MIKESCHOBER.DEV

Do you like JavaScript (in twenty minutes!)

A very simple, project-based introduction to JavaScript

ROADMAP

- JS variables
- JS data types
- JS functions
- JS math functionality
- the DOM
- Developer tools
- Roadmap for building the project

JS VARIABLES

- In JS, data can be stored in variables
- Examples of variable declarations:

```
let randNum = 0;

const numNoChg = 7;
```

- Variables declared with ‘let’ can be reassigned

```
randNum = 14; (this would be allowed)
```

- Variables declared with ‘const’ cannot be reassigned

```
numNoChg = 14; (this would NOT be allowed)
```

JS DATA TYPES

- Among others, data types in JS include:
    - Number (exactly what it sounds like)
    - String (sequence of characters)
    - Boolean (true/false)

JS FUNCTIONS

- Functions are reusable blocks of code
- Functions can take input(s) and return an output, or can just perform an action
- Functions allow us to avoid repeating ourselves in code, simplifying our codebase
- Example of a JS function:

```
function printName(yourName) {
    console.log(yourName);
};
```

- Invoking the above function and its output:

  Input: printName(‘mike’);

  Output: mike

JS FUNCTIONS

- What is “abstraction”?
  - Basically the hiding of complexity within code modules
  - A very important concept in programming

- We will use functions to abstract away a lot of our logic
  - Allows us to simplify our code
  - Allows us to focus on the structure of our program

JS FUNCTIONS

- “Functions” versus “methods”
  - A function is an independent code block that can return a value or accomplish an action
  - A method is just like a function, except it is designed to act only on specific data structures (basically, only designed to act on specific elements within our program)

- JS has many built-in methods that we can use to accomplish common tasks. For example:

  `Array.push();`
  (adds an element to an array)

  `String.toUpperCase();`
  (makes all letters in a string uppercase letters)

- So, for now, we will define our own functions, and we will use built-in JS methods

JS MATH OBJECT

- JS has built-in math functionality, accessed from within a data structure called an “object”
    - The object contains methods that we can use to perform various math functions
    - Example syntax:

```
Math.random();
Math.floor();
```

- We will use both of the above methods within our project

DOM = DOCUMENT OBJECT MODEL

- The DOM is basically the structure of an HTML document
- The DOM gives us access to methods and properties that allow us to use JS to make changes to HTML elements
  - Examples of these:

```
document.getElementById();
document.querySelector();
someElement.textContent = ‘Hello’;
```

- Using JS to make changes to an HTML doc is called “DOM manipulation”

