A round up of useful things

This lecture covers:

- Web technologies, options in general
- Some debugging advice

All the lab solutions

- There are solution files available on Moodle
- Remember that they won’t run “out of the box”. You will at least need **npm install** to create node_modules
- Remember that Dynamic Web is a pre-requisite for this module. It is fine to use technologies that you used there.

Hidden files for configuration

- There are some files and folders on your computer that you might not be able to see.
- These files and folders start with a **.**
- Examples:

  .git

  .env

  .bashrc

Why should I care about hidden files?

.git

- Because if you ever want to get rid of that GitHub configuration…all you need to do is delete **.git**
- If you want to know what your git command lines are **actually** doing, it'll all be stored in there (but you will want the CLI to interpret it)

Package Managers

- We’ve been using **npm** throughout for node package management
- Packages are collections of script that expose some functionality
- There are alternatives: e.g. **yarn** (not covered in this module)

What does npm do?

- “Manages” packages!
- It downloads them from the node package repository to node_modules (for local install)
- It can install specific versions of packages, and it can uninstall them, too
- It makes the packages work. So if they need other packages (dependencies), it will download them, too.

npm packages for everything?

- The story of “leftpad” and how one open source developer’s flounce “broke the internet” in 2016: https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code
- Moral: *sometimes* you might want to write the code yourself rather than relying on a package (especially if it’s only 11 lines).

What does npx do?

- npx is for executing node package commands
- You can get the same thing from npm by declaring the script that you want to run inside the scripts section in your package.json file
- But npx can be faster and will even let you run things without downloading a local copy

React - what do I need to know?

- React is a package (written by Facebook)
- You get it from npmjs.com
- It’s for creating user interfaces (it’s front end)
- Other packages are available which help React do things (e.g. Next.js)

React

- React is component-based
- It efficiently renders objects on a UI, making use of a virtual DOM
- “Modern” React uses JSX (an abstraction of JavaScript that adds html) or ES6, and tools such as Babel **transpiles** this into JavaScript that can be interpreted by any browser.

React

- React is “stateful”: there is a stored state (accessed within component classes using this.state, or via *useState* hook)
- React takes properties as input (accessed within component classes using this.props)
- React components are now often functions in a single file (but can be classes in a single file)
- See more at https://reactjs.org

React (downsides)

- React gives a LOT of freedom in how you structure your code. So it can quickly end up a hot mess.

- React uses a virtual DOM. That makes it RAM-hungry on the client side. AND including the whole virtual DOM makes for a fat download.

- React is a library, and you gain some functionality by using other React packages. In a framework (like Angular), this functionality comes built in.

