Front-End Tools & Portfolio

Git & GitHub, Visual Studio Code, Emmet, & Portfolio Development

Learn by doing step by step exercises.

Includes downloadable class files that work on Mac & PC.

EDITION 1.1

noble desktop

Published by:
Noble Desktop
185 Madison Ave, 3rd Floor
New York, NY 10016
nobledesktop.com

Table of Contents

Table of Contents

Downloading the Class Files

Thank You for Purchasing a Noble Desktop Course Workbook!
These instructions tell you how to install the class files you'll need to go through the exercises in this workbook.

Downloading & Installing Class Files

1. Navigate to the **Desktop**.
2. Create a **new folder** called **Class Files** (this is where you'll put the files after they have been downloaded).
3. Go to **nobledesktop.com/download**
4. Enter the code **fe-2207-20**
5. If you haven't already, click **Start Download**.
6. After the **.zip** file has finished downloading, be sure to unzip the file if it hasn't been done for you. You should end up with a **Front-End Tools and Portfolio Class** folder.
7. Drag the downloaded folder into the **Class Files** folder you just made. These are the files you will use while going through the workbook.
8. If you still have the downloaded .zip file, you can delete that. That's it! Enjoy.

About Git

What is Git?

Git is the most commonly used version control system. Git tracks the changes you make to files, so you have a record of what has been done, and you can revert back to specific versions should you ever need to. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

So regardless of whether you write code that only you will see, or work as part of a team, Git will be useful for you.

Git is software that runs locally. Your files and their history are stored on your computer. You can also use online hosts (such as **GitHub**) to store a copy of the files and their revision history. Having a centrally located place where you can upload your changes and download changes from others, enables you to collaborate more easily with other developers. Git can automatically merge the changes, so multiple people can even work on different parts of the same file and later merge those changes without losing each other's work!

Git was created by Linus Torvalds in 2005 for the development of Linux (a clone of the Unix operating system). Linux and Unix are commonly used to run web servers, and macOS is based on Unix.

Ways to Use Git: Command Line vs. Desktop GUI Apps

Git is software that you can access via a command line (terminal), or a desktop app that has a GUI (graphical user interface).

In this book we'll cover how to use Git in **Visual Studio Code**, a popular free code editor we use in our **web development classes**. This makes Git much easier to learn for beginners.

If you later want to learn the command line version of Git the workflow is the same, but you'd need to learn the specific commands to type into the terminal.

Installing Visual Studio Code

Visual Studio Code is a great code free editor for Mac and Windows and it has built-in Git features so it's what we'll be using in this book. If you already have it, there's nothing else you need to do. If you don't have it, here's how to get it:

1. Visit **code.visualstudio.com** and download **Visual Studio Code**.
2. To install the app:
   - Mac users: Drag the downloaded app into your **Applications** folder.
   - Windows users: Run the downloaded installer.
     During installation check on the **Add "Open with Code" …** options.

Installing Git

Visual Studio Code integrates with the Git software installed on your computer. So to use Git, you must first install it. Installing Git is different on Mac and Windows, so follow the appropriate instructions below depending on your operating system.

Mac: Download & Install Git

1. Go to **Applications > Utilities** and open **Terminal.app**.
2. In the Terminal window, type the following and press **Return**:

   ```
   git --version
   ```

3. If it tells you a git version number, you already have Git installed. If you do not have it installed, you'll be prompted to download and install the Xcode developer tools (which include Git). Proceed with the download/install process until it's done.

Windows: Download & Install Git

1. Open a web browser and go to **git-scm.com/download**
2. Click the download button.
3. Launch the installer you downloaded and:

   Most of default options should be find but on the **Choosing the default editor used by Git** screen choose **Use Visual Studio Code as Git's default editor**.

   Otherwise just keep clicking **Next** to go through the installer.

Setting Up Git

Git helps you to work with other developers. Because you want to know who worked on a file and how to contact them, you must tell Git your name and email.

Tell Git Your Name & Email

1. To access the command line, do the following instructions for your platform:
   - Mac: Go to **Applications > Utilities** and open **Terminal.app**.
   - Windows: Launch the **Git Bash** app, which may be an icon on your Desktop, or in the Windows Start menu (probably in a **Git** folder).
2. Enter the following command, replacing **Your Name** with your actual first and last name (but keep the quotes around it):

   ```
   git config --global user.name "Your Name"
   ```

3. Enter the following command, replacing **you@example.com** with your actual email (but keep the quotes around it):

   ```
   git config --global user.email "you@example.com"
   ```

   Keep in mind that changing the name and email will only affect future work.

Checking Your Git Setup (Name & Email)

To check the name or email currently set in git, you can run the same commands, but without the value in quotes. So you'd use these commands:

```
git config --global user.name
git config --global user.email
```

Git Repositories

A Git **repository** (or **repo** for short) contains all of the project files and the entire revision history. You'll take an ordinary folder of files (such as a website's root folder), and tell Git to make it a repository. This creates a **.git** subfolder, which contains all of the Git metadata for tracking changes.

On Unix-based operating systems such as macOS, files and folders that start with a period (.) are hidden, so you will not see the **.git** folder in the macOS Finder unless you show hidden files, but it's there. You won't ever need to go into the **.git** folder, so don't worry about not being able to see it.

TIP: On macOS you can show or hide invisible files by hitting **Cmd–Shift-Period(.)**

Initialize a Git Repo

1. Open a project folder in Visual Studio Code.

   You can do this by going to **File > Open Folder**, navigate to the folder, select it, and hit **Open** (Mac) or **Select Folder** (Windows).

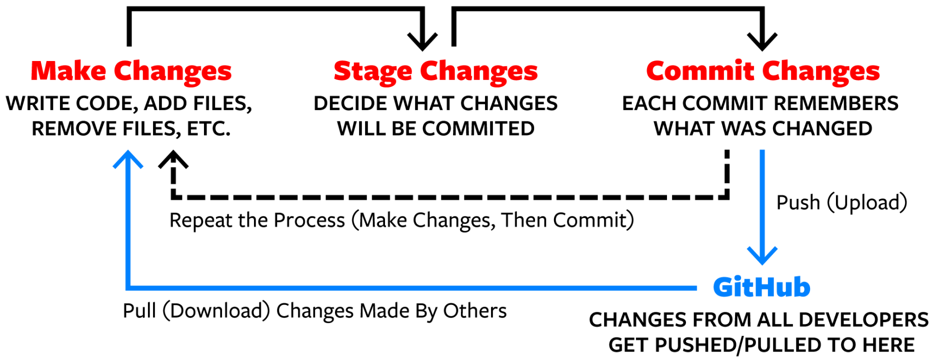
2. Open the **Source Control** panel on the left of the window.

