Week 3

COMP1531

hi again, its me - only 7 more weeks

UPDATES

Iteration 0:

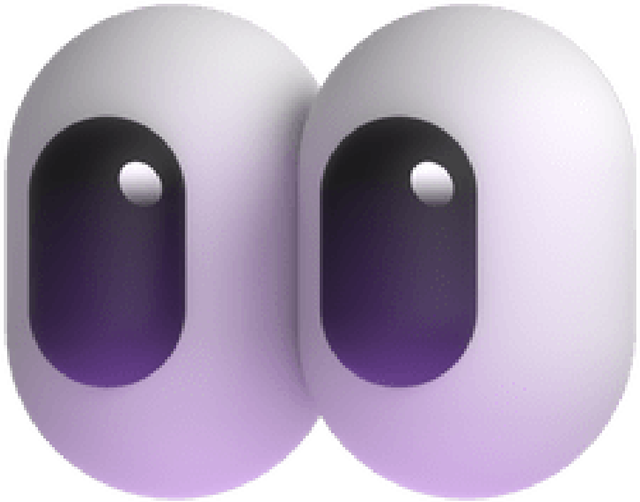
- Manual marking feedback
  - During the lab + post on Teams
- Good job! Well done!

Iteration 1:

- Released! Make sure to merge in the MR
- Get started! Read spec, watch intro video
- Standup, meetings, task boards!

```
> 0 > null
false
> 0 >= null
true
> 0 == null
false
> 0 <= null
true
> 0 < null
false
> BRUH
```

TODAY...



01 PACKAGE MANAGEMENT

02 TESTING

03 MULTIFILE IMPORT / EXPORT

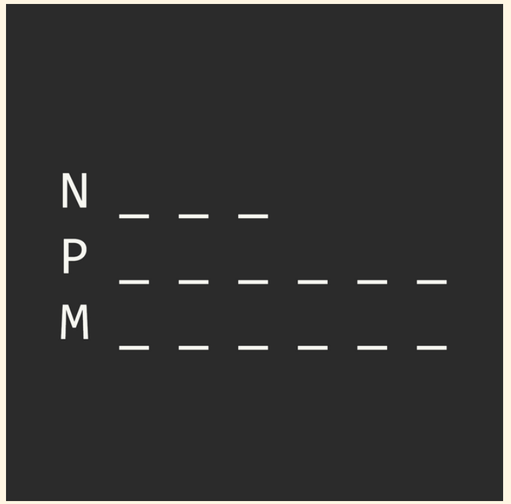
PACKAGE MANAGEMENT

1. What are packages? Why do we use them?
2. Why do we need to manage them?
3. What tool(s) do we use to manage our packages?

25M+

DOWNLOAD WEEKLY

NPM express - NodeJS



https://www.hangmanwords.com/play/custom?g=bm9kZSUyMHBhY2thZ2UlMjBtYW5hZ2Vy

In groups, consider the interface below:

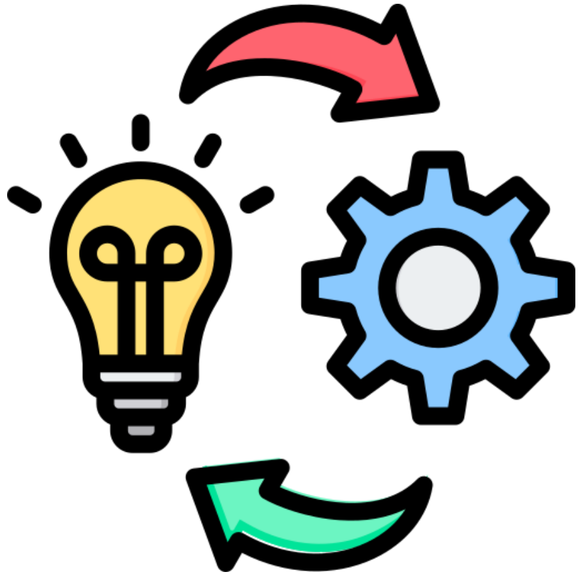
Interface: Functions

Name & Description	Parameters	Return Object	Errors
`stamp` Given an email, stamp it with a unique `identifier` and a `timeString`. The `identifier` should be universally unique (e.g. someone from across the world has an improbable chance of generating the same string). It should not rely on the characters in the email.	(email)	`{identifier, timeString}`	Return `{error}` when the email is invalid