https://sredmond.medium.com/why-we-need-to-stop-using-react-1ae3ec7959a4

Scripts

- Scripts are just a series of instructions put together

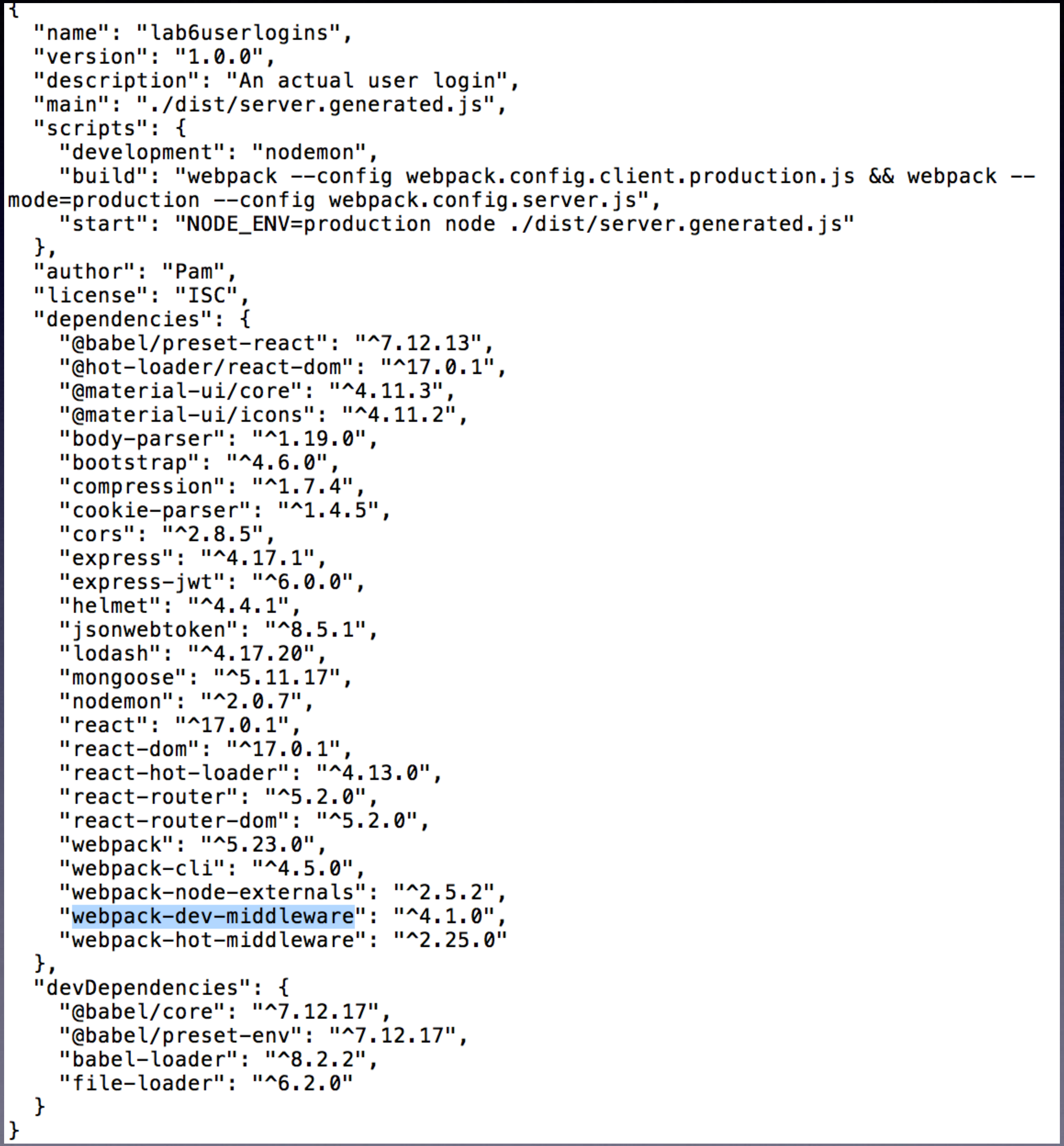
- The instructions execute one line after another

- You could do the same thing by typing the instructions one after another at the command line (but it’s boring and error prone)

- You can store scripts in a file (eg .sh .bat .ps1)

package.json Scripts

- YOU define them
- Pay attention because this is where tool chains are defined
- But you could get the same effect with npx



```
{
  "name": "lab6userlogins",
  "version": "1.0.0",
  "description": "An actual user login",
  "main": "./dist/server.generated.js",
  "scripts": {
    "development": "nodemon",
    "build": "webpack --config webpack.config.client.production.js && webpack --
mode=production --config webpack.config.server.js",
    "start": "NODE_ENV=production node ./dist/server.generated.js"
  },
  "author": "Pam",
  "license": "ISC",
  "dependencies": {
    "@babel/preset-react": "^7.12.13",
    "@hot-loader/react-dom": "^17.0.1",
    "@material-ui/core": "^4.11.3",
    "@material-ui/icons": "^4.11.2",
    "body-parser": "^1.19.0",
    "bootstrap": "^4.6.0",
    "compression": "^1.7.4",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "express-jwt": "^6.0.0",
    "helmet": "^4.4.1",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.20",
    "mongoose": "^5.11.17",
    "nodemon": "^2.0.7",
    "react": "^17.0.1",
    "react-dom": "^17.0.1",
    "react-hot-loader": "^4.13.0",
    "react-router": "^5.2.0",
    "react-router-dom": "^5.2.0",
    "webpack": "^5.23.0",
    "webpack-cli": "^4.5.0",
    "webpack-node-externals": "^2.5.2",
    "webpack-dev-middleware": "^4.1.0",
    "webpack-hot-middleware": "^2.25.0"
  },
  "devDependencies": {
    "@babel/core": "^7.12.17",
    "@babel/preset-env": "^7.12.17",
    "babel-loader": "^8.2.2",
    "file-loader": "^6.2.0"
  }
}
```

Scripts on my computer

- If you’re on a Unix-based system with a bash terminal (i.e. mac, Linux), you can write your own scripts in a file ending in **.sh**
- If you’re on Windows cmd, a **.bat** file should do the same thing. But **.sh** files will also run in Powershell (or use .ps1)
- You might need extra permissions to run your own scripts (especially on a Unix-based system)

Why would I want my own script?

- A lot of things to type on the command line (one instruction is quite complicated)
- A lot of instructions to be run
- Because if you can do loads of things with ⇧ and return ⮥
- By the way, you know to use the →| tab key, right?

