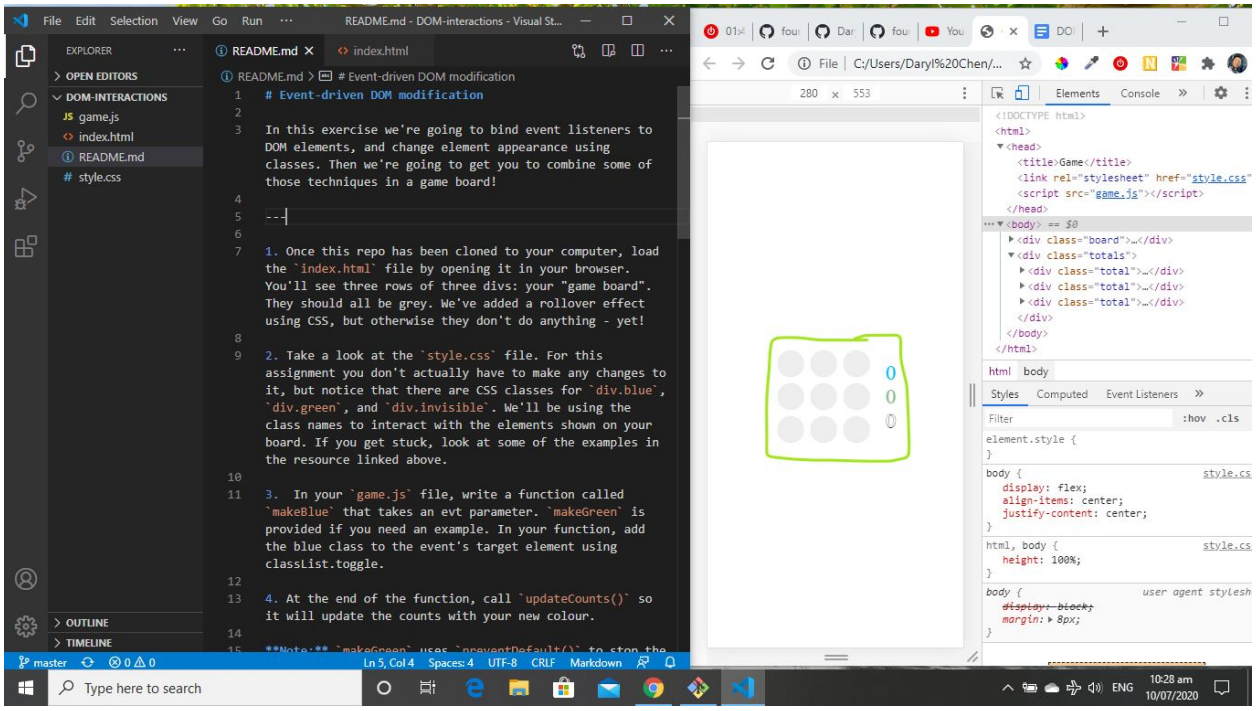
DOM-interactions

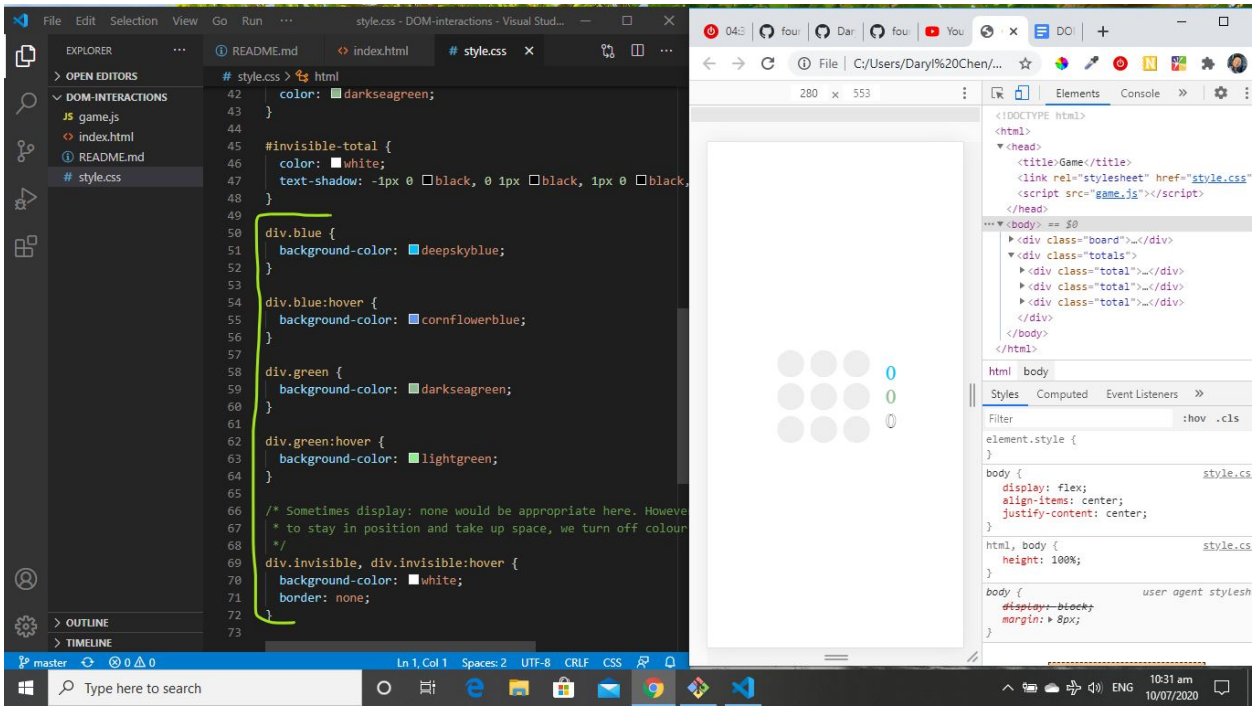
```
# Event-driven DOM modification

In this exercise we're going to bind event listeners to DOM elements, and change element appearance using classes. Then we're going to get you to combine some of those techniques in a game board!

---
```




```
2. Take a look at the `style.css` file. For this assignment you don't actually have to make any changes to it, but notice that there are CSS