Class 30 - Deep Dive Into GitHub & Version Control

Class 30 Course Content

Preparation

GOALS

By the end of this lesson, you will be able to:

1. **Create new Repositories via the Command Line**
2. **Create and work on branches**
3. **Work through and Resolve Merge Conflicts**

CONCEPTS

- **Branch:** A *Branch* is a copy of a code line, managed in a version control system that helps teams work parallel on different features simultaneously.
- **Pull Request:**A *Pull Request* is when a developer asks for changes on a branch to be committed to an external repository by pushing their code up for review, hopefully, to be accepted and merged into the repositories main source code.
- **Merge Conflict:** A *Merge Conflict* is when multiple developers make concurrent changes around the same lines of code and create pull requests. The merge conflict must be resolved by choosing to be accepted in the area where the conflicting code occurs.

Walkthrough

STEP 1: CREATE THE PERFECT COMMIT

Aim: Craft the perfect commit by adding the right changes and composing a good message

| *https://github.com/new* |

- [ ] **Create a new GitHub Repository**
  - Navigate to GitHub - Create a New Repo
  - Enter in the Repo name and click the green "Create repository" button
  - copy the commands for "create a new repository on the command line"

```
# Copy these commands or save them on another screen
echo "# test2" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
```

```
git remote add origin https://github.com/WilderDev/test2.git
git push -u origin main
```

🔻

| *Terminal* |

- [ ] **Create a New Project via Command Line & Initiate Git**
  - Navigate to a good spot on your computer to create a new coding project
  - Create a folder and run the commands we copied earlier
  - Inside the folder, create an HTML, CSS, and JavaScript file
  - Check the git status

```
#Navigate to the proper folder on your machine
cd Desktop/Coding_Projects
mkdir Codefi_Git_Practice
cd Codefi_Git_Practice

# Initialize the Repository Globally & Locally
echo "# test2" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/WilderDev/test2.git
git push -u origin main

# Create New Files
touch index.html
touch styles.css
touch main.js

git status
```

🔻

- [ ] **Commit the New Files**

```
git add -A
git status
git commit -m "INIT: Main Files (HTML, CSS, JS)"

git push -u origin main
```

🔻

- [ ] **Add the boilerplate HTML & a few simple CSS rules**

```
code .
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <link rel="stylesheet" href="./styles.css" />
    <title>Home Page</title>
  </head>
  <body>
    <h1>Home Page</h1>
  </body>
</html>
```

```
h1 {
  font-size: 4rem;
}
```

🔻

- ☐ **Check the current state and push the new changes**

```
git status
git add index.html
git status

git add -p styles.css
y

git commit -m "ADD: Home page HTML & CSS"
git push -u origin main
```

STEP 2: GIT BRANCHING

Aim: Master the essentials and concepts around git branches

| *Terminal* |

- ☐ **Create a New Feature Branch**

```
git checkout -b FEATURE_About-Page_Wilder
```

🔻

- [ ] **Create the About Page**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <link rel="stylesheet" href="./styles.css" />
    <title>About Page</title>
  </head>
  <body>
    <h1>About</h1>
  </body>
</html>
```

🔻

| *Terminal* |

- [ ] **Commit your new changes to the current working branch**

```
git add -A
git commit -m "ADD: About Page HTML"
```

STEP 3: PULL REQUESTS

Aim: Create a pull request and merge the changes from the About Page Branch into the Main Branch

| *Terminal* |

- [ ] **Publish the feature branch to GitHub**

```
git push -u origin FEATURE_About-Page_Wilder
```

🔻

| *https://www.GitHub.com/yourName/yourRepository* |

- [ ] **Create a pull request and accept the changes**

- On the GitHub Repository for this project, create a pull request with the new branch
- Write the pr message
- Accept changes and merge them into the main branch

STEP 4: MERGE CONFLICTS

Aim: Learn about and release the fear of the inevitable merge conflicts

NOTE: You will need two people to do this section successfully.

| *PERSON 2: https://www.GitHub.com/yourName/yourRepository && index.html* |

- [ ] **Have a code coach or another coder fork the repository you have been creating**
  - After forking, have them create a new feature branch called `FEATURE_HP-Main-Content_LastName`
  - They should make changes on the main page HTML

```
<body>
  <main>
    <h1>Merge Conflicts Demo</h1>
    <h2>Hello World from Person 2</h2>
  </main>
</body>
```

🔻

| *PERSON 1: Terminal && index.html* |

- [ ] **Create another new feature branch w/ conflicting changes**
  - Create a branch called `FEATURE_HP-Blog-Post_Wilder`
  - Make changes to the Home Page HTML that will conflict with what person 2 is coding

```
<body>
  <article>
    <h2>Sample Blog Post</h2>
  </article>
</body>
```

🔻

| *PERSON 2: Terminal* |

- [ ] **Publish Branch and Merge Changes into Main**
  - Use the terminal to add your changes and publish them to the GitHub Repository
  - On GitHub, Create the pull request and merge the changes into the main branch

```
git add -A
git commit -m "ADD: Home Page Main Content"
git push -u origin FEATURE_HP-Main-Content_LastName
```

🔻

| *PERSON 1: GitHub Desktop* |

- [ ] **Walkthrough GitHub Desktop Branch Merge to Main Flow**
  - Navigate to the main branch in GH Desktop
  - Pull down any new changes
  - Click on the branches dropdown, and at the very bottom, there is a button to merge a branch into main
  - Choose the current Feature Branch you are working on
  - Navigate to the branch you are working on and solve the merge conflicts
  - Commit your changes and create a pull request and merge into the main branch

---

Additional Notes

EXERCISES / GAMES

Possibly try pairing students up and working through the GitHub branch flow.

Otherwise, for the remainder of the class, try and complete as many lessons in this tutorial as possible!

Learn Git Branching *(Interactive Tutorial)*

RESOURCES

Using Git Branches *(Articles)*

Git Branching and Merging: A Step-By-Step Guide *(Article)*

Learn Git in 15 Minutes *(Video)*

Git & GitHub Tutorial For Beginners *(Video Series)*