Before we begin...

- Open up these slides:
  - https://slides.com/threequal/jsd_01_command_line

Git, JS & the Command Line

Learning Objectives

- **Explain** the terminal and what it is used for
- **Use** commands to perform common tasks using the terminal
- **Understand** the role of Git and how to **use** it effectively
- **Create** the file structure for this course
- **Use** GitHub to remotely store repository changes
- **Explain** JavaScript and its use
- **Run** basic JavaScript on the command line with Node

Agenda

- Terminal: Background & Usage
- Git: Background & Usage
- GitHub: Background & Usage
- JavaScript: Background
- Basic JavaScript with Node time!

A quick review

Terminal

GENERAL ASSEMBLY

What is it?

A way to manipulate and interact with your computer

It's entirely text-based

Not the **W.I.M.P** (Windows, Icons, Menus and Pointers) style!

Why use it?

- It's (eventually) very fast
- It's automatable and flexible
- No interruptions
- It gives us what we expect
- Sometimes it is the only way
    - Command Line Interaction (C.L.I.)
    - Web servers

How do we use it?

Well, let's talk about how a computer actually works...

But, basically we interact with a **Shell** (we will be using the Bash Shell)

The Bash Shell?

- Bash is a regular program on your computer
- It was created to take commands from you
  - We talk to it using the **Bash Shell Language**
- When I say "shell", it's just that program we were talking about before
  - It's an interface to interact with other programs

Are there other shells?

- Bash "Bourne Again shell"
- C shell
- Z shell "zsh"
- Korn shell
- Bourne shell
- Debian's Almquist shell "dash"

What can you do with it?

Most of you will have a lot of experience with the WIMP (Windows, Icons, Menus, Pointer) style of system

That's not the only way. We are going to be using a text-only "console" or "terminal"

This is going to seem alien and primitive but you will soon see the power!

What can you do with it?

- Anything! Run programs to make all sorts of changes
  - Editing files and images
  - Converting files between types
- Creating back-ups
- Making and copying files
- Downloading, compiling, and running programs
- We can do a lot more with the Terminal
- `telnet towel.blinkenlights.nl`

How do you work with it?

- Non-interactively
  - Running scripts. We are already doing this!
- Interactively
  - Opening up a **REPL**

Common Commands

```
pwd         # Where am I? The programmer's "um"
ls          # List all files in the current directory
cd          # Change Directories
mkdir       # Make a Directory
rmdir       # Remove an empty directory
rm          # Remove a file or a directory
touch       # Create a file
open        # Open a file in the default application
code        # Open the VSCode Editor (atom will open in Atom)
say         # Make your computer talk
```

A Tiny Bit of Markdown

What is it?

- A plain text format
- An easy way to generate HTML
- Most commonly ends in the file extension ***.md***
- GitHub and Slack both use it
- See here and here for an introduction to it
- Why am I showing this...

The Basics

```
# Main Heading
## Sub-Heading

This is a paragraph.

**This is bold text**

_This is italicized text_

* This is a list item
* This is the second list item
```

The Basics

~~~~
`  4; `

```

Code can go here!
```


```js
// You can optionally add a language

var isThisJS = true;
```
~~~~

Exercise

Set up your file structure for this class using only the terminal!

```
JavaScript_Course/

    class_00/
        README.md
        homework/

    class_01/
        README.md
        homework/

    ...
```

Resources

Resources

- Watch these Code Academy videos
- Read these
  - Quick Left's Tutorials - start from the bottom!
  - Learn CLI the Hard Way
- Track down the Terminal City Murderer
- Some other useful links
  - 40 Terminal Tricks and Tips
  - 25 Useful Find Examples
  - Terminal Cheatsheet

Git

GENERAL ASSEMBLY

History

History of Git

- Made in 2005 by Linus Torvalds
  - Before that, he made the Linux Kernel
  - Here is a Ted talk
  - Here is his GitHub
  - Here is the source code for Git

