Battleship Project Evaluation Criteria

So how does this work anyway?

In each category you will get a mark of either, **Exceptional**, **Acceptable**, **Needs Work** or **Unacceptable**. Scoring a single **Unacceptable** in any category will immediately mean you **will not pass** this project, so make sure you're reading through each of the categories requirements and hitting the mark in each category. **Exceptional**, **Acceptable** and **Needs Work** are all worth 120%, 100% and 60% of the weighting of that category respectively. Your total score will be added up from each category and is scored out of 100 and in order to pass this project you'll need to achieve at least an **80% overall**. Good luck!

	Weight	**Exceptional**	**Acceptable**	**Needs Work**	**Unacceptable**
		120%	**100%**	**60%**	**0%**
Version Control	**7.5%**	Commits were made frequently and deal with small, atomic changes. Commit messages clearly explain the changes made and are succinct. All files and directories that don't belong in the repo have been excluded using `.gitignore` file.	Commits were made at sensible milestones, and for the most part commit messages clearly explain the changes made. Most files and directories that don't belong in the repo have been excluded using `.gitignore` file.	Commits were generally made at sensible milestones in the project. Commit messages only sometimes clearly explain the changes made. Some files and directories that don't belong in the repo have been excluded using `gitignore` file.	All changes were added to the repo in one or two big commits, and/or commit messages are for the most part not useful.
Functional Requirements	**40%**	All functional requirements have been met, including some of the stretch goals. Some minor bugs may be present.	All non-stretch functional requirements have been met. Some minor bugs may be present.	Most functional requirements have been met, but there are some styling issues or some non-trivial bugs may be present.	Little to none of the functional requirements have been met.
Code Style	**7.5%**	Code follows the included style guide, i.e. there are no ESLint errors and no rules have been disabled.	Code generally follows the style guide. There are some small ESLint errors. There are less than 10 ESLint warnings	Code follows some of the style guide, and there are some major ESLint warnings. There are more than 10 ESLint warnings.	Code doesn't follow the included style guide, or ESLint is not setup.

Commenting	**2.5%**	Where necessary, comments are used to clearly explain what the code is doing and why, and are in an easy-to-understand and consistent style.	Comments are generally useful in helping a reader understand the code	Comments mostly serve as a distraction to the reader or are missing where they are necessary.	Comments are not used to help the reader understand the code (they are unclear, redundant or missing).
JavaScript, HTML and CSS Best Practices	**17.5%**	JavaScript, HTML and CSS code consistently follows best practices. jQuery being used to build out elements.	JavaScript, HTML and CSS code generally follows best practices. jQuery being used to build out elements.	JavaScript, HTML and CSS code sometimes follows best practices. jQuery being used to build out elements.	JavaScript, HTML and CSS code does not follow best practices. Strings are being used to build out elements.
Code Modularity and Reusability	**17.5%**	Code is sensibly modularized. Functions are single-serving, have no side-effects wherever possible, and helper functions are defined in appropriate cases.	Functions are generally single-serving and for the most part side-effect free. Some helper functions have been defined but not always in appropriate cases.	Functions are sometimes single-serving and may have side-effects that could be avoided. Little or no helper functions have been defined, or are used in inappropriate cases.	Code is not modularized or reusable
Documentation	**7.5%**	A properly formatted README is defined, explaining the purpose of the project and which features are implemented, how to play the game and has example images. An example page is setup on GitHub Pages.	README is missing a few section, but includes explanation of how to use the API, and implemented features.	README is missing most sections, but at least includes API documentation.	No README file exists in the repo, or README file is empty.