Code Fellows

Class 04
Data Modeling

seattle-javascript-401n14

Lab 03 Review

Code Challenge 03 Review

Vocab Review!

What is a call stack?

What is call stack?

The call stack is a part of every running program, where functions we want to run are pushed onto the stack when we encounter it in our program, and popped off the stack when the function returns something. Items on the call stack are resolved in order, and can be blocking.

What is a callback?

What is a callback?

A callback is something we pass to another function, with the intent that it will be called when the function completes. This is the most generic way to handle asynchronous operations - do the asynchronous function, and then call a synchronous function when it’s done.

What is a Promise?

What is a Promise?

A Promise is a refined way to handle asynchronous code. You store the asynchronous action in a class object, and then can easily append response actions using `.then()`

What is async/ await?

What is async/ await?

`async` and `await` are keywords applied to functions and operations that essentially create Promises behind-the-scenes. It’s a much simpler way for us to code asynchronous actions.

What is a buffer?

What is a buffer?

A buffer is a stream of raw data, that can later be interpreted and converted into a specific data type (string, integer, video file). JavaScript lets us work with buffers by exposing the `Buffer` class.

What is a mock?

What is a mock?

A mock is a hard coded mock-up of a package module. When we test our code, we don’t want our tests to fail or run slow because of something in an external module. So we fake the classes, functions, etc in a package module by ensuring our fake versions take in the same parameters, but return hard-coded data

What is a database?

What is a database?

A database is a structured collection of data, usually stored outside our applications altogether. Databases want to be secure, so they limit how our applications can access them.

Data in Applications

- Many applications are connected to a **database**
- A database allows an application to **Create**, **Read**, **Update** and **Delete** data and have those operations **persist** even when the application isn't running
    - We shorten these actions into the acronym **CRUD**

Other variations [edit]

Other variations of CRUD include:

- BREAD (Browse, Read, Edit, Add, Delete) [5]
- DAVE (Delete, Add, View, Edit)[6]
- CRAP (Create, Replicate, Append, Process)[7]

Okay Wikipedia...

CRUD?? Why?

- CRUD operations are the primary way to communicate with a database, and therefore the primary way to access your application’s data
- We want to start thinking about our applications as communicators with a database
- The communication language is CRUD

