Your Next Week

Tuesday March 24	Wednesday March 25	Thursday March 26	Friday March 27
6:30 PM **— DUE Class 03 Code Challenge** **— DUE Class 03 Lab** **— DUE Class 04 Reading** — Class 04A	*6:30 PM* — Class 04B *MIDNIGHT* **— DUE Class 04 Learning Journal**	*6:30 PM* — Lab 04 & Code Challenge 04 Co-Working	
Saturday March 28	**Sunday March 29**	**Monday March 30**	**Tuesday March 31**
9 AM **— DUE ALL PRE-WORK** **— DUE Class 04 Code Challenge** **— DUE Class 04 Lab** **— DUE Class 05 Reading** — Class 05 *MIDNIGHT* **— DUE Class 05 Learning Journal**	*MIDNIGHT* **— DUE Career: Accountability Partners** **— DUE Class 04-05 Feedback**		*6:30 PM* **— DUE Class 05 Lab** **— DUE Class 06 Reading** — Class 06A

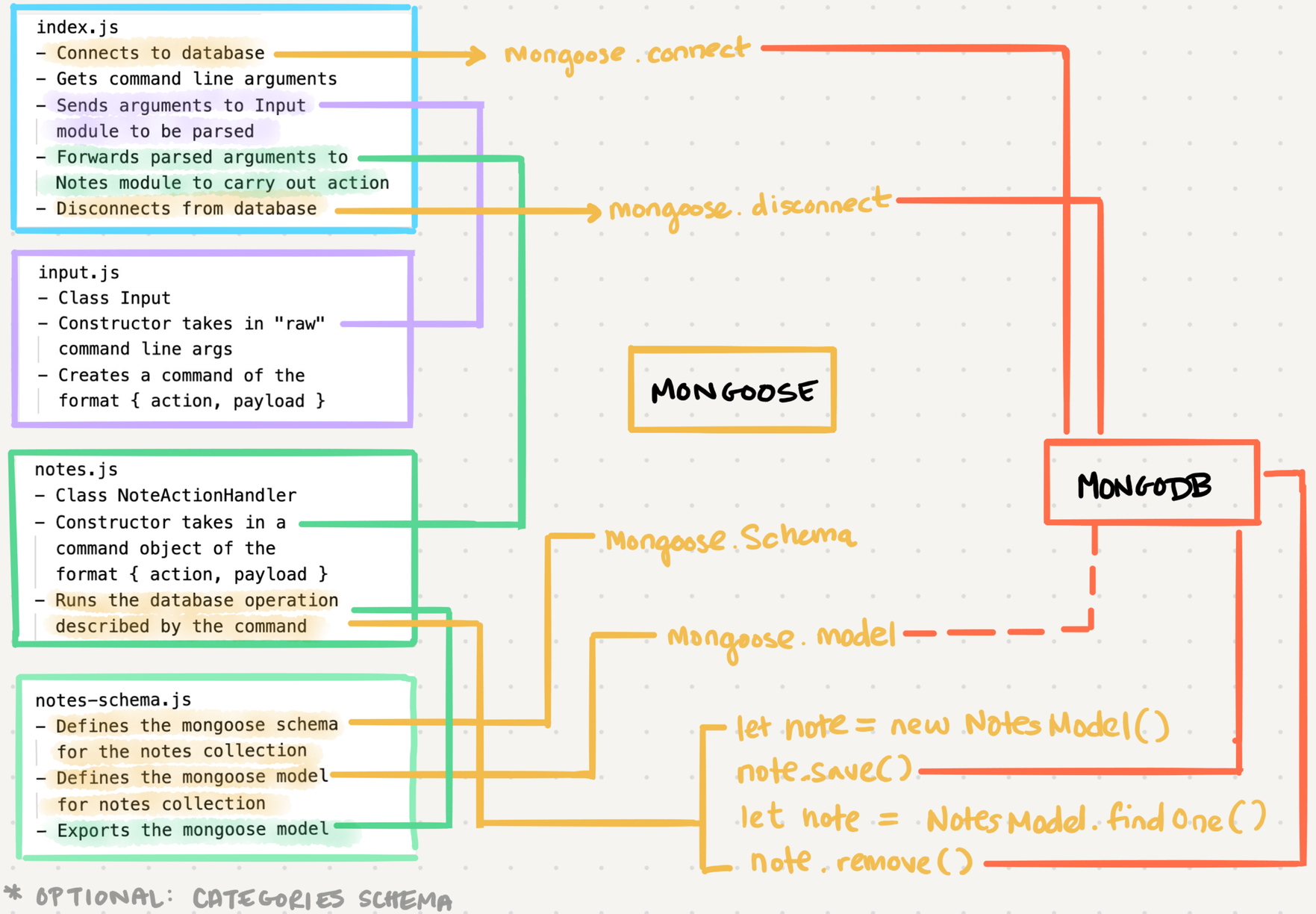
Class 04

Advanced MongoDB/ Mongoose

seattle-javascript-401n16

Lab 03 Review

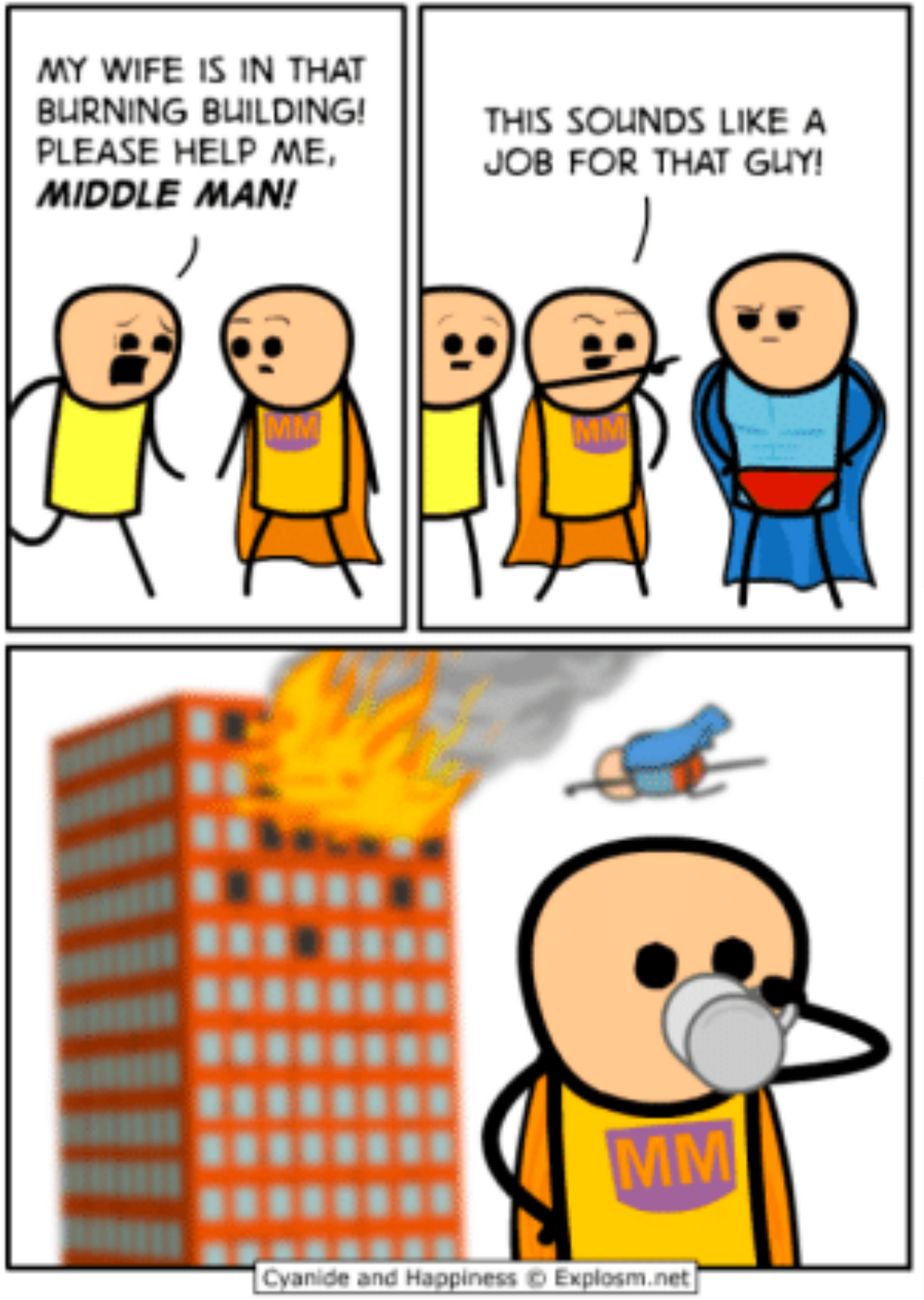
Lab 03 Possible Implementation



Code Challenge 03 Review

Middleware

- There's *hardware*, *software* and *middleware*
  - Ware = tool
- **Middleware** lets two applications communicate
- Mongoose helps our application communicate with MongoDB
- Any tool / function / process between two larger tools / functions / processes




Application Code
Create
Update
delete
Read
Mongoose Model
.save()
.updateOne()
.deleteOne
.findOne()
.find()
MongoDB

Middleware within Middleware

- Mongoose itself can have middleware that runs between processes
- Two distinct types:
  - **Pre** middleware: runs before a database command completes
  - **Post** middleware: runs after a database command complete
- We create pre / post middleware using pre / post **hooks**

