**Design Paradigm**

A paradigm in the software world guides the way that developers view a given problem and organize a solution.

The project we inherited was written in JavaScript and designed using HTML. All the code was comprised in one JavaScript file with no classes but multiple functions. This leads us to believe that the design paradigm that the group chose was Function – Oriented Design. No class objects were created in the logic of the code. For our 2nd project we continued the Function – Oriented Design approach and added new methods to implement the AI and the power shot. It would have been a major change to convert the design of this code to an object-oriented design and so we avoided it.

Another feature we noticed in the code was that the requirements were broken down into functions that each performed specific action. The requirements that were provided were a big picture of what was required to be done. Every requirement had to be broken down into smaller segments in various ways.

The code had a separate function that dealt with the various boards, ensuring the hits were registered correctly on the board, ships were placed correctly and legally, updating the required boards and so on. While building on the code we added functions for the easy, medium, and hard level shots for the AI and a function that dealt with the power shot (our custom addition) so we could keep the Function-Oriented Design consistent. As for the Top-down Functional Decomposition the main requirement was to create a battleship game. This requirement had smaller requirements such as legal placement of ships, shooting in the correct place, or drawing the boards. Each had been implemented individually, then called from each other to make a cohesive game.