COMP 1950

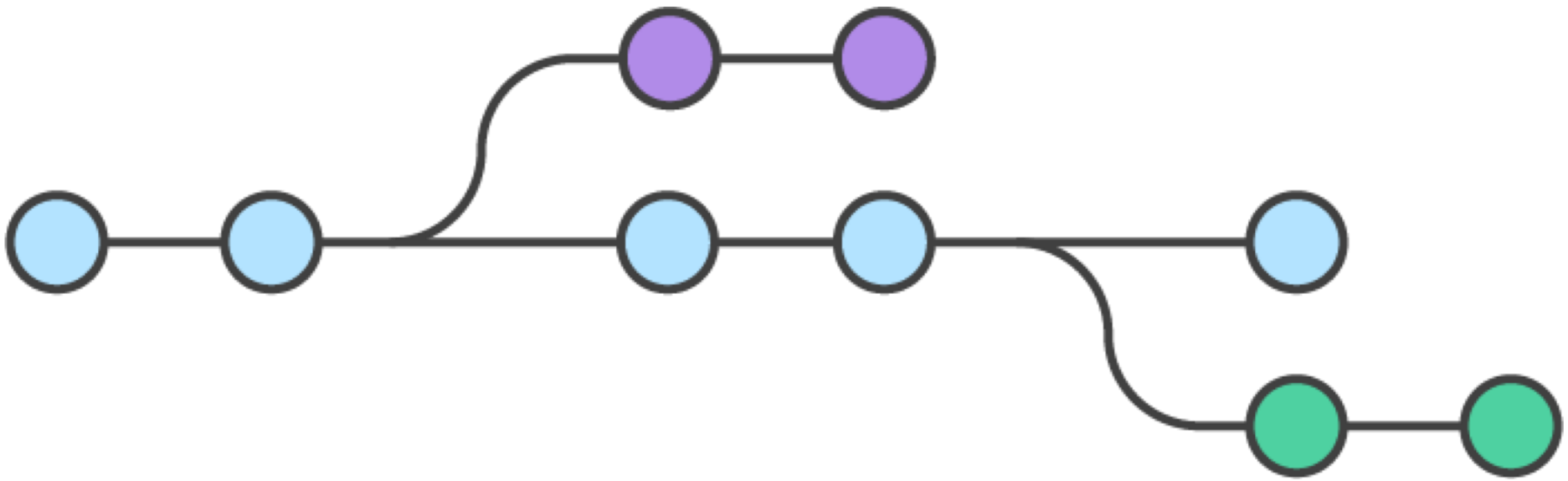
Web Development and Design 2

Day 09

Agenda

- Final Project
- Version control, Git and GitHub
- Final Project Lab time

Version Control



What is Version Control?

- Version control is the management of changes to documents, graphic files, computer programs, web development files and other collections of information

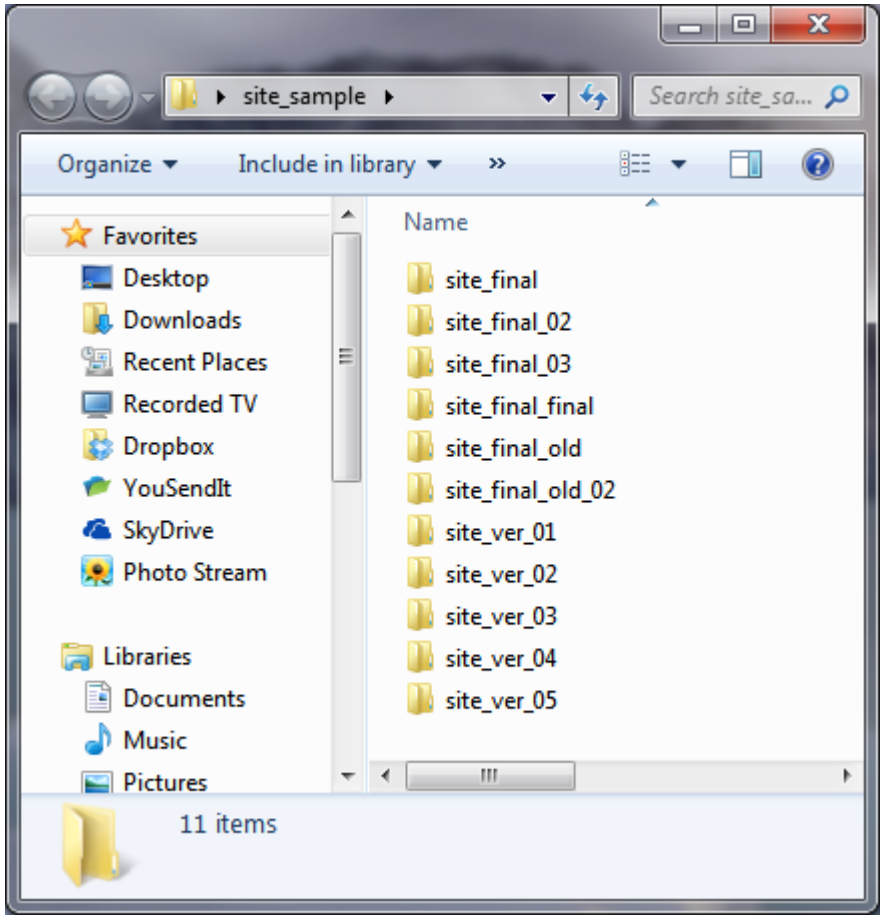
Why Should You Use Version Control?

