



GIT & GITHUB

WELCOME BACK

GOALS FOR THIS UNIT

1. Review
2. Command Line
3. Version Control
4. Git & Github
5. Hands-on
exercises

QUESTIONS?

REVIEW

THE COMMAND LINE

GETTING STARTED

WHICH OS?

Mac users: Open up Terminal.

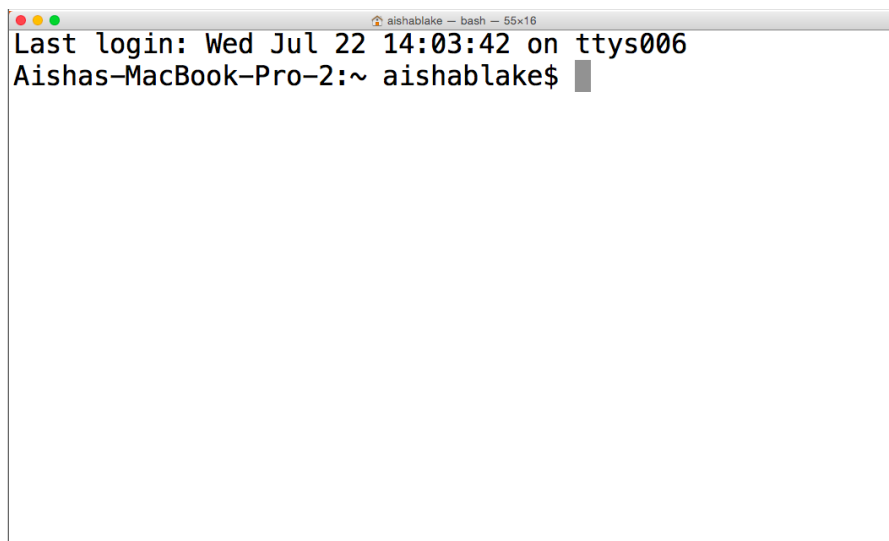
Windows users: Open up the command prompt application you've been using.

If you're on a PC, you can use the command line tool that came with your computer, but know that certain commands will be different from the Mac/Linux commands.

BACK TO BASICS

You've been using the command line since Day 1, but I want to make sure we take the time to solidify those basic commands and learn some new ones. This will eventually become second nature!

TERMINAL PROMPT

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, and a status bar on the right that reads 'aishablake - bash - 55x16'. The terminal content shows a login message: 'Last login: Wed Jul 22 14:03:42 on ttys006'. Below this, the prompt 'Aishas-MacBook-Pro-2:~ aishablake\$' is displayed, followed by a dark gray rectangular cursor block.

```
aishablake - bash - 55x16
Last login: Wed Jul 22 14:03:42 on ttys006
Aishas-MacBook-Pro-2:~ aishablake$
```

Many command line prompts will end with a dollar sign (\$). This is your signal that it's okay to type a new command.

TERMINAL CHEAT SHEET

There are lots of things we can do with the command line. Once you get used to it, doing things this way can be a lot faster than switching between the mouse and the keyboard, pointing and clicking.

We're going to go over a few very important commands that will be particularly useful to us, but you should feel free to refer back to this **cheat sheet** as often as necessary.

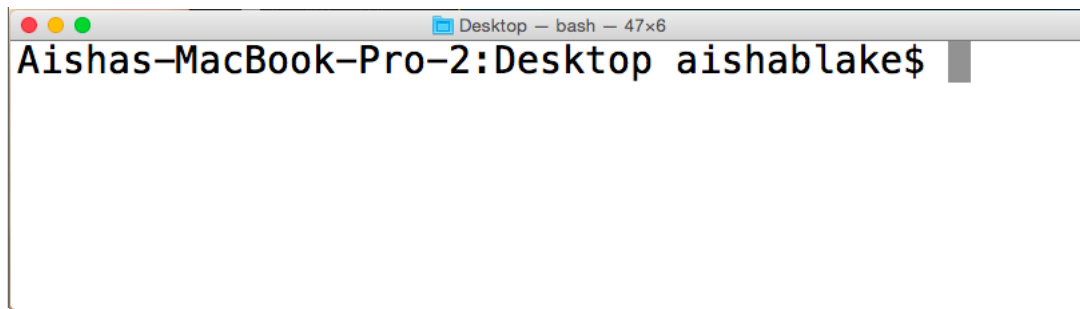
Windows users can use this **cheat sheet**.