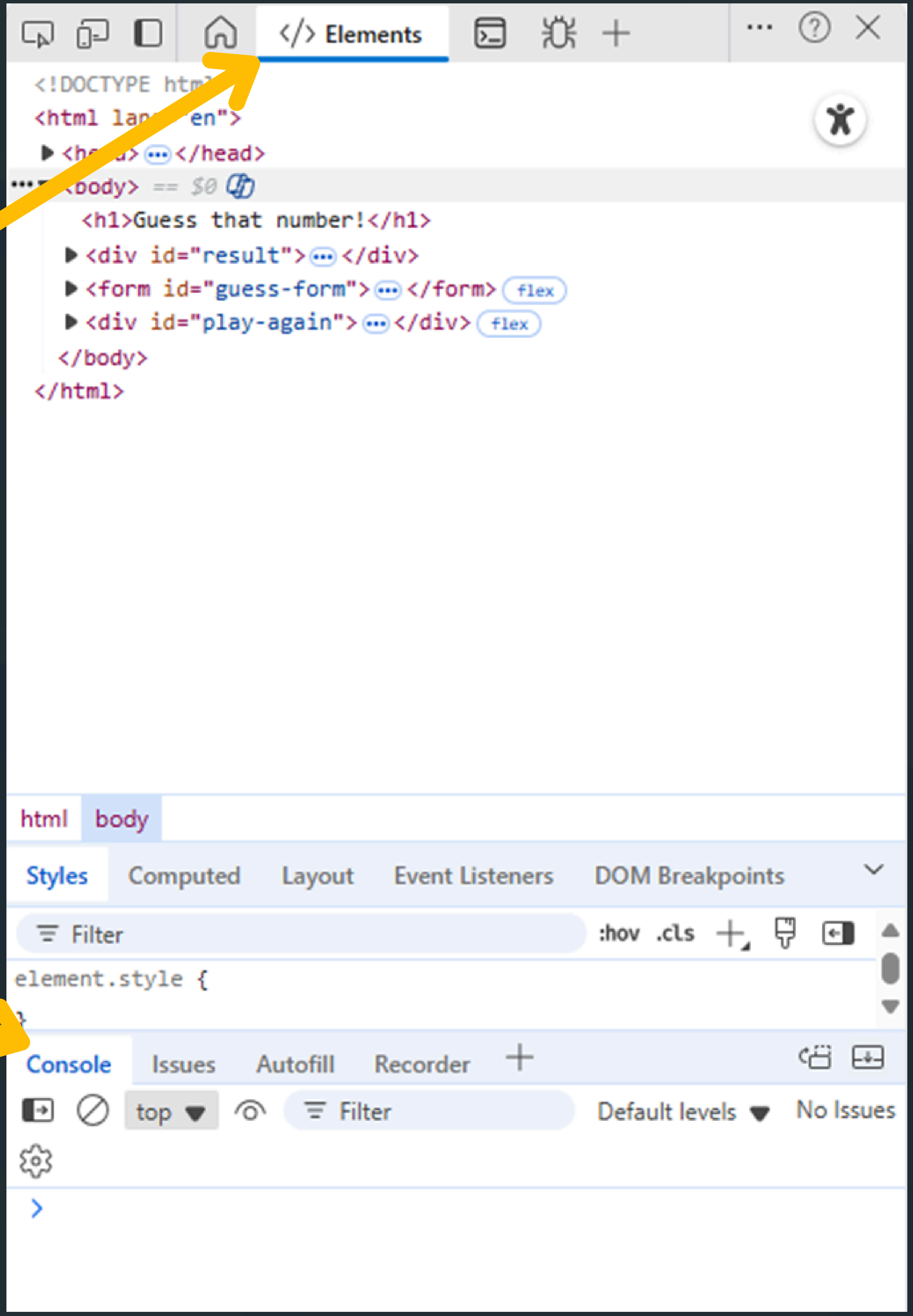
GREAT WEBSITES FOR REFERENCE!

- I use the following websites when I need to look something up or understand something better:
  - https://www.w3schools.com/js/
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
  - https://stackoverflow.com/questions

  Check them out!

DEVELOPER TOOLS

- To access your browser's built-in developer tools, open up a web page and press CTRL-SHIFT-J
- We want both the Elements tab and the Console tab to be visible
    - If you don't see the Console tab, press Esc and it should open below the Elements tab



```
<!DOCTYPE htm
<html lan   en">
▶ <h  d> ··· </head>
··· ▼ <body> == $0
    <h1>Guess that number!</h1>
  ▶ <div id="result"> ··· </div>
  ▶ <form id="guess-form"> ··· </form> flex
  ▶ <div id="play-again"> ··· </div> flex
  </body>
</html>
```