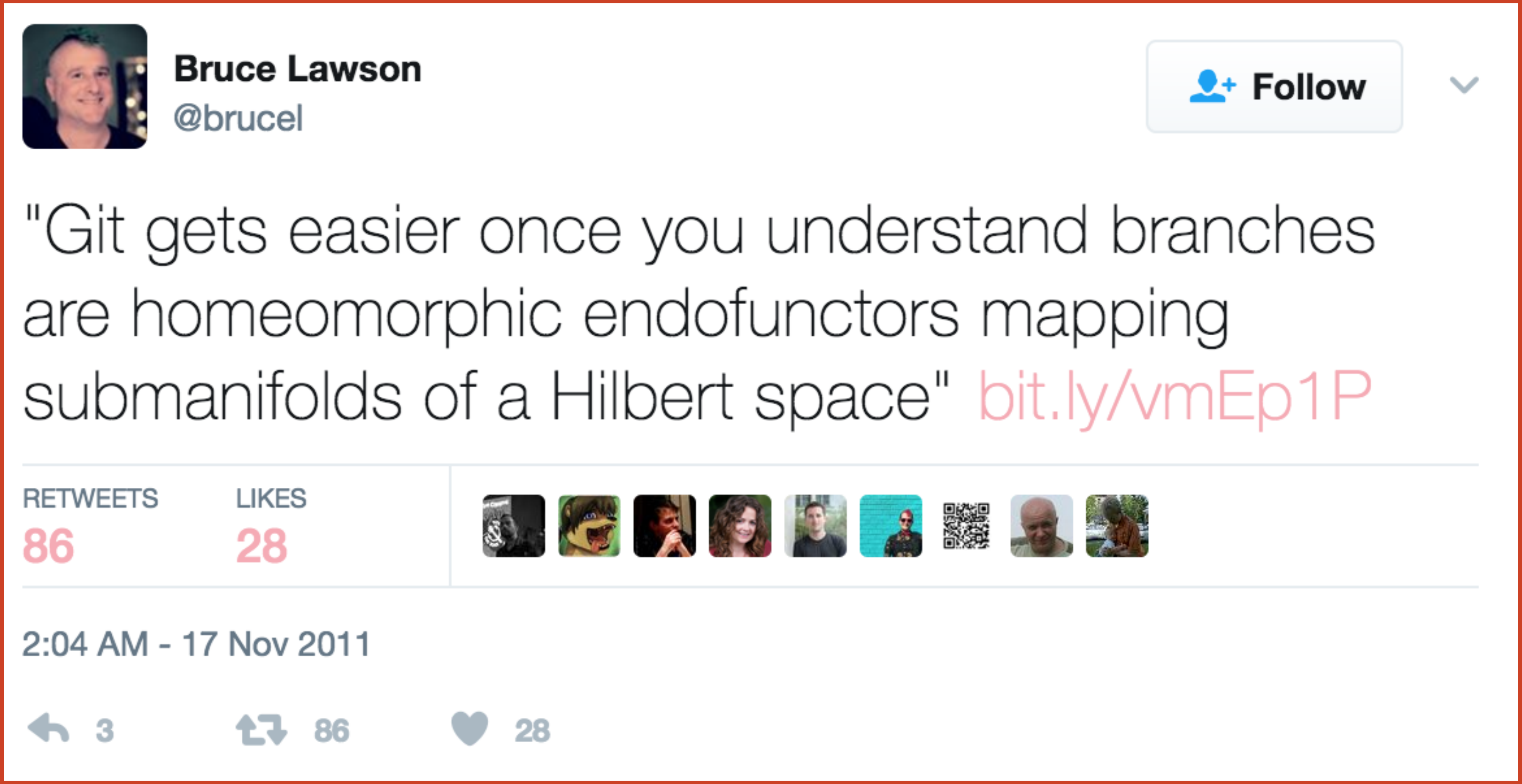
Why is it called Git?

> *“ I'm an egotistical bastard, and I name all my projects after myself*
>
> *- Linus Torvalds*

A Warning!

Warning!

- ***"Git is infuriating"*** - Mandy Brown
- It takes a long time to feel comfortable with it
- Most explanations of it get very technical very quickly
    - Focus on the concepts



Bruce Lawson
@brucel

"Git gets easier once you understand branches are homeomorphic endofunctors mapping submanifolds of a Hilbert space" bit.ly/vmEp1P

RETWEETS 86
LIKES 28

2:04 AM - 17 Nov 2011

What is it?