NAVIGATION

CHANGE DIRECTORY

The `cd` command allows us to define a path to the directory we wish to navigate to. Executing this command will change where we are in the folder structure. This change should be reflected in our command prompt.

GETTING ORIENTED



The terminal prompt will give us some indication as to where we are within the folder structure.

CURRENT DIRECTORY

A single period (.) is used to indicate the current directory.

`ple.txt`

PARENT DIRECTORY

Two periods (..) indicates the parent directory. This is like moving one level "up" within the folder structure.

```
ther-example.txt
```

ROOT DIRECTORY

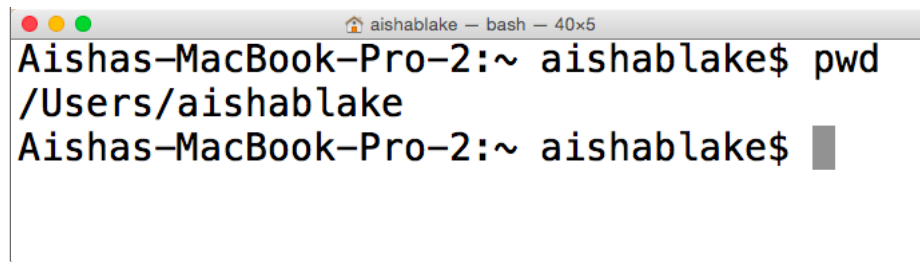
A tilde (~) is used to indicate the root directory. We're essentially saying "go back to the beginning and start from scratch". On a Mac, this is your Home.

INFORMATION

PRESENT WORKING DIRECTORY

The `pwd` command returns a path showing exactly where we are within the folder structure.

YOU ARE HERE

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, followed by a home icon and the text 'aishablake — bash — 40x5'. The terminal content shows the prompt 'Aishas-MacBook-Pro-2:~ aishablake\$' followed by the command 'pwd'. The output of the command is '/Users/aishablake'. Below the output, the prompt 'Aishas-MacBook-Pro-2:~ aishablake\$' is shown again with a grey cursor block at the end.

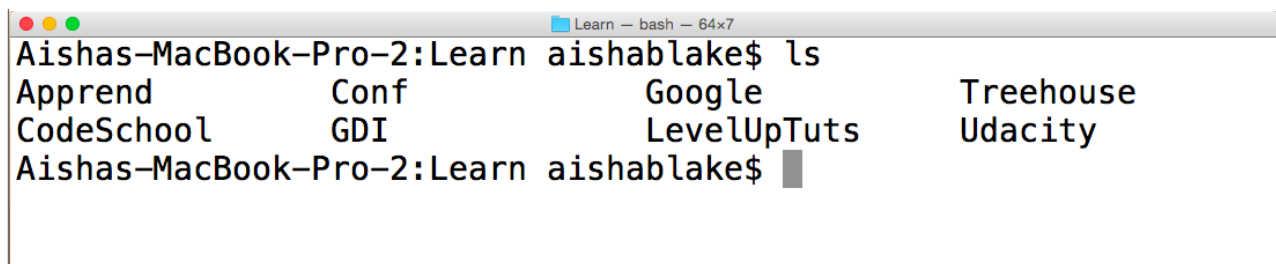
```
Aishas-MacBook-Pro-2:~ aishablake$ pwd
/Users/aishablake
Aishas-MacBook-Pro-2:~ aishablake$
```

LIST

Use `ls` to see the contents of the current directory.

Windows users can type the `dir` command.

