Monopoly Readme *- 30 April 2013*

Andrew Mittereder (amittere), Peter Marino (pmarino), Tyler Hedrick (thedrick)
Source: www.github.com/thedrick/Monopoly
Site: http://cmuopoly.cloudapp.net/
Video: http://www.youtube.com/watch?v=vLegMq71gJE

Overview

Our term project is a recreation of the classic game 'Monopoly,' adapted to Carnegie Mellon. To use our game, begin by navigating to the website listed above on a mobile device for each player, and also on one or more desktop/laptop computers that will act as the boards. Follow the on-screen prompts as one player hosts a game and the others join it. Once everyone has joined, click 'Start Game' when you are ready to play. Playing the game should follow very closely to the game of Monopoly that you're familiar with, with your mobile phone acting as anything you'd have in your hand during the game and the desktop/laptop client acting as the board.

Note that hopefully all features you enjoy from the classic Monopoly game are replicated in our version. Be sure to try out rolling (and rolling doubles), buying properties, buying houses and hotels, a variety of chance and community chest cards, jail (and proper jail rules including bail and get out of jail free cards), owing more than you have on hand in cash forcing you to sell developments or mortgage properties, actually going bankrupt and losing the game, and winning the game (if you have time to play that long!)

If you are in need of a little help, you can use a special function that we created for testing purposes. When you are on the roll screen, use the function forceRoll(int, bool) where int represents the number of spaces to move (please only use natural numbers) and bool represents whether or not you wish to have the roll count as a 'doubles' roll. If you can't access the console on your phone, you can start a mobile client on a regular laptop by navigating directly to the mobile.html page located in our root directory -- it will allow you to join the game as if you were playing on a phone, although obviously the design elements suffer as this is not how the game was intended to be displayed.

We hope you enjoy our final project. We certainly enjoyed creating it, and have worked extremely hard to make it as close to the original experience as possible -- while also attempting to make it truly a 'cross-platform mobile web app.'

Technologies

- Javascript
    - Found in thousands of lines in all .js files in our js folder as well as our app.js file. Specifically, view monopoly.js, which contains object information for properties, players, and games. Javascript is ubiquitous in our game.
- HTML
    - Also very widespread -- HTML, including forms, is used in every single screen in our game, from powering our login screens to helping to design the board itself. HTML is also dynamically created using javascript and jQuery on page visits.
- CSS
    - CSS is used to style every HTML page, with a wide variety of specific selectors and techniques that allow for a clean but hopefully board-game-esque feel to our design. This includes media queries for landscape vs portrait orientations on mobile.
- DOM Manipulation
    - DOM manipulation is used often, whether it be to populate the trade, inspect, or manage screens. Check any of these HTML & js files to see this in action.
- jQuery
    - In order to successfully manipulate the DOM, we use jQuery. Anywhere that DOM manipulation can be found will also include a decent amount of jQuery calls. jQuery is also used for AJAX calls.
- AJAX-server
    - We utilize AJAX calls primarily during the game creation stage of our implementation. This can be found in the first hundred lines of our app.js file.
- AJAX-client
    - On the other side, our mobile.html and desktop.html consume this API when trying to start to host a game. This can be found in their corresponding javascript files.
- node.js & Express server
    - A basic Express server is used in order to serve our html files, redirect as necessary, and to send our socketserver information.
- Websockets
    - Socket.io is the bread-and-butter of our game, as it requires an immense amount of communication in the direction from server->clients, as the server computes all of the logic for the game and the clients merely display this information meaningfully.
- Caching and LocalStorage (line 289 (mobile.js), before each load of FBSDK)
    - LocalStorage is used to store chat and event logs on the board side, as well as cache Facebook data to remove the Facebook SDK dependency if it has already

been loaded. LocalStorage is necessary for trading to have each client store information about the other and deliver data back to the server.

- Server-side Databases (MongoDB) (top section of the app.js file, grep for “query”)
    - MongoDB is used to preserve game states, to maintain a collection of users and boards, and to keep all of the property card, chance card, and community card data the same across all games.
- Facebook API & Authentication (in document.ready function for most client js files)
    - In order to ensure users are real users and keep user accounts safe and separate, we have used the Facebook API to handle our login process. This also allows us to use profile pictures and player names instead of having users enter them manually.
- underscore.js
    - This independent library allows for some functional programming concepts to be used in javascript, such as map, reduce, filter, and etc. We use this where appropriate in some of our server functions. Specifically, checkMonopoly() found line ~1360 of app.js uses it extensively.