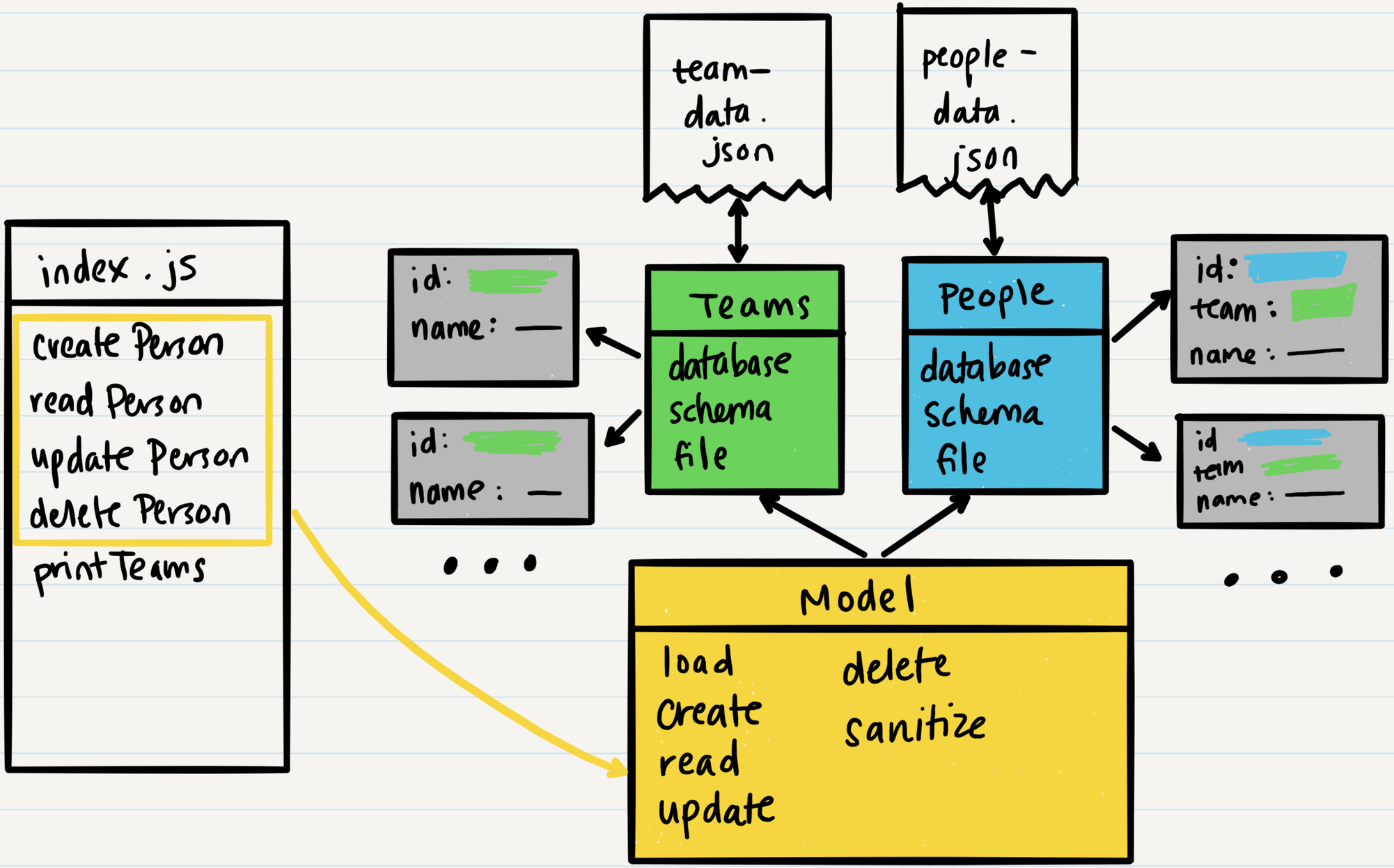
Data Modeling

- We've already been doing this with UML diagrams and classes
- Data modeling is a way to define types of data and the relations between them in your application
- "**Data Models**" are usually defined by classes, and they contain functions to hook into the database for CRUD operations

Let’s Model Some Data!

Requirements

- We need to create a simple app that stores **people** and **teams**
- People can only be on one team
- Teams must have at least one person
- We need to keep track of who is on which team, and we need to be able to add more people or teams
- Each person and team should have a **unique identifier**
- We need to **persist** this data


team-data.json
people-data.json
index.js
createPerson
readPerson
updatePerson
deletePerson
printTeams
id:
name:
id:
name:
Teams
database
schema
file
People
database
schema
file
id:
team:
name:
id
team
name:
Model
load
create
read
update
delete
sanitize

```
    this.database = JSON.parse(contents.toString().trim());
    return this.database;
  }

  // CRUD: create
  async create(item) {
    let record = this.sanitize({ id: uuid(), ...item });

    if (record) this.database.push(record);

    await writeFile(this.file, JSON.stringify(this.database));

    return record;
  }
```