3. In the **Source Control** panel click **Initialize Repository**.

Tracking Changes with Git (The Git Workflow)

Git keeps a list of changes (files added, deleted, or modified). So how do we tell Git to record our changes? Each recorded change is called a commit.

Here's an illustration of the Git workflow:


Make Changes
WRITE CODE, ADD FILES, REMOVE FILES, ETC.
Stage Changes
DECIDE WHAT CHANGES WILL BE COMMITED
Commit Changes
EACH COMMIT REMEMBERS WHAT WAS CHANGED
Repeat the Process (Make Changes, Then Commit)
Push (Upload)
GitHub
CHANGES FROM ALL DEVELOPERS GET PUSHED/PULLED TO HERE
Pull (Download) Changes Made By Others

Before we make a commit, we must tell Git what files we want to commit (new untracked files, modified files, or deleted files). This is called staging (we add files to the stage). Why must we do this? Why can't we just commit something directly? Let's say you're working on two files, but only one of them is ready to commit. You don't want to be forced to commit both files, just the one that's ready. We add files to a staging area, and then we commit what has been staged. Even the deletion of a file must be tracked in Git's history, so deleted files must also be staged and then committed.

Check the Status of Your Files

Let's first check the status of files in our Git repo.

1. Open a project folder in Visual Studio Code.

   You can do this by going to **File > Open Folder**, navigate to your folder, select it, and hit **Open** (Mac) or **Select Folder** (Windows).

2. Open the **Source Control** panel on the left of the window.

3. In the **Source Control** panel, if any work has been done, under **Changes** you'll see files listed with their status:

   M = modified
   U = untracked
   D = deleted

   NOTE: Within a file, a colored bar on left (by the line numbers) indicates that line has changed.

Stage & Commit Files

Stage Files to Prepare for Commit

In the **Source Control** panel :

- To stage all files, hover over **Changes** and click the plus (+) that appears to its right.
- To stage individual files, hover over the file name and click the plus (+) that appears to its right.
- To unstage a file (if you accidentally staged it), hover over it and click the minus (–) that appears to its right.

Commit Files

After you've staged files and are ready to commit:

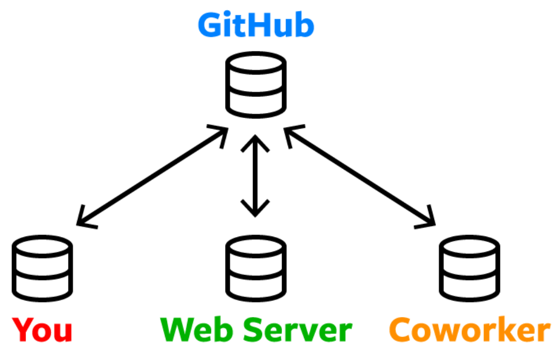
1. At the top of the **Source Control** panel type in a commit message.

   Commit messages should NOT be written in the past tense, such as "I made headings blue". Use language like "Make headings blue", as if you are giving orders to the codebase. The reason for this is when you work with other people, your code may not be automatically approved. You'll request that they pull your changes into the codebase. When they read the commit messages they will do know what your code **will** do. Your change will "Make headings blue".

2. Do one of the following:
   - At the top of the **Source Control** panel click the **Commit** button.
   - Or hold **Cmd** (Mac) or **Ctrl** (Windows) and hit **Return** (Mac) or **Enter** (Windows). If you get a message asking if you'd like to stage all your changes and commit, click **Yes**.

Pushing (Uploading) Your Changes: Remote Repositories & GitHub Explained

When you have a local repository that you want to share with others (or create an online backup), you need to create a remote repo on a service like GitHub so you can upload (push) your files and revision history to it.



Storing a copy of your Git repo with an online host such as **GitHub** (which is run by Microsoft) gives you a centrally located place where you can upload your changes and download changes from others. This lets you collaborate with other developers, and serves as a backup of your code (and the Git history of changes).

Push for the First Time to Create a New GitHub Repo (Publishing from Visual Studio Code)

1. Windows Users Only: To ensure GitHub's authorization will work with VS Code, you must set your your default browser Google Chrome. For instructions on how to do that visit **tinyurl.com/def-brow**
2. Open your local Git repo folder in Visual Studio Code.
3. At the bottom left of the VS Code window, click the **Publish to GitHub** button ☁.

4. If this is the first time you're using GitHub in VS Code you'll see a message wanting to sign into GitHub, click **Allow** and:
   - In the web browser that opens, when asked to Authorize Visual Studio Code, click **Continue**.
   - If asked, sign into your GitHub account.
   - Click **Authorize github**. If you get an error, do the sidebar below.

If An Error Occurred

1. Switch back to Visual Studio Code.
2. Hit the **Esc** key.
3. At the bottom left of the window, click **Signing in to github.com**.
4. Hit the **Esc** key.
5. Now we can try again (it often works the second time). At the bottom left of the VS Code window, click the **Publish to GitHub** button.
6. Click **Allow**.
7. In your web browser, click **Continue** and hopefully this time it works! Proceed with the remaining steps below.

   - If asked about allowing this page to open Visual Studio Code, click **Allow**, **Open Visual Studio Code**, or whatever your browser says.
   - Back in Visual Studio Code, if you're asked to allow an extension to open this URI click **Open**.

5. In the panel that appears at the top of the window, choose **Publish to GitHub private repository**.

   NOTE: You could use a public repository if you want anyone to be able to see and contribute to it, such as and open source project.

6. At the bottom right of the window you'll see the upload progress. During the upload, if you're asked to sign into GitHub, click **Sign in with your browser**:
   - Switch back to your web browser and click **Authorize GitCredentialManager**.
   - When you see **Authentication Succeeded** switch back to Visual Studio Code.