- Allows you create and maintain separate versions of a digital project
- Allows you to go back to previous versions of a project
- Allows you try out new features with out affecting the main program or web site

You may already be using version control

- If you have worked on a digital project of any kind you may have been performing your own manual forms of version control by saving files with different version file names
- Manually version controlling your files, while better than no version control, can be error prone.
- You may accidentally overwrite a previous version, or you loose track of which version is the main version
- With the multitude of files required in web development, manually version controlling your project quickly becomes tedious

Manual Version Control



Automating Version Control

- Version control is tedious, so let a computer keep track of your versions for you
- Many version control systems to choose from. Below are just a few of many examples:
  - SVN
  - TFS
  - Git
- For this class we will use the Git version control system
  - Git is the most popular form of version control used in web development

Git Version Control System

- Git is an open source (free) version control system
- Created by Linus Torvalds in 2005 for use on the development of Linux

Installing Git

- BCIT computers for this class already have Git installed
- For Windows users, you have many options:
  - Simplest option is to visit the link below:
    - https://gitforwindows.org/
    - Click the download button, launch the installer and follow the install instructions
- For Mac users, visit the link below and the Git software should start to download (if it does not, click the manual download link)
  - https://git-scm.com/download/mac
  - Once downloaded, launch the installer and follow the install instructions

Git on the Command Line vs a GUI

- We will learn how to use Git on the command line. Which simply means entering Git text commands into a Terminal window (Mac) or a Command Prompt window (Windows)
- Git can also be used via a GUI (Graphical User Interface). Instead of entering text commands you simple click on buttons to perform common Git tasks
- Some developers prefer the command line method and some prefer a GUI. The choice is yours.
- Even if you do prefer a GUI, it is still a good idea to know the basic Git text commands in case you have to go to the terminal to fix issues

Git GUI Options

- BCIT computers already have the GitHub Desktop software installed which can be used to use Git
- For Windows and Mac users, a good free option is the GitHub Desktop software.
  - You can download the free GitHub software at the link below:
    - https://desktop.github.com/
- FYI
  - If you install the GitHub desktop software, it will automatically install Git for you as well, so no need to also install Git separately
- We will explore how GitHub relates to Git in the upcoming slides

Setting up Git

- Git requires very little setup once installed
- The following two global commands only need to be set once per computer.
- Once these settings are set, they will apply to all your future Git projects on your current computer
- BCIT computers get scrubbed, so you will need to re-enter these commands on BCIT computers for each class

Git Global Settings – user.name

- Set your name that Git will use to attach to your commits (more on what commits are on upcoming slides)
- Enter the following command in your Terminal window (Windows user's launch Git BASH or PowerShell)
- FYI
  - Don't include the "$". The "$" represents your command prompt

```
$ git --global user.name "Michael Whyte"
```

- Replace my name with your name

Git Global Settings – user.email

- Set your email address that Git will use to attach to your commits (more on what commits are on upcoming slides)
- Enter the following command in your Terminal window (Windows user's launch Git BASH or PowerShell)

```
$ git --global user.email "info@mwhyte.ca"
```