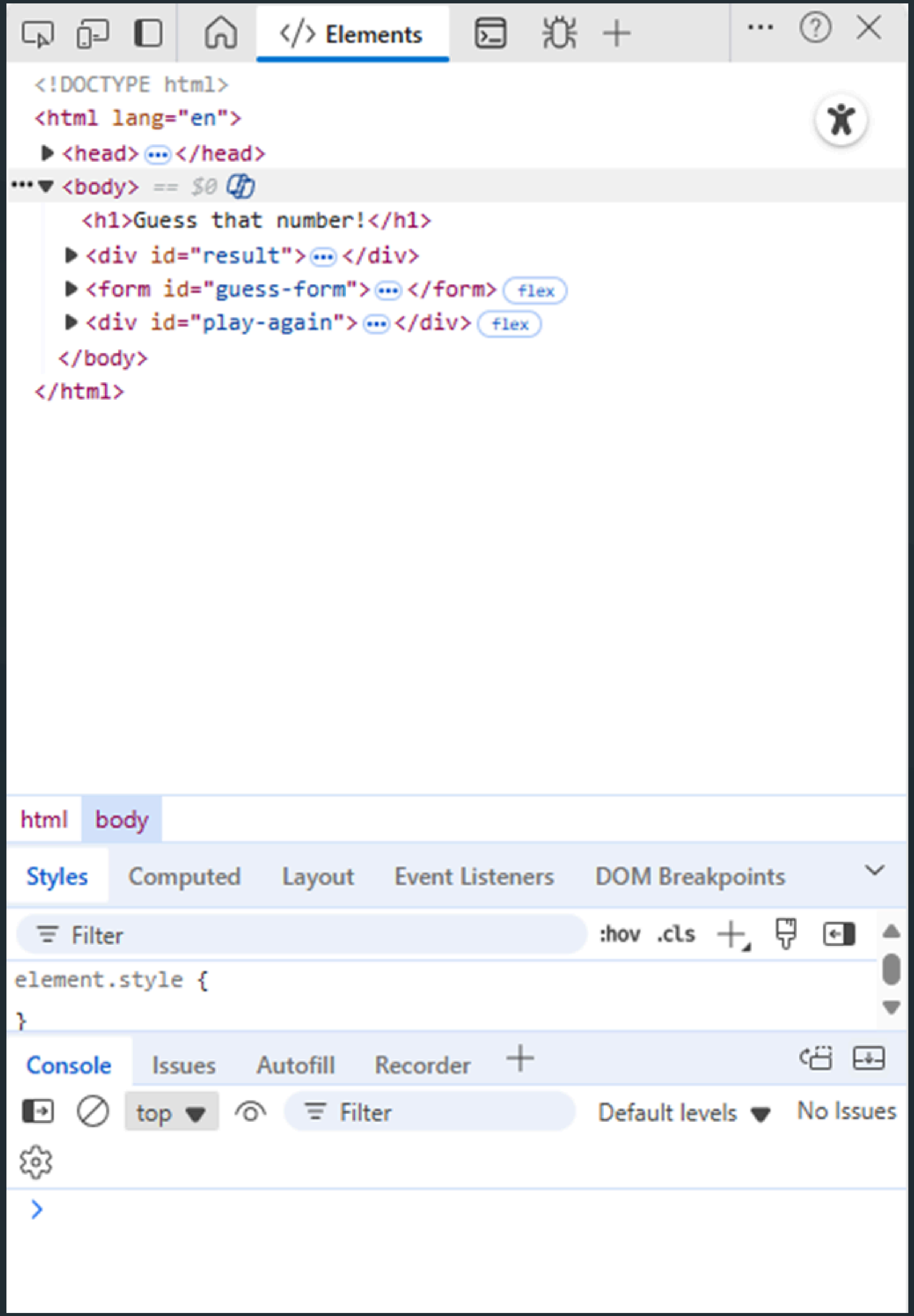
html body
Styles Computed Layout Event Listeners DOM Breakpoints
Filter :hov .cls

```
element.style {
}
```

Console Issues Autofill Recorder
top Filter Default levels No Issues

DEVELOPER TOOLS

- The Elements tab
  - Shows us the HTML structure of the web page
- The Console tab
  - Allows us to execute JS code
  - We will be writing all of our code here for now

