What is Git and Why Use it?

In this lesson, we'll discuss the basics of version control with git!

We'll cover the following

What is version control?

Managing your project's files. For example, when building a website, you will have a lot of files, like CSS, HTML, JS, images.

What is Git?

Git is the world's most popular version control system.

Why Git?

Git maintains a complete history of the changes made to any project, it makes collaboration simpler and more productive, and it allows working on multiple features at once.

History

Git keeps track of all the changes we make to our project. Suppose you change part of the CSS, and the layout looks great, but a few months later, that change causes the layout to break in a few pages. Instead of having to remember what you changed a few months ago, Git would have a record of exactly which lines were removed/added/modified, which makes fixing the issue a lot

easier. Furthermore, you can revert changes with Git. Nothing is ever lost or is ever final.

Collaboration

Git makes collaboration in a team very easy and increases productivity.

Suppose you and a team member are working together on some HTML code. In a Git-less scenario, you would write your part of the code, email the file to them; they would write their part and send it back to you. However, while they are doing their part, you won't be able to continue working on the code. If you do, you'd have to figure out what exactly your fellow team member changed when they send it back to you and somehow merge that onto your file with your new code. This combing for changes and merging is not only tedious work but is prone to human error. Furthermore, the time spent doing so can be spent more productively on actually writing code. The good news is that Git manages these changes and merges them for you so you and several team members can work on projects simultaneously without ever having to worry about accidentally deleting someone else's work or merging incorrectly.

Feature branches

Git allows you to work on and ship multiple features simultaneously. For example, suppose you want to update the way your website's navigation bar looks, and you want to add a feature that allows users to comment on your posts. These constitute as separate features: the new navigation bar and the comments. Now in a Git-less situation, if you start working on both features and you finish the navigation bar in a week but need 3 more weeks to enable comments, you won't be able to make your navigation bar live until the comments feature is done as well. So you would have to work on features sequentially if you want to make them live as they get completed, which means a loss in productivity.

Git, however, has the ability to make *separate branches* for each feature. These branches can be worked on simultaneously, and when once done, it can be *merged* back into the main branch.

Git in the industry

Git is the industry standard, and most employers assume that you are well versed in Git if you apply for a software job. Furthermore, it can make your time more productive and less prone to data loss even if you are working alone.

Let’s look at how to actually use Git in the next lesson!