

Pro Dev Session



Pro Dev Session

You may want to write this down.

Collaboration is an essential part of being a great teammate and developer. To be a good collaborator, we must master our collaboration tools, namely git and GitHub.

Today we will get more practice with git.



Pro Dev Session

You may want to write this down.

We previously looked at the `add`, `commit`, `log`, and `checkout` commands.

Today we will continue down in this learning journey by learning the `status`, `stash`, and `branch` commands.

Pro Dev Session

You may want to write this down.

The `status` command allows us to see which files are being tracked, which changes have been committed, which branch we are on, and more.

This is a command you should use often, especially when you run into errors and you aren't sure what's wrong.

Pro Dev Session

You may want to write this down.

The `stash` command allows you to temporarily save work that you don't want to save forever.

Remember when you used the `commit` command to permanently save your project exactly as it was at that moment in time? Well `stash` is like `commit`'s baby sister. It saves your work but more temporarily and without a record that's tied to your project.

Pro Dev Session

You may want to write this down.

But why would you want to save something temporarily? To understand this, we first need to take a detour and understand branching.

Git allows us to create isolated areas to work on new features for our project. This allows developers to safely build new functionality without fear of messing up their master copy or interfering with other features that they may be working on.

These isolated areas are called branches. You can create one using the `branch` command.

Pro Dev Session

You may want to write this down.

On an actual project, you'll often have many working branches on your machine at once. The clean working production ready code that mirrors what's on GitHub is in a branch named master. You will get to name all of the additional branches that you add.


Pro Dev Session

You may want to write this down.

So why would you want to save something temporarily?

Git fundamentally does not want you to lose things, so it won't let you move from one branch to another without some form of saving. If you want to change branches but your work is in a messy state that you don't want to permanently save or you're on a branch that you would like to leave so you can delete it, you still have to save your work first.

The `stash` command allows you to save it in a non-permanent way.



Pro Dev Session

You may want to write this down.

Sidebar: Git takes time to master. We'll be going back over the concepts from the last 2 days again in the next class before we add any significant new knowledge.

Git Practice

Work individually but with the help of your classmates to complete all of the goals below.

Goals:

- Create a new repository on GitHub and clone it to your local machine.
- Create a main method and add some print statements.
- Assume this is your working production ready code and your boss asks you to add a new chat feature.
- Type `git branch chat` to create a new branch called chat (This allows you to safely build the chat feature without risk of messing up the production ready code).
- The previous command created the new branch but you aren't on it. Type `git status`. This will show you the branch you are on and which files are unmonitored (not added) and which monitored changes are not committed.
- Type `git add -A` and `git commit -m "initial commit"`.
- Now that you have committed your changes, you can change branches. Type `git checkout chat`.
- You are now on the chat branch. Make some changes in the main method. Add and commit these changes.
- Make some more changes. Do not commit them.
- Suppose your boss decides to scrap the chat feature. Let's go back to master. Type `git checkout master`.
- You can't change branches. Use the `status` command to see what's going on? You have changes that have not been saved, but you don't want to permanently save them.
- Type `git stash` then `git checkout master`. You are now back in your master branch!



**20
minutes!**

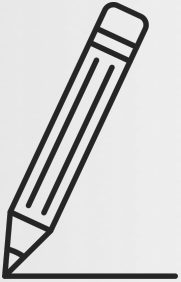
Stand Up!



Review

Review Time

TAMING THE TORNADO OF INFORMATION



Notebooks Ready? It's time for a review.





Stay Seated & Take 3 Deep Breaths.

RELAX.

Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back in a few minutes.

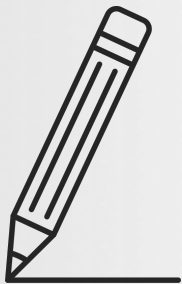


Full-Stack



Full Stack

A LOOK AHEAD



Notebooks Ready? It's time for a little lecture.



Full Stack

A LOOK AHEAD

Recall that APIs are an interface through which separate systems interact.

The APIs that we've been building allow client computers to interact with our server (and ultimately our database)

We're going to take a short detour to see how these two systems work together.



Full Stack

A LOOK AHEAD

This syntax may be mostly unfamiliar to you. Don't worry. We'll spend weeks learning this stuff. Today is just a preview of what's to come.

This is about understanding how your learnings thus far fit into the larger picture of web development.



Full Stack

A LOOK AHEAD

Recall that, the client makes a request:



Server

GET `"/products"`



Client

Full Stack

A LOOK AHEAD

...and the server sends a response. This response is usually in JSON format.



Server

Some JSON response

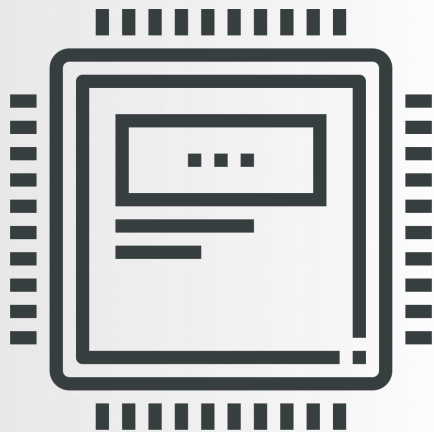


Client

Full Stack

A LOOK AHEAD

The browser renders this response data in a meaningful way for the user.



Server



Client

lunch.

Practice

API Practice

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in TEAMS to complete all of the goals below.

Goals:

- Complete all the requirements in the slacked out instructions.



**60
minutes!**



Stay Seated & Take 3 Deep Breaths.


RELAX.

Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back shortly.



Sneak Peek

Note that next class will start with the sneak peek topic. You are NOT expected to master this today.



Error Handling

A LOOK AHEAD

Next class we will begin to explore error handling. You've already seen a sneak peek of this concept.

What errors do you think could occur in your Cinder application?





Stay Seated & Take 3 Deep Breaths.

RELAX.

Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back shortly.



Wrap Up

Module 3 Lesson 3

HOMEWORK

You don't have to submit your nightly homework, but you are expected to complete it.

.

Continue working on Cinder App





No Daily Assessment

You're off the hook! But please hang around for help.

