Full Stack Web development

Lecture 5: GitHub, IDEs, Linters, and Troubleshooting

Today:

- Git
- Integrated Development Environments (IDEs)
- Linters
- Troubleshooting HTML and CSS using Browser Web Development Tools

Git

- Version Control System
  - What does that mean?
  - Why do we need it?
    - Tracks changes by one or more users
  - Is Git the only option?
    - SVN
    - Mercurial
  - What is GitHub and how is it different from Git?
    - SaaS (software as a service)
    - Free for individuals
    - Free for open-source projects

Common Terms:

- Repository – collection of files and folders that make up a single project
- Branch – an active line of development. A repository can have many branches
- Commit – a single set of code changes that usually are tied to a feature or bug fix.
- Checkout – switching to a specific branch
- Clone – copying a branch of a repository to a computer or remote server.

Git – Getting Started - Option 1

- Install Git on your computer (https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)
- Create an account at GitHub (https://github.com/join)
- Create a new directory on your PC called ECC-Webdev-Task1*
- Create a new repository on GitHub (best to keep the same name but not required)
- Initialize the repository on your computer

```
echo "# ECC-WebDev-Task1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/<USER>/ECC-WebDev-Task1.git
git push -u origin master
```

*Repository name is just a suggestion, name it whatever you like

Git – Getting Started – Option 2

- Install Git on your computer (https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)
- Create an account at GitHub (https://github.com/join)
- Create a new repository on GitHub called ECC-Webdev-Task1*
  - Be sure to add a README
- Clone the new repository onto your PC

```
git clone https://github.com/<USER>/ECC-WebDev-Task1.git
```

*Repository name is just a suggestion, name it whatever you like

Git – Making a Commit

- Create a new file, or copy one of our previous files into your folder where you setup the Git repository

`git status` –Shows you if there are any changes that need to be added.

`git add FILENAME` –adds the file listed to be ready to commit

`git commit -m ”add new file"` – creates the commit

`git push -u origin master` – push change to remote branch (GitHub)

Git – A Word of Caution

These two lines:

`git add FILENAME` –adds the file listed to be ready to commit
`git commit -m ”add new file"` –creates the commit

Can be combined into:

`git commit -am “add new file”`

The -a flag means add <u>ALL</u> changed files to the commit. Use this with caution as it can cause you to commit files you do not intend to!

Git – Pulling In Changes

- If someone else is working on the same codebase, or you are working from a different computer, you always want to pull down the most recent changes before making your own.

```
git pull -p
```

This pulls down changes from the remote branch you are working on.

Note on `-p`: this flag tells git to clean up any dead branches; branches that only exist locally and not on the remote

Git – Creating a new branch

- If there are a lot of changes you want to make to the code, but you don’t want them to go into master until they are built and tested (this is the correct way.. hint.. hint..), you create a new branch.

```
git checkout -b branch-name
```

This creates a new branch based on the current branch that you can commit code to.

Git – Creating a new branch cont.

- When to create a new branch
  - Adding a feature
  - Making a large change that will need to be tested
  - Fixing/Testing a bug that can upset application stability
- Branch naming
  - Working on an issue ticket? Put it in the branch name
  - **Never** use something generic like new-branch or branch-1
  - Branch prefixes:
    - bugfix/<ticket-number>-<description-of-bug-or-title-of-ticket>
    - feature/<ticket-number>-<feature-name>
    - hotfix/<ticket-number>-<title-of-ticket>

Git – Switching branches

- Sometimes it makes sense to have different branches, but you want to switch between them.

```
git checkout dev
git checkout main
```

The first command switches to the dev branch, the second command switches back to main.

Git – Merging branches

- The easiest way to merge branches is to do it on the command line

Merging dev branch INTO master:

```
git checkout master
git merge dev
git push
```

The first command switches to the master branch, the second command merges dev into the master branch and the third command pushes the merge to GitHub.

Git – Merging branches with Pull Requests

- Another way, the *preferred* method is to create a **Pull Request**.
- A pull request or PR allows for code review and clean merging
- Most repository sites have this option (if they don’t, go somewhere else)
- Cool! But Why?!
  - **Accountability**: everyone checks for errors (code review)
  - **Security**: dangerous vulnerabilities should be caught in the branch not in the main trunk which might sneak into production
  - **Stability**: new features or fixes can be tested and tried on a branch BEFORE being merged into main, thereby protecting from possibly unstable code sneaking into a different branch

Git – Merging branches with Pull Requests

- Got it!! So how?!?

