Object Relational Mapping

In this assignment, you will learn to use the Objection ORM to issue queries to a database.

1 Learn

- Watch Objection.js a SQL ORM, a talk from the London Node User Group, May 2018.
- Read the documentation for Objection.

2 Set Up

Set up a Node project and install the necessary modules:

```
npm init
npm install --save objection knex pg
```

3 Review

Initialize Knex and Objection like this:

```
const knex = require('knex')({
  client: 'pg',
  connection: {
    host: 'faraday.cse.taylor.edu',
    user: 'readonly',
    password: 'nerds4christ',
    database: 'dvdrental'
  }
});

objection = require('objection');
const Model = objection.Model;
Model.knex(knex);
```

Here's a simple definition of a `Model` object that will connect to the `country` table in the `dvdrental` database.

```
class Country extends Model {
  static get tableName() {
    return 'country';
  }
}
```

Finally, we use the `Country` model to fetch data from the database and print out select countries.

```
Country.query()
  .then(countries => {
    countries.forEach(country => {
      if (country.country.startsWith('F')) {
        console.log(country.country_id,
                    country.country);
      }
    });
    knex.destroy();
  })
  .catch(err => console.log(err.message));
```

Here's the output:

```
32 'Faroe Islands'
33 'Finland'
34 'France'
35 'French Guiana'
36 'French Polynesia'
```

Important: The function call `knex.destroy()` closes down KNEX's connection to POSTGRESQL. If you don't do this, NODE will not exit because it's waiting for KNEX to close the database connection.

Normally, you would only call `destroy` *once* at the very end of your program's execution.

4 Code

Meet the following requirements.

1. Define an OBJECTION `Model` class called `Actor` that provides access to the `actor` table of the `dvdrental` database.
2. Define a JAVASCRIPT function called `actors_starting_with` that takes a single argument called `prefix`. The function should:
   (a) Use the `Actor` model to retrieve *all* actors from the database.
   (b) Filter the list of actors to include only those whose *first* or *last* name begins with the value of `prefix`.
   (c) Construct a list the full names of matching actors. Use a template literal to construct the full names using the `first_name` and `last_name` fields from the database.
   (d) Return a `Promise` that resolves to the list of actors' full names. Remember that both the OBJECTION `query` method and the standard `then` method of a promise both return promises!

3. Write code that runs `actors_starting_with('F')` and outputs the list of actors' full names to the console. You may use the returned promise directly (with `.then`) or may wrap the function call in a `async` function and use `await`.

5 Submit

Submit to the course web site your source code from Section 4.