classes for `div.blue`, `div.green`, and `div.invisible`. We'll be using the class names to interact with the elements shown on your board. If you get stuck, look at some of the examples in the resource linked above.
```



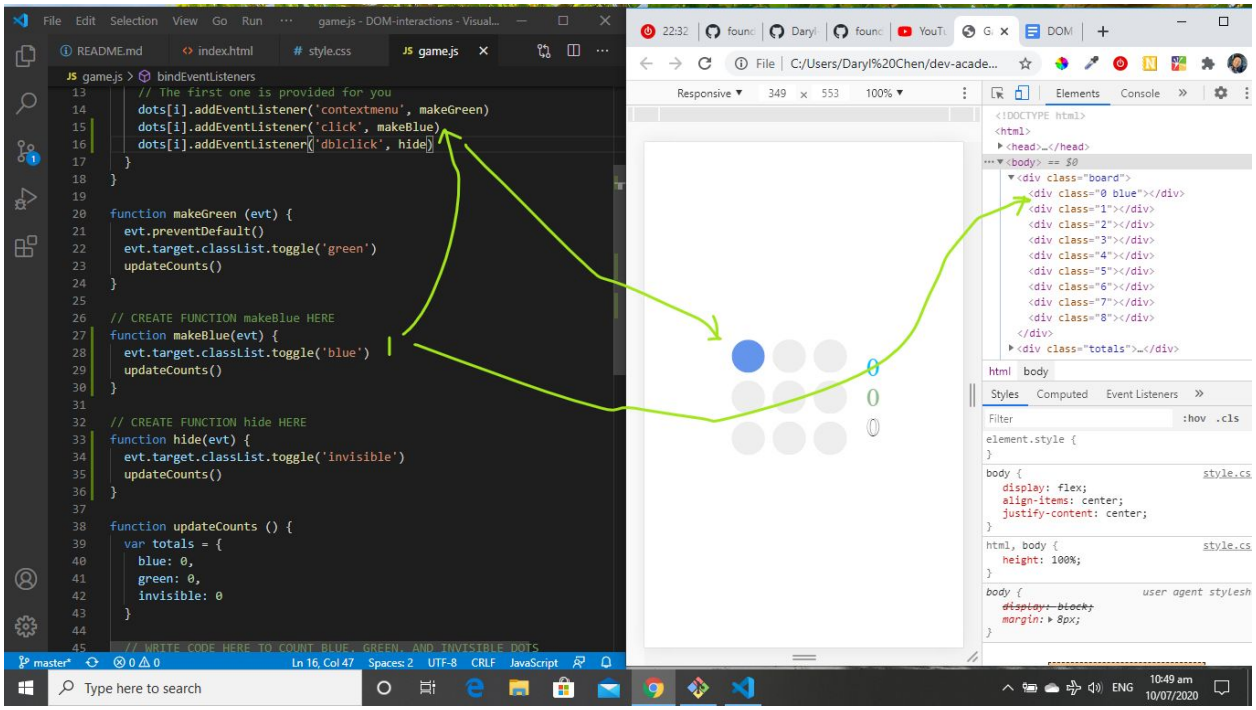
3. In your `game.js` file, write a function called `makeBlue` that takes an evt parameter. `makeGreen` is provided if you need an example. In your function, add the blue class to the event's target element using classList.toggle.