What’s Postman

- https://www.postman.com/downloads/

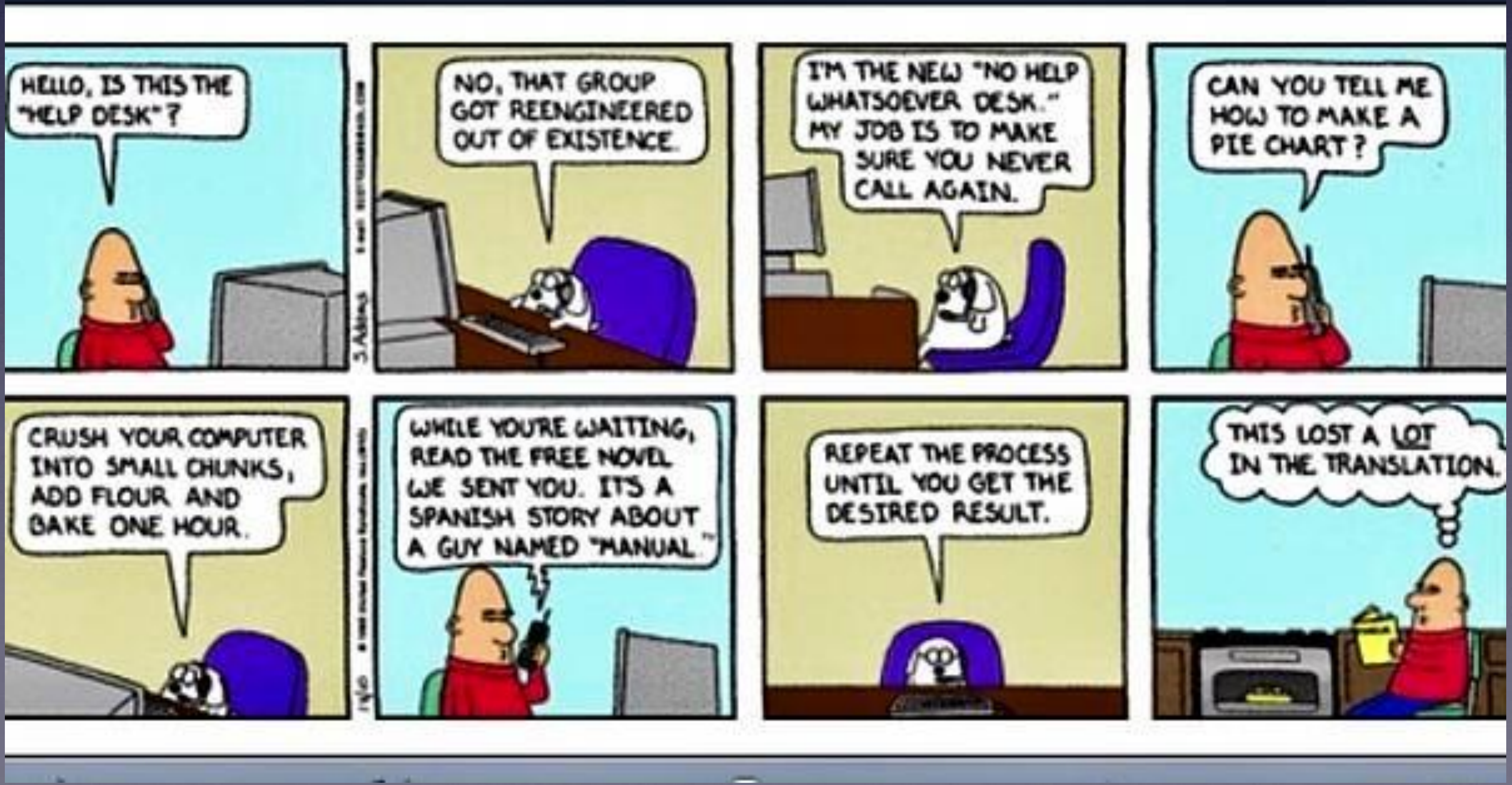
- For creating (and sending) HTTP requests and picking up the responses.

- Alternatives are available (cURL is the command line equivalent or **curl.exe on PowerShell**).

- You use it to test the back end when you don’t have a front end (and it will help you to understand/design the API).

What's <insert npm package>?

- You can look all of these up: https://www.npmjs.com
- You will find documentation, number of recent downloads (i.e. how popular a package is), code samples…


HELLO, IS THIS THE "HELP DESK"?
NO, THAT GROUP GOT REENGINEERED OUT OF EXISTENCE.
I'M THE NEW "NO HELP WHATSOEVER DESK." MY JOB IS TO MAKE SURE YOU NEVER CALL AGAIN.
CAN YOU TELL ME HOW TO MAKE A PIE CHART?
CRUSH YOUR COMPUTER INTO SMALL CHUNKS, ADD FLOUR AND BAKE ONE HOUR.
WHILE YOU'RE WAITING, READ THE FREE NOVEL WE SENT YOU. IT'S A SPANISH STORY ABOUT A GUY NAMED "MANUAL."
REPEAT THE PROCESS UNTIL YOU GET THE DESIRED RESULT.
THIS LOST A LOT IN THE TRANSLATION.

nodemon

“nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.”