What is it?

- A **version control system** (or **VCS**)
    - It takes snapshots of our projects
    - Gives us a project-wide undo button!
- A collaboration tool
    - It merges differences in our code for us
- A local development tool
- Supports non-linear development

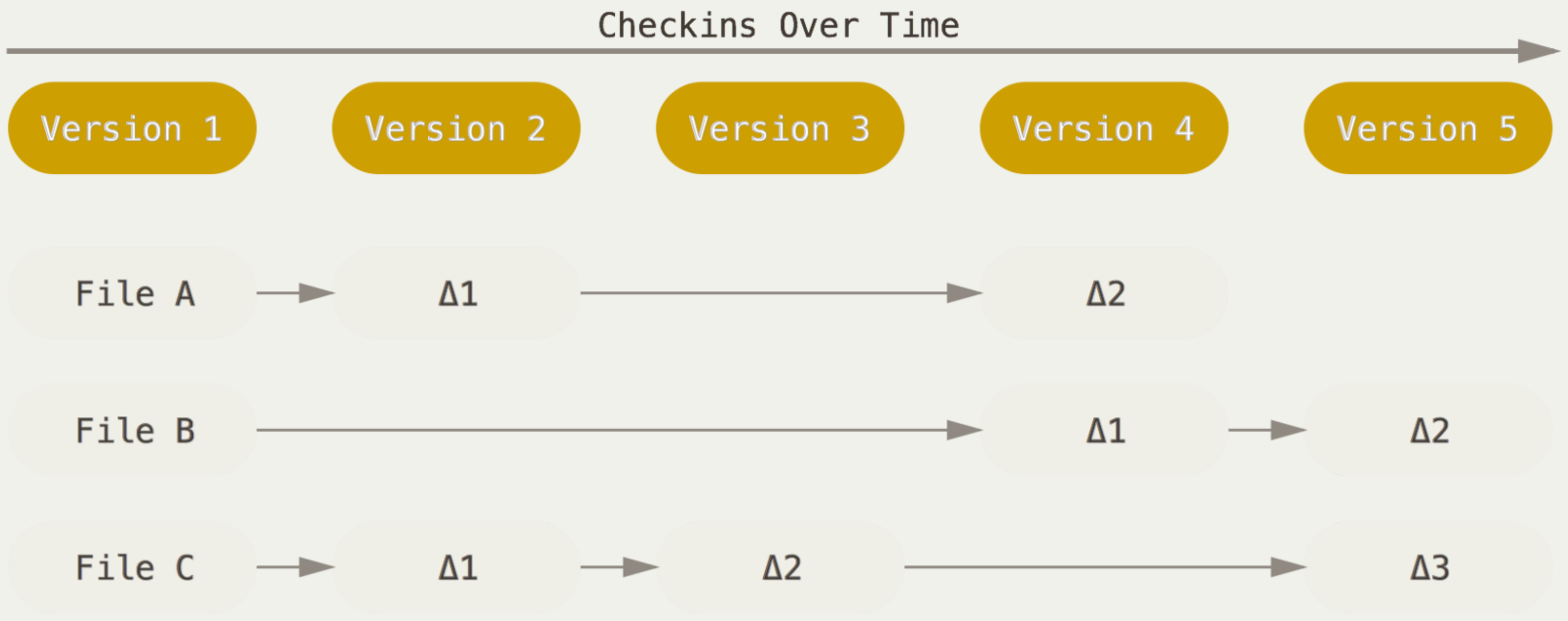
What is it?

It's a tool for modern-day teamwork - collaboration among people, working asynchronously, on a shared body of work.

It saves us from moving floppy disks around, or saving lots of copies of the one file.

More people === more likely to use it

Version Control



Why use it?

Why use it?

- You make a change and realise it was a horrible mistake? ***Git can undo it***
- You want to figure out where everything went wrong? ***Git will show you***
- You want to try out a new innovative feature that will probably destroy everything? ***Git can protect you***
- You want to work with a bunch of people? ***Git will make that easier***

Remember the audience

The ultimate audience of Git is you.

