Intro to Git and Github

Hyper Island - FED27

Per Enström – 2025-09-05

Git

What is it

- Version management
  - Instead of creating assignment.docx, assignment-2.docx, assignment-final.docx, assignment-final-2.docx, assignment-final-final-for-real.docx
  - Like the version history in Google Docs, or Dropbox
  - Like the history in Photoshop
- Manual control over every step in the history, we decide ourselves when and what to add as a history step

Git

Why do we use it?

- Makes it easy to see when something was introduced or removed in your code base
- Allows you to work on different branches of your code
- Makes collaborating on code seamless
- Allows us to easily release new versions when we’re done with a feature, instead of always doing changes in the published web page

Git

Important!

Don’t add sensitive information (API Keys, passwords, etc) to git!

While we *can* rewrite the git history, it’s a pain, and once a password has been added to the history it’s visible forever.

Git

Repository

A project in Git is called a *repository*. The repository contains a hidden folder that contains all information related to git, the full history, different branches, etc.

We create a new repository for every project or web site.

To initialize a repository in a folder on our computer we use the command `git init`. This creates an empty repository in that folder, ready to accept our changes.

Git

Commit

- Every change in a git history is called a *commit*. It’s a change we *commit* to.
- Every commit is stored as a difference between the previous step and the next, i.e. we don’t store all code in all steps, only what has changed.
- These differences are called *diffs*.
- These changes can then be used to go back in time to any point in the code base’s history.
- Every commit has an accompanying *commit message* that we write ourselves
  - The commit message should be brief and descriptive of the change
  - *The Will Rule*, any commit message should make sense if we say "This commit will" before the message
    - Good: "Add an image", "Fix bug in login", "Rework the home page"
    - Bad: "Added image", "Fixed bug", "Reworked the home page", "fix", "asdfasdfa"

Git

The stage, and tracking

- To commit changes to a file, we need to tell Git to *track* that file
- All files start out untracked
- Every uncommitted change we are working on needs its file to be tracked, and then added to *the stage*, before we can commit it
- We can – and often do – stage only part of what we've changed and commit that
- Tracking and staging are conveniently done in one command, the `git add` command
  - `git add` takes a third parameter which is the file we want to track and stage
  - If we want to track and stage a file called `index.html`, we use the command `git add index.html`
  - We can also track and stage all changed files at the same time with `git add .` (notice the period)

Git

How often do we commit?

- General rule is to commit small and often
  - This could be several times an hour, or a couple of times a day

- A goal is that the code base is always working at every commit
  - This means that when we're building web apps in Javscript, we should not commit code that is in a state that doesn't work
  - Doing many small commits is very helpful when collaborating, it makes it easy to track how files have evolved over time, and makes combining different changes from different people much easier