7. At the bottom right of the window you'll see the upload progress. When it's done click **Open on GitHub**.

   Your web browser will open and you'll see your remote repo!

8. Switch back to Visual Studio Code and you can continue your work.
9. If you're asked **Would you like Code to periodically run 'git fetch'?** (at the bottom right of the VS Code window) we recommend clicking **No**.

How You Typically Push Changes to GitHub

Once the local Git repo knows about the remote repo (GitHub), you can use the Push command. Here are some different ways to do that:

- At the bottom left of the Visual Studio Code window, click the Push commits button 0↓ 1↑ to pull and then push (we explain what pulling is in the next exercise).
- At the top right of **Source Control** panel click the **More Actions** button ··· and from the menu choose **Push**.

Pulling (Downloading) Your Changes

After someone else makes changes to the remote repo (GitHub), you can download (pull) their changes into your local repo.

How to Pull Changes from GitHub

Once the local Git repo knows about the remote repo (GitHub), you can use the Pull command. Here are some different ways to do that:

- At the bottom left of the Visual Studio Code window, click the Synchronize Changes button .
- At the top right of **Source Control** panel click the **More Actions** button ··· and from the menu choose **Pull**.

When to Pull

It's a good idea to pull:

- before you start to work (so you have the latest changes)
- before you push (in case something has changed while you worked)

Contributing to Someone Else's Repo

When you want to start contributing to a Git repo that has already been created, you can clone it (download a copy) from GitHub. If it's a private repo, they need to invite you.

Clone a Git Repo

1. First you'll need the URL of the Git repo. When viewing the repo on GitHub, click the **Code** button to see the URL. Copy that URL by clicking the button to its right.
2. In Visual Studio Code choose **File > New Window**.
3. Open the **Source Control** panel on the left of the window.
4. In the **Source Control** panel click **Clone Repository**.
5. In the panel that appears at the top the window, paste the URL (or git clone command) and hit **Return** (Mac) or **Enter** (Windows).
6. You'll be asked where you want to store the repo.

   Navigate to the folder where you want to store the repo (a new folder for the repo will be created here) and click **Select as Repository Destination**.
7. If you're asked **Would you like to open the cloned repository?** (at the bottom right of the VS Code window) click **Open**.

Giving Someone Access to Your Private GitHub Repo

If you have a private GitHub repo that you want someone else to contribute to:

1. On github.com go to the page for your repo.
2. Click the **Settings** tab.
3. In the left sidebar, click **Collaborators**.
4. Click **Add people**.
5. Enter the person's email address and follow any remaining prompts.

Merging Changes

Sometimes you'll get a conflict when pulling, reverting commits, merging branches, etc. For example, you and another developer unknowingly both work on the same part of a file. The other developer pushes their changes to the remote repo. When you then pull them to your local repo you'll get a merge conflict. Luckily Git has a way to handle conflicts, so you can see both sets of changes and decide which you want to keep.

What To Do With a Merge Conflict

1. When you get a merge conflict, open a conflicted file. They are listed in the **Source Control** panel under **Merge Changes**.
2. In the conflicted file, look for these conflict markers:

   `<<<<<<< HEAD` (Current Change)
   Marks the start of the changes.

   `=======`
   Divides your changes from the changes in the other branch.

   `>>>>>>> branch-name` (Incoming Change)
   Marks the end of the changes.
3. Now you must decide how to resolve the conflict. Above the change you can choose various ways to handle it:
   - Accept Current Change
   - Accept Incoming Change
   - Accept Both Changes
   - Compare Changes

   Look at the code and choose whichever is appropriate for this case.
   After accepting the change, you can make further edits to the code as needed.
4. Save the file.
5. After you've handled all the conflicts, in the **Source Control** panel stage the changes by hovering over **Merge Changes** and clicking the plus (+) that appears to its right.
6. At the top of the **Source Control** panel click the **Continue** button.
7. Push/Sync whenever you're ready to upload those changes to GitHub so others can get them.

Viewing Changes & Undoing Changes

Sometimes you make a mistake and want to go back to a previous version. Here's how to rollback changes.

Undo Local Changes That Have Not Been Committed

If you have made changes that you don't like, and they **have not been committed** yet, do as follows:

1. In the **Source Control** panel you'll see files that have been changed.
2. Hover over any file that contains changes you don't want to keep and click **Discard Changes** button that appears.
3. When it confirms, click **Discard Changes**.

   That file has now been reverted to the way it was at the previous commit (before your changes).

Install the Git Graph Extension for Visual Studio Code

To add some useful Git features to Visual Studio Code, you should install the **Git Graph** extension:

1. On the left side of the Visual Studio Code window, click on the **Extensions** icon .
2. In the search field type: **Git Graph**
3. Under the **Git Graph** click **Install**.

View a List of Commits

You'll need the **Git Graph** extension installed for this to work.

1. Towards the bottom left of the window, click **Git Graph**.

   Alternatively, at the top of **Source Control** panel you can click the **View Git Graph** button .
2. In the **Git Graph** tab that opens you can:
   - Read through the list of commit messages to find a commit.
   - Click on the commit to see which files were changed.
   - Within the commit, click on a file name to see the changes.

Undo Your Last Commit (That Has Not Been Pushed)

If you made a mistake on your last commit and have not pushed yet, you can undo it. For example, maybe you added some files and made a commit, and then immediately realized you forgot something. You can undo the commit, and then make a new (correct) commit. This will keep your history cleaner.

1. At the top right of **Source Control** panel click the **More Actions** button ⋯ and from the menu choose **Commit > Undo Last Commit**.
2. Your latest commit will now be undone. Your changes remain in place, and the files go back to being staged so you can unstage those files and make additional changes, add missing files, discard changes, etc. before making your next commit.

Undo a Specific Commit (That Has Not Been Pushed)

If you have made local commits that you don't like, and they **have NOT** been pushed yet, you can reset things back to a previous good commit. It will be as if the bad commits never happened.

You'll need the **Git Graph** extension installed for this to work.