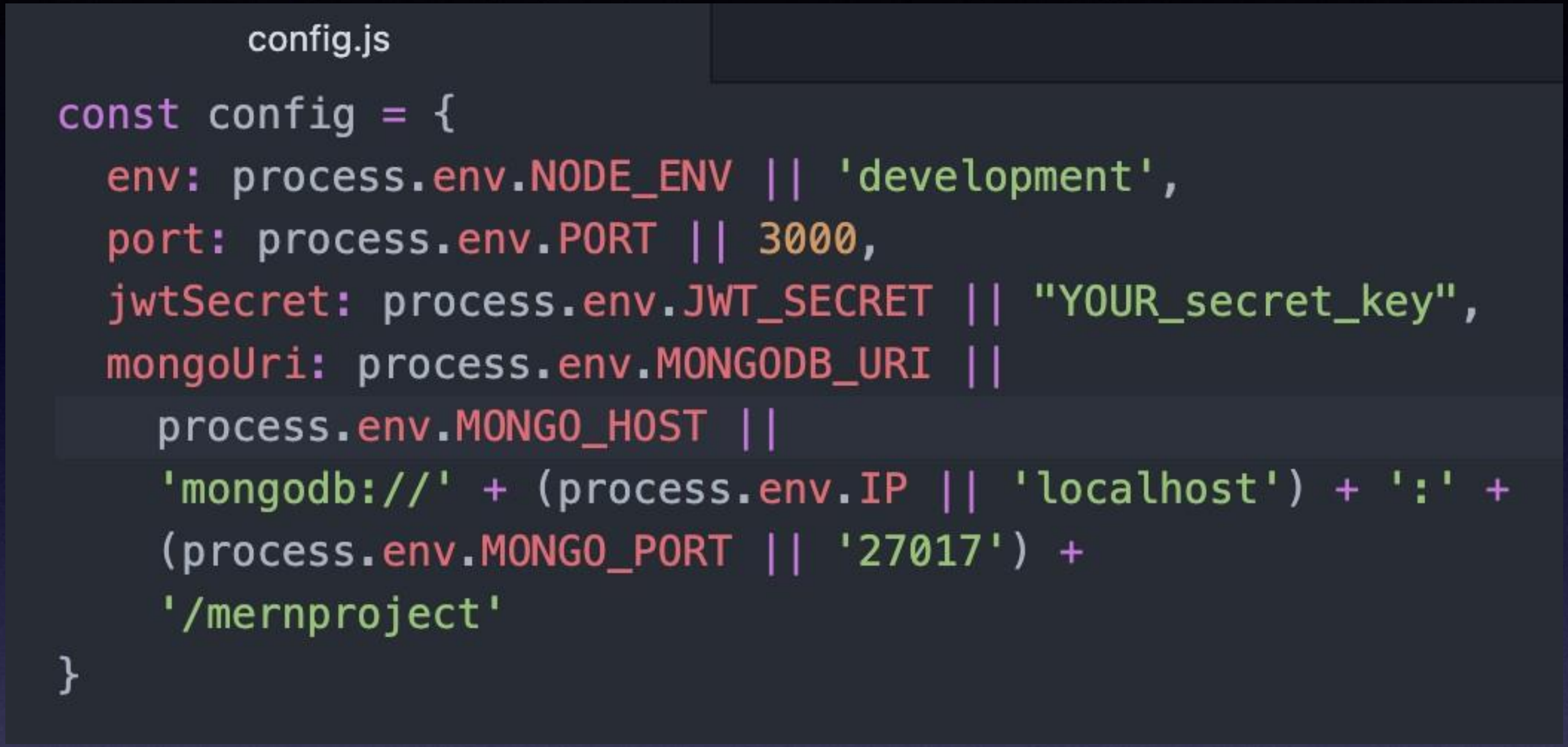
So should I use nodemon?

