GIT BASICS

SEE NYCDA.COM/GIT FOR A STEP-BY-STEP GUIDE

LEARNING OBJECTIVE

> UNDERSTAND WHAT `git` IS AND WHERE IT IS USED

> MAKE YOUR FIRST, "COMMIT"

GETTING SETUP

PLEASE DOWNLOAD GIT, IF YOU DON'T ALREADY HAVE IT:
HTTP://GIT-SCM.COM/DOWNLOADS

IF YOU ARE A WINDOWS USER, USE GIT BASH (INCLUDED WITH GIT) FOR ALL TERMINAL AND GIT INTERACTIONS. TO LAUNCH, START MENU -> PROGRAMS -> GIT -> GIT BASH

WHERE IS GIT?

> ONCE YOU'VE INSTALLED GIT, YOU CAN VERIFY IT HAS BEEN INSTALLED BY OPENING UP THE TERMINAL AND TYPING `git`

```
$ git
```

> IF YOU SEE A 'COMMAND NOT RECOGNIZED' ERROR, YOU PROBABLY HAVEN'T INSTALLED GIT

> WE CAN SOLVE YOUR ISSUES ONCE THE LECTURE IS OVER - LOOK ALONG WITH A FELLOW STUDENT FOR NOW!

WHAT IS `git`?

> A VERSION CONTROL SYSTEM MEANT TO MAKE IT EASIER TO HAVE MULTIPLE VERSIONS OF CODE, SOMETIMES ACROSS MULTIPLE DEVELOPERS OR TEAMS

> AT ITS MOST SIMPLE, `git` HELPS WITH THE 'INDEXV1.HTML, INDEXV2.HTML, INDEXV3FINAL.HTML' PROBLEM

> AT ITS MOST COMPLEX, GIT ALLOWS DEVELOPERS TO WORK TOGETHER WORLDWIDE ON CODE WITHOUT STEPPING ON EACH OTHER'S TOES

WHY LEARN git?

> ALLOWS YOU TO SEE THE CHANGES YOU MAKE TO YOUR CODE AND EASILY REVERT THEM

> BY PUSHING YOUR GIT REPOSITORIES TO GITHUB, YOU'LL HAVE A DEVELOPER PORTFOLIO AUTOMAGICALLY!

> YOU'LL EVENTUALLY PUSH YOUR CODE TO A REMOTE SERVER USING git - YOUR APPLICATION'S HOST, SO YOU CAN RELEASE YOUR WEBPAGE OR WEB APPLICATION TO THE WORLD

DON'T GET CONFUSED - `git` VS GITHUB.COM

> **GIT** IS A VERSION CONTROL SYSTEM THAT TAKES SNAPSHOTS OF YOUR CODE AT CERTAIN POINTS IN DEVELOPMENT

> THESE SNAPSHOTS ARE STORED IN A 'REPO', OR 'REPOSITORY' ON YOUR LOCAL MACHINE

> **GITHUB.COM** IS A WEBSITE THAT HOSTS GIT REPOSITORIES ON A REMOTE SERVER

> WE'LL BE ABLE TO TROUBLESHOOT YOUR ISSUES MORE EASILY IF YOU'RE ABLE TO IDENTIFY WHICH OF THESE TWO SEPARATE THINGS YOU'RE HAVING TROUBLE WITH

CONFIGURING GIT

YOUR COMMITS WILL HAVE YOUR NAME AND EMAIL ATTACHED TO THEM. BEFORE WE PUT ANYTHING UP IN PUBLIC, LET'S MAKE SURE THEY'RE CORRECT!

```
$ git config --global user.name
> should be your name, i.e. Zach Feldman
$ git config --global user.email
>should be your email, i.e. zach@nycda.com
```

TO FIX EITHER, JUST ADD THE DESIRED VALUE IN QUOTES AFTER THE COMMAND:

```
$ git config --global user.name "Zach Feldman"
$ git config --global user.email "zachfeldman@gmail.com"
```

GETTING STARTED

> BEFORE USING `git`, YOU'LL NEED A PROJECT TO USE IT WITH

> A PROJECT IS ANY DIRECTORY (FOLDER) FULL OF FILES - THIS COULD BE A WEBSITE, WEB APPLICATION, OR REALLY ANY COLLECTION OF FILES

> USE THE `cd` COMMAND TO GET TO THE DIRECTORY YOU'LL BE USING GIT IN

GETTING STARTED: `git init`

> ONCE YOU'RE IN YOUR PROJECT'S DIRECTORY, RUN THE `git init` COMMAND TO INITIALIZE GIT

`$ git init`

> YOU ONLY NEED TO RUN THIS COMMAND ONCE PER PROJECT

> UNSURE IF YOU'VE INITIALIZED YET? TRY RUNNING `git status` - IF YOU GET A `fatal error`, YOU HAVEN'T RUN `git init` YET

