETTING UP A PROJECT

Locally

New or old

1. Create a new project in VS Code
    - Add a few files / folders

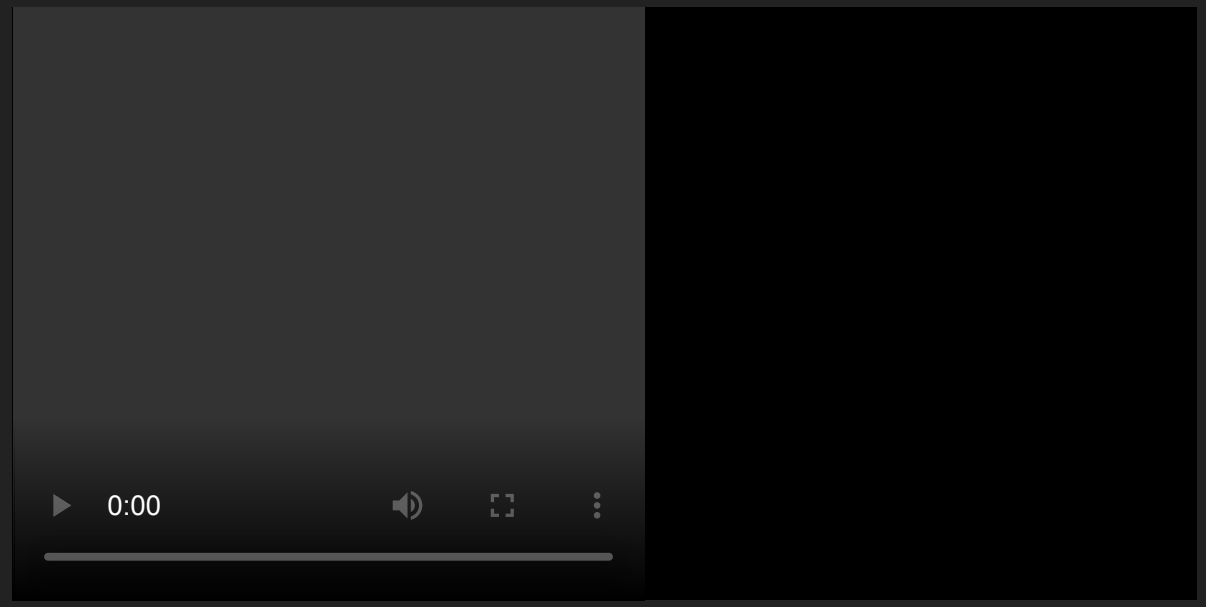


2. Initialize it as a git project

- Open the "Git Tab"
- Click "Initialize Repository"

Sorry about the sound :-)

3. In VS Code, `add` (+) all the files you need
4. Write a meaningfull message and click `commit` (V)



Now you have version control locally.

Unless your harddisk dies, you've got a snapshot of your code

EXERCISE #2

1. Split into groups (2-3 people)

2. Player A: follow the steps above, talk each other through the process. Then...

3. Player B: follow the steps above, talk each other through the process. Then ...

4. Player C: follow the steps above, talk each

HOOKING IT UP TO GITHUB

HOOKING IT UP TO GITHUB

Backing up

HOOKING IT UP TO GITHUB

Backing up

Preparing for collaboration

We need to create a remote repository to store our code

There are several ways we can do this, but this is what I found to be the easiest

5. In VS Code, press `Cmd/Ctrl + Shift + P` to open the Command Palette
6. Start typing "Publish to GitHub" and select the option when it appears
   You probably have to login.
7. Choose either a ~~private~~ or **public** repository
8. You can click "Open in GitHub" to see you new

- This will also issue your first `push`
- Now your code is backed up on GitHub
- Your project is set up, congrats

That was the setup, with practice, this can be done in 5 minutes

EXERCISE #3

We'll spread out into groups again

All members try to connect their repository to GitHub

My suggestion: Player A goes first and the other to help out. When the first one is working, the other two do it

A SHORT INTERACTIVE TOUR

EXERCISE #4

GITFOGRAPHIC<3

https://jonasholbech.github.io/gitfographic3/

Complete the rest

30 minutes, go through it now

Try to define the following words

- `branch`
- `merge`
- `reset`
- `checkout`
- `feature` (not a git word)
- `commit message`
- `GitHub`

WORKING

WITH GIT, ALONE

MANAGING A SINGLE TIMELINE

1. add
2. commit
3. push
4. checkout
5. reset

add

Do you get it?