4. At the end of the function, call `updateCounts()` so it will update the counts with your new colour.

Note: `makeGreen` uses `preventDefault()` to stop the right mouse button's context menu from appearing. You don't need it for the other event handlers.

5. When you've written makeBlue, add an event listener for it in the bindEventListeners function beneath the one for `makeGreen`. It should look very similar to the one above it, but use the 'click' event and your `makeBlue` function.

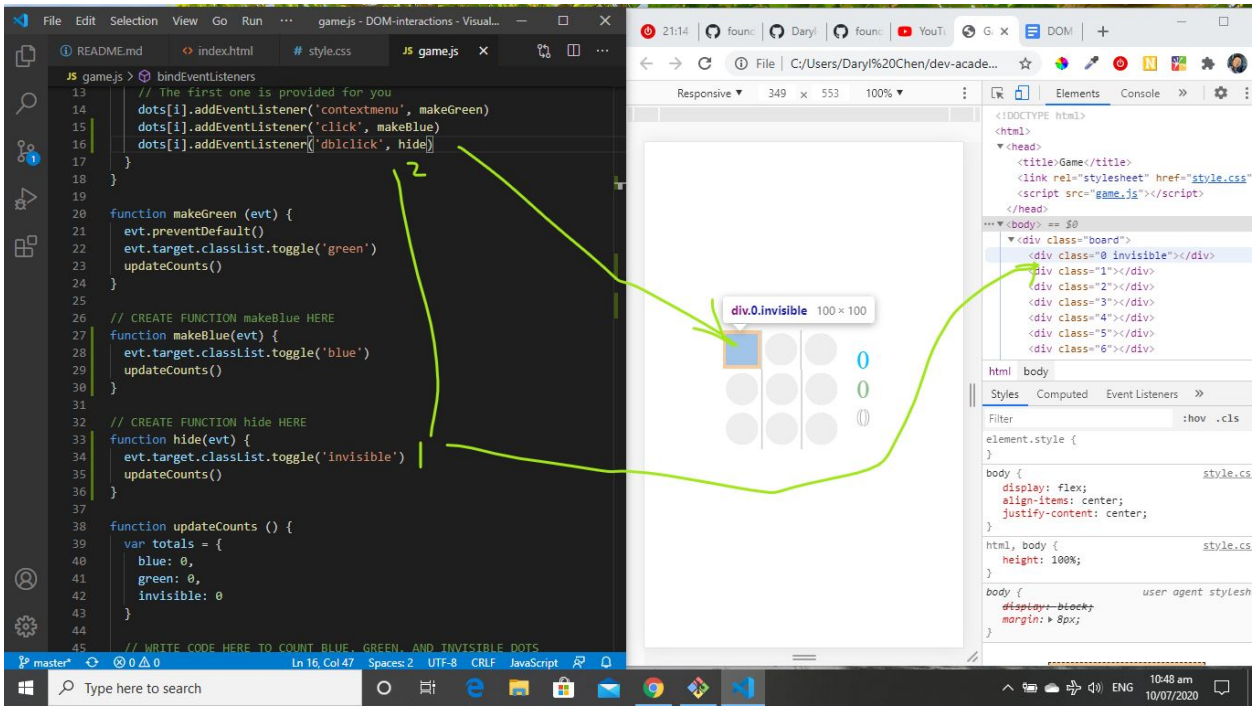
Reload the page in your browser and test the changes. If all goes well, when you click the left mouse button on a dot it should go blue, and go green if you click the right mouse button!