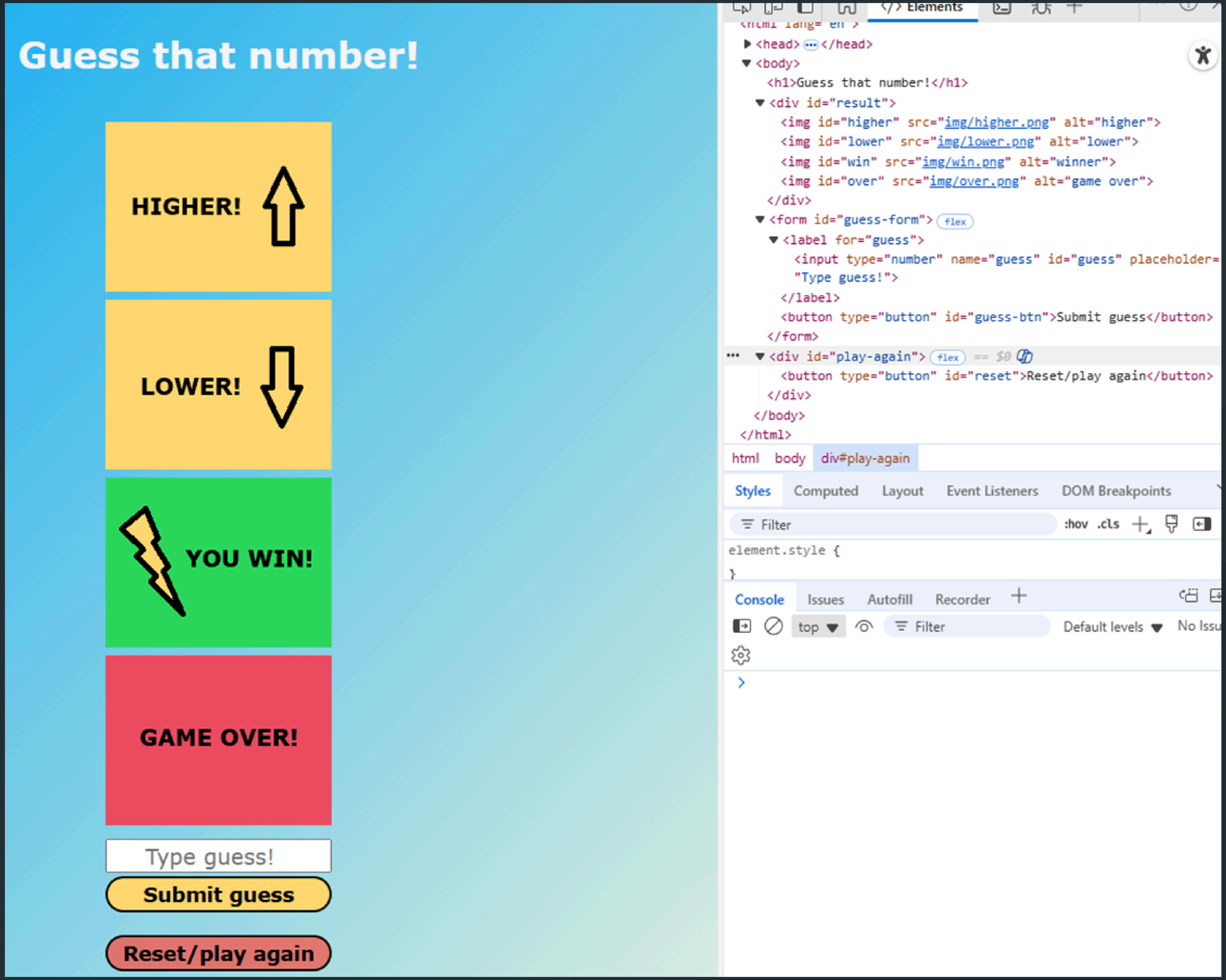


```
<!DOCTYPE html>
<html lang="en">
  <head> ... </head>
  <body> == $0
    <h1>Guess that number!</h1>
    <div id="result"> ... </div>
    <form id="guess-form"> ... </form> flex
    <div id="play-again"> ... </div> flex
  </body>
</html>
```

```
element.style {
}
```

OUR PROJECT

- This is a view of our project page
- We will add JS code to make this a functioning game!

