Overview

Discuss following technologies:

- Node Package Manager `npm`
- Mongo db

Node Package Manager

- Track dependencies of project transitively.
- Handles multiple versions of same package.
  1. Assume project requires v1 of packages A and B.
  2. Package A requires v1 of package C.
  3. Package B requires v2 of package C.

  npm sets things up so as to have both v1 and v2 coexistent at runtime. This can occasionally cause trouble, but usually works seamlessly.

npm Dependencies

- Distinguish different types of dependencies:
  - Runtime dependencies.
  - Development dependencies (use option `-D` or –save-dev).
    - Build-time dependencies like `webpack`.
    - Testing dependencies like `mocha`.

Config File package.json

- Config file `package.json` is JSON: hence no comments allowed!!
- Sample package.json.
- `npm install PACKAGE` will install `PACKAGE` and all its dependencies in `node_modules` directory.
- Above command will create a `package-lock.json` which specifies the exact versions of the dependencies found.
- `npm install` will install all dependencies from `package.json`.
- I use `npm ci` to get exactly the same versions as `package-lock.json`.

MongoDb

- One of many nosql databases. No rigid relations need to be predefined.
- Allows storing and JS data serialized in "documents".
- Allows indexing.
- Provides basic **Create**-**Read**-**Update**-**Delete** (CRUD) repertoire.

Mongo CRUD

All operations asynchronous. Set up to return a `Promise` when called without a handler.

Create	`insertOne()` and `insertMany()`.
Read	`find()` returns a `Cursor`. Can grab all using `toArray()`.
Update	`updateOne()` and `updateMany()`. Also, `findOneAndUpdate()` and `findOneAndReplace()`. Can combine insert and update functionality using `upsert` option.
Delete	`deleteOne()` and `deleteMany()`.

Asynchronous DB Operations

- As mentioned, most DB operations are asynchronous.
- Usually a good idea to cache DB connection as opening a DB connection is an expensive operation.
- Create an object which wraps a database. Cache database connection within object when object is first created.
- Getting a connection to a DB is an asynchronous operation.
- Can we build object using an `async constructor`??
- Use a static factory method instead.

Setting up a Mongo Project

```
$ npm init -y #creates package.json
...
$ npm install mongodb     #old versions of npm required --save
npm notice created a lockfile as package-lock.json...
...
added 6 packages from 4 contributors ...
$ ls -a
.gitignore  node_modules  package.json
package-lock.json ...
$
```

Mongo Shell Log

Allows interacting with mongo db. Following log assumes that collection `userInfos` in db `users` is loaded with simpsons data.

```
$ mongo
MongoDB shell version v3.6.3
...
> use users
switched to db users
> db.userInfos.find({})
{ "_id" : "bart", "id" : "bart", ... }
{ "_id" : "marge", "id" : "marge", ... }
{ "_id" : "lisa", "id" : "lisa", ... }
{ "_id" : "homer", "id" : "homer", ... }
> db.userInfos.find({"firstName": "Bart"})
{ "_id" : "bart", "id" : "bart", ... }
> db.userInfos.find({}).length()
4
```

Mongo Shell Log Continued

```
> db.userInfos.deleteOne({"firstName": "Bart"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.userInfos.find({}).length()
3
> db.userInfos.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 3 }
> db.userInfos.find({}).length()
0f
```