Interface: Data Types

If the variable name	It is of type
is **error**	`string`, with value being a relevant error message of your choice
is **email**	`string`
is **identifier**	`string` that is globally unique
is **timeString**	`string` in the format `'WEEKDAY - hh:mm:ss [am/pm]"`. e.g. `'Saturday - 06:03:54 pm'`

1. Before researching, discuss with your group the kinds of tools you'd likely need to solve this problem.
2. Look through the npm registry (https://www.npmjs.com/) or other sources for any package that meets your needs. Note them down. If you have time, try to complete the function



IMPLEMENTATION

```
import validator from 'validator';
// Formats: https://date-fns.org/docs/format
import { format } from 'date-fns';
import { v4 } from 'uuid';

function stamp(email) {
  if (!validator.isEmail(email)) {
    return { error: `invalid email: '${email}'` };
  }
  return {
    identifier: v4(),
    timeString: format(new Date(), 'EEEE - hh:mm:ss aaa'),
  };
}

console.log(stamp('invalid@@email'));
console.log(stamp('valid@email.com'));
```

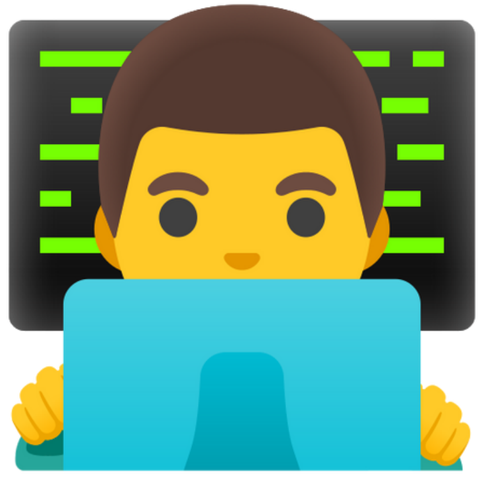
CODING DEMO

Testing, Coding, Multi-file import/export demo

Below is the MVP interface for a movie data system.

Variable	Type
is exactly **error**	`string`, with the value being a relevant error message of your choice
contains suffix **Id**	`number`, specifically integer
is exactly **title**	`string`
is exactly **director**	`string`
is exactly **movies**	Array of objects, where each object has type `{movieId, title, director}`

Name & Description	Input Parameters	Returned Object	Errors
`movieAdd` Adds a movie to the data store.	(title, director)	`{movieId}`	`{error}` when any of: • title is an empty string, "" • director is an empty string, ""
`movieEdit` Edits a movie in the data store.	(movieId, title, director)	`{}`	`{error}` when any of: • movieId does not refer to an existing movie • title is an empty string, "" • director is an empty string, ""
`moviesList` Lists all movies in the data store.	()	`{movies}`	N/A
`clear` Delete all movie data and return an empty object.	()	`{}`	N/A



CODING TIME

Name & Description	Input Parameters	Returned Object	Errors
`movieAdd` Adds a movie to the data store.	(title, director)	`{movieId}`	`{error}` when any of: • title is an empty string, "" • director is an empty string, ""
`movieEdit` Edits a movie in the data store.	(movieId, title, director)	`{}`	`{error}` when any of: • movieId does not refer to an existing movie • title is an empty string, "" • director is an empty string, ""
`moviesList` Lists all movies in the data store.	()	`{movies}`	N/A
`clear` Delete all movie data and return an empty object.	()	`{}`	N/A

01 Testing

In b.movies/src/movie.test.js, write tests for each of the functions listed in the interface. Your test suite should include:

- basic error cases
- return value of each function
- behaviour and side effects

02 Implementation

Complete the implementation for the interface functions. Ensure that they pass your tests.

03 Automarking

Let's see if we are correct!

CLIENT

Your tutor is a client who will interact weekly with your group to check in and learn about the current state of the backend application.

1. In groups, identify and discuss what kind of information you should collect or track each week to keep your client informed and satisfied.
2. Your client will now explain the requirements and expectations for iteration 1 in further detail. In particular, the

- Task
- Git practices (commits, branches, merge requests, issue boards)
- Data storage/structure
- Black-box testing & and implementation of features
- Communication, meeting minutes, frequent standups
- Assumptions
- Marking Criteria

High Expectations for your team :3

Reach out during the lab for clarification!

✓ Summary

By the end of today you should...

- Know how to use npm to install packages and use them in your code.
- Understand the basics of testing with Jest.
- Have a clear understanding of iteration 1