- Replace my email address with your email address

The .gitignore File

- A .gitignore file is a plain text file which lists out files and folders that you do not want Git to track and version control
- Files that you probably do not want Git to keep track of include:
  - System files created by the OS
  - node_modules folder
  - Computer generated files that are generated through a compile program
    - This includes compiled CSS files generated by SASS
    - Since compiled files can easily be created by running your compiler, their really is no need to keep track of these files. The underlying source files that the compiler uses to generate the compiled files should be version controlled

Creating a .gitignore file

- To create a “.gitignore” file simply open your project in your text editor of choice and create a new file and save it as “.gitignore” in the root of your project
- Note to Mac users:
  - Macs hide files that start with “.” by default.
  - To see your “.gitignore” file in your project folder in Finder, simply press “Cmd+Shift+.”. To re-hide hidden files press “Cmd+Shift+.” again

Editing a .gitignore file

- To tell Git to ignore files or folders in your project, simply list them out in your “.gitignore” file
  - To ignore a file:
    - file-to-ignore.txt
  - To ignore a folder:
    - folder-name/
- Use “#” in front of a line you wish to make a comment
- Use “*” as a wild card.
  - For example to ignore all “.txt” files located in a folder called “info” which is located in the root of our project, we would add this line to our “.gitignore” file:
    - foo/*.txt

A Reference “.gitignore” File

- Click the link below to download a reference “.gitignore” file that works well for most web development projects
- Note:
  - The “.gitignore” file ignores a “styles” folder and all ”.css” files inside it. This is useful if you are using SASS. If you are not using SASS simply delete or comment out the lines that refer to the “styles” folder and the “.css”
- Link to a reference “.gitignore” file:
  - https://goo.gl/Af9RjL

Initialize a Project to Use Git

- To have Git start to keep track of your project, simply navigate to your project folder in Terminal and enter the following command:

```
$ git init
```

- After entering this command, Git will start tracking changes to all files inside your project folder

Checking the Status of a Git Repository

- A handy Git command is ”git status”
- Running “git status” in the Terminal will let you know the current status of the Git repository
- The “git status” command let will you know if any untracked files have been added to the repo or if any changes made to the files must be committed to the repository
- Run the following command in your project folder to see the status of your Git repository:

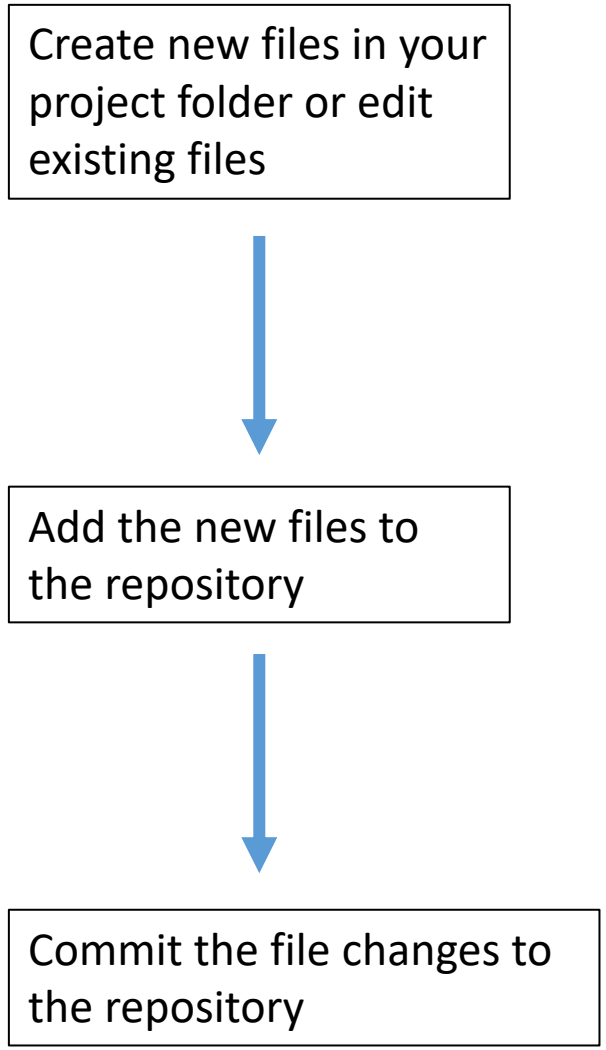
```
$ git status
```

Git does not track empty folders

- If you are starting a new project, you will quickly notice that Git does not track empty folders. This is by design, as Git really only tracks files and their changes not folders
- To have Git track folders, you must add a file to every folder you want Git to track

