GIT FOR TEAMS

Objectives

- Understand Git collaboration best practices
- Deal with merge conflicts
- Use feature/branch workflow in a team
- Rebase and merge branches

Pirate Code

ACTIVITY: COLLABORATE ON A FILE (15MIN)

One person per table create a repo with html, css, and js files
Add everyone at table as collaborators
Everyone add a section about themselves
Merge conflicts, galore!

WHAT DOES A MERGE CONFLICT LOOK LIKE? HOW DO YOU RESOLVE THEM?

```
<<<<<<< HEAD
color: red
=======
color: blue
>>>>>>>
```

HARD RULES

- Do NOT commit directly to master EVER
- Master should ALWAYS be in a working state
- Create feature branches for tasks (good: book-crud, bad:craig-branch)
- Rebase branches BEFORE merging to master

OTHER GUIDELINES

- Use good branch names -- no spaces, no capital letters
- Merge conflicts should be handled frequently.
- Merge conflicts should be handled on the feature branch.
- Communicate and isolate who is working in different sections of the code.

DEMO

» feature branch | rebase | merge

FEATURE/BRANCH WORKFLOW (WITH REMOTE REPOSITORY)

1. git checkout -b <branch_name> (this creates a new branch and checks out to that same branch)
2. do work
3. commit (add, commit)
4. git checkout master
5. git pull origin master
6. git checkout <branch_name>
7. git rebase master
8. fix conflicts! (tells you on command line where to look in editor)
9. git push origin <branch_name>
10. git checkout master
11. git merge <branch-name>
12. git push origin master
13. Tell teammates to git pull origin master

REPEAT ACTIVITY (15 MIN)

» User feature/branch work flow | rebase | merge

DISCUSSION

RESEARCH A GIT COMMAND YOU HAVEN'T USED BEFORE (5 MIN)

Git-SCM - git bible

Atlassian Git Tutorials

Git Flow

Git Team Exercises