- It’s useful for development.
- It can save you restarting your server every time (because it does it automatically or when you type **rs**)

systemctl

- For starting and stopping services
- Works for mongod
- Might be useful somehow

Environment variables?


config.js

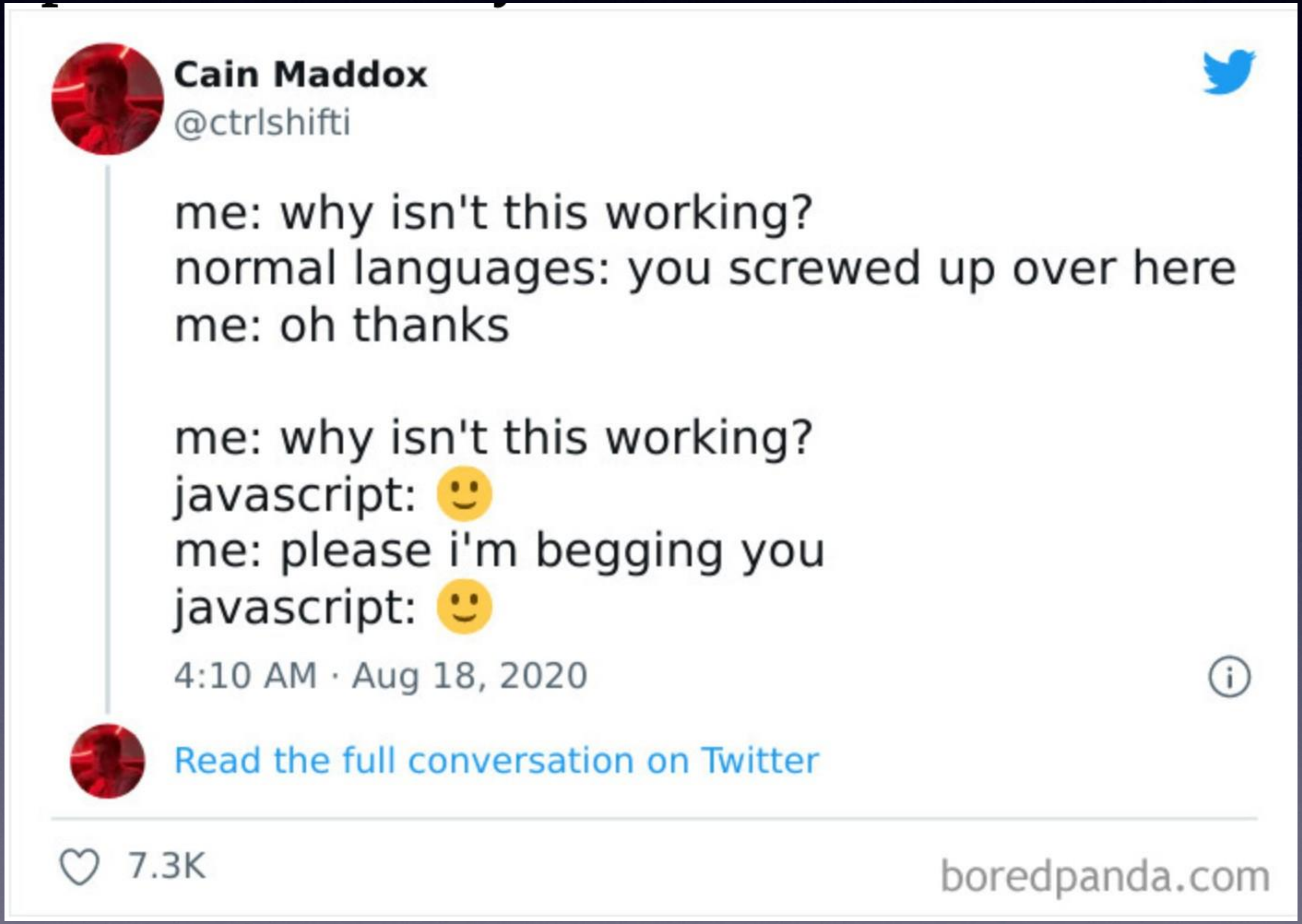
```
const config = {
  env: process.env.NODE_ENV || 'development',
  port: process.env.PORT || 3000,
  jwtSecret: process.env.JWT_SECRET || "YOUR_secret_key",
  mongoUri: process.env.MONGODB_URI ||
    process.env.MONGO_HOST ||
    'mongodb://' + (process.env.IP || 'localhost') + ':' +
    (process.env.MONGO_PORT || '27017') +
    '/mernproject'
}
```

- This uses a big fat load of alternative environment variables

- || is pronounced “or”

- This is how we make the thing work regardless

Debugging



Cain Maddox
@ctrlshifti

me: why isn't this working?
normal languages: you screwed up over here
me: oh thanks

me: why isn't this working?
javascript: 🙂
me: please i'm begging you
javascript: 🙂

4:10 AM · Aug 18, 2020

Read the full conversation on Twitter

7.3K

boredpanda.com

Read the Error Message!

- Easier said than done…
- Get an entire stack dump
- I can’t even feed this to StackOverflow

~~Read~~ FIND the Error Message!

