GitHub Flow and Computational Thinking

Contents

Prepare for Lab

1. Create an account on GitHub.com. learn more about GitHub accounts (5 minutes)
2. Accept this assignment to create your first GitHub repository and preview the basic vocabulary that we will use during the lab. (15 minutes)

Lab Agenda

Part 1

1. Discussion and Goals
2. Initialize Repos with an assignment
3. Vocabulary Review (quiz or prismia)
4. Change index.md and view impact on the site
5. Examine parts and discuss the structure
6. Create an issue
7. Close issue by updating the site
8. Create Pull Request on a partner's repository
9. Review the pull request, update and merge
10. History
11. Git Blame

Part 2

1. Review and AMA
2. Use git on a command line
3. Clone
4. make a change
5. add
6. commit
7. push

Next Steps with GitHub Flow

- Create your profile README. Let the world know a little bit more about you! What are you interested in learning? What are you working on? What's your favorite hobby? Learn more about creating your profile README in the document, "Managing Your Profile README".
- Go to your user dashboard and create a new repository. Experiment with the features within that repository to familiarize yourself with them.

:octocat: GitHub Flow

The GitHub flow is a lightweight workflow that allows you to experiment and collaborate on your projects easily, without the risk of losing your previous work. We will learn this in lab, to prepare, here is a preview of the vocabulary.

Repositories

A repository is where your project work happens–think of it as your project folder. It contains all of your project's files and the history of each one. You can work within a repository alone or invite others to collaborate with you on those files.

Cloning

When a repository is created with GitHub, it's stored remotely in the ☁️. You can clone a repository to create a local copy on your computer and then use Git to sync the two. This makes it easier to fix issues, add or remove files, and push larger commits. You can also use the editing tool of your choice as opposed to the GitHub UI. Cloning a repository also pulls down all the repository data that GitHub has at that point in time, including all versions of every file and folder for the project! This can be helpful if you experiment with your project and then realize you liked a previous version more. To learn more about cloning, read "Cloning a Repository".

Committing and pushing

Committing and **pushing** are how you can add the changes you made on your local machine to the remote repository in GitHub. That way your instructor and/or teammates can see your latest work when you're ready to share it. You can make a commit when you have made changes to your project that you want to "checkpoint." You can also add a helpful **commit message** to remind yourself or your teammates what work you did (e.g. "Added a README with information about our project").

Once you have a commit or multiple commits that you're ready to add to your repository, you can use the push command to add those changes to your remote repository. Committing and pushing may feel new at first, but we promise you'll get used to it 🙂

Instructor Guide

Preparation

Activity

1. Discussion and Goals *(10 min)*
    - how we think in CS
    - role of GitHub
    - CS is more than just writing code, it's using tools
2. Initialize Repos with an assignment *(5 min)*
    - confirm that all students have the repo
    - parts of the page on GitHub, basic navigation
3. Vocabulary Review (quiz or prismia) *(10min)*

    - repo
    - issue
    - pull request
    - commit
4. Change index.md and view impact on the site *(5min)*
    - also check settings and turn on rendering
    - view in browser
    - view gh pages branch
5. Examine parts and discuss the structure *(15 min)*
    - notice parts of html
    - what was in repo? what came from elsewhere?
    - why is htat good?
6. Create an issue *(5min)*
    - programming is teamwork
7. Close issue by updating the site *(10 min)*
    - github links the work with the reason for the work
    - why do we want this?
8. Create Pull Request on a partner's repository *(20 min)*
    - anyone can help improve open source content
9. Review the pull request, update and merge *(15 min)*
    - code review is essential
10. Commit History *(5 min)*
    - we can see a summary of what has changed
    - why might this be good
11. Git Blame *(5 min)*
    - we can see who changed what when
    - why might this be good
12. Review by adding more pages with issues and a PR *(remaining time if available)*

    ```
    ```{toctree}
    :maxdepth: 2
    :hidden:
    pagename
    ```
    ```

13. Feedback *(5min)*
    - one thing you learned
    - one htink you have a question about

Key Learning Outcomes

Learning Outcomes

1. Define and use correctly key GitHub terminology
2. Identify patterns in CS thinking: modularity, reuse
3. Idenitfy why version control is important
4. Identify advantages to GitHub flow for collaboration
5. Apply problem solving skills to determine what a system does by observing how different inputs change the outputs

Key points

1. GitHub is a useful tool for collaboration, version control, backup and webhosting
2. Computer Science is more than writing code from scratch
3. Template engines allow for well structured websites that are easier to maintain, with less repeated code.