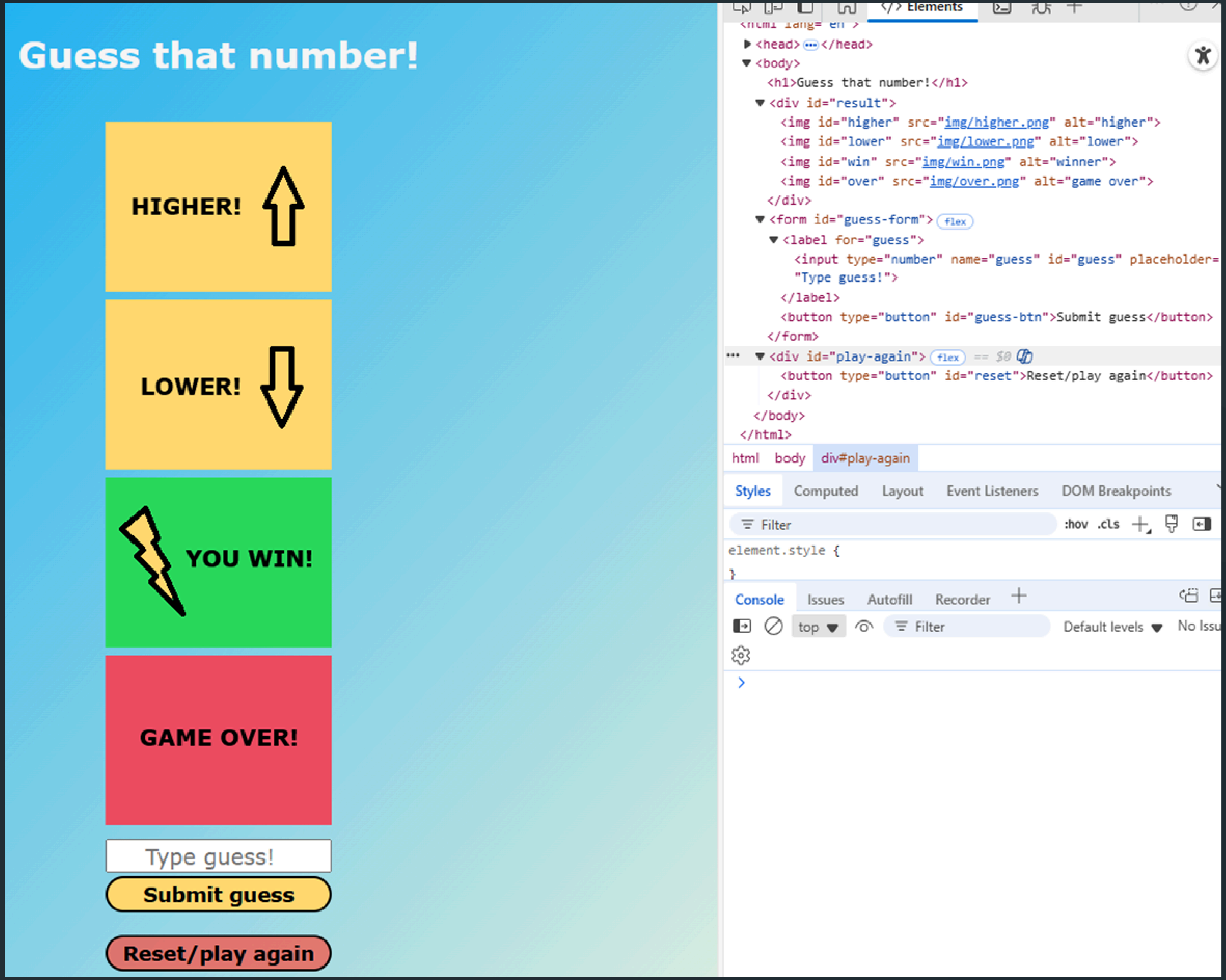


```
<html lang="en">
  <head> … </head>
  <body>
    <h1>Guess that number!</h1>
    <div id="result">
      <img id="higher" src="img/higher.png" alt="higher">
      <img id="lower" src="img/lower.png" alt="lower">
      <img id="win" src="img/win.png" alt="winner">
      <img id="over" src="img/over.png" alt="game over">
    </div>
    <form id="guess-form"> flex
      <label for="guess">
        <input type="number" name="guess" id="guess" placeholder=
        "Type guess!">
      </label>
      <button type="button" id="guess-btn">Submit guess</button>
    </form>
    <div id="play-again"> flex == $0
      <button type="button" id="reset">Reset/play again</button>
    </div>
  </body>
</html>
```

```
element.style {
}
```

