Missing Semester - Branching and Merging (Week 3)

Introduction to Branching and Merging in Git

Welcome to today's lesson on Branching and Merging in Git! Git is a powerful version control system that allows developers to efficiently manage their codebase and collaborate with others. One of the key features of Git is its ability to create branches, which are independent lines of development that allow you to work on different features or bug fixes simultaneously.

In this lesson, we will explore the concept of branching in Git and learn how to create, switch between, and delete branches. We will also delve into the process of merging branches, which allows us to combine the changes made in one branch with another. Understanding branching and merging is crucial for effective collaboration and maintaining a clean and organized codebase.

Whether you are a beginner or have some experience with Git, this lesson will provide you with a solid foundation in branching and merging. So let's dive in and discover the power of Git's branching and merging capabilities!

Git is a powerful version control system that allows developers to work on different versions of a project simultaneously. One of the key features of Git is its ability to create branches, which are independent lines of development. This allows developers to work on different features or bug fixes without interfering with each other's work. Once the changes on a branch are complete, they can be merged back into the main branch,

combining the changes from multiple branches into a single cohesive version.

Branching

Branching in Git is a lightweight and efficient process. It allows developers to create a new branch based on an existing branch, typically the main branch (often called "master" or "main"). This new branch is an exact copy of the existing branch, including all the files and commit history.

To create a new branch, you can use the `git branch` command followed by the desired branch name. For example, to create a branch named "feature-branch", you would run:

```
$ git branch feature-branch
```

Once the branch is created, you can switch to it using the `git checkout` command:

```
$ git checkout feature-branch
```

Now you are on the new branch and can start making changes to the codebase without affecting the main branch.

Merging

Merging is the process of combining changes from one branch into another. It allows developers to integrate their work back into the main branch or merge changes from one branch to another.

To merge changes from one branch into another, you can use the `git merge` command. For example, to merge the changes from the "feature-branch" into the main branch, you would run:

```
$ git checkout main
```

```
$ git merge feature-branch
```

Git will automatically merge the changes from the "feature-branch" into the main branch, combining the commit history and resolving any conflicts that may arise.

It's important to note that conflicts can occur when Git is unable to automatically merge changes. Conflicts usually happen when two branches have made conflicting changes to the same file or lines of code. In such cases, Git will mark the conflicting areas in the affected files and prompt you to manually resolve the conflicts.

Conclusion

Branching and merging are essential concepts in Git that enable developers to work on different versions of a project simultaneously and combine their changes seamlessly. By creating branches, developers can work on new features or bug fixes independently, without interfering with each other's work. Merging allows the changes from different branches to be integrated back into the main branch, creating a unified and up-to-date codebase. Understanding and effectively using branching and merging in Git can greatly enhance collaboration and streamline the development process.

Resources and Exercises for Further Studies.

1. [WEB] Learn all about Git Branching through this awesome website - https://learngitbranching.js.org