TABLE OF CONTENTS



```
Learn — bash — 64x7
Aishas-MacBook-Pro-2:Learn aishablake$ ls
Apprend          Conf            Google          Treehouse
CodeSchool       GDI             LevelUpTuts     Udacity
Aishas-MacBook-Pro-2:Learn aishablake$
```

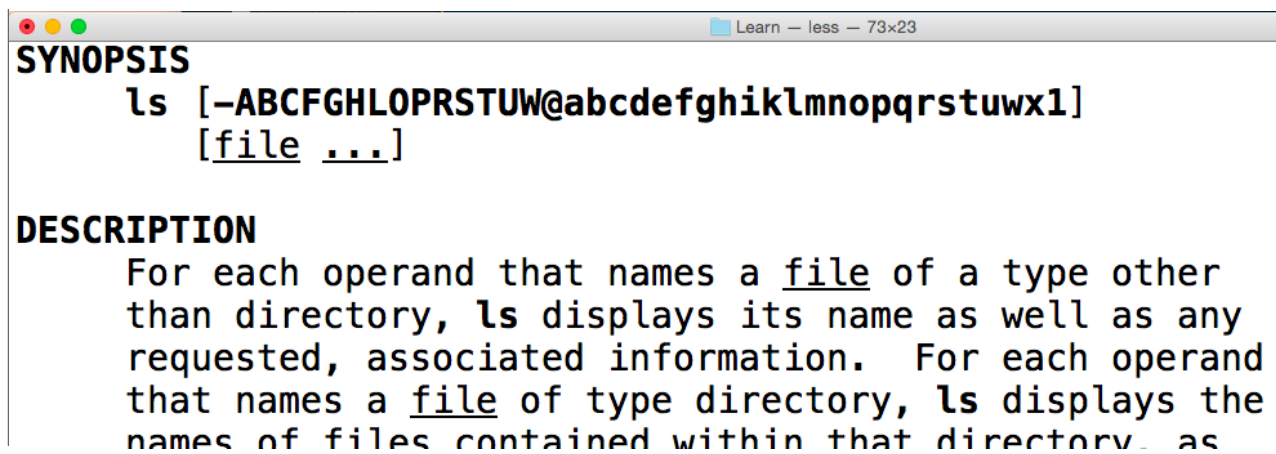
HELP

When you're not sure what your options are, typing `help` will list all possible commands.

MANUAL

You've figured out what you *can* say, but not what the commands do. Use `man` with any command to see the manual for that command. Hit the 'q' key to escape.

NOBODY READS THE MANUAL



The image shows a terminal window titled "Learn — less — 73x23". The window displays the synopsis and description of the 'ls' command. The synopsis shows the command 'ls' followed by options in brackets: [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz1] and arguments in brackets: [file ...]. The description explains that for each operand that is a file (not a directory), 'ls' displays its name and any requested information. For each operand that is a directory, 'ls' displays the names of files contained within that directory.

```
SYNOPSIS  
ls [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz1]  
    [file ...]  
  
DESCRIPTION  
For each operand that names a file of a type other  
than directory, ls displays its name as well as any  
requested, associated information. For each operand  
that names a file of type directory, ls displays the  
names of files contained within that directory. as
```

CREATION

NEW FILE

Create a new file by typing the `touch` command followed by the name of the file you wish to create. That file will be added to the current directory.

```
le.txt
```

Windows users can type the `echo [string] > [file name]` command.

NEW FOLDER

Create a new directory (or folder) by typing the `mkdir` command followed by the name of the folder you wish to create. That folder will be added to the current directory.

ff

PRO TIPS

PREVIOUS COMMANDS

We can cycle through any previously executed commands using the up and down arrows. This is useful for a number of reasons:

- Correcting small errors in long or complicated commands
- Retyping frequently used commands
- Recalling the steps that led to the current state

MULTIPLE COMMANDS

We can string multiple commands together with two ampersands (&&). These commands will be executed in order.

```
ff && cd my-stuff
```

CLEARING THE SCREEN

The `clear` command (you guessed it) *clears* the terminal window. We can also use the keyboard shortcut `Cmd+K` on a Mac.

Windows users can type the `cls` command.

VERSION CONTROL

Version control (sometimes called source control) is a system that manages changes to a program, website, or other collection of files.

Version control facilitates two key processes in development:

- Collaboration
- Managing changes to the codebase

COLLABORATION

Version control allows you to work on the same project simultaneously with other people and helps to avoid and manage the conflicting changes.

MANAGING CHANGES

Version control systems allow for seamless integration of new code, easy viewing of previous changes, and makes it easy to undo a change set. It also allows us to revert documents back to a previous state.

