Exercise Thursday

Intermediate JavaScript (/intermediate-javascript) / Test-Driven Development and Environments with JavaScript (/intermediate-javascript/test-driven-development-and-environments-with-javascript) / Haiku Creator, RPG, Sudoku Solver (Two-Day Project) - Part 2

Text

Goal: Continue to practice testing with Jest and implementing ES6 features along the way. You should use ES6 classes, since those will be required on the code review. Test all of your business logic using TDD and Jest.

Warm Up

- Why should we separate our business logic into multiple files?
- Why would we use the `class` keyword in our code? What does it do?
- When would we use an arrow function in our code? Think back to the `Box` object example and using `Array.prototype.forEach()` in the `Box` object's method to `addJunk` to our box.

Code

Pick one of the following projects to complete. If you finish with more time on your hands, pick another project.

Weekday Calculator

This prompt will give you the chance to try out working with the JavaScript `Date` object (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date).

Create an app for calculating the day for any given date. The user should be able to enter in a date and see what day of the week that date occurred on.

If you enjoy brain teasers, try to work out your own algorithm. If you need help getting started, check out the Wikipedia article (https://en.wikipedia.org/wiki/Determination_of_the_day_of_the_week). The program should be able to handle past dates, future dates, and invalid dates (user should be notified if the date is invalid).

Make sure to use `class` syntax and, as always, use TDD to develop your business logic before writing any user interface logic.

Haiku Checker/Creator

A haiku is a poem that consists of three lines. The first has five syllables, the second has seven, and the third has five. Start by creating an application that checks whether a poem is in fact a haiku. If you have time, build out your application so that it can randomly generate haikus.

This project provides an excellent opportunity for TDD with Jest.

- Your logic should verify that the poem has three lines.
- Your logic should verify English syllable rules (and exceptions) one at a time. A quick Google search will provide information

on English syllable rules.
- If you successfully complete a Haiku checker, continue to build out your application to randomly generate haikus.

Make sure to use `class` syntax and, as always, use TDD to develop your business logic before writing any user interface logic.

Build Your Own RPG

An RPG (Role Playing Game) is a game where players assume the roles of characters in a fictional world. Build and fully unit test the business logic for a Medieval Role Playing Game (or another genre that you prefer: sci-fi, cyberpunk, '80's high school).

Logic could include the following:

- **Character creation:** Use ES6 classes to generate different character types. Be creative with your character types — use warriors, wizards, scientists, prom queen, whatever! Characters should have specific attributes. For instance, in a medieval RPG, characters might have `strength` and `intelligence` attributes among others. In an '80's high school RPG, characters might have `snark` and `charm`.
- **Battle system:** Many RPGs have a battle system so characters can fight monsters, though that could just as easily be a high school 'battle' system where the prom queen has a dance-off with the theater aficionado. Determine conditions for winning a battle, whether that's defeating monsters (with swords and spells doing damage), accumulating dance-off style points, or any other system you think of.
- **Level up:** Determine a leveling system. Characters should be able to go from Level 1 to Level 2 and so on. Generally each level comes with new abilities. How do characters level up in your game? What attributes and powers do they gain? Does their `strength` go up or do they learn new spells?

- **Inventory:** Characters should be able to have items that enhance their abilities. Maybe the Magic Armor increases their defense power or legwarmers increase their dance-off ability. Create a limit to the number of items a character can have. Characters should be able to add, drop, buy and sell items.

Feel free to build out your RPG as you see fit. The goal is to use `class` syntax and TDD to create well-tested business logic. You are not expected to have a functioning game in the browser. Instead, focus on testing and business logic!

Peer Code Review

- Tests were committed before code.
- Business logic has 100% line coverage with Jest.
- Project utilizes ES6 classes.
- Dependencies are managed with npm.
- webpack is used to lint, bundle, and process code.

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/es6-maps-and-sets)
Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/whiteboard-practice-javascript-strings)