**Project #1**

**User Story Tic Tac Toe Example**

**As a user:**

* Should be able to see game page with a start button
* I should be able to click start New Game
* A playable board should appear on screen
* I should be able to see who’s current player
* I should be able to click on a playable square and fill it with current player eg. X or O
* The game should stop if a draw or a winner is found.
* Should display game message displaying who the winner was
* Should be able to start another New Game

**Technical Requirements**

Your app must:

* **Render a game in the browser**
* **Switch turns** between two players
* **Design logic for winning** & **visually display which player won**
* **Include separate HTML / CSS / JavaScript files**
* Stick with **KISS (Keep It Simple Stupid)** and **DRY (Don't Repeat Yourself)** principles
* Use **Javascript or jQuery** for **DOM manipulation**
* **Deploy your game online**, where the rest of the world can access it
* Use **semantic markup** for HTML and CSS (adhere to best practices)

**Necessary Deliverables**

* A **working game, built by you**, hosted somewhere on the internet
* A **link to your hosted working game** in the URL section of your Github repo
* A **git repository hosted on Github**, with a link to your hosted game, and frequent commits dating back to the very beginning of the project
* **A readme.md file** with explanations of the technologies used, the approach taken, installation instructions, unsolved problems, etc.

**Suggested Ways to Get Started**

* **Break the project down into different components** (data, presentation, views, style, DOM manipulation) and brainstorm each component individually. Use whiteboards!
* **Use your Development Tools** (console.log, inspector, alert statements, etc) to debug and solve problems
* Work through the lessons in class & ask questions when you need to! Think about adding relevant code to your game each night, instead of, you know... *procrastinating*.
* **Commit early, commit often.** Don’t be afraid to break something because you can always go back in time to a previous version.
* **Consult documentation resources** (MDN, jQuery, etc.) at home to better understand what you’ll be getting into.
* **Don’t be afraid to write code that you know you will have to remove later.** Create temporary elements (buttons, links, etc) that trigger events if real data is not available. For example, if you’re trying to figure out how to change some text when the game is over but you haven’t solved the win/lose game logic, you can create a button to simulate that until then.

Inspo:

* Opposite arrow game
* Snake (perhaps too complex using JS)
* How many can you guess right in 60 seconds
* Escape game?
* Sudoku? Magic number = 45
* Memory game (LARGE!!!)
* Memory order game