Kinds of version control:

- Centralized
- Distributed

Centralized version control systems are run on one central server infrastructure. Each collaborator checks out the code from and merges changes into the main server.

Distributed version control systems allow each collaborator to maintain a separate repository of the code which can be periodically reconciled with other peer repos.

VOCABULARY

WHAT DOES IT ALL MEAN?!

We've perhaps already thrown around a couple of words that didn't make sense at the time. Well, now is the time for all to become clear!

Note Remember that you can (and should) always just ask if we say something you don't understand!

REPO

A location where data is stored and managed

UNTRACKED

When we talk about "untracked" files, we mean that Git isn't tracking the changes made to such files. It's aware that they exist, but that's about it.

TRACKED

Tracked files are monitored by Git. It keeps tabs on every little change made to tracked files.

LOCAL

As with local files, a local repo exists on your computer. It is your personal copy of the project.

REMOTE

The remote repo is one hosted online. For our purposes, this is the copy of the repo that is hosted on GitHub.

GIT, GITHUB & THE COMMAND LINE



GIT & GITHUB

Git is an open source, distributed version control system originally developed by Linus Torvalds.

GitHub is a social coding service that offers hosting for software projects that use Git as their source control.

GIT & GITHUB

The combination of the two have become the gold standard in the OSS community and startup scene. It is gaining major ground in the enterprise space as well.

GIT

Open your command prompt:

- Command Prompt in Windows
- Terminal in OSX

INSTALLATION

VERIFY YOUR INSTALL

n

If you don't get some kind of error, you're good.

```
git version 2.4.1
```

IF YOU'RE NOT GOOD

This was covered in the pre-work, but just in case...

If you've never used Git before, you may need to download it. If you're on a Mac, you'll already have Git installed out of the box but may want to upgrade to the latest stable version.

Get Git!

While we're at it, if you haven't already...

Create a GitHub account!

BASIC CONFIGURATION

```
-global user.name "Your Name"  
alt name for git to use when you make a commit  
  
-global user.email "you@email.com"  
alt email for git to use when you make a commit
```

GIT COMMANDS

8 GIT COMMANDS YOU NEED

This will cover most of what you need to do on a daily basis.

- `git init`
- `git clone`
- `git status`
- `git add`
- `git commit -m`
- `git push`
- `git pull`
- `git reset`

GIT INIT

Creates a new git repository in the current folder including all child folders

```
objects folder >  
mo && cd git-demo
```

NEW FILES

Create a new file in the new empty repo.

Call it < yourname >.js

js

GIT STATUS

Reports the current status of the repo, such as whether any files have been modified or new files have been created.

GIT ADD

Add untracked files to the repo AND adds a tracked file's changes to the staging area making it ready to be committed. `git add` must be called with a parameter that is the path to the file(s) you wish to add.

`a.js`

STAGING AREA

There are 4 states a file can exist in a repo.

- Untracked
- Tracked
- Changed
- Staged
- Committed (which is really just back to 'Tracked')

GIT ADD

- After you add the file, check your status again. You should see that the file is now being tracked and is ready to be committed.
- Change the file in your editor in some way and save it. Run another `git status`. What happened?
- `git add` the file again. Check your `git status` again. What happened?

GIT COMMIT -M < MESSAGE >

- Once you have code staged, you can commit the change to your repo. This will create an anchor point that you can build from or return to if needed.

Protip: Don't forget the `-m`. If you do, you will be thrown into an in-terminal editor and that will ruin your whole day.

GIT COMMIT -M

Commit the new file to your git repo.

```
m "I added a new file!"
```

I check my status like a crazy person. Usually to make sure I'm not committing something I don't want to. `git status` will be one of your most used commands.

BONUS!

`git log` lets you view your repo's commit history

(you may need to press 'q' to exit the log)

There are tons of options you can add to the `git log` command to customize how it looks.

GITHUB

GITHUB

In order to push our code to GitHub, we will need to create a remote repository for i to live.

Code with me!

PUSHING TO A REMOTE

Once your changes are committed locally and you have a remote repository you can push to, you can do a `git push` to push your code to that repo.

