Game Closure Coding Challenge

Introduction

Whether you're a frontend, backend, or full stack JavaScript engineer, this challenge is for you! Implement a clone of Adventure Capitalist in your favorite JavaScript stack, for fun and profit.

Adventure Capitalist

Adventure Capitalist is an idle business sim-game. The basic idea is to purchase a business, win capital from that business, upgrade the business and then purchase more businesses.

The way to win capital in Adventure Capitalist is once you've purchased a business, you need to click on the business and it takes some time to gain the capital. Once done, you can click again.

In order to automate this, you can hire a manager who can run the business for you, so you don't have to click manually anymore. Then you can upgrade the business and gain even more money.

A version of the game can be found on the web, and it's also available on mobile devices: http://en.gameslol.net/adventure-capitalist-1086.html

The full game has many different features and extensions, but we're looking for the basic gameplay with upgrading businesses and hiring managers.

The game is idle, so it progresses while you are away: If you have a manager, the game should continue playing while you're gone.

Spec Requirements

- Buy and upgrade businesses. There should be several business types to choose from.
- Make money from a business. (i.e. you click on a business and in a certain amount of time you get money – see web implementation above.)
- Hire managers, so that money is made automatically.
- When you close the game, next time you open it, you should see the money that your businesses made for you. (Businesses continue to make progress while you're away.)

Tech Requirements

- The UI can be anything: Visual, textual, even a fullscreen CLI works.
- Must be built in a dialect of JavaScript: ES5, ES6, ESNext, TypeScript, etc.
    a. We recommend against using Cocos Creator for this challenge.
- We expect you to put focus on at least one of the following topics:
    a. Visual polish: Beautiful design and/or graphics.
    b. Extra features: Going beyond MVP functionality.
    c. Server component: Creating a full-stack app.
- Write your README as if it was for a production service, including:
    a. Description of the problem and solution.
    b. Whether the solution focuses on back-end, front-end or if it's full stack.
    c. Reasoning behind your technical choices, including architectural.
    d. Trade-offs you might have made, anything you left out, or what you might do differently if you were to spend additional time on the project.
    e. Link to the hosted application, if applicable.

How We Review

We value quality over featurefulness. It is fine to leave extra features aside provided you call them out in your project's README. The goal of this coding challenge is to help us identify what you consider production-ready code. You should consider this code ready for final review with your colleague, i.e. this would be the last step before deploying to production.

The aspects of your code we will assess include:

- **UX:** Is the interface understandable and pleasing to use?
- **Clarity:** Does the README clearly and concisely explain the problem and solution? Are technical trade offs explained?
- **Correctness:** Does the application do what was asked? If there is anything missing, does the README explain why it is missing?
- **Code quality:** Is the code simple, easy to understand, and maintainable? Are there any code smells or other red flags? Does object-oriented code follow principles such as the single responsibility principle? Is the coding style consistent throughout the codebase?