Creating a Wrapper Interface

- Knowing the intricacies of Mongoose shouldn’t be required to do CRUD actions
- What if we wrapped the Mongoose code in an easier-to-use class?
  - Standardize operations
  - One place where we can change what those operations do
- If we ever change from MongoDB to Postgres, it will be super easy!
  - Interface will stay the same

Our Model Class

- We'll have the mongoose model still as-is
- Then we'll wrap the mongoose model in our own model class
- Can feel redundant and not always necessary!
    - More useful for larger applications/teams

Testing a Database

- We don’t want to change any *actual* data
- When we run our tests, we want to tell the tests to run a “fake” or “**mock**” database
  - Starts off empty
  - Our test files insert some dummy data at the start
- New dependency: `npm install @code-fellows/supergoose`

What is a Mock?

- A fake version of the real thing
- Inputs are the same, and outputs are *somewhat* the same
- Our mock database has the same ability to do CRUD operations, but it’s not actually persisting anything
- You can make a mock of functions too!
    - Useful for testing code that uses external modules/functions that you DON’T need to test

Vocab Review

mock

mock

A mock allows you to skip testing some piece of your code while still maintaining a roughly similar input/output path. This way, you can skip calling certain functions and still test your application flow. And by skipping some functions, you can save on time and prevent redundant or unnecessary tests.

middleware

middleware