PUSHING TO A REMOTE

`git push` pushes a local repo to a remote location.

```
git push
```

PULLING FROM A REMOTE

`git pull` pulls changes from a remote repo and attempts to merge them with your local changes. You must be in sync with the latest changes to a remote before you can push changes up to it.

```
git pull
```

CLONING A REPO

`git clone` creates a local copy of a remote repository in your local file system allowing you to make local changes. Note: This will make a new directory of the repo's root folder whenever you run this command.

(do this in another directory, not your current project.)

```
tps://github.com/kroysemaj/git-demo
```

UNDOING BAD THINGS

`git checkout` allows you to remove the changes to a file before it has been staged.

Make some changes to your js file and save them.

`aisha.js`

All of your changes will be reset to the state of the file in the last commit.

UNDOING BAD THINGS

`git reset` is your major league do-over button. You can undo changes you've made or completely destroy all of your changes if you make a real mess of things.

Make some more changes to your js file. Now stage them.

```
AD aisha.js
```

This will discard all of the staged changes to your file.

BRANCHES

GO YOUR OWN WAY

There will be times that you want to make a major change to your code. It's generally best to keep these types of changes separate from your working version, especially if it's live!

HOW IT WORKS

Think of the history of your project like a tree with branches that can diverge and merge back in with the main trunk (or master branch). We can have branches for new features we're working on, branches to separate development and production code, even branches for different team members.

GIT BRANCH < BRANCH >

Create a new branch with the specified name.

```
ool-feature
```

GIT CHECKOUT < BRANCH >

Switch over to the specified existing branch.

```
cool-feature
```

GIT CHECKOUT -B < BRANCH >

This is a shortcut! With just one line, we can create a new branch and switch over to it.

```
-b cool-feature
```

MERGING

WHEN TO MERGE

At some point, you'll want to move all your glorious changes stored in the new branch to your master branch so that the world can benefit from your genius. This is the point at which you must merge.

GIT MERGE < BRANCH >

Merge the specified branch into the current branch.

```
master  
ol-feature
```

CONFLICT RESOLUTION

Depending on which files were changed within the branch, merge conflicts may arise. If the same file has been changed in the same place on two different branches, Git won't know what to do and will defer to your judgment.

GETTING YOUR HANDS DIRTY

To fix a merge conflict, we need to dive into the code and manually delete the parts we don't want.

```
1 <<<<<< HEAD
2
3 Here is the original change.
4 =====
5 Here is the modified change.
6 >>>>>> 58326c301d09b58f3ac23d616e73f7b478424cc5
7
```

CLEANUP

Once we've made the hard decisions, we just need to add and commit the files we've changed just as in our normal Git workflow!

FURTHER READING

If you want more info on branches, checkout (see what I did there?) this **tutorial!**

WRAP UP

Git is awesome and you will be learning it in the best way-- you're being forced to. We will use Git for all of our project work. I encourage you to do the same for your personal projects and labs as well.

OTHER RESOURCES

If you need a refresher, trying going back to these resources:

- **git immersion**
- **Try Git**

EXERCISE!

INSTRUCTIONS

Break into groups and build a skeleton of a seat reservation app.

1. Create a repo for the project.
2. Each member should clone the repo to their local machine.
3. As a group, begin building out the HTML and CSS according to the instructions (two slides from now).

SEAT RESERVATION APP

You own a small theatre. With just 24 seats, it's a very intimate space. You want to create a simple app that will allow users to reserve seats for the upcoming show. Since you really don't have much money, this will be a simple project.

INSTRUCTIONS

- Display each seat as a small box in the browser. There should be 24 arranged in rows and columns. The number of rows and columns is up to you, but should remain the same no matter the width of the viewport.
- There should be some visible indication that certain seats are available and others are not.

INSTRUCTIONS (CONT.)

- Clicking on available seats should cause a simple form to appear on the page below the boxes.
- The form should ask for the user's name and email address.
- Include at least one heading, a brief paragraph of instructions, and at least one image.

FIGURE IT OUT

Find an open source project that needs help. Search for beginner-friendly projects. Often, this will mean making small changes to the front-end, fixing typos, or working on documentation. Fork the project, solve an issue, and make a pull request.