Design

A general note:

This project, even for "merely" Tic Tac Toe, took a ***heck*** *of a lot* longer to implement than I thought it would. The reason? Design. I went through essentially five "rewrites" just to be able to implement a two-player version (not shown here), and successfully incorporating the AI required a complete overhaul of the hopelessly jumbled structure. (This was, in fact, the general inspiration for my project's subtitle.)

Having some class(es):

Because there were many components that come together in creating and playing out the game, it was wonderfully convenient that Javascript happened to be an OOP. Although I couldn't really find Javascript "classes" in the traditional OOP sense of the word, there were ways to quasi-use them based on the fact that pretty much *everything* in Javascript is an object. So I created definitions for a Location object, a Move object (that incorporates a Location object), and a Board object. It might seem a bit confusing at first why I'd bother having both Location and Move objects; after all, they're pretty much the same thing in this implementation. However, I created this framework with the idea of flexibility – that more complicated games could use this same basic structure. A move in checkers, for instance, could involve a sequence of locations, a list of pieces captured while jumping from location to location, the promotion of a normal piece to a king piece – certainly more than one location. The framework that I have in place here, though, could be potentially generalized to incorporate all these different ideas.

The board is least boring:

At least in Tic Tac Toe, however, the definitions of Move and Location were pretty straightforward to write. Of all the quasi-classes, defining the Board was by far the most difficult task. The Board sees everything and is uniquely tied with the fundamental attributes of the game; it knows the positions of all the Xs and Os on the board, whose turn it is, whether the game is over (and if so, if there is a winner), among other properties. It makes sense to include all these properties with the Board because, in general, the board defines the game. All the calculations (i.e., for the AI) are done on the board based on whose turn it is to play on the board and consider the possible moves to be played on the board. It makes sense why much of the action happens on the board.

In the beginning was the Index:

At least, most of the *logical* action – the internal thinking about the game – happens on the board. This component is mostly separate from the rest of the program – like handling the appearance of the board during the game and after the game, the rendering of the difficulty status and the game result status, handling human errors in making game inputs – but everything comes together in index.js, which deals with these aforementioned tasks and how they connect with the internal logic. And all of this comes visually together in index.htm, which just displays all the buttons that get this Tic Tac Toe party started.