ROADMAP

1. Define our variables
   - Variables to hold the DOM elements that we want to manipulate
   - Variable to hold the number that will be guessed
2. Write JS functions
   - Function to pick a random number
   - Function to handle guess submit
   - Function to hide all images
   - Function to reset game
3. Add "event listeners"
   - Use the addEventListener() method to listen for clicks on our buttons and invoke the functions



```
<html lang="en">
  <head>…</head>
  <body>
    <h1>Guess that number!</h1>
    <div id="result">
      <img id="higher" src="img/higher.png" alt="higher">
      <img id="lower" src="img/lower.png" alt="lower">
      <img id="win" src="img/win.png" alt="winner">
      <img id="over" src="img/over.png" alt="game over">
    </div>
    <form id="guess-form"> flex
      <label for="guess">
        <input type="number" name="guess" id="guess" placeholder=
        "Type guess!">
      </label>
      <button type="button" id="guess-btn">Submit guess</button>
    </form>
    <div id="play-again"> flex == $0
      <button type="button" id="reset">Reset/play again</button>
    </div>
  </body>
</html>
html  body  div#play-again
```

```
element.style {
}
```

DEFINING VARIABLES

<u>Variables to hold DOM elements</u>

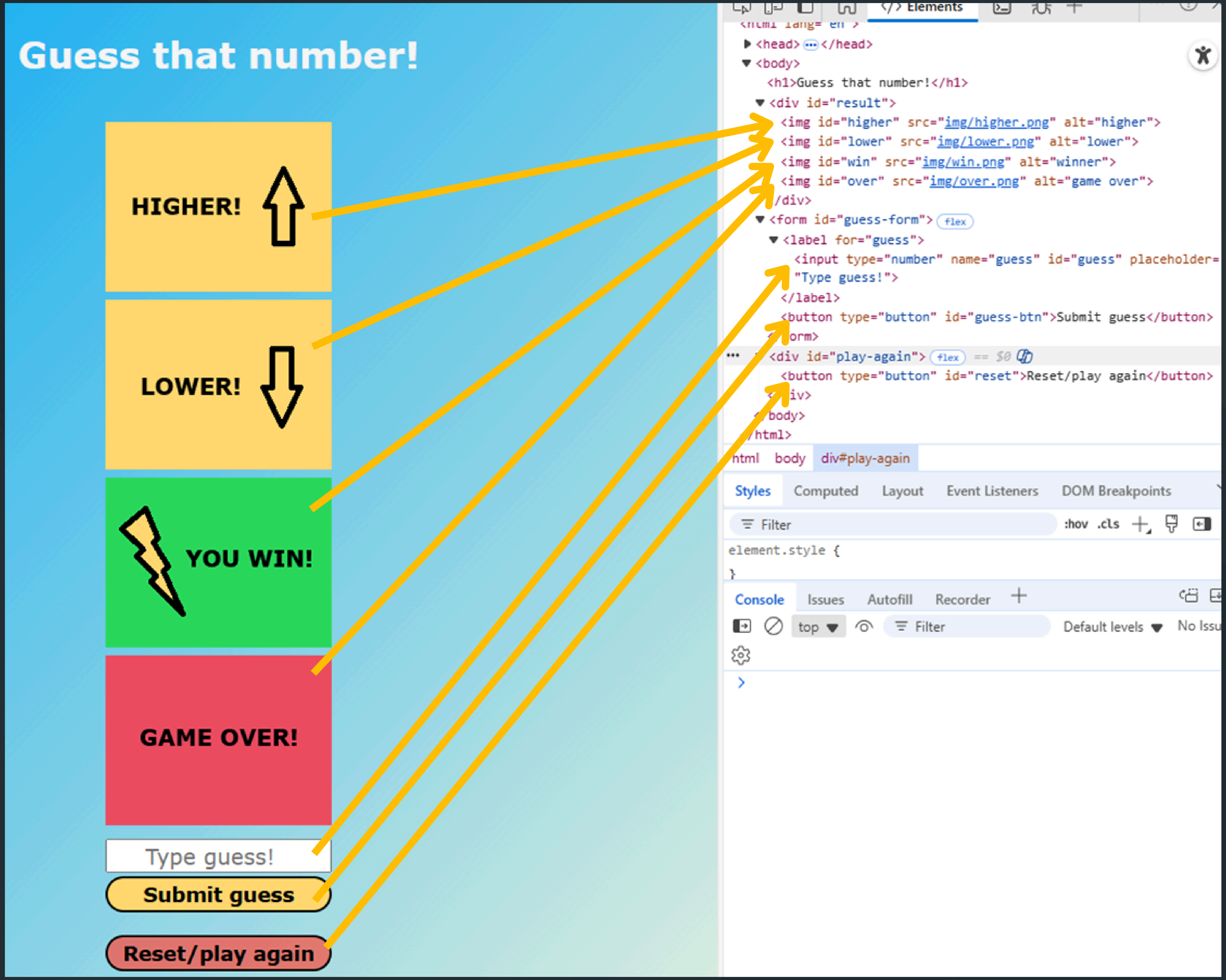
```
const guessInput = document.getElementById('guess');

const higherImg = document.getElementById('higher');
const lowerImg = document.getElementById('lower');
const winImg = document.getElementById('win');
const overImg = document.getElementById('over');

const guessButton = document.getElementById('guess-btn');
const resetButton = document.getElementById('reset');
```