If it’s a stack dump, look for things that you DO recognise.

```
TypeError: req.params is not a function
    at /Path/server.js:79:24
    at Layer.handle [as handle_request] (/Path/node_modules/express/lib/router/layer.js:95:5)
    at next (/Path/node_modules/express/lib/router/route.js:137:13)
    at Route.dispatch (/Path/node_modules/express/lib/router/route.js:112:3)
    at Layer.handle [as handle_request] (/Path/node_modules/express/lib/router/layer.js:95:5)
    at /Path/node_modules/express/lib/router/index.js:281:22
    at Function.process_params (/Path/node_modules/express/lib/router/index.js:335:12)
    at next (/Path/node_modules/express/lib/router/index.js:275:10)
    at expressInit (/Path/node_modules/express/lib/middleware/init.js:40:5)
    at Layer.handle [as handle_request] (/Path/node_modules/express/lib/router/layer.js:95:5)
```

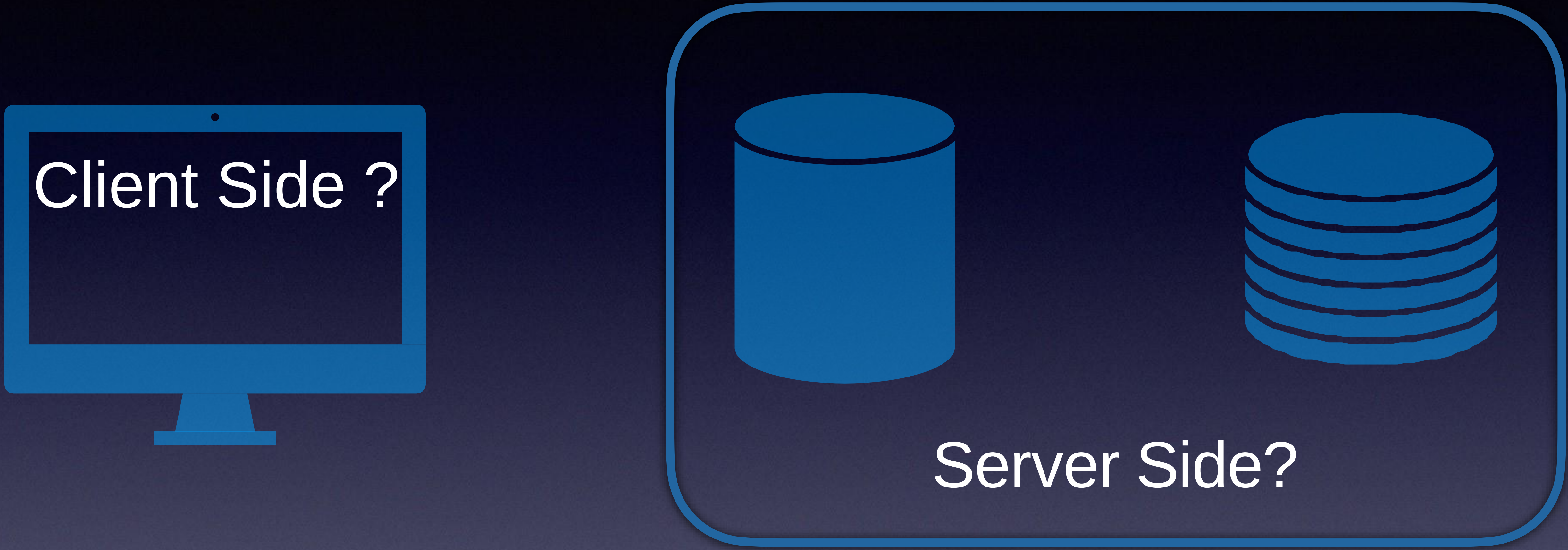
Don’t give your variables or files common names

myInt instead of int

myNode.js instead of node.js

Understand what is **yours** and what is someone else’s.

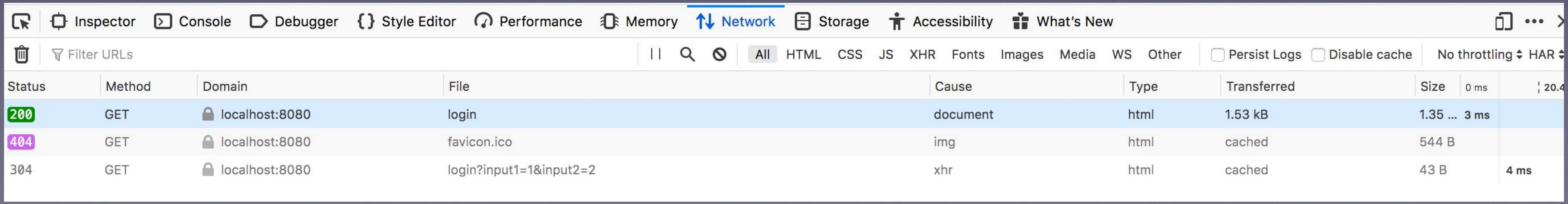
Where is the bug?