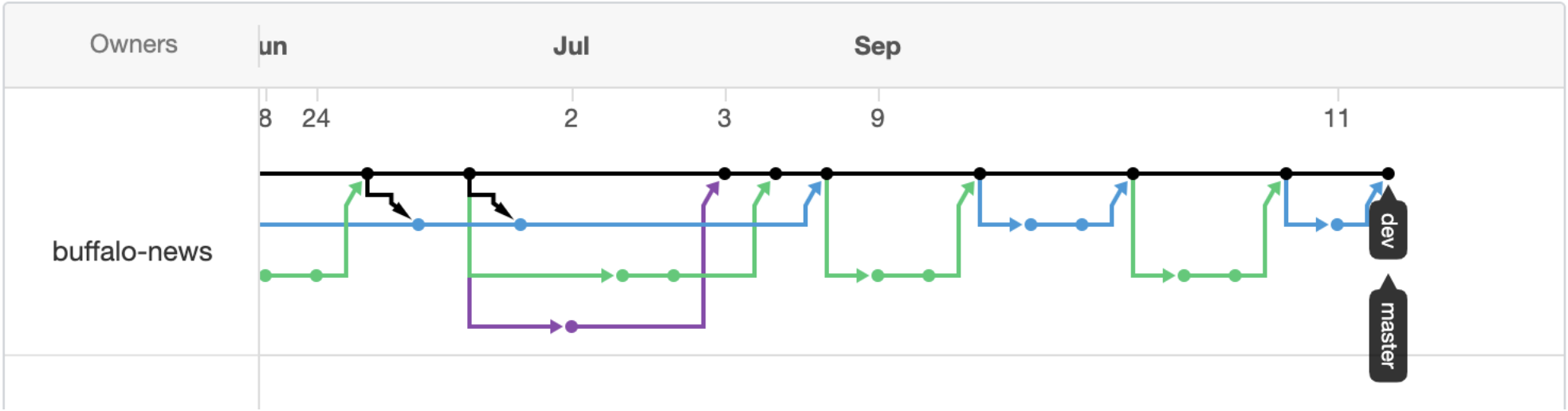
1) Create a branch
2) Make changes
3) Push branch to remote
4) Site dependent but *usually* either the git message returned from the remote will provide a link OR open the remote repo site (github.com/user/reponame) navigate to either the branch or pull request section and create a new one.

- DEMO TIME!!