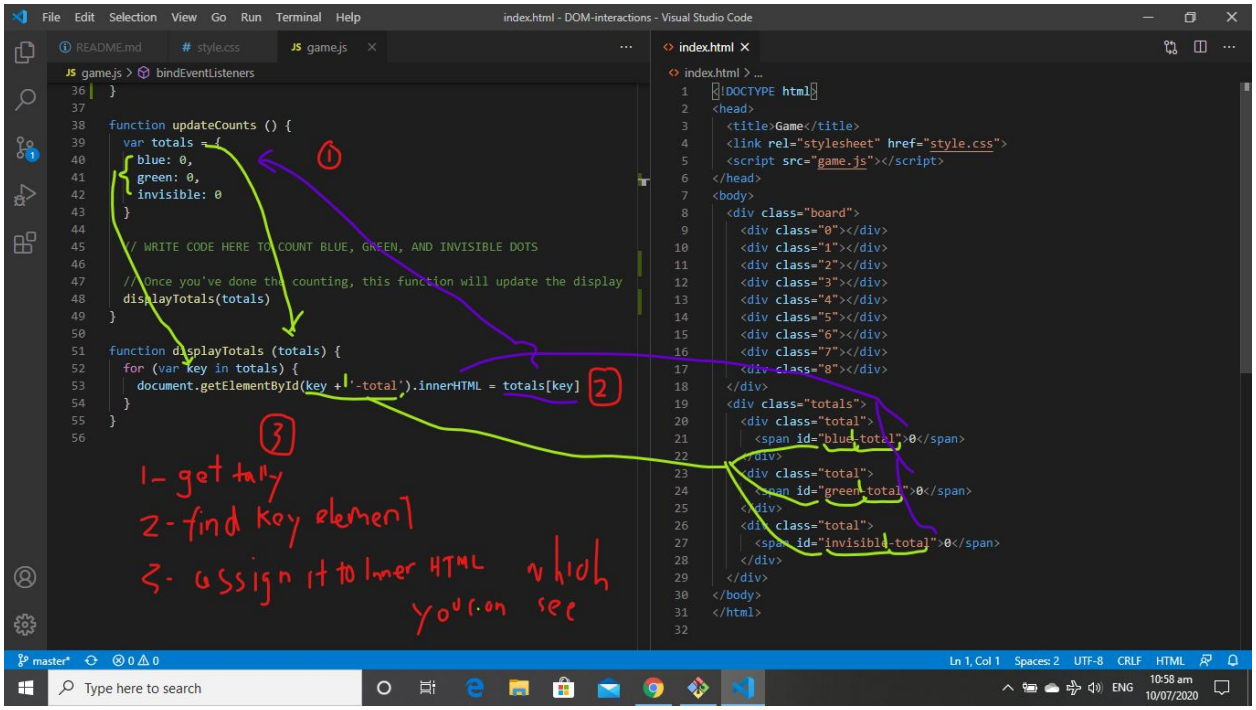
6. Write a function `hide` that takes an evt parameter. It'll look almost exactly the same as the first two event handlers, but add the class `invisible`. Don't forget to call updateCounts() at the end of the function.

```
// CREATE FUNCTION hide HERE
function hide(evt) {
  evt.target.classList.toggle('invisible')
  updateCounts()
}
```