(We use the id property to access each HTML element)

<u>Variable to hold num to guess</u>

```
let randNum = 0;
```

(Defined with 'let' so we can reassign it)

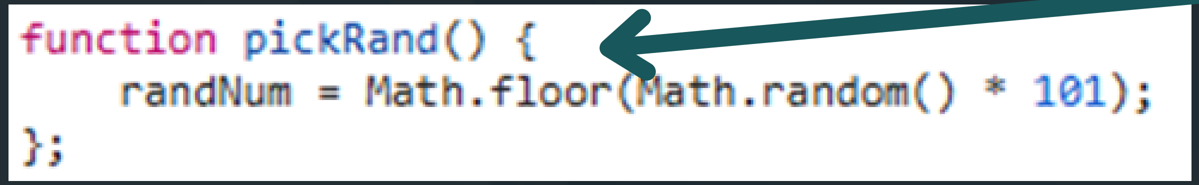


```
<html lang="en">
▶ <head> ••• </head>
▼ <body>
    <h1>Guess that number!</h1>
  ▼ <div id="result">
      <img id="higher" src="img/higher.png" alt="higher">
      <img id="lower" src="img/lower.png" alt="lower">
      <img id="win" src="img/win.png" alt="winner">
      <img id="over" src="img/over.png" alt="game over">
    /div>
  ▼ <form id="guess-form"> flex
    ▼ <label for="guess">
        <input type="number" name="guess" id="guess" placeholder=
        "Type guess!">
      </label>
      <button type="button" id="guess-btn">Submit guess</button>
    orm>
••• <div id="play-again"> flex == $0
      <button type="button" id="reset">Reset/play again</button>
    iv>
  body>
 /html>
```

```
element.style {
}
```