How Git Works

- To have Git keep track of changes a file must first be “added” to the repository
- After a file or group of files have been added to the repository there changes must be committed to Git



Add a single file to Git

- Create the file in your project folder
- Run “git status” in the Terminal window
- After running “git status” you will see Git telling you that your newly added file needs to be “added” to the repository in order for Git to track the file
- Run the following Git command to have Git add a file to the repository

```
$ git add [the name of your file]
```

Adding Multiple Files

- If you add or create multiple files at once, it would be tedious to add them one by one with “git add”
- Git provides a shortcut to add multiple files
- Run the following command to add all new files

```
$ git add .
```

Committing your Changes

- Once you have added your files, you must “commit” the changes you have made to the files and the repository to Git
- Once changes have been committed you can return to them at a later time by checking them out via Git
- You must commit your changes to have them added to the version history of the repository
- Commits must contain a message which describes the changes you have made
  - Good messages are important, especially if you need to revert to a previous version of your project

The Git Commit Command

- To commit your changes, enter the following command into the Terminal window

```
$ git commit –m “Describe your changes”
```

Combining Add and Commit

- If your files have already been added by Git in a previous command, and you make further changes to the file(s) you can combine the ”git add” and “git commit” commands into a single command
- After making changes to an already tracked file(s), enter the following command to both add it and commit the changes

```
$ git commit –a –m “Describe your changes”
```

More Git Commands

- The Git commands ”status”, “add”, “commit” are all you need most of the time when working on local projects
- We will go over many of the more common Git commands in class today
- In the day’s folder I have included a “Git” cheat sheet with all the common Git commands
  - Look for the file called “git-cheat-sheet.pdf”

Git Learning Resources

- Want to learn more?
- Try the following resources:
  - Book
    - Git for Humans
      - https://abookapart.com/products/git-for-humans
  - Web Site
    - Git Tower Learn
      - https://www.git-tower.com/learn/

GitHub

- Git and GitHub are two separate things
- Git is a version control system
- GitHub is a code hosting service that uses Git to keep track of changes made to a GitHub repo (project)
- GitHub stores a copy of your project in the cloud, so even if you loose your local copy you can restore from the copy on GitHub
- GitHub allows multiple developers to easily work on the same repository (project) at the same time

Create a Free GitHub Account

- GitHub has both free and paid accounts
- Free accounts allow you to create unlimited public repositories
  - Public repositories can be viewed (but not edited) by anyone with a browser and an internet connection
  - To edit a public repository you must be the project owner or have invited another GitHub user(s) as a collaborator of the project
  - Public repositories are fine for open source projects and class projects for education, but may not be appropriate for client work
- Paid accounts allow you to create private repos that can only be viewed and edited by team members
- For this class, a free account is all you need

Create a GitHub Account

- To create a free GitHub account visit the link below, and sign up for a free account
  - https://github.com/

Create a GitHub Repo

- You can create a local Git Repo than “push” that up to GitHub, but an easier way is to first create the repo on GitHub and then “clone” it down to your local computer
- To create a GitHub repo, simply log into GitHub on the web and click the “Start a Project” button

Clone a GitHub Repo

- To clone a repo, means to make a copy with all the version history of the project to your local machine
- To clone a GitHub repo to your local machine do the following:
  - Visit your GitHub repo on the web
  - Click the green “Clone or download” button and then click the “Copy to Clipboard” button located beside the path to your remote repos Git file
  - On your local computer open the Terminal and change the directory to a location on your computer where you would like to store your project
  - When the above steps are done, enter the following command into the terminal to clone the remote GitHub repo to your local computer

```
$ git clone [path-to-your-github-repo-git-file]
```

Keeping a Local and Remote Repo In-sync

- Changes made to a cloned repo are not automatically synced to the remote repo on GitHub
- Keep your remote repo in-sync with your local copy you have to “push” your local changes to the remote repo
- To push your changes to GitHub, make sure your have committed your changes locally
- Once all local changes have been committed, enter the following command into the Terminal and enter any required GitHub username and passwords

```
$ git push
```

Keeping a Local and Remote Repo In-sync

- If you are working on a GitHub repo as part of a team, than changes that other team members make and push up to GitHub will not automatically be reflected on your local copy of the repo
- To get remote changes to your local copy, enter the following command into the Terminal

```
$ git pull
```