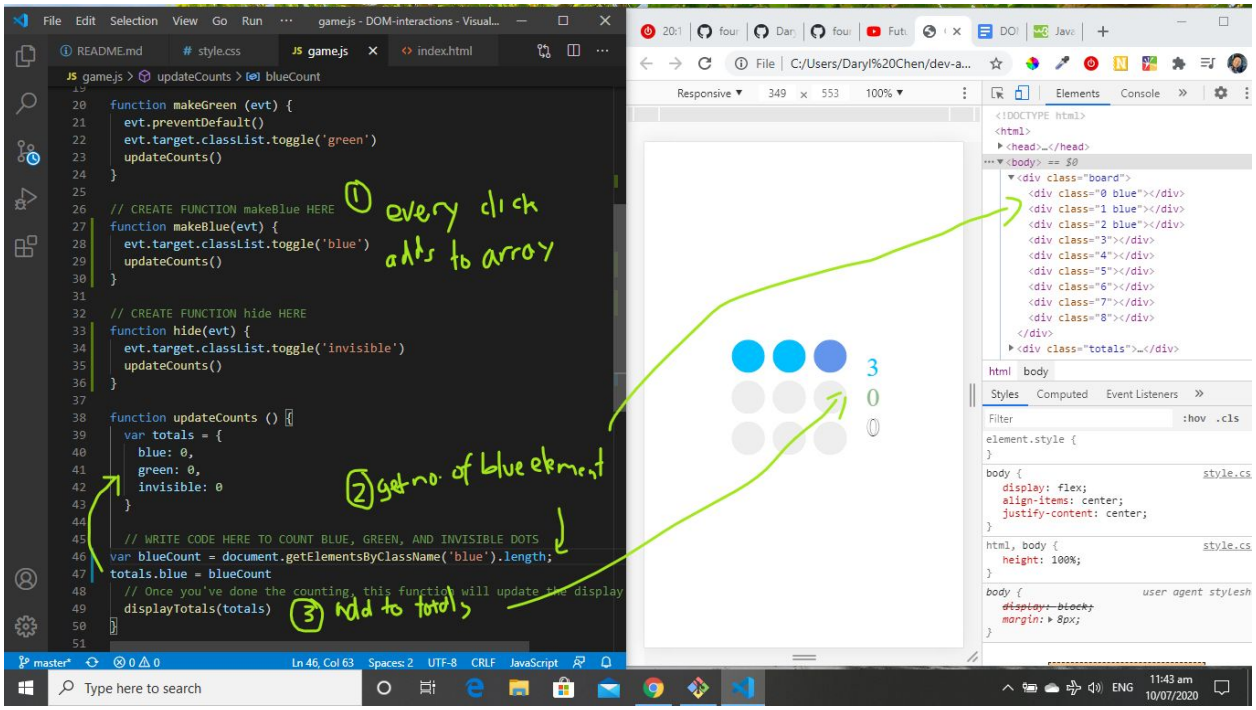
7. Add an event listener for your `hide` function to bindEventListeners. Use the 'dblclick' event. Reload in the browser and test it out. A double-click should make dots disappear, and they should reappear with a second double-click in the same place.



You're now going to update the display counter on the page to match the number of dots that are `blue`, `green`, or `invisible`. Take a look at the `displayTotals` function. It has been written for you but see if you can understand what it's doing. `displayTotals` has been called in our `updateCounts` function and passed an object called `totals` that contains the number of each colour of dot. \



8. Complete the `updateCounts` function so that it increases the count of each property in the `totals` object, depending on how many of the corresponding type of dots there are on the page. You might like to use `getElementsByClassName` which returns an array, because there could be many DOM elements with the same class, and then count how many items are in that array. The [MDN documentation on `getElementsByClassName`](https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByClassName) might be helpful here. (*hint:* search the web for "how to check the length of an array"). Then assign that number to the right property in the `totals` object.



9. Repeat that for all 'blue', 'green' and 'invisible' classes. Reload the browser periodically to see if your changes are working. If they are, the counts to the right of the board should start going up!