1. Towards the bottom left of the window, click **Git Graph**.

   Alternatively, at the top of **Source Control** panel you can click the **View Git Graph** button.
2. To figure out what the commits do (and thereby which commit you want to undo):
   - Read through the list of commit messages to find a commit.
   - Click on the commit to see which files were changed.
   - Within the commit, click on a file name to see the changes.
   - Close the changed file tab so you end up back in the **Git Graph** tab.
3. Once you know which commit (that has not been pushed) you want to remove, in the **Git Graph** tab, **Ctrl–click** (Mac) or **Right–click** (Windows) on the name of the commit and choose **Drop**.
4. Confirm that you want to drop it.

   That commit will be removed and the changes reversed. Your files have been updated so it's as though those changes (and that commit) were never made.

Undo a Specific Commit (That Has Been Pushed)

If you have made a local commit that you don't like, and it **has been** pushed, you can reset things back to a previous good commit. It will be as if the changes made by the bad commit never happened (although you will still see the commit was made, and then reverted).

You'll need the **Git Graph** extension installed for this to work.

1. Towards the bottom left of the window, click **Git Graph**.

   Alternatively, at the top of **Source Control** panel you can click the **View Git Graph** button.

2. To figure out what the commits do (and thereby which commit you want to undo):

   - Read through the list of commit messages to find a commit.
   - Click on the commit to see which files were changed.
   - Within the commit, click on a file name to see the changes.
   - Close the changed file tab so you end up back in the **Git Graph** tab.

3. Once you know which commit you want to revert, in the **Git Graph** tab, **Ctrl–click** (Mac) or **Right–click** (Windows) on the name of the commit and choose **Revert**.

4. Confirm that you want to drop it.

   A new commit will be created that reverses the original commit.

   NOTE: If there were any conflicts, you'll have to manage them and then make a new commit.

5. Push whenever you're ready to upload those changes to GitHub so others can get them.

Another Git Extension for Visual Studio Code

If you want even more Git features in Visual Studio Code, you may want to check out GitLens.

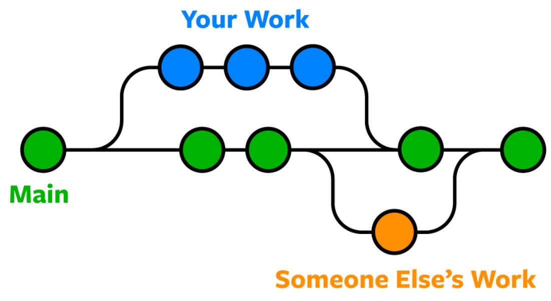
1. On the left side of the Visual Studio Code window, click on the **Extensions** icon.
2. In the search field type: **GitLens**
3. Under the **GitLens** click **Install**.

Here are a few things the GitLens extension does for you:

- Adds new sections at the bottom of the **Source Control** panel such as Commits, File History, Branches, etc.
- Adds new Git buttons at the top right of the window for navigating changes, etc.
- When your cursor is in a line of code, it adds a comment (at the end of the line) about who and changed that line of code and when.

Git Branches

Git lets you branch out from the original code base. This lets you more easily work with other developers, and gives you a lot of flexibility in your workflow.



Here's an example of how Git branches are useful. Let's say you need to work on a new feature for a website. You create a new branch and start working. You haven't finished your new feature, but you get a request to make a rush change that needs to go live on the site today. You switch back to the **main** branch, make the change, and push it live. Then you can switch back to your new feature branch and finish your work. When you're done, you merge the new feature branch into the main branch and both the new feature and rush change are kept!

NOTE: **Main** branches used to be called **master** branches.

Working on a Branch: Commit, Push, Pull, etc.

Keep in mind that whenever you commit, push, pull, etc. you're doing so on a branch. So make sure you're on the correct branch you doing any Git commands.

See What Branch You're On

The current branch is shown at the bottom left of the Visual Studio Code window (**main** is the default branch name).

Create a New Branch

1. At the bottom left of the Visual Studio Code window click the current branch name (**main** is the default branch name).
2. In panel that appears at the top of the window, choose **+ Create new branch**.
3. Type in a name for your new branch and hit **Return** (Mac) or **Enter** (Windows)

   You're now ready to commit to this branch.

Switch to a Branch In Your Local Repo

1. At the bottom left of the Visual Studio Code window click the current branch name (**main** is the default branch name).
2. A panel will appear at the top of the window listing the branches. Choose the branch that you want to switch to.

Switch to a Branch In a Remote Repo

1. To get the list of branches from GitHub, at the top right of **Source Control** panel click the **More Actions** button ··· and go into **Pull, Push** and choose **Fetch**.
2. At the bottom left of the window click the current branch name (**main** is the default branch name).
3. A panel will appear at the top of the window listing the branches. The ones that start with **origin/** are remote branches. Choose the branch that you want to switch to.

Fetch versus Pull

Fetch downloads new data from the remote repo, but does not merge that into your files.

Pull downloads new data from the remote repo (using fetch), and merges it into your files.

Merge a Branch

1. Switch to the branch that you want to merge another branch into (changes will be merged into this branch).

   Most commonly you'll merge changes from another branch into the **main** branch, so you'll typically switch to the **main** branch (at the bottom left of the window click the current branch name and choose the desired branch).

2. In the **Source Control** panel make sure you don't have any changes that need to be committed.
3. Now you can merge another branch into the current branch. At the top right of **Source Control** panel click the **More Actions** button ··· and from the menu choose **Branch > Merge Branch**.

4. Select the branch you want to merge (into the current branch) and you're done.

   NOTE: When you merge, there may be a conflict. Refer to **How to Handle Merge Conflicts** to learn what to do.

Delete a Local Branch

If you're done with a branch and you don't plan on using it again, you can delete it as follows:

1. You can't delete the current branch, so switch to different branch than you want to delete.
2. At the top right of **Source Control** panel click the **More Actions** button ··· and from the menu choose **Branch > Delete Branch**.
3. Select the branch you want to delete.

   Confirm that you want to delete it, and you're done.

Editing a GitHub Repo Directly With Nothing to Download

Visual Studio Code was built with using web tech, so you can use it in a web browser directly on github.com without having to download an app or the repo.

1. Log into github.com
2. Navigate to the GitHub repo you want to edit.
3. Hit the period key (.) to open the web version of Visual Studio Code right in your browser. Nothing must be downloaded!
4. Open files, make any edits, etc.
5. When you commit, the changes will automatically be pushed to the repo. Sweet!

