Week 1-3 Side Project

A Challenging Stretch Project for Advanced Students

Introduction & Background

Unlike traditional education, every student that attends Lighthouse Labs has had a very different educational and professional background. Some of our students have a background or even work experience with software, while others will be learning to code for the first time.

Despite being a very intense, fast-paced learning environment, we try to do our very best to cater to these various levels and backgrounds. This stretch project is one such example. We had received feedback that our prep material on Compass, including the stretch work it contains, is not challenging enough for some of our more advanced students. Stretch projects such as this allow us to push them further.

Prior Knowledge & Experience

While it may seem otherwise at first glance, this project does not assume that you have a background with technology or with programming.

That said, this project is intentionally a very big jump from the challenges and project you completed in the prep work. It is not meant to be easy nor a natural progression to the content covered earlier as it is designed in order to challenge and test your ability to rapidly learn and grow without regular amounts of direction.

The focus here is on problem solving while learning new concepts. It is for individuals who are already comfortable with the fundamentals of code (functions, objects, arrays, conditionals, loops).

You may find yourself completely stuck on this exercise, not knowing how to solve a particular problem or fix a particular error, please do not waste hours trying to tackle it. We suggest that you instead take a break and focus on some of the stretch goals for the curriculum projects. This will help you flex some of your programming muscles and make sure you're still working through the bootcamp material.

Duration

This project should take you 40 to 60 hours to complete. We strongly suggest allocating large (3+ hour) focused sessions as is common practice and necessity amongst professional developers.

Assessment

This stretch project is of course optional. Should you choose to complete it, we also have a process for having it reviewed by our instructors. This review allows us to help you improve your code with tangible feedback while also serving as an assessment tool for us.

In order to be tangible and objective with our assessment, an evaluation rubric has been sent along with this document. Use this as a guideline for how you will be assessed for this project.

Deadline

In order to have your submission reviewed in a timely manner you must submit your work **by 9pm on the Sunday preceding your 4th week of bootcamp**.

General Rules

- No collaboration of any kind (peers, mentors, friends, etc) is allowed. Think of this as a solo challenge / test
- Similarly, you may **not** post questions or answers to this project on external forums.
- Do not attempt to directly google solutions to this problem (eg: “example code for HTML battleship game”). The goal is to problem solve and create the logic yourself. That said, you are of course allowed to google and read up on technologies or concepts that you may be unfamiliar with, such as jQuery.
    - Keep track of which tutorials, example solutions, reference documentation or other material you used in order to learn along the way. You will be asked to submit this list.

About The Project

This project will consist of building a battleship game, once again using only HTML, CSS and JavaScript. jQuery will still be used to manage and manipulate the DOM.

There are intentionally no visuals and no assets provided with this project, and many aspects of the UI are left intentionally vague. We are leaving you freedom to get creative with the UI. We encourage you to make the game look presentable and even add extra flair with things like CSS transitions and animations.

Battleship Rules

- Each player has a 10x10 board on which the player is able to place 5 ships:
    - A Carrier, which is 5 tiles long
    - A Battleship, which is 4 tiles long
    - A Cruiser, which is 3 tiles long
    - A Submarine, which is 3 tiles long
    - A Destroyer, which is 2 tiles long
- Each ship can be placed either horizontally or vertically on the board, and cannot be placed partially off the board.
- Each tile is denoted by a coordinate, A-J for columns and 1-10 for rows
    - i.e. the top left corner would be at coordinate A1
- Each player then takes turns picking a tile on the opposing player's grid, taking a shot at that tile.
    - If the tile contains a ship, the shot is a HIT
    - If the tile does not contain a ship, the shot is a MISS
- A ship is sunk if all the tiles for that ship have been marked as a HIT.
- The game ends when one player has sunk all of the opposing players ships.

Functional Requirements

- Allow the player to start a new game.
    - This should reset the board and allow the user to place new ships onto the board.
        - Ships should be able to be rotated so they can be either vertically or horizontally placed.
    - Once the player is content with the ship positioning, they should be able to start the game. Picking a starting player at random or through an option. Come up with a creative way for this to work.
- Allow the player to take a shot on their turn only.
- Allow the player to play against an AI.
    - It's up to you on how complicated to make this AI player.
    - [Stretch] Allow the user to pick the difficulty of the AI player.
- Have a leaderboard for games against the AI

    - This should show how many games the current player has won against the AI, compared to all players
        - There is no need to have the user log in to accomplish this, just let them pick a username to use.
- [Stretch] Allow the player to play against another person on a different computer
    - This will require you to extend your server-side application which will allow for connection between players.
- [Stretch] Allow the player to setup options before starting the game
    - Number of ships
    - Number of shots per turn
    - Board size
- [Stretch] Allow the player to save a replay of the game
    - You should also allow a player to load a replay and step through each turn.

Display Requirements

- Players should be able to see the other players board and their own.
    - The players board should show the following:
        - The players ship placement
        - Any shots the opposing player has made.
    - The opponents board should show the following:
        - Any shots made by the player, and whether it was a hit or a miss
    - Both boards should show:
        - Coordinates of the cells, up to you on how this should be displayed
- The game should also show who's turn it currently is, the player or the opponent.
- Show a list of ships for each player
    - When a ship is destroyed, indicate this in the list of ships
- Show a log of shots and messages to the player.
    - This log should show:
        - All the shots taken by each player
        - A message when a player sinks another ship
    - e.g.:
      Player 1 shoots at A1: HIT
      Player 2 shoots at B5: MISS
      …
      Player 1 shoots at A3: HIT
      Player 1 has sunk a submarine!

Tips

- Read through the jQuery documentation or find some simple tutorials on jQuery. This will help you get a handle on how to access and create DOM elements using jQuery.
- Commit at every step. No commit is too small, but at the same time commit code that is not going to throw an error. One massive commit with all your work is going to result in an unsatisfactory submission.

- Don't look at the code for other games to see how they're implemented, because it will either be overwhelming, overkill, or cheating.
- This should be completed in only HTML, CSS and JavaScript w/ jQuery. You may find references to using SVG or Canvas for your solution. However we strongly advise that you stay away from those approaches.
- Try to break your solution down into small functions that will work together to solve the problem. One massive function will not be accepted, for example.
- Think about how the game will work, what are the various phases the game will progress through.

Submission Requirements & Process

- Setup a project on Github, this will be used to submit your solution to this project.
- Add a linter into your text editor of choice, there is ESLint for Atom, ESLint for Sublime Text or ESLint for VSCode. Otherwise you can use the command line by running `eslint` from the command line. Take a look at the ESLint documentation for instructions. Also use the Lighthouse config for ESLint available here.
- Add a README.md file with the following information to your project. The README should be in Markdown format:
    - About
        - Give some context to what your project is for
    - Example Screenshots (embedded within the readme as image tags)
    - Detailed instructions on how to get the application started.
    - Explain how the game works, and how the user is expected to play the game.
    - Explain how the AI works, what techniques it uses to make a move.
    - A Feature list of your game (options it supports, etc)
    - A list of known issues / bugs
    - A list of features that are on the roadmap but haven't been implemented yet
    - A list of all the external resources (tutorials, docs, example code, etc) that you encountered and used to help you create this library

Once completed, please submit your project code to us via this Google Form.

If you have any questions about this project, please send us an email at web-prep-stretch@lighthouselabs.ca.

That's all for now. Thank you and Good Luck!