`commit`

Do you get it?

`push`

Do you get it?

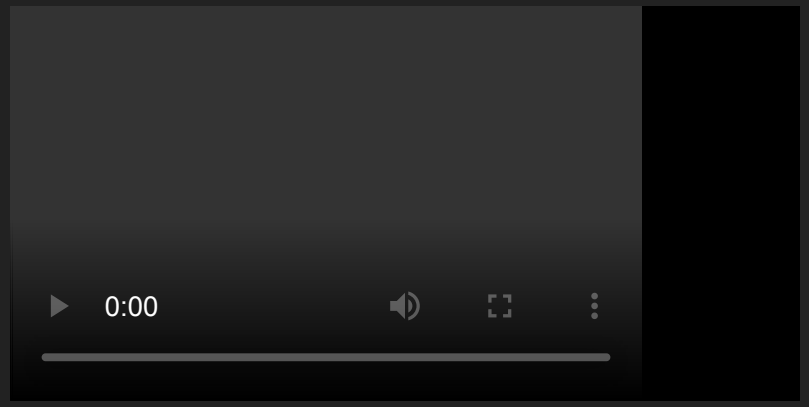
DEMO?

1. add
2. commit
3. push

`checkout`

Looking at a point in time

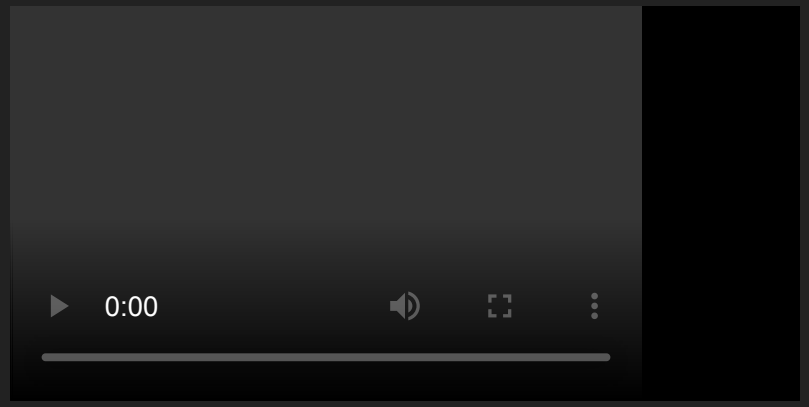
You can look, but you can't (shouldn't) touch



(No audio)

GOING BACK TO OUR TIMELINE

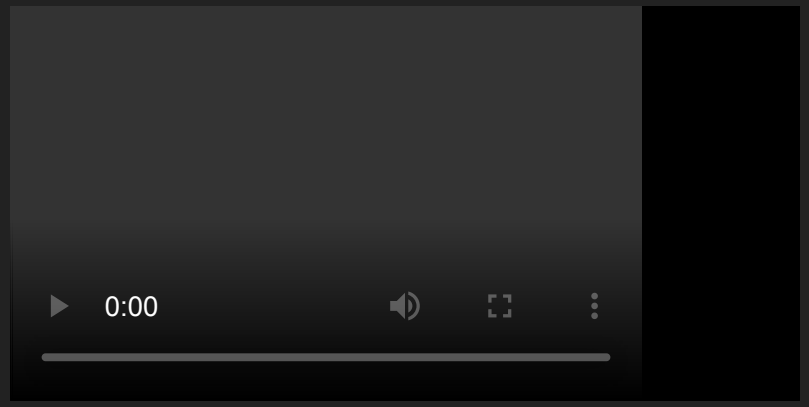
Once we've looked, we can go back to our own timeline (master)