It takes a long time to get used to!

Concepts

Terminology

- **Repository** - A project
- **Branch** - A version of your project
- **Origin** - A place where your code is stored
- **Add** - Tell Git to pay attention to a file(s)
- **Commit** - Tell Git to take a snapshot of a file(s)
- **Push** - Tell Git to take all of the code that it has locally and put it up on GitHub

Terminology

- **Merge Conflict** - When two pieces of code can't be automatically merged, you get one of these - you need to decide what you want
- **Fork** - Your copy of someone else's GitHub repository
- **Pull Request** - When you request to have a project include your code
- **Clone** - When you take code from GitHub and get an exact local copy on your computer

How do we use it?

How to use it?

- The Command Line
- Applications
  - GitHub Desktop
  - SourceTree
  - GitKraken
  - Plus more

Git commands

```
git init

git add README.md

git add -A

git commit -m "Your commit message"

git status

git log

git reset --hard ...............
```

How to go back?

See here

- `git checkout ..................`
  - Temporarily go back to view a snapshot of your code
- `git reset --hard ............`
  - Delete the changes you have made and go back to a snapshot of your code

Exercise

Take the folder you created before, and turn it into a **git repository**

Make sure Git is keeping track of all files in the folder! (**Note**: it won't keep track of empty folders unless there is a ***.gitkeep*** file in the folder)

Resources

Resources

- Atlassian: Learn Git
- Official GitHub Git Tutorial
- CodeSchool
- Code Academy
- Git & GitHub for Poets
- Git For Humans

GitHub

GENERAL ASSEMBLY

What is it?

What is it?

- It is a website that uses Git behind the scenes to allow developers to share their projects amongst each other
- A **Graphical User Interface** (GUI)
- Helps us perform common tasks
- The Dropbox or Google Drive for code

Why do we use it?

Why do we use it?

- To share our code with other computers
- For collaboration (Pull Requests, Forks etc.)
- It acts as a portfolio
- To visualise what is going on
- As a project management tool (Projects)
- An error reporting system (Issues)
- Documentation (Wiki)
- Free hosting (GitHub Pages)

What will we use it for?

- For you:
  - You can access my code and slides
  - You can collaborate with each other
  - You can share your code effectively (debugging will be a lot easier)
- For me:
  - I can see your homework etc.

How do we use it?

How do we use it?

Once we have a local Git repository...

- Create a repository on GitHub
- We need to tell Git where the code should be stored
  - `git remote add origin URL`
- We need to push (or upload) all of the code
  - `git push origin master`

How do we use it?

Once we have a local Git repository...

- We need to pull (or download) all of the code
  - **`git pull origin master`**
- We can also clone a repository
  - **`git clone URL`**

A Typical Upload Workflow

```
git init
# Only necessary the first time

git add -A

git commit -m "Made changes"

git remote add origin URL
# Only necessary the first time

git pull origin master
# Check for code that you don't have

git push origin master
```

