Updated: 12 Sep 2018

A2 Problem Statement

Due: 18 Sep 2018 @ 11:59 a.m. — Submission Form

Overview & Objectives

In this assignment, you'll build a browser-based version of John Conway's Game of Life, a cellular automaton that simulates the evolution of an organism using a simple deterministic rule.

The main purpose of the assignment is to give you more practice writing high-quality code in JavaScript, using the idioms you've learned in class. These are: abstracting iteration with functionals, using closures to structure data types, and structuring control flow with events.

In applying these idioms, you should focus on achieving two goals: separation of concerns and control of module dependencies.

Separation of concerns means factoring out different aspects of the functionality so that they can be understood and modified separately. For example, in this assignment, some concerns might include: the rules of the game, the layout of the board, how the user interacts with the game, how the game is displayed, and so on.

A dependence exists when changes to the text of one module requires changes to another. A primary aim in software design is to minimize the dependencies between modules, so that changes can be made locally. Reducing dependencies is achieved by designing interfaces that are as minimal as possible, by ensuring that implementation details are encapsulated, and by avoiding unnecessary references between modules. A good separation of concerns is vital to achieving minimal dependencies also.

Specification

Your task is to build a browser-based version of John Conway's Game of Life. You should build a browser-based version of Game of Life that allows the user to select one of a collection of preset starting states, or create an arbitrary starting configuration. Moreover, a user should be able to start and stop an instance of the animation while it is running.

In order to display the progression of each life, your grid should be at least 24 cells by 24 cells (larger board sizes are allowed and encouraged!), and may be fixed in size. When choosing how long each step of the game should be, be sure that the game state at each step is visible — steps should not progress instantaneously, and the user should be able to pause the game at any given state.

Your implementation should be loaded as a web page, and run without a server. This project requires client-side code only. You should only be writing HTML and JavaScript files that run directly in a web browser, and there is no need to provide any server-side code to generate these pages.

Your code should include documentation of public interfaces, and should be commented appropriately. Testing graphical behavior is hard, and we don't expect you to do this automatically — eyeballing the output is enough. Although you are not writing tests for this assignment, the other aspects of your code should be structured so that they could be tested automatically.

Deliverables

- Your code, comprising a main file (index.html) which when loaded plays the game, and separate JavaScript files containing the game functionality, which are imported by the HTML file.
- A README.md file at the top level of the directory, that includes a brief commentary explaining
  - (a) What concerns you identified, and how you separated them
  - (b) How you exploited functionals in your code
  - (c) Any interesting design ideas you had or tradeoffs that you made
- Fill out the submission form.

Grading

See the rubric.

Hints

Documentation of Public Interfaces: You are not required to use any particular JavaScript annotation tool (if you'd like to, JSDoc is a popular tool), nor any particular styling of comments. Recall from 6.031 that to document your interfaces, you wrote brief, stylized comments for each function and, for each module, a very brief summary of what the module provides.

Use of Functionals: This assignment provides many opportunities to use functionals, as taught in lecture, to eliminate duplication in the code and to make it more succinct and more elegant. In particular, you may want to think about how you iterate over the neighbors of a cell, and whether this can be abstracted. One of the former 6.170 TAs (Harihar Subramanyam) made a guide about common misuses of functionals. It's highly recommended that you read it before starting the pset.

Additional References: The Wikipedia article on Game of Life; an implementation using an external graphics library, with simple source code (but you can do better!); a video of Conway talking about the game; a longer discussion of the game, its properties, as well as different objects and their properties (including stationary, oscillating, and gliding objects).

The Grid: We suggest using CSS Grid (here's an example), but you can implement it any way you wish, so long as you make use of DOM manipulation as taught in class.

Edge Behavior: When live cells reach the edge of the grid, there are a number of ways they could behave, including: cells off the board could simply all be "dead" cells, or "live" cells could wrap around the edge of the board. Any behavior is acceptable, so long as your implementation is consistent and reasonable.

Markdown: When writing your README.md file, check out this cheat sheet for information on how to style markdown.

Updated: 12 Sep 2018

Testing: Although you will not be assessed on writing tests (graders will not run them), if you choose to write tests, we suggest using Jest.