Code that is used to bridge the gap between applications or tools. Often described as “software glue”, middleware makes it much simpler for two differing systems to communicate with one another. Broadly, middleware can be described as any code running between two other processes.

supergoose

supergoose

A package developed by Code Fellows, supergoose is used to help set up a mock database for your tests. This can otherwise be tedious to do on your own, and many companies use similar packages to spin up a fake database for testing.

pre hook

pre hook

With Mongoose, you can write your own middleware that runs between a Mongoose command being called, and the MongoDB receiving that command. To write this middleware, Mongoose exposes a ‘`pre`’ function, often called the pre hook since it hooks your middleware code up to the correct part of the process.

post hook

post hook

With Mongoose, you can write your own middleware that runs between a MongoDB command being called, and the response to Mongoose on your application. To write this middleware, Mongoose exposes a ‘`post`’ function, often called the post hook since it hooks your middleware code up to the correct part of the process.

in-memory

in-memory

When something is in-memory, it is only accessible while a program is running.

Any in-memory data is not persisted after the application ends.

interface

interface

Like middleware, an interface is a shared space where two systems can exchange information. Interfaces allow two unrelated entities to interact. Interfaces receive a certain kind of input and translate that into the commands needed to get the expected output.

Lab 04 Overview

Code Challenge 04 Overview