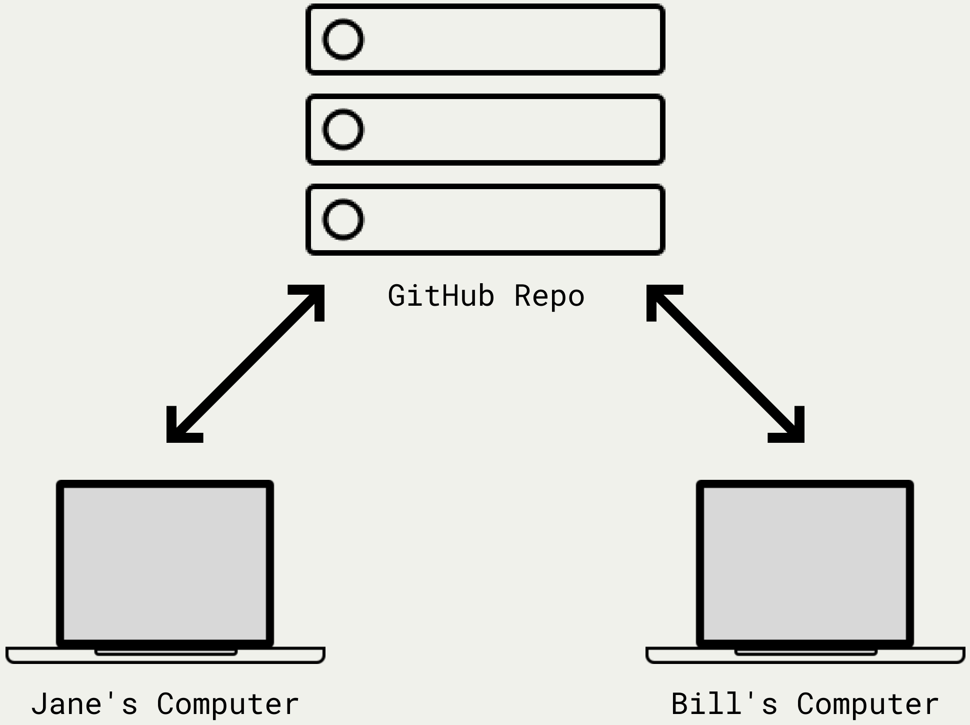
A Typical Clone Workflow

```
git clone URL
# Only necessary the first time

cd Directory
# Replace Directory with the project name

git pull origin master
# Every time you want to re-download
```

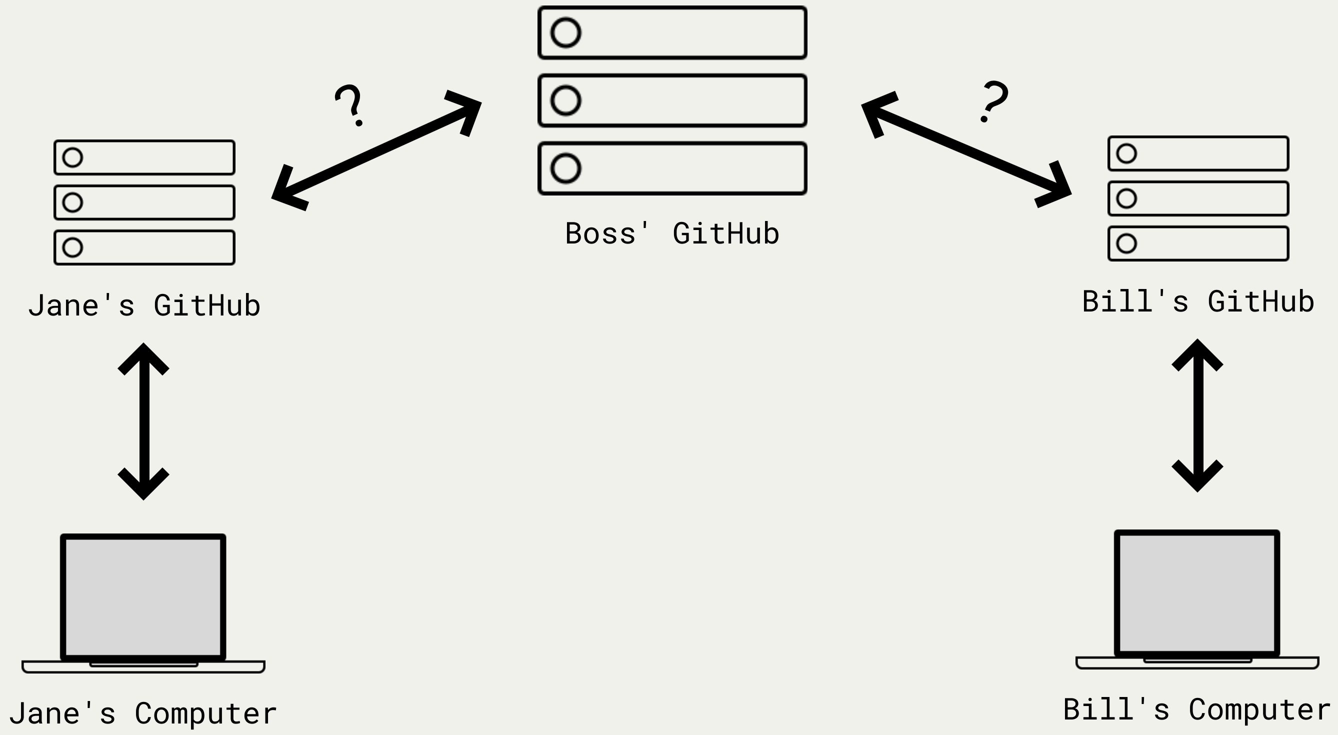
Note: This is how you will be getting my code from code and my slides (plus things from any extra sessions)