Somewhere else entirely?

Client Side

- Run (or reproduce) it locally
- Use the developer tools in your favourite browser
- Look at the Inspector to see if you can see what is wrong with the code
- Look at the Network to see what is going in and out of the server



Status	Method	Domain	File	Cause	Type	Transferred	Size	0 ms
200	GET	localhost:8080	login	document	html	1.53 kB	1.35 ...	3 ms
404	GET	localhost:8080	favicon.ico	img	html	cached	544 B	
304	GET	localhost:8080	login?input1=1&input2=2	xhr	html	cached	43 B	4 ms

Client Side (server interaction)

- A server response to a client side request may indicate a malformed API request
- If a server’s API is well written and informative, it can give you some clue as to what is wrong with your client code

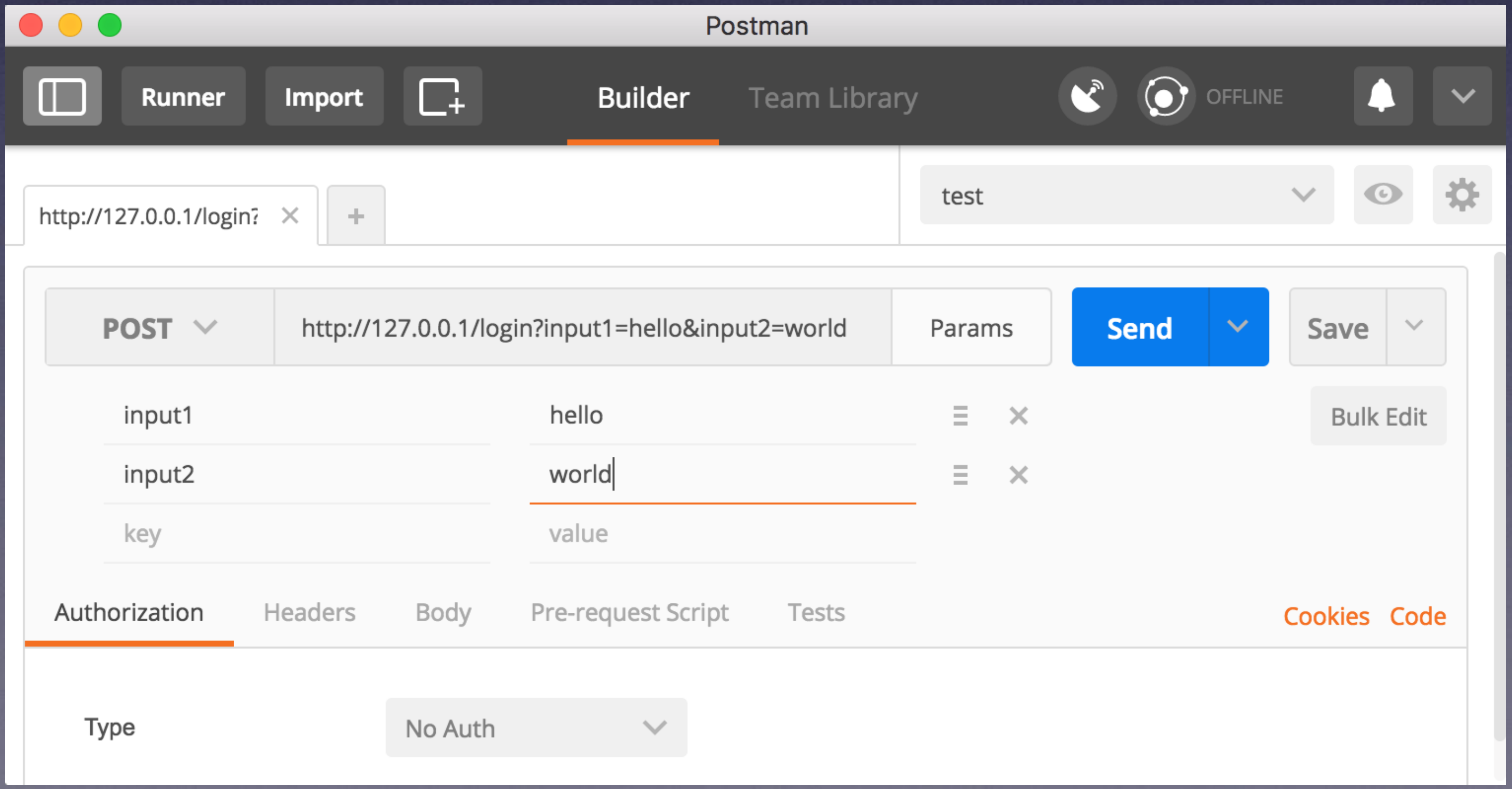
Server Side

- Read the console logs
- Or just the console where you started your server
- Write to the console logs (so you can tell which code is executed and which fails)
- Print nice words to the console logs so you can find them

Server Side

"Mimic" the client:

- Handcraft URLs
- Use apps like Postman to craft more complicated ones



http://127.0.0.1/login?

