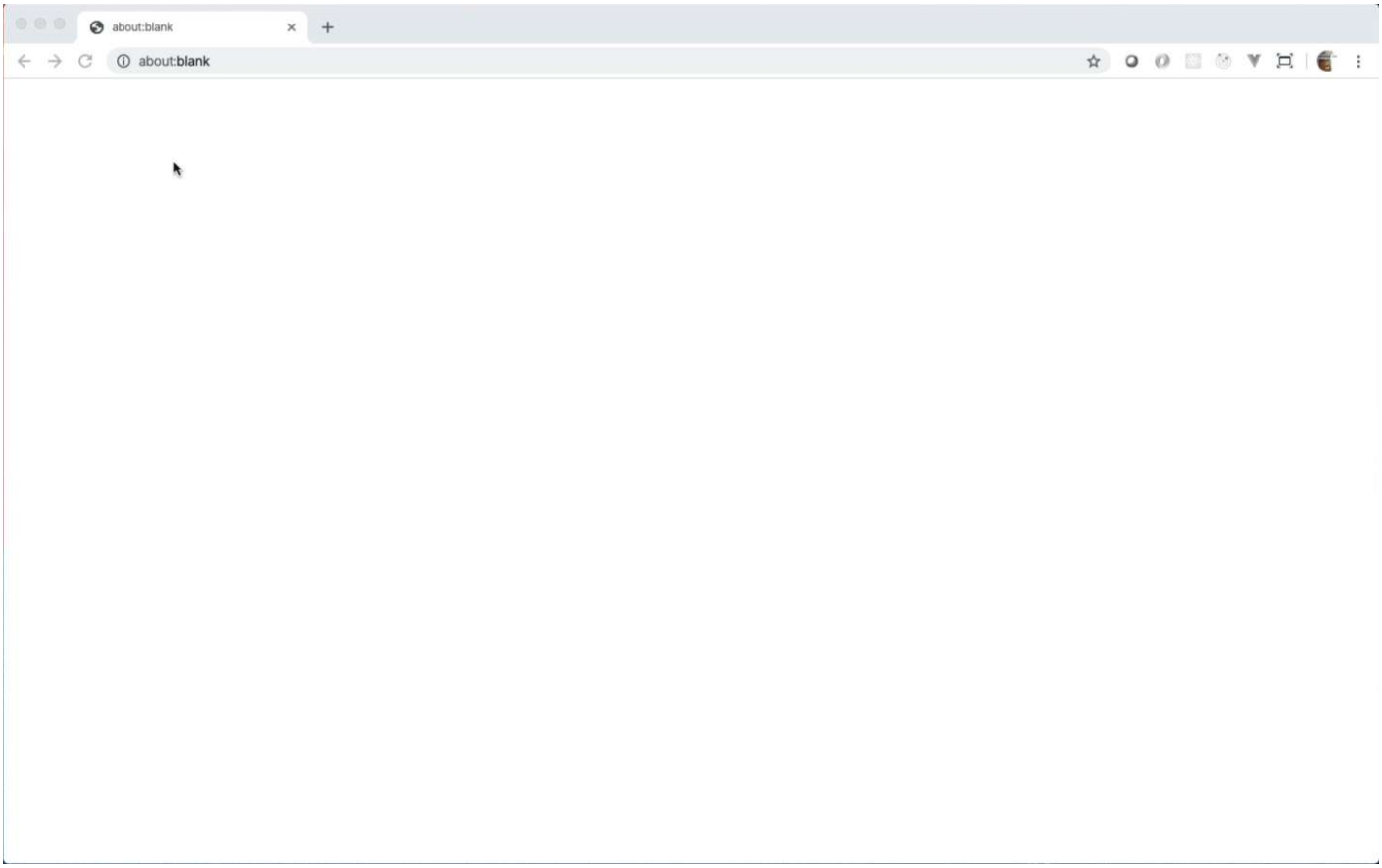
Creating A Game



In this requirement, you will get the "new game" functionality to work.

You are now familiar enough with CSS selectors, that these instructions will now start using them for shorthand. If you see instructions like *Add the class "is-invisible" to* `#board-holder`, then that means to add the class "is-invisible" to the element that has the id "board-holder". Since ids are supposed to be unique in the HTML, you should only find one element with any specified id value.

You will implement that user story by performing the following steps. There are quite a few here because you're getting some of the plumbing in place for the first time.

Make sure you're on HTTP

Make sure that you're serving this from HTTP rather than just from a file. Again, those instructions are:

1. Type `python3 -m http.server` in your terminal. Note the port number that the HTTP server is using.
2. Open a browser.
3. Type the URL `http://localhost:«port number»` in the address bar where the value for "port number" is replaced by the port number reported by Python in the first step.

HTML element class name manipulation

Remember that when adding and removing class names from an element, it helps to use `add` and `remove` methods of the `classList` property of the element. Check out the documentation to remind yourself how to do that.

The "Game" class

You're going to create a class named `Game`. It will have the single responsibility of managing the state of the game. In future steps, you will create more classes that it will use to help delegate behavior. In this step, `Game` will manage player names as well as creating the name of the specific game.

The steps to create a game

These steps will help guide you through creating a game.

- In **index.html**
  - Add the "is-invisible" class to `#board-holder`.
  - Add the "disabled" attribute to `#new-game`.

- In **connect-four.js**: Add an event handler for the window's "DOMContentLoaded" event. In that handler, have it:
  - At the top of the file, declare a global variable named `game` and set it to `undefined`.
  - Create an event handler for the "keyup" event of `#player-1-name`. In the event handler, have it set `#new-game`'s "disabled" property to `false` if both `#player-1-name` and `#player-2-name` have non-empty content, enable `#new-game`. Otherwise, disable `#new-game`.
  - Create an event handler for the "keyup" event of `#player-2-name`. In this event handler, do the exact same thing you did in the `#player-1-name` handler. Now, think, did you copy and paste some code? If so, can you refactor it somehow to remove duplication because the intended behavior is identical?
  - Create an event handler for the "click" event of `#new-game` that, when clicked:
    - Sets the global variable `game` to a new instance of the `Game` class passing in the two players' names.
    - Sets the values of the two player name input elements to empty strings.
    - Sets the disabled property on `#new-game` to `true`, thereby disabling it. (See if you have this functionality already written somewhere and, if you do, somehow reuse it so prevent code duplication.)
    - Calls a function named `updateUI()`.

  - Declare a function named `updateUI()` after the `game` variable declaration and before the event listener for "DOMContentLoaded". In that function, put the following logic:

    - If `game` is `undefined`, have it add the "is-invisible" class to `#board-holder`.
    - if `game` is not `undefined` have it remove the "is-invisible" class from `#board-holder` and set the inner HTML of the `#game-name` element to the value returned by the `getName()` function of the object stored in the `game` variable.

- At the top of the file, import the `Game` class from the file at the path **./game.js** using `import { Game } from './game.js';` because you want to load a file that will contain the `Game` class. Remember that you have to use the ".js" on the file name because you're loading these directly in the browser.

- Create a file named **game.js** in the same directory as the **connect-four.js** file. In that file, declare and export a class named `Game` that has
    - A constructor that takes the names of the two players and sets them to instance variables on the object. (Remember that creating instance variables requires the use of the `this` keyword, such as `this.name1 = playerOneName;`, for example, creates an instance variable named `name1` and sets it to the value of `playerOneName`.)
    - A method named `getName()` that returns a string of "*Player 1 Name* vs. *Player 2 Name*".