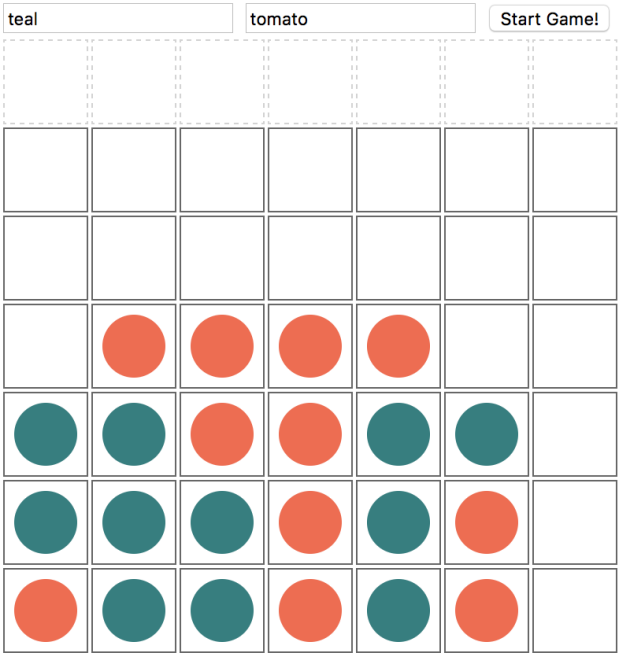
Connect Four OO

Download starter code <../connect-four-oo.zip>



<_images/connect4-oo.png>

In this exercise, you'll turn a non-OO-designed version of the game Connect Four into a more featureful, OO version.

You can try out our solution <http://connect4-oo-rithm.surge.sh>

> **Warning: Start With Our Starter Code**
>
> Instead of using your own code for non-OO Connect Four, **start with the starter code in our zip file!**

Part One: Make *Game* Into a Class

Right now, our Connect Four is a bunch of disconnected functions and a few global variables.

This can make it hard to see how things work, and would make it hard to restart a game (quick—which variables would you have to reset to start a game?)

Let's move this to being a class.

Initially, we'll start with one class, ***Game***. The players will still just be numbers for player #1 and #2.

- What are the instance variables you'll need on the ***Game***?
  - for example: height, width, and the board will move from global variables to instance attributes on the class. What else should move?
- Make a constructor that sets default values for these

- Move the current functions onto the class as methods
  - This will require mildly rewriting some of these to change how you access variables and call other methods

You should end up with all of the code being in the ***Game*** class, with the only other code being a single line at the bottom:

```
new Game(6, 7);   // assuming constructor takes height, width
```

Part Two: Small Improvements

Make it so that you have a button to “start the game” — it should only start the game when this is clicked, and you should be able to click this to restart a new game.

Add a property for when the game is over, and make it so that you can’t continue to make moves after the game has ended.

Part Three: Make Player a Class

Right now, the players are just numbers, and we have hard-coded player numbers and colors in the CSS.

Make it so that there is a ***Player*** class. It should have a constructor that takes a string color name (eg, “orange” or “#ff3366”) and store that on the player instance.

The ***Game*** should keep track of the current player *object*, not the current player number.

Update the code so that the player pieces are the right color for them, rather than being hardcoded in CSS as red or blue.

Add a small form to the HTML that lets you enter the colors for the players, so that when you start a new game, it uses these player colors.

Further Study

If you have more time and would like more tasks, here are some things to play with:

- Make it so that you can have more than two players
- The look-and-feel is very sparse: add animations, better graphics for the board or pieces, and other CSS ideas. You could even use bootstrap for things like modals for the start-new-game form.
- Make a very simple computer player: it could pick a random column and place a piece there. Can you do this in an object-oriented way, so there is a ***ComputerPlayer*** class?

- Want something ambitious? Try to build another game using OOP! Here are some ideas to get you started:
  - Checkers <https://en.wikipedia.org/wiki/Draughts>
  - Othello <https://en.wikipedia.org/wiki/Reversi>

Solution

You can view our solution <solution/index.html>