(this one has audio)

reset

Reset was destructive, so that means move back permanently (sort of)



(this one has audio)

`reset` is a "last option" other (and better) ways exists

`push --force`

Let's imagine we've pushed everything

Then we reset locally

Now the local and remote repos are out of sync

If we try to push again we get an error

If we are REALLY sure we want to go back

We can do a forced push

But that's not for now

GENERAL THOUGHTS, PART #1

"SAVE GAMES"

Commits are points in history we can go back to

That's called checking out or resetting

"SAVE GAMES"

Commits are points in history we can go back to

That's called checking out or resetting

In general, if something is working, do a commit

Try to do one thing per commit

If we haven't already, let's do it

Let's go to the whiteboard

If we haven't already, let's do it

Let's go to the whiteboard

Any code produced in this example is not really important, focus on git

FORKING AND CLONING

CLONING

Cloning downloads a copy of a GitHub repository

If it's **your** repo, you can now push

If it's someone elses, you can **not** push

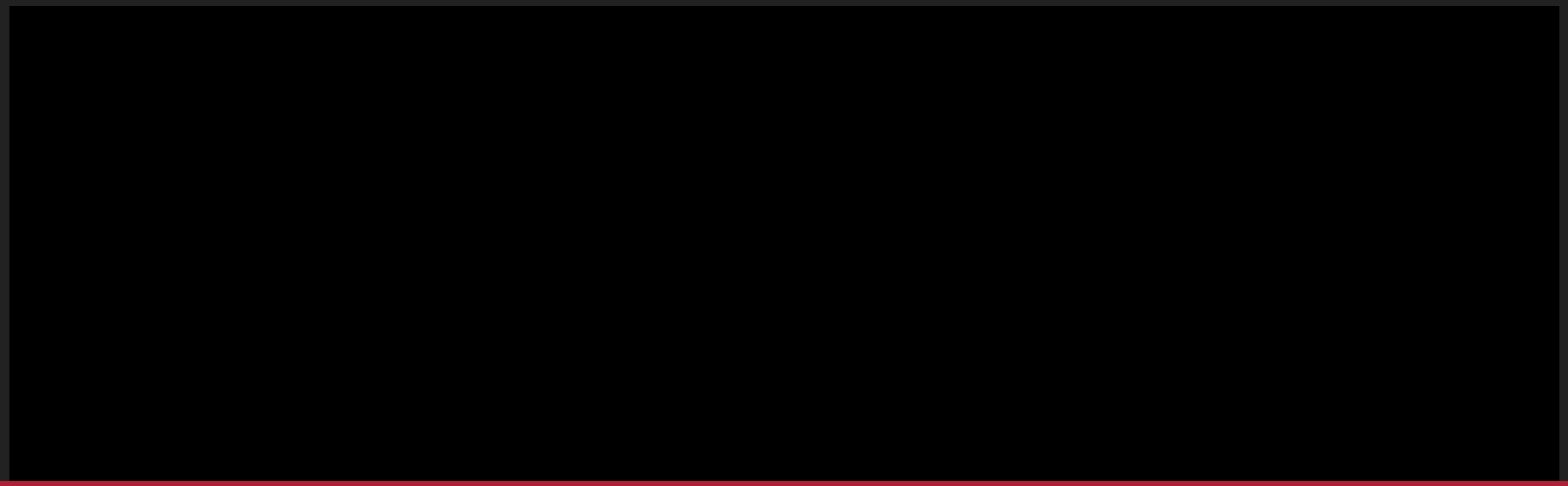
FORKING

Forking (technically not a git thing) means:

"Copy this cool repository to my profile so that it is now mine"

What's the difference then?

FORKING



CLONING

To `clone` is to take all code from a GitHub repository and downloading it

It's THE best way to share code

And it's not hard

1. Find the repository you would like to clone
2. Grab the `.git` url
3. In VSC type `Cmd/Ctrl + Shift + P` and find `Git: clone`
4. Paste in the URL
5. Find a folder you would like to store the code in
6. That's it