Women Techmakers

Git, GitHub & Web Hosting Workshop
WTM Hamburg

Git, GitHub & Web Hosting Documentation

During our Workshops we're going to develop parts of our WTM Hamburg Website together.

At this point, we'll continue collaborating all as a team via version control using Git, and hosting our website with GitHub to go live.

This document will give you a general definition of what Git, GitHub and Web Hosting are and what they are used for.

You can refer to this documentation at all Workshops, during the development of your tasks.

What is Version Control and why should you use it?

Version control is an important part of the everyday software development process. We need version control to keep track of the changes made to code and documents which make up software projects.
There is a lot of good reasons for using a version control systems, as for instance when a file is deleted by mistake and needs to be recovered, which is made possible by version control.

Also imagine working on a project with other people and having to tell every single one of them that you are currently editing a certain project file. Your co-workers would have to stop working on the same file because they might otherwise overwrite your changes.

This would be a pretty ridiculous workflow, right? Version control solves this problem by supplying each developer with their own local version of the code that he or she edits, so that nothing is overwritten.

Read further:
https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

VSC (Version Control System) terminology

The changes tracked by a version control system (short: **VCS**) are identified by a code made up of numbers and letters called the **revision number**.

This number identifies the state of the code when the developer makes a **commit.** You can imagine commits to be a “snapshot” of the state of the files that make up a software project at a certain point in time.

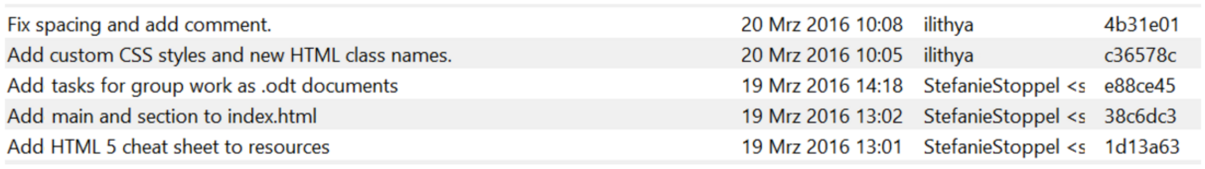
A commit, or "revision", is an individual change to a file (or set of files). When you save a file, it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who. Commits usually contain a commit message which is a brief description of what changes were made.

You should always add **commit messages** to the commits because it makes it easier for you and others to keep track of the changes made to the code without having to go through all of it.

Commit messages should be always meaningful, and as best practice should always be written in English and in present tense.
E.g. Instead of “I added tests for” or “Adding tests for,” use “Add tests for.”

Read further on commit guidelines here:
https://git-scm.com/book/ch5-2.html

In the screenshot below you can see a few commits made on the WTM HH website project. The first column displays the commit message. The second and third column show when and by whom the change was made. The last column contains the revision number.



Fix spacing and add comment.	20 Mrz 2016 10:08	ilithya	4b31e01
Add custom CSS styles and new HTML class names.	20 Mrz 2016 10:05	ilithya	c36578c
Add tasks for group work as .odt documents	19 Mrz 2016 14:18	StefanieStoppel <s	e88ce45
Add main and section to index.html	19 Mrz 2016 13:02	StefanieStoppel <s	38c6dc3
Add HTML 5 cheat sheet to resources	19 Mrz 2016 13:01	StefanieStoppel <s	1d13a63

Info:
Once you start using a VCS you should get used to making commits of your code as often as possible. This helps keeping track of even the smallest changes in a software project and is generally considered good practice.

Usually there is a server where the files and their commit histories are stored. This central code hub is called a **repository.**

When you start working on an already existing project you have to **clone** the code from the repository. This process creates a local **working copy** and a local repository of the code on your computer.

When making changes to the code only the working copy changes at first. To make the changes available to others and create a “snapshot” of the code at a certain stage you need to commit it.

In some VCS this is enough to make the code accessible to others (E.g. Subversion). With the VCS we’re using (**Git**) we still need to **push** the changes to the **central repository** so that others can access it from there by **pulling** our changes into their working copies.