Working with Folders

- Drag a folder onto VS Code to view it in the sidebar. You can also choose **File > Open Folder** and select a folder you want to open.
- Quickly open/switch to a file in the current folder:

  – Choose **Go > Go to File** or hit **Cmd–P** (Mac) or **Ctrl–P** (Windows).

  – Start typing the name of a file (use the **Down/Up Arrow** keys to move the selection up or down).

  – Hit **Return** (Mac) or **Enter** (Windows) to open the selected file.
- Search within the current folder: Hit **Cmd–Shift–F** (Mac) or **Ctrl–Shift–F** (Windows) or choose **Edit > Find in Files**.

Sidebar Features

Here are some useful things you can do in the sidebar:

- **Ctrl–Click** (Mac) or **Right–Click** (Windows) on a folder in the sidebar (or an empty area) and choose **New File** or **New Folder**.
- **Ctrl–Click** (Mac) or **Right–Click** (Windows) a file or folder in the sidebar and choose **Rename**.

  TIP: You can also select a file or folder in the sidebar and hit **Return** (Mac) or **Enter** (Windows) to make the name editable so you can rename it.

Switching Between Files (Tabs)

- Mac: Hit **Ctrl–Tab** (add **Shift** to cycle in the opposite direction).
- Windows: Hit **Ctrl–Page Down** or **Ctrl–Page Up** (Windows).

Outline View

1. In the file **Explorer** sidebar, at the bottom click on **Outline** to see the structure of the current file (such as the tags in HTML).
2. Click on anything in the outline to go to that code.

Using the Command Palette

VS Code's Command Palette lets you quickly find and apply commands. To open the Command Palette, choose **View > Command Palette** or hit **Cmd–Shift–P** (Mac) or **Ctrl–Shift–P** (Windows).

Here's an example of how to use it:

1. Select some text that you want to convert into title case (where the first letter of each word is capitalized).
2. Hit **Cmd–Shift–P** (Mac) or **Ctrl–Shift–P** (Windows) to open the Command Palette.
3. Start typing **title** until **Transform to Title Case** appears and is selected. (If needed, use the **Down/Up Arrow** keys to move the selection down or up.)

   NOTE: You can also find this command by typing **ttt** (for **t**ransform **t**o **t**itle case) because the search is very good at matching!
4. Hit **Return** (Mac) or **Enter** (Windows) to execute the command.

Some Useful Commands

- **Sort Lines Ascending** or **Sort Lines Descending**
- **Update Image Size**

Visual Studio Code Tips and Tricks

To help you learn and get the most out of it, Microsoft (the maker of VS Code), created an illustrated guide: **code.visualstudio.com/docs/getstarted/tips-and-tricks**

Keyboard Shortcut Guides

Here are the official keyboard reference sheets:

- Mac: **code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf**
- Windows: **code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf**

Adding/Removing a Comment

- To add/remove a line comment, hit **Cmd–/** (Mac) or **Ctrl–/** (Windows) or choose **Edit > Toggle Line Comment**.

  A comment appropriate for the current language (HTML, CSS, JS, etc.) will be intelligently added or removed. You can even place the cursor anywhere in a line and the entire line will be commented out!

- To add/remove a block comment, hit **Option–Shift–A** (Mac) or **Alt–Shift–A** (Windows) or choose **Edit > Toggle Block Comment**.

  Block comments let you select something and comment it out, but it does not have to be an entire line.

Move Lines Up or Down

1. Select the line(s) you want to move. Putting the cursor in a line also works.
2. Hit **Option–Up/Down Arrow** (Mac) or **Alt–Up/Down Arrow** (Windows) to move the line(s) up or down.

Copy a Line

Place the cursor anywhere in a line. Then hit **Option–Shift–Down Arrow** (Mac) or **Alt–Shift–Down Arrow** (Windows) to copy the line below the current line (use **Up Arrow** to add the copy above the current line).

Insert a New Line

You can insert a new line above or below the current line without having to move the cursor to the end of the line!

- Insert a new line below: **Cmd–Return** (Mac) or **Ctrl–Enter** (Windows)
- Insert a new line above: **Cmd–Shift–Return** (Mac) or **Ctrl–Shift–Enter** (Windows)

Delete a Line

Place the cursor anywhere in a line. Then hit **Cmd–Shift–K** (Mac) or **Ctrl–Shift–K** to delete the entire line.

Delete a Word (Work in most apps, not just code editors)

- Delete word to left of cursor: **Option–Delete** (Mac) or **Ctrl–Backspace** (Windows)
- Delete word to right of cursor: **Option–Fn–Delete** (Mac) or **Ctrl–Delete** (Windows)

Quickly Cut or Copy a Line

You can cut (and paste) an entire line of code without selecting it:

1. Place the cursor anywhere in a line. You do not have to select the entire line!
2. Hit **Cmd–X** (Mac) or **Ctrl–X** (Windows) to cut the line.
3. Place the cursor wherever you want the code, and hit **Cmd–V** (Mac) or **Ctrl–V** (Windows) to paste it.

NOTE: You can also **copy** a line of code the same way. Just use **Cmd–C** (Mac) or **Ctrl–C** (Windows) instead.

Add Next Item to Selection

- Hit **Cmd–D** (Mac) or **Ctrl–D** (Windows) to select the next occurrence of the whatever you currently have selected.
- Repeat the keystroke to continue adding more items to the selection.
- If it selects something you don't want, hit **Cmd–K** (Mac) or **Ctrl–K** (Windows) and then hit **Cmd–D** (Mac) or **Ctrl–D** (Windows) to skip that item.

Navigating Code

- Hit **Cmd–Shift–O** (Mac) or **Ctrl–Shift–O** (Windows) (the keystroke for **Go > Go to Symbol in Editor**) to navigate certain filetypes (such as CSS or JS). For example, in CSS files you can type in the name of selectors, or in JS use function names.
- To jump to a specific line of code, hit **Ctrl–G** (Mac and Windows) and type in the line number. Hit **Return** (Mac) or **Enter** (Windows) to go to the line.
- Move to beginning or end of a **line**:

  Move to beginning of a line: **Cmd–Left Arrow** (Mac) or **Home** (Windows)

  Move to end of a line: **Cmd–Right Arrow** (Mac) or **End** (Windows)
- Jump **words**:

  Jump to previous word: **Option–Left** (Mac) or **Ctrl–Left** (Windows)

  Jump to next word: **Option–Right** (Mac) or **Ctrl–Right** (Windows)

Fold (Hide) a Section of Code

To fold/hide a section of code, place the cursor anywhere in the code and hit the following keystroke (repeat it to unfold/show the code).

