# Knex migrations & seeds

I THINK WE SHOULD BUILD AN SQL DATABASE.
UH-OH
E-mail: SCOTTADAMS@AOL.COM
S.Adams
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
© 1996 United Feature Syndicate, Inc.(NYC)
12/17
WHAT COLOR DO YOU WANT THAT DATABASE?
I THINK MAUVE HAS THE MOST RAM.

# Objectives

- Explain what the Knex migration system is.
- Explain why the Knex migration system is useful.
- Use Knex to migrate a PostgreSQL database.
- Explain what the Knex seed system is.
- Explain why the Knex seed system is useful.
- Use Knex to seed a PostgreSQL database.

# The good news!

- you've already done something *very* similar (writing .sql files into psql server), but with knex migrations we will have a bit more control over data flow.

  ie) **psql my_DB_name < schema.sql**

# Research:

(4 min)

* what is a schema migration?
* why is it useful?

# Features of a migration:

- named by timestamp
- define an 'up' & 'down' behavior
  (what how to go **forward** and **back** in time)

# Before we can get started...a bit of setup

1. make sure knex is installed globally

   ```
   npm i -g knex
   ```

2. `npm init`

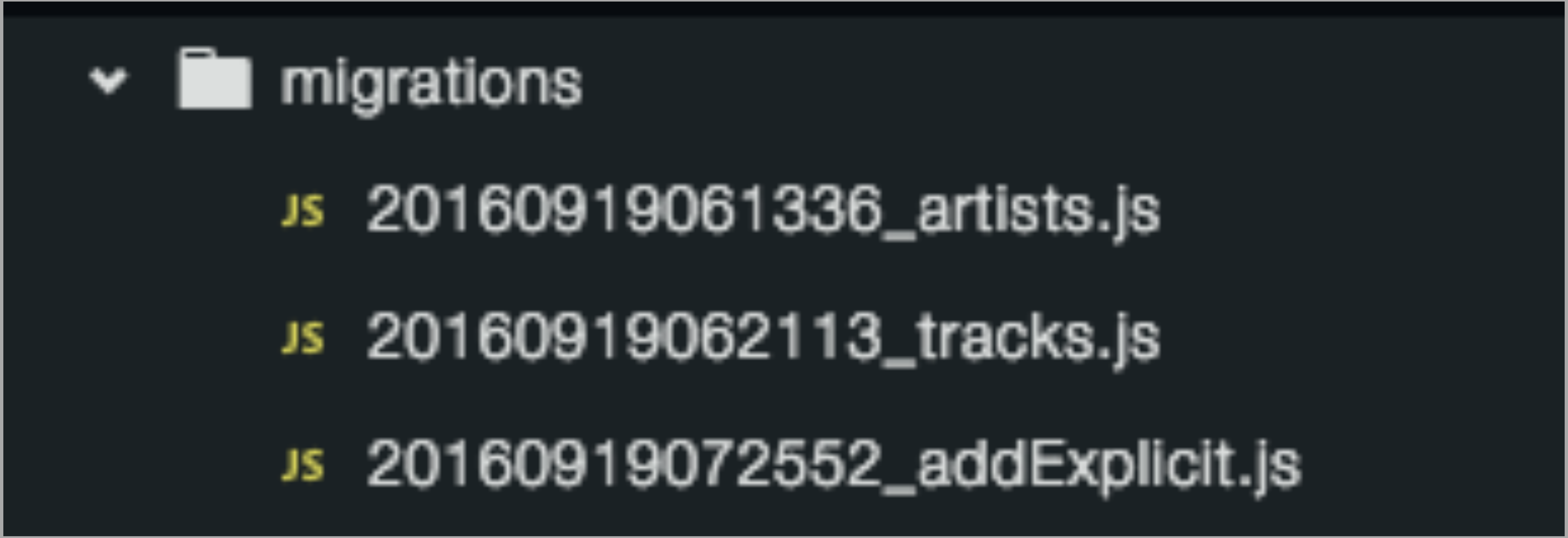
3. `npm i --save pg knex`

4. `knex init //creates a knexfile.js`

5. `createdb db_name`

```
//...in knexfile.js
'use strict';
module.exports = {
  development: {
    client: 'pg',
    connection: 'postgres://localhost/your_db_name'
  }
};
```

# migration file naming



```
migrations
    JS 20160919061336_artists.js
    JS 20160919062113_tracks.js
    JS 20160919072552_addExplicit.js
```

# Defining *up* and *down*... (migrate/rollback) syntax

```
'use strict'

exports.up = function(knex) {
  return knex.schema.createTable('artists', (table) => {
    table.increments();
    table.string('name').notNullable().defaultTo('');
    table.timestamps(true, true);
  })
};

exports.down = function(knex) {
  return knex.schema.dropTable('artists');
};
```

My barn having burned
to the ground
I can now see the moon

Mizuta Masahide
1657 – 1723

# Important Knex CLI commands (migrations)

```
knex migrate:make migration_name
```

* be descriptive in migration name...adds migrations folder

```
knex migrate:latest
```

* migrates your latest migration to psql server

```
knex migrate:currentVersion
```

* shows the last migration you made (timestamp)

```
knex migrate:rollback
```

* go back in time...rollback your migration

# Important Knex CLI commands (seeds)

```
knex seed:make seed_name
```

* be descriptive in seed name...adds seeds folder with boilerplate

* the order your seeds are run matters! use numbers!

```
knex seed:run
```

* migrates your seed data to psql server

# PRO Tip:

- if you have knex installed globally, you can skip the 'npm run' command...this differs from the article's instructions
- also if knex is globally installed, use **knex init** to create a knexfile.js

# Review:

- What does the knexfile.js file do? Why is it important?
- Why is the knex CLI super handy for creating migration files?
- What is a database schema?
- Why do migration file names start wtih a UTC timestamp?
- Why is it important to use the knex CLI rollback function to undo a migration?
- Why does knex add an exports.down function to your migrations?

fin