MAKING YOUR FIRST COMMIT

> A COMMIT IS A 'SNAPSHOT' OF YOUR PROJECT AT A CERTAIN TIME

> IT TRACKS ALL OF THE CHANGES YOU'VE MADE SINCE THE LAST COMMIT

> TWO STEPS TO MAKE THE COMMIT:

> STAGE YOUR FILES: CHOOSE WHICH FILES TO COMMIT

> COMMIT TELL GIT TO MAKE YOUR COMMIT WITH AN ACCOMPANYING DESCRIPTIVE MESSAGE

STAGING FILES

> TO STAGE AN INDIVIDUAL FILE, USE `git add`

```
$ git add index.html
```

> TO STAGE EVERY FILE/CHANGE SINCE YOUR LAST COMMIT, JUST USE -A OR '.'

```
$ git add -A
-- OR --
$ git add .
```

MAKING A COMMIT

> ONCE YOUR FILES ARE STAGED, YOU CAN MAKE A COMMIT USING THE `git commit` COMMAND

> ALWAYS USE THE `-m` FLAG TO ADD A DESCRIPTIVE COMMIT MESSAGE

```
$ git commit -m "Initial commit."
```

TIP: USE DESCRIPTIVE COMMIT MESSAGES

IF YOU USE DESCRIPTIVE COMMIT MESSAGES, IT'LL BE MUCH EASIER TO SEE HOW YOUR PROJECT HAS PROGRESSED LATER ON

```
$ git commit -m "Initial commit."

$ git commit -m "Added 'about' to the navigation bar and a page for it."

$ git commit -m "Closes #15 by adding a blue background on hover."
```

TIP: IF YOU FORGET A COMMIT MESSAGE...

IF YOU FORGET TO ADD A COMMIT MESSAGE AND HIT ENTER AFTER
`git commit`

`$ git commit [Enter]`

YOU MAY END UP IN WHAT'S CALLED VIM. TO EXIT THIS, SIMPLY HIT
THE ESCAPE KEY.
THEN TYPE THE FOLLOWING.

`$ :q!`

git status

> YOU CAN ALWAYS RUN THIS COMMAND 'FOR FREE', MEANING WITH NO REPERCUSSIONS

> WILL HELP YOU FIGURE OUT IF

> `git` IS INITIALIZED FOR THIS PROJECT

> WHAT FILES ARE STAGED

> WHAT FILES HAVE BEEN CHANGED SINCE THE LAST COMMIT

TIP: YOU NEED FILES!

> A COMMON BEGINNER MISTAKE IS TO GET EXCITED ABOUT STARTING YOUR NEW PROJECT AND TRY TO `git init` BEFORE YOU HAVE ANY FILES

> GIT DOESN'T WORK UNLESS YOU HAVE FILES TO TRACK!

TIP: USE `git log`

> CURIOUS WHAT COMMITS HAVE BEEN MADE SO FAR?

> `git log` WILL SHOW YOU WHAT COMMITS HAVE BEEN MADE SO FAR

```
$ git log
commit 4038fb143edfc068264479cce855619730d6edca
Author: Zach Feldman <zach@nycda.com>
Date:   Tue Nov 25 17:05:28 2014 -0500

    GA tracking stuff.

commit 74ee59894ef22fd714bf3ffb06f2ef4cf43be0bc
Merge: de4b141 6c991aa
Author: Zach Feldman <zachfeldman@gmail.com>
Date:   Tue Nov 25 13:01:54 2014 -0500

    Merge pull request #201 from nycda/classes-page-custom-field-fix
```

EXERCISE: GETTING TO KNOW GIT

> CREATE A NEW PROJECT FOLDER WITH AT LEAST TWO FILES, PERHAPS `index.html` AND `style.css`

> INITIALIZE A NEW GIT REPOSITORY IN THIS FOLDER

> MAKE YOUR INITIAL COMMIT

> MAKE A CHANGE TO ONE OR BOTH OF THE FILES, THEN MAKE ANOTHER COMMIT

> REPEAT THIS PROCESS STARTING FROM SCRATCH A FEW TIMES TO MAKE SURE YOU HAVE IT DOWN

RESOURCES

CODECADEMY

LEARN GIT

TEAMTREEHOUSE

GIT BASICS – GETTING STARTED WITH GIT

MORE

QUIZ

1. WHAT IS `git` AND WHAT IS ITS PRIMARY USE?

2. NAME A GIT COMMAND THAT CAN ALWAYS BE RUN WITH NO CONSEQUENCES.

3. WHAT ARE THE STEPS TO MAKE YOUR FIRST COMMIT IN A PROJECT?