- Mac: **Cmd–K** then **Cmd–L**
- Windows: **Ctrl–K** then **Ctrl–L**

You can also use these shortcuts:

- Mac: **Cmd–Option–[** to fold or **Cmd–Option–]** to unfold
- Windows: **Ctrl–Shift–[** to fold or **Ctrl–Shift–]** to unfold

Multiple Cursors & Selections

- Place multiple cursors: **Option–Click** (Mac) or **Alt–Click** (Windows)
- Multiple selections (discontinuous): Hold **Option** (Mac) or **Alt** (Windows) while dragging.
- To make a single discontinuous selection area across multiple lines: Hold **Option–Shift** (Mac) or **Alt–Shift** (Windows) while dragging.
- Create a multi-line cursor: Hold **Cmd–Option** (Mac) or **Ctrl–Alt** (Windows) and then hit the **Up or Down Arrow** keys.
- If you have multiple selections or cursors, you can make edits: delete, type, paste, etc. Press **Escape** (**esc**) to return to one cursor.

Setting Preferences

1. In Visual Studio Code do the following:
   - Mac users: Go into the **Code** menu and choose **Preferences > Settings**.
   - Windows users: Go into the **File** menu and choose **Preferences > Settings**.
2. You can either look through the preferences or use the search field to find the preference you want.

   For example, you may want to change the **Editor: Tab Size** preference from **4** (the default amount) to **3** so tabs are not so wide.

Changing the Default Font

Source Code Pro is a really nice coding font from Adobe. You can download it for free on fonts.google.com. After installing the font, you can set Visual Studio Code to use it as follows:

1. In Visual Studio Code do the following:
   - Mac users: Go into the **Code** menu and choose **Preferences > Settings**.
   - Windows users: Go into the **File** menu and choose **Preferences > Settings**.
2. In the search field type: **font family**
3. Find **Editor: Font Family** and add **Source Code Pro,** (don't miss the comma) at the beginning of the font list.

   You should end up with something like: **Source Code Pro, Menlo, Monaco, 'Courier New', monospace**

Recommended Extensions

Visual Studio Code has some great free extensions. Here are some we recommend:

open in browser

In the setup instructions of this book we recommended this extension. It lets you open an HTML file in a browser by hitting **Option–B** (Mac) or **Alt–B** (Windows).

Live Server

Tired of refreshing a browser every time you update your HTML and CSS?
This extension lets you **Ctrl–Click** (Mac) or **Right–Click** (Windows) on an HTML file and choose **Open with Live Server**. Your web browser will open (using a local server from the extension) and will refresh automatically when you save in VS Code!

Live Preview

Tired of refreshing a browser every time you update your HTML and CSS?
This extension adds a Live Preview button at the top right of the window, which will open a browser window directly in VS Code! As you edit HTML and CSS files, the changes will instantly show up without even having to save your files!

FontSize Shortcuts

In VS Code you can change the font size by holding **Cmd** (Mac) or **Ctrl** (Windows) and pressing **plus (+)** or **minus (–)** to make the text bigger or smaller. While this is useful, it also scales up the interface which we don't like. This extension changes those keystrokes to only scale up the text, not the interface!
NOTE: **Cmd–0** (Mac) or **Ctrl–0** (Windows) resets to the normal font size.

HTML End Tag Labels

Adds a comment-like label for each HTML end tag (so you know which particular tag it's ending).

Image preview

Shows an image thumbnail in the gutter (by the line numbers).

Git Graph

Adds a visual graph of your commits, branches, etc with features for reverting to a commit and much more.

GitLens

Adds even more Git features to VS Code.

About Emmet

Emmet (which is built into VS Code) lets you type HTML and CSS code faster.

Its main feature is expandable code abbreviations. You type a few characters and Emmet expands the abbreviation into the full code.

It also has commands (which don't have keystrokes by default, although you can assign them) to make it easier and faster to code HTML and CSS.

Emmet Documentation

Visit **emmet.io** for lots of good info and demos. It also has a great **Cheat Sheet** listing all the abbreviations and what they expand to: **docs.emmet.io/cheat-sheet**

For more information on Emmet in VS Code, visit **code.visualstudio.com/docs/editor/emmet**

Using Emmet's HTML Abbreviations

Emmet includes a lot of short abbreviations that you can expand into longer HTML code. Type an abbreviation, then hit **Tab** to expand the abbreviation into the full HTML code. Emmet's Cheat Sheet has a complete list of the abbreviations: docs.emmet.io/cheat-sheet

Below are examples of some useful abbreviations and the code they expand into after you hit **Tab**. Many use a CSS-like syntax.

h1

```
<h1></h1>
```

p

```
<p></p>
```

img

```
<img src="" alt="">
```

div

```
<div></div>
```

#myElement

```
<div id="myElement"></div>
```

### .myElement

```
<div class="myElement"></div>
```

NOTE: **div** is assumed. If you want the class on a different tag (such as a **ul** tag) refer to the next example.

ul.myList

```
<ul class="myList"></ul>
```

ul>li

```
<ul>
    <li></li>
</ul>
```

ul>li*3

```
<ul>
   <li></li>
   <li></li>
   <li></li>
</ul>
```

ul.myList>li*3

```
<ul class="myList">
   <li></li>
   <li></li>
   <li></li>
</ul>
```

h1+p

```
<h1></h1>
<p></p>
```

TIP: After typing content into the **h1** tag, hit **Tab** to jump inside the **p** tag!

div#myContent>h1+p*3

```
<div id="myContent">
    <h1></h1>
    <p></p>
    <p></p>
    <p></p>
</div>
```

ul>li.item$*5>a

```
<ul>
   <li class="item1"><a href=""></a></li>
   <li class="item2"><a href=""></a></li>
   <li class="item3"><a href=""></a></li>
   <li class="item4"><a href=""></a></li>
   <li class="item5"><a href=""></a></li>
</ul>
```