Some words of advice:
At the beginning of your path as a developer all these words and terms can be very confusing. Don’t worry though, we have all been there and know what it’s like. You’re gonna get the hang of it eventually! There is a lot of very good tutorials and resources on version control to help you. We have included some of them inside this file but be sure to also search online yourself.

What is Git?

Git is a Version Control System, an open source program for tracking changes in text files. It was written by (amongst others) the author of the

Linux operating system — Linus Torvalds. It is a so-called “distributed VCS”.

This means, that every developer has their own local working copy of the code, plus a local repository where all the changes made to the code in form of commits are saved.

When the developer thinks it is time to make the commits available to others, he or she pushes them to the central repository where others can access them (by pulling the changes).

Read about git and other kinds of version control systems:
https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Branches

A branch is a parallel version of a repository. It is contained within the repository, but does not affect the primary or master branch allowing you to work freely without disrupting the "live" version.

When you've made the changes you want to make, you can merge your branch back into the master branch to publish your changes.

To find out more about branching in git, read this:
https://www.atlassian.com/git/tutorials/using-branches/

That tutorial also contains information about an important feature when working with branches: ***merging.***

Git commands overview

To remember the different Git commands it is a good idea to have a cheat sheet**:**
https://www.git-tower.com/blog/content/posts/54-git-cheat-sheet/git-cheat-sheet-large01.png

It is good to know some of the most frequently used commands for Git Bash (also called Git Shell). Git Bash is a command line client that comes with GitHub for desktop.

The following commands are often used when working with Git:

git status
Shows the local changes in your working copy of the repository.

git clone <url>
Clones an existing repository from the specified URL and creates a working copy of it on your computer.

git init
Creates a new Git repository inside the directory Git Bash is pointing at.

git checkout <branch>
Changes from the current to the specified branch. All changes made from here on will be made on the newly checked-out branch.

git add . / git add -p <file>
1) Add all local changes to be staged for the next commit.
2) Add changes in a specific file to be staged for the next commit.

git commit -m “A message”
Commit all staged changes and add a commit message in present tense.

git pull <remote> <branch>
Pull changes from a remote repository branch into your local repository.

git push <remote> <branch>
Push changes (commits) from your local repository to the remote repository branch.

Read further:
https://help.github.com/articles/github-glossary/

GitHub

GitHub is a web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.

Git vs. GitHub

A lot of people get these two things mixed up:

- **Git** is an open source version control system that keeps track of changes to a project's files.
- **GitHub** is an online service for hosting software projects.

Git is the core technology that GitHub, the social and user interface, is built on top of.

Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

Web Hosting

Web hosting or server is much like the space that you rent out to have your business in. It's merely the space itself.

It does not include furnishings like shelves for your products, just as the web hosting account doesn't include a site for you to sell your products.

Without the hosting services, you won't have a place for your files to reside, so your domain would then become like a disconnected phone number in the phone directory, and your site files would have nowhere to stay.

Read further:
https://en.wikipedia.org/wiki/Web_hosting_service

Domain Name

A domain name is a unique name that identifies a website.

Domain names on the internet are much like entries in a phone book. The phone book tells people looking for people or a business what the entries are just as a domain tells people (i.e. their computers) that a domain is hosted on the server.

Without a domain, you would have to tell people that your site is located at a temporary url such as 123.456.789.123/~mysite instead of using a domain name such as mysite.com.

All domain names have a domain suffix, such as .com, .de, .net, or .org. The domain suffix helps identify the type of website the domain name represents. For example, "*.com*" domain names are typically used by commercial website, while "*.org*" websites are often used by non-profit organizations.

What is the difference between domains vs. web hosting?

Domain Name
When you have a site visitor, they use your domain name to view your website.

Web Hosting Service
When a site visitor enters your domain name into a browser, the domain is then translated into your server IP address, then the server sends that user your site files, which their browser represents to them as a typical web page.

Happy Coding!
Women Techmakers HH Team