Git 01

Why do we need git?

- Seeing the history of what has been changed
- Jump back in time
- Working collaboratively, remote and even offline
- Easier integration of new features
- Having different versions (branches)

Why not Dropbox?

…or any other cloud based file manager

Problems with Dropbox

- Files will not be merged automatically
- You can not track the exact changes that have been made
- It is only possible to roll back single files, not a whole directory
- There are no fixed, named points you can return to (only a date)
- If you are not online while editing, there is even more trouble ahead
- There is no way to maintain two or more different versions of a file

diff to the rescue!

- Parses a files line by lines and compares them
- Outputs lines that are different
- Has a long history in rolling out patches to source code
- git has fully been build on the principal of diff

git config

- Set your name: git config --global user.name „John Doe“
- Set your email: git config --global user.email „john@doe.com“
- Set the default editor: git config --global core.editor "code --wait -r"
- You can also set a name and an email for each repository

git init

- Initializes a new repository
- Creates the hidden folder „.git“

git clone

- Clones an existing git repository
- Receives the whole history of everything ever done in that repository
- There is no such thing as a „master“-repository

git status

- Where are you in your repository?
- Are there any changes made to the code?
- Are any files staged?
- Is everything pushed to the remote repository?

git add; git commit

- git add is used to „stage“ changes for a commit
- git commit creates a commit of the staged changes
- Commits are just a collection of diffs (with a name and a checksum)
- Follow those rules:
  - Try to stay below 50 characters
  - Think in your mind „This commit will…“ and complete the sentence
  - Do not end subject line with a period

git reset

- Unstages all staged files
- Can also be used with „--hard“ to reset all changes in the code

git log

- See a list of past commits
- Find out who has done what and when
- Check if your commits are as intended

git checkout

- Switches the HEAD to any commit
- Is used to switch between branches
- Can also get a single file of a commit

git branch

- Branches are useful for
  - Features
  - Hot fixes
  - Different versions of the same project
  - Keeping track of what should go live and what still needs testing
  - Collaboration
  - Reviews
- Branches are cheap, so use them plenty!

git push

- Remote repositories can be everywhere, even on the same machine
- Bring all your changes back to the remote repository
- Do this often
  - better for colleagues to follow your progress
  - serves as a backup
  - bus factor

git fetch; git merge (or: git pull)

- A fetch will get all the changes made in the remote repository
- Your local branches will stay untouched
- A merge gets changes from one branch into another
  - those can be just local branches
  - or it can be changes from a remote branch into a local branch
- pull is a short-hand for fetching and merging in one go (fast-forward)

Try it on your own

- Make changes to a file and/or add a new file
- Check your changes with „git status“
- „git add .“ and „git commit“ those changes
- Verify all your changes have been commited with „git status“
- Check your commit with „git log“
- Update remote repository with „git push“
- Repeat 5 times

- Advanced:
  - Make several changes, but do not commit everything
  - Make a change through the Github website and „git pull“ – verify your local files