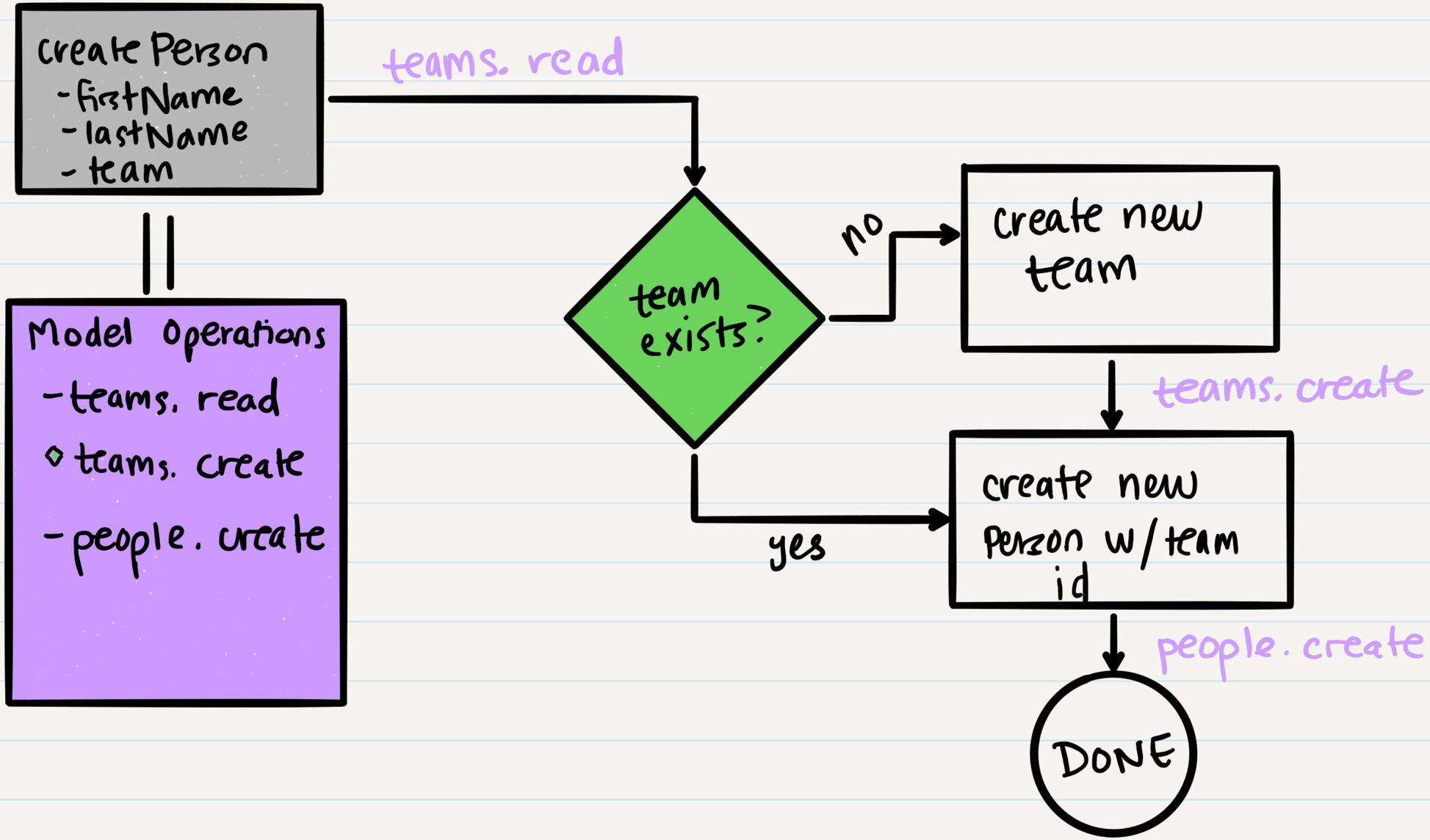
```
==========================================
Team:  Green Buffalo
Garth Constantine
Walden Phinn
Frederik Suche
==========================================

==========================================
Team:  Red Heron
Tim David
Alex Leyburn
```

Demo

class-01/ demo/models

We limit ourselves when it comes to effecting our database, but we can create really unique actions on top of that to define when we should Create, Read, Update and Delete.


createPerson
-firstName
-lastName
-team
teams.read
Model Operations
-teams.read
◇teams.create
-people.create
team exists?
no
yes
create new team
teams.create
create new Person w/team id
people.create
DONE

What's Next:

- Due by Midnight tomorrow: **Learning Journal 03**
- Due by Midnight Thursday: **Code Challenge 04**
    - **We're going to do this in pairs!**
- Due by 9am Saturday:
    - **Lab 04**
    - **Read: Class 05**
    - **Read: DSA 01**
- Next Class:
    - **DSA - Linked Lists**
    - **Class 04 - Data Modeling With NoSQL Databases**

Questions?