Git Workflow

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

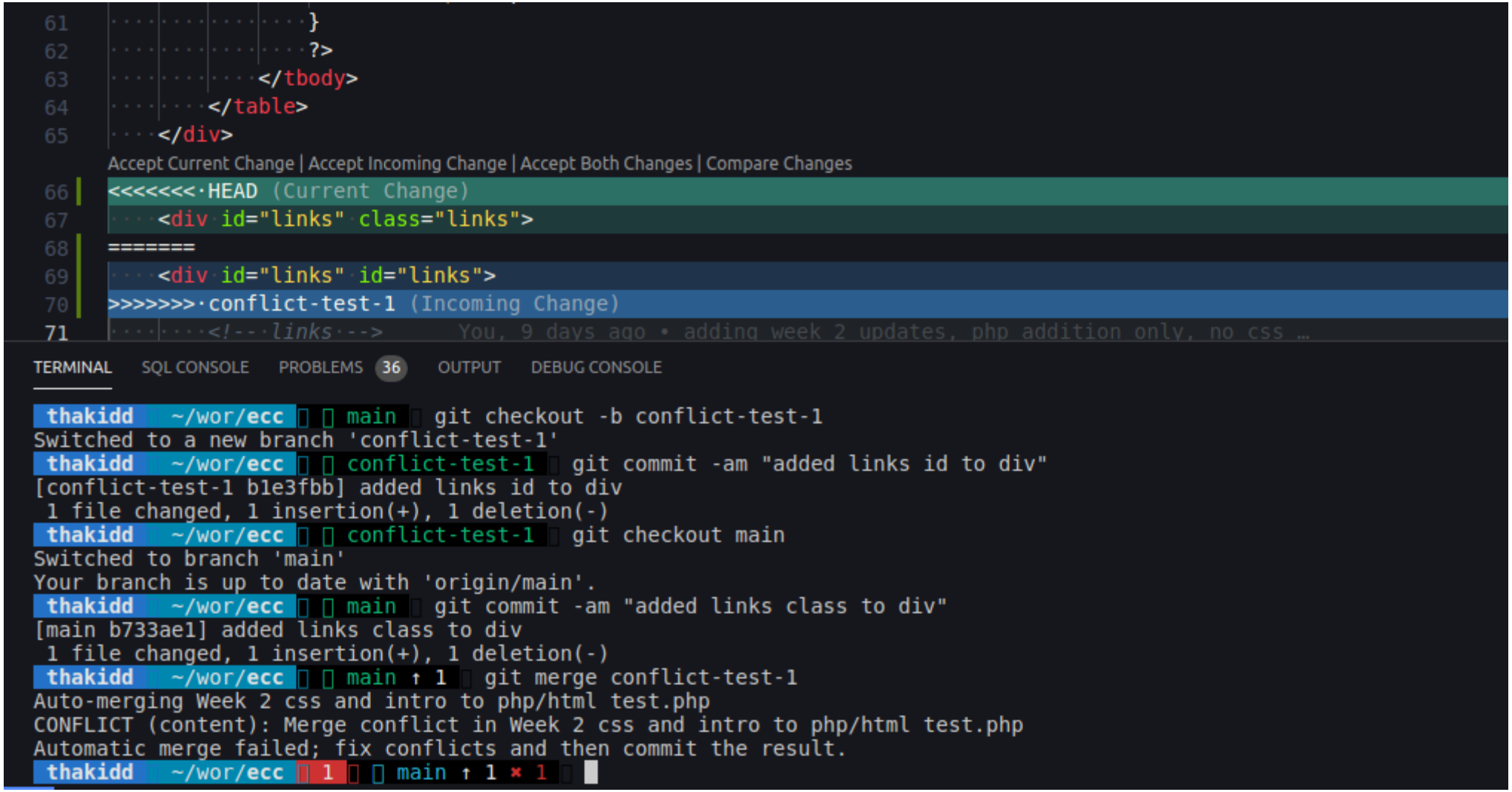


Git – Merge Conflicts

- If you change the same file on two different branches, and then you try to push your commit to GitHub, you create a merge conflict that git cannot resolve.

```
dhcp-10-101-250-155:merge_conflict SUYEONSON$ git push
To git@github.com:suyeonson/merge_conflict.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:suyeonson/merge_conflict.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
dhcp-10-101-250-155:merge_conflict SUYEONSON$
```

Git – Merge Conflicts



Git – Fixing Merge Conflicts

- Finding and fixing merge conflicts
  1) `git status`
  2) Find the file(s) labeled ‘both modified’
  3) Search for <<<<<<< HEAD
  4) Update with correct changes
  5) `git add FILENAME`
  6) `git merge --continue`
  7) Push the changes

- DEMO

Integrated Development Environments (IDEs)

- An integrated development environment is a tool used in development that helps you to program more efficiently and to help find errors more quickly.
- In this class we use VSCode, however today (and only today) I will talk about and explain some others, as well as speak to the differences.

VSCode

- Demo
  - Plugins
  - Packages
  - Projects
  - Terminal
  - Git Integration

Other IDEs

- Eclipse
  - Open source
  - Popular for many different languages
  - Runs on java so it can eat up ram

Android Studio

  - Useful for building Android apps

- Xcode
  - Popular for developing IOS apps
- Visual Studio
  - Useful for Microsoft development
  - Paid version of VSCode.
- IntelliJ
  - Java IDE
- PyCharm
  - Python IDE
- PHPStorm
  - PHP IDE
- Atom
  - Basic fast text editor
  - Created by GitHub
- Notepad++
  - Similar to Atom

LINTERS

- A linter is a special IDE plugin that helps analyze source code in real-time and find syntax and stylistic bugs.
- Extremely useful on teams in order to ensure coding standards
- Coding Standards
  - Tabs vs Spaces
  - Correct indenting of tags
  - Capitalization and case structure
- DEMO VSCode Linters

Troubleshooting – Developer Tools

Troubleshooting code is one of the hardest jobs we face as developers. We spend hours writing code and then when we test things don’t work correctly. Browser based developer tools are key to that troubleshooting process and can save you hours of work down the road.

- Demo - Troubleshooting on your own, Guess/Test method.

- Chrome Developer Tools Demo

- Firefox has a very similar option also