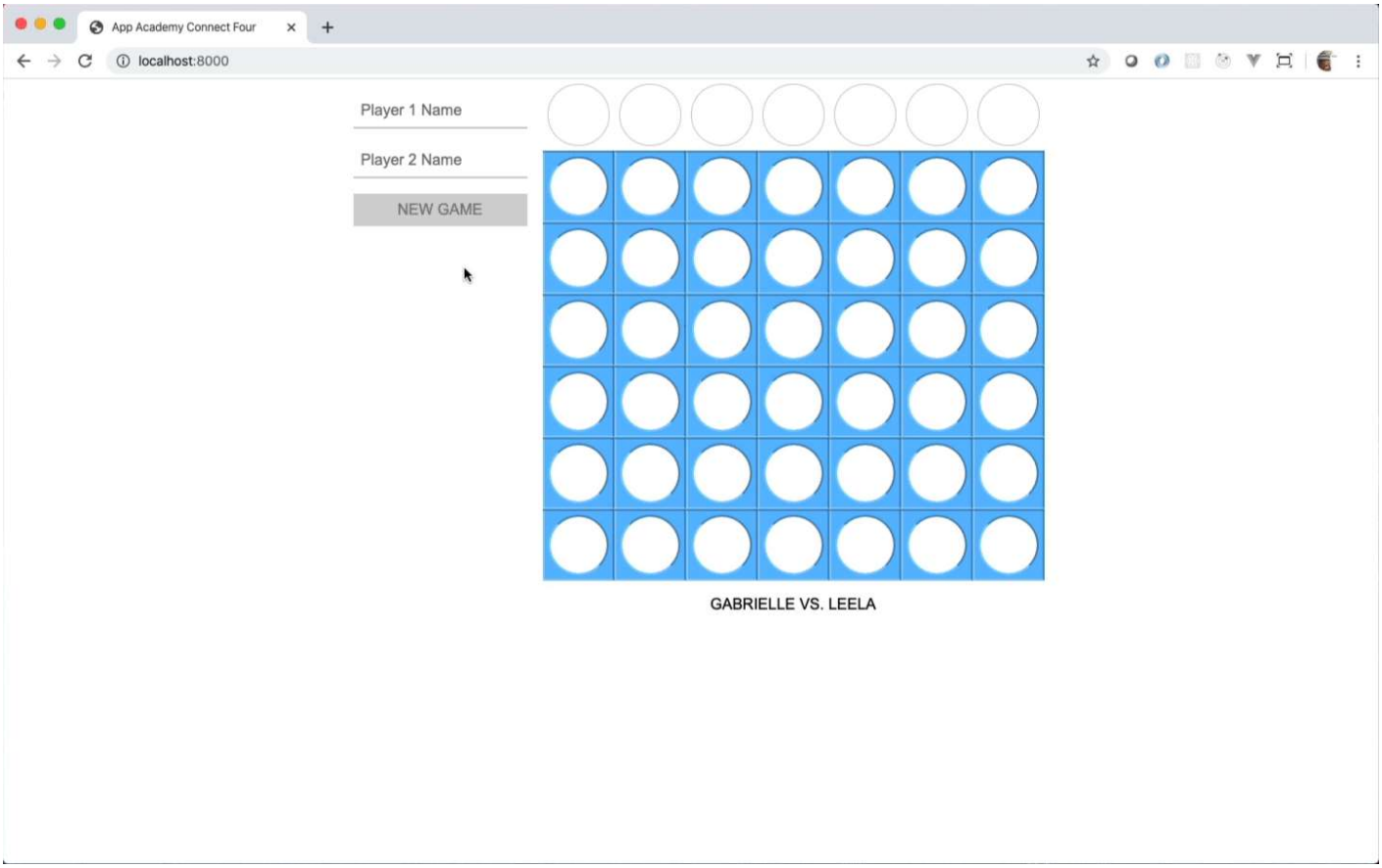
Indicating The Current Player



Now, it's time to let the players make their moves. They do this by clicking on the click target at the top of the column. When they do that, it's up to the game to determine where that token should "fall" to, switch to the next player, update the tokens on the board, and update the color of the hover of the click targets.

Again, make sure you're still on HTTP.

In this step, you'll make it so that the current player can see the color of their token when they hover over the click targets above the board. First, you'll update the game to track the current player. Then, you'll use that state to update the UI.

You're extending the responsibility of the game to track the current player. This is still within the scope of responsibility of what a `Game` should do. So, adding that data directly to the class for it to manage is ok and normal.

- In **game.js**:
  - In the constructor of the `Game` class, create a new instance variable to track the current player and set it to `1` to indicate that its the first player.
  - Create a method called `playInColumn()` that will handle the click of the user. In that method, change the value of the current player to the other player. If it was `1`, then make it `2`; otherwise, make it `1`.

- In **connect-four.js**:
  - In the event handler for "DOMContentLoaded", register an event handler for the "click" event on `#click-targets`.
  - In the `#click-targets` event handler that you just created, have it call the `playInColumn` method of the object in the global variable `game`. Then, have it call the `updateUI` method.
  - In the `updateUI` method, in the portion of the code that uses the `game` object, have it get the current player from the instance variable that you created in the constructor of the `Game` class. Use that value to determine if you need to add "red" and remove "black" from the `#click-targets` element for player two, or if you need to add "black" and remove "red" for player one from that element.