NOTE: **$** outputs the current number of the repeated element.

ul>li.item$*5>a[href="/details/"]

```
<ul>
   <li class="item1"><a href="/details/"></a></li>
   <li class="item2"><a href="/details/"></a></li>
   <li class="item3"><a href="/details/"></a></li>
   <li class="item4"><a href="/details/"></a></li>
   <li class="item5"><a href="/details/"></a></li>
</ul>
```

NOTE: The **`a[href=""]`** above uses the CSS attribute selector syntax. If you haven't used attribute selectors, learn more them at **css-tricks.com/attribute-selectors**

ul>li.item$*5>img[src="img/photo$.jpg"]

```
<ul>
   <li class="item1"><img src="img/photo1.jpg" alt=""></li>
   <li class="item2"><img src="img/photo2.jpg" alt=""></li>
   <li class="item3"><img src="img/photo3.jpg" alt=""></li>
   <li class="item4"><img src="img/photo4.jpg" alt=""></li>
   <li class="item5"><img src="img/photo5.jpg" alt=""></li>
</ul>
```

`lorem20`

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt illo non
dolor nobis eveniet dolore consectetur nostrum, sequi officia qui!
```

NOTE: Change the number after lorem to be how many words you want!

Using Emmet's CSS Abbreviations

Emmet's abbreviations are not just for HTML. Emmet has a lot of abbreviations for CSS code too! Once again, you type an abbreviation, then hit **Tab** to expand the abbreviation into the full CSS code. Emmet's Cheat Sheet has a complete list of the abbreviations: **docs.emmet.io/cheat-sheet**

Below are examples of some useful abbreviations and the code they expand into after you hit **Tab**. Many are based on the first letter of a property (or the first letter of the words used in the property and value).

p

```
padding: ;
```

pt

```
padding-top: ;
```

m

```
margin: ;
```

mt

```
margin-top: ;
```

tac

```
text-align: center;
```

dib

```
display: inline-block;
```

m20

```
margin: 20px;
```

m20% (or m20p)

```
margin: 20%;
```

m20-40

```
margin: 20px 40px;
```

NOTE: If no unit is specified, pixels are assumed.

m20p40p

```
margin: 20% 40%;
```

NOTE: If you specify the type of unit (such as %, p) don't include a hyphen (-) separator. Adding a hyphen separator would give you negative values.

fz2e

```
font-size: 2em;
```

bg

```
background: #000;
```

bd

```
border: 1px solid #000;
```

c

```
color: #000;
```

`cr`

```
color: rgb(0, 0, 0);
```

TIP: After expanding the abbreviation, hit **Tab** to jump between the rgb values!

`cra`

```
color: rgba(0, 0, 0, .5);
```

TIP: After expanding the abbreviation, hit **Tab** to jump between the rgb values!

`poa`

```
position: absolute;
```

`bxsh`

```
box-shadow: inset hoff voff blur #000;
```

TIP: After expanding the abbreviation, hit **Tab** to jump between the various values. Both sets of values will be highlighted so you only have to type them once!

View Possible Emmet Commands

Emmet isn't only about expanding abbreviations, it has additional features.

1. Hit **Cmd–Shift–P** (Mac) or **Ctrl–Shift–P** (Windows) to open the Command Palette.
2. Type **emmet** to see a list of features (the corresponding keystroke is shown on the right if if has one).

Enabling Keystrokes for Emmet Commands

Emmet's numerous features do not all have keystroke assigned by default. If you find an Emmet feature that you use a lot and you'd like to have a keyboard shortcut, here's how to set up a keystroke for it:

1. In Visual Studio Code do the following:
   - Mac users: Go into the **Code** menu and choose **Preferences > Keyboard Shortcuts**.
   - Windows users: Go into the **File** menu and choose **Preferences > Keyboard Shortcuts**.
2. In the search field type in the feature's name (or type **emmet** to see all its features).
3. Double–click on the desired feature.
4. Press your desired keystroke, then hit **Return**.

   NOTE: It will warn you if there are other features already using that keystroke.
5. You're done, so close the **Keyboard Shortcuts** tab.

Emmet: Wrap with Abbreviation

Wrapping a tag around a selection is such a common task. In the **Before You Begin** section near the start of this book are instructions for assigning our preferred keystroke for this: **Option–W** (Mac) or **Alt–W** (Windows). Here's how to use it:

1. Select some code. For best results select from the start to end of the code, without including whitespace before or after.
2. Hit **Option–W** (Mac) or **Alt–W** (Windows) and a panel will appear at the top the window.
3. Type in an abbreviation for a tag, class, etc. For example, type **.container** to wrap something in a div with a container class.

   NOTE: As you type the abbreviation you'll see the new code to confirm it's working.
4. Hit **Return** (Mac) or **Enter** (Windows) to finish (or hit **Esc** to cancel).

Emmet: Remove Tag

Emmet can remove a tag but keep its contents.

1. Place your cursor in a tag you want to remove (or somewhere in the content within the tag).
2. Use the Command Palette to find the **Emmet: Remove Tag** command.

   This also works on elements that don't have end tags (like an **img** tag).

Emmet: Update Image Size

Emmet can look up the size of an image file for you.

1. Click anywhere in an **img** tag that does not have width or height attributes (such as `<img src="img/logo.png">`) or an **img** tag that has the wrong width and height.
2. Use the Command Palette to find the **Emmet: Update Image Size** command.

Emmet: Evaluation Math Expression

Emmet can do math for you.

1. Select some math, such as **158/2**
2. Use the Command Palette to find **Emmet: Evaluation Math Expression**.

How to Create a New HTML File with Emmet

1. Create a new file.
2. Save the file as **some-file-name.html** so the app knows this is an HTML file.
3. Type **!** (that's an exclamation point) and hit **Tab** to expand it.

   All the standard tags (like head and body) will be created.
4. Hit **Tab** until the title is highlighted, then enter your title.
5. Hit **Tab** again to jump into the body tag and you're ready to code!
6. You can delete the following line, which is old code for Internet Explorer:

   ```
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   ```

Adding HTML Around a Plain Text List

1. Select a multi-line plain text list such as:

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

2. Hit Emmet's wrap tag keystroke: **Option–W** (Mac) or **Alt–W** (Windows). Please note this keystroke only works if you defined it (as we've previously explained and had you set up using the instructions in the **Before You Begin** section near the start of this book).