Collaboration Approaches

Origins



Forks & Pull Requests



Exercise

Take the Git repository you created earlier, and put it onto GitHub!

Once you have it on GitHub, add a file locally and upload that new file to GitHub

JavaScript Background

What is it?

What is it?

- The most popular programming language in the world (according to GitHub, GitHut and Stack Overflow)
- A very flexible language
  - In browsers
  - On the back end - Node.js
  - Lots of other places
- A "weird, poorly designed" language...
- ...That is everywhere

What is it?

It is **asynchronous**

- There are two main types of language:
  - Synchronous (runs one step at a time, waits for the previous line to complete - this is *blocking*)
  - Asynchronous (runs one step at a time, doesn't wait for the previous line to complete - this is non-*blocking*)

History

History

- Built in 10 days by Brendan Eich
  - Twitter
  - Github
- Released in May 1995
- Named Mocha -> LiveScript -> JavaScript
  - Just a marketing move!
- Current Version: ES2017
  - It's based on something called ECMAScript

Versioning

- **ES1** - 1997
- **ES2** - 1998
- **ES3** - 1999
- **ES3.1** - 2009 (renamed to ES5)
- **ES2015** - 2015 (also called ES6)
- **ES2016** - 2016
- **ES2017** - 2017
- **ES2018** - Soon!

What can it do?

What can it do?

- Validating information
- Live updating pages
- Adding interactivity
- Adding animations (e.g. TweenMax)
- Internet of Things and Hardware
- Visualise data (e.g. D3.js, DeckGL)
- Can be used for art (e.g. P5.js, PaperJS)
- 3D (e.g. ThreeJS, ReGL), Games (e.g. Phaser), AI, Augmented/Virtual Reality (e.g. Aframe, AR.js, MozVR)
- Plus more!

How does it do it?

How does it do it?

- JS gives us a/an:
  - Syntax
  - Data Types
  - Way to save, access and manipulate data
  - Way to use APIs

Data Types & Inheritance

- When we create data, we get:
  - Properties (accesses data)
  - Methods (runs an operation on data)

```
// Properties
"Hello".length;

// Methods
"Hello".toUpperCase();
```

Basic JavaScript with Node

GENERAL ASSEMBLY

What is Node?

- It's JavaScript, but on the back-end!
    - It is the ECMA Script syntax, but with access to the file system, databases etc.
- It's a program on our computer!
    - We run it using the terminal

A JavaScript Console

node

- We run this in our terminal
- It opens up a **REPL** (**R**ead, **E**valuate, **P**rint, **L**oop)
- **CTRL + D** exits this REPL!

Running a file

```
node main.js
```

- Once we have created our file (replace the name of the file), we can then execute the JS in the file
- To get something printed to the terminal, we need to use:
  - `console.log("Hello");`

Exercise

- Upload the code we just wrote to GitHub
- Send the link to your homework repository in Slack
- Bonus: Add **ga-wolf** as a collaborator to your homework repository

Homework

- Review the Terminal and Git
  - Do the Terminal City Murderer exercise (Optional)
- Look for JavaScript's presence on some of your favourite pages. Share some inspiration sites
- Prepare for next lesson!

Homework (Extra)

- Read You Don't Know JS: Types & Grammar
- Read Eloquent JavaScript
- Read Speaking JavaScript
- Do the JS video tutorials on CodeAcademy and Dash

What's next?

- Data Types in JavaScript
- Variables
- Conditionals & Loops

Questions?

Feedback time!

Lesson 01: ***Command Line***

https://ga.co/js05syd

Thanks!