POST http://127.0.0.1/login?input1=hello&input2=world

input1	hello
input2	world
key	value

Type No Auth

Exactly what do you have installed/running?

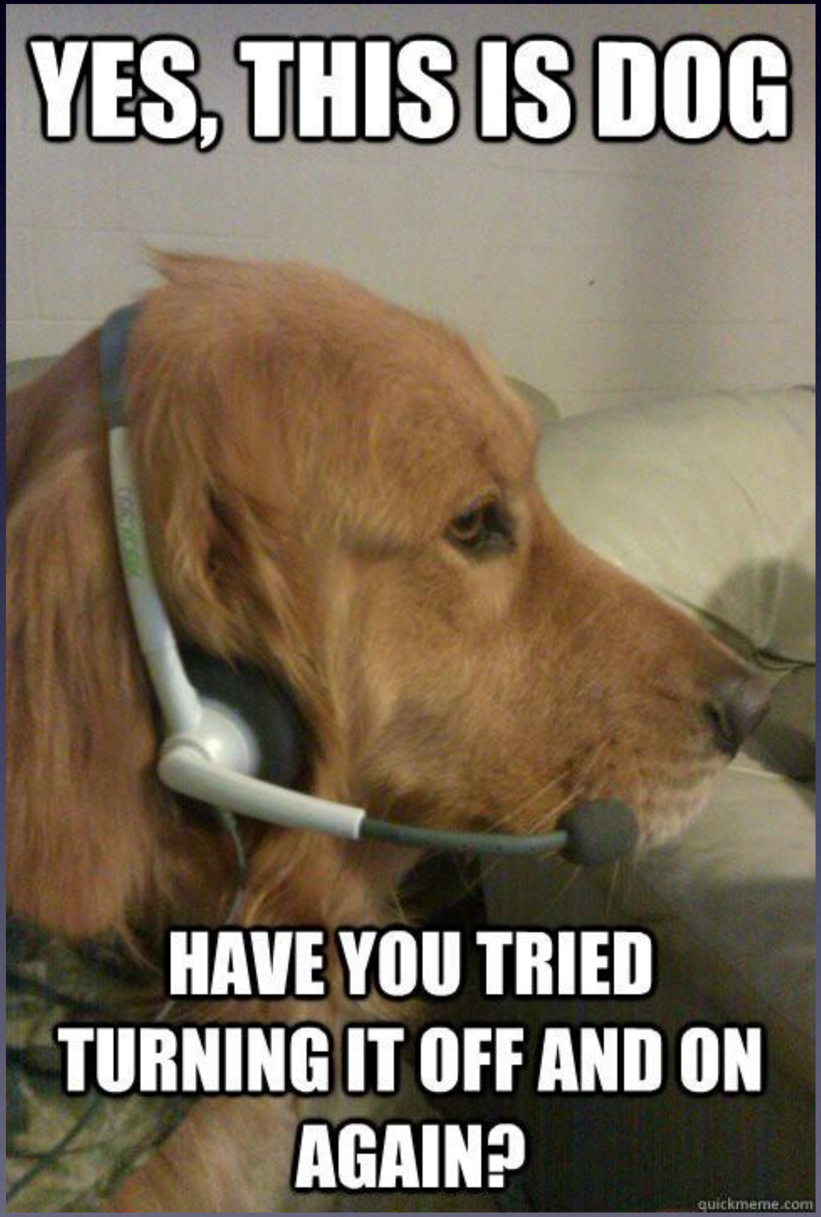
- Check your software version for everything
- Should you be using that version?
- Do you have a database running?
- Just restart your servers

Insanity: doing the same thing over and over again and expecting different results.

Somewhere else?

What **can't** you control?

- Networks
- Firewalls
- Browser type?
- What processes do you need to restart?



Be sure of your procedure

- You need to know that when you change the code and “run it again” you are actually having an effect.
- If in doubt, make a simple change that will have a visible effect on code output (add a log statement, change some visible text, put in a swear word).

More than one bug or not?

Entities should not be multiplied without necessity.

Occam’s razor