3. Type **ul>li*** to see it wraps all the list items like this:

```
<ul>
    <li>Monday</li>
    <li>Tuesday</li>
    <li>Wednesday</li>
    <li>Thursday</li>
    <li>Friday</li>
    <li>Saturday</li>
    <li>Sunday</li>
</ul>
```

4. If you add **>a** so you have **ul>li*>a** you'll also get links:

```
<ul>
    <li><a href="">Monday</a></li>
    <li><a href="">Tuesday</a></li>
    <li><a href="">Wednesday</a></li>
    <li><a href="">Thursday</a></li>
    <li><a href="">Friday</a></li>
    <li><a href="">Saturday</a></li>
    <li><a href="">Sunday</a></li>
</ul>
```

5. Hit **Return** (Mac) or **Enter** (Windows) to finish.

Wrapping Text on Multiple Lines (Convert a list into JavaScript Array)

We can quickly turn a plain text list like this:

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

Into a JavaScript Array like this:

```
days=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday']
```

Here's How To Do It

These instructions are for Visual Studio Code:

1. Select the multi-line plain text list (such as the days list above).
2. Choose **Selection > Add Cursors to Line Ends**.
3. To select the entire line, hold **Shift** and hit the **Home** key. If you don't a Home key, hold **Shift** then **fn** (the function key) and hit **Left Arrow**.
4. Type a **single-quote (`'`)** to wrap them in quotes.
5. Press **Right Arrow** twice to move cursor after the quote.
6. Type a **comma**.
7. Press **Right Arrow** to go to next line.
8. Hit **Delete** (Mac) or **Backspace** (Windows) to pull everything onto one line.
9. Delete the extra **comma** at the end of the line (if there is one).
10. Select all the list code that's now on a single line.
11. Type a **left square bracket [** to wrap it in brackets.
12. In the front of the bracketed code type **days=** and you should end up with:

```
days=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday']
```

Noble Desktop Training

Learn live online or in person in NYC

Front-End Web Development	Motion Graphics & Video Editing
Full-Stack Web Development	Adobe Premiere Pro
JavaScript	Adobe After Effects
Python	InDesign, Illustrator, & Photoshop
Software Engineering	Web, UX, & UI Design
Data Science & Data Analytics	Figma
SQL	Digital & Social Media Marketing
WordPress	and much more...

nobledesktop.com

Common Git Workflows

As a handy reference, below are a few common git workflows you'll use.

Edit, Stage, Commit, & Push

1. pull (to make sure you're working with the latest code)
2. do your coding and any file changes
3. stage files
4. commit
5. push

Switch to an Existing Branch & Pull Latest Changes

1. pull (ensures we get a list of all branches from the remote)
2. checkout the new branch
3. pull

Create a New Branch & Push to It for the First Time

1. pull (to make sure you're working with the latest code)
2. checkout the new branch
3. stage files
4. commit
5. push

Pull Requests

Pull requests are a way to discuss changes before merging them into your codebase. Let's say you're managing a project. A developer makes changes on a new branch and would like to merge that branch into the **main** branch. They can create a pull request to notify you to review their code. You can discuss the changes, and decide if you want to merge it or not.

Pull Requests on GitHub

GitHub has documentation on how to collaborating with pull requests at **docs.github.com/en/github/collaborating-with-pull-requests**

In particular here are the 2 parts you'll probably be most interested in:

- Create Pull Request: **help.github.com/articles/creating-a-pull-request**
- Merge Pull Request: **help.github.com/articles/merging-a-pull-request**

Pull Requests in Visual Studio Code

To work with pull requests inside of Visual Studio Code, check out the extension **GitHub Pull Requests and Issues**.

Adding Commits to a Pull Request

Let's say the person reviewing the pull request wants you to make changes before they accept it. Make the changes, commit them, and push them to the branch for the pull request.

ReadMe Files

A ReadMe file is a standard place for instructions or documentation that you want to share with people about a repo. Here's how to add a ReadMe file to your Git repo.

Create a ReadMe File

1. Using Visual Studio Code, choose **File > New File**.
2. Choose **File > Save As**.
3. Save the file as **README.md** into the base (root) folder of your Git repo.
4. Add any instructions or documentation that you want to share with others.
   Use Markdown to format headings, lists, links, etc.

   Here are some guides for the Markdown syntax:

   - **daringfireball.net/projects/markdown/syntax**
   - **guides.github.com/features/mastering-markdown**

   To see a nicely formatted preview of your Markdown file within Visual Studio Code, at the top right of the window click the **Open Preview** button.
5. When done, save the file, commit the changes, and push to the remote repo. GitHub will display the nicely formatting ReadMe on the project page for the repo.

What is Markdown?

Markdown can be converted into other formats (such as HTML) and has been incorporated into many things. It's the standard format for ReadMe files in Git.

John Gruber of Daring Fireball (**daringfireball.net**) created Markdown. He says "Markdown is intended to be as easy-to-read and easy-to-write as is feasible. Markdown's syntax is intended for one purpose: to be used as a format for writing for the web."

Ignoring Files

There will be some files that you don't want Git to track. You can use a .gitignore file to list the files and/or folders that Git should ignore.

Create a Git Ignore File

1. Using Visual Studio Code, choose **File > New File**.
2. Choose **File > Save As**.
3. Save the file as **.gitignore** into the base (root) folder of your Git repo.

   NOTE: On Unix-based operating systems such as macOS, files that start with a period (.) are hidden. You won't see the **.gitignore** file in the macOS Finder (unless you show hidden files), but you will be able to see it Visual Studio Code.

4. In the .gitignore file you list any files you want to ignore. Here's an example of some files everyone would probably want to ignore:

```
# Mac
.DS_Store
.DocumentRevisions-V100
.Spotlight-V100
.Trashes

# Windows
Thumbs.db
```

Syntax for .gitignore

To learn about the syntax for this file, refer to:

- git-scm.com/docs/gitignore
- atlassian.com/git/tutorials/saving-changes/gitignore