DEFINING FUNCTIONS

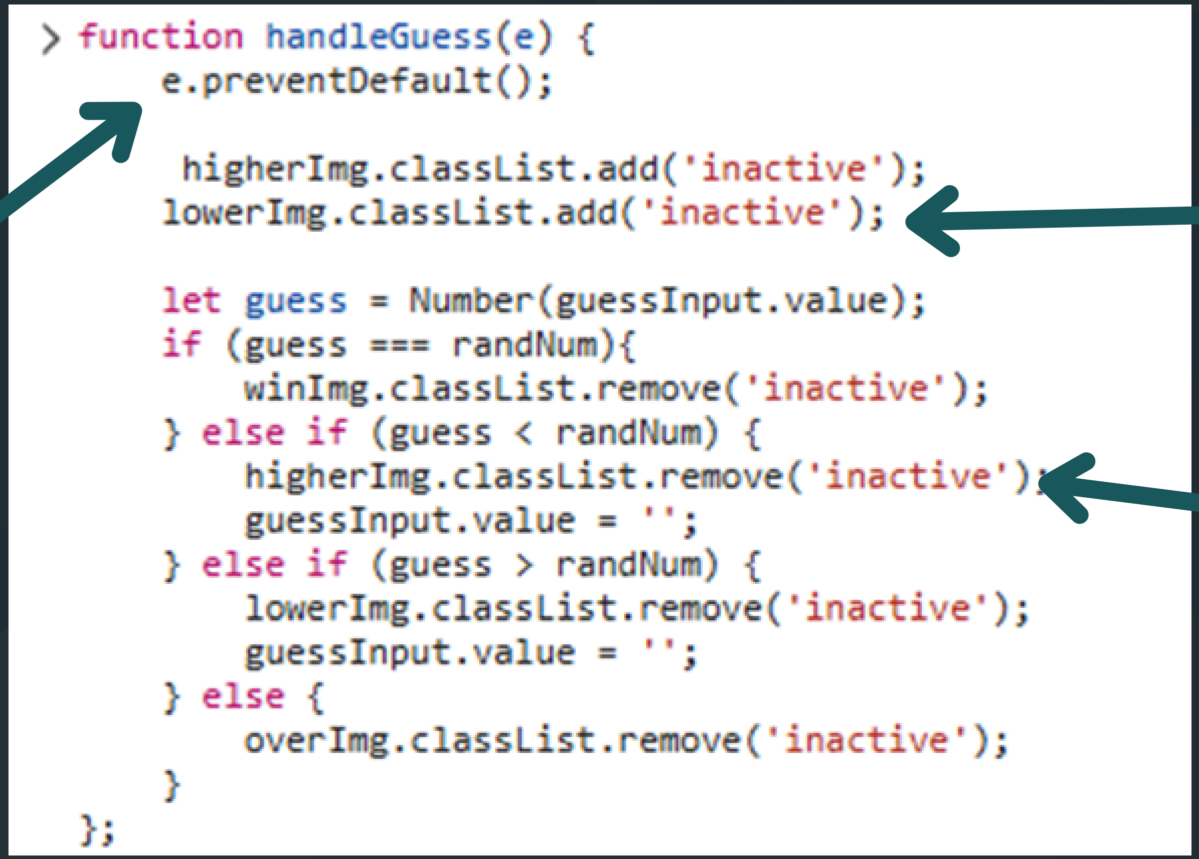
- Function to pick random number



```
function pickRand() {
    randNum = Math.floor(Math.random() * 101);
};
```

Defines our randNum variable with a random number (number to be guessed)

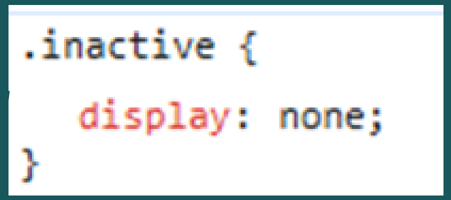
- Function to handle guess submit



```
> function handleGuess(e) {
      e.preventDefault();

       higherImg.classList.add('inactive');
      lowerImg.classList.add('inactive');

      let guess = Number(guessInput.value);
      if (guess === randNum){
          winImg.classList.remove('inactive');
      } else if (guess < randNum) {
          higherImg.classList.remove('inactive');
          guessInput.value = '';
      } else if (guess > randNum) {
          lowerImg.classList.remove('inactive');
          guessInput.value = '';
      } else {
          overImg.classList.remove('inactive');
      }
  };
```

Built-in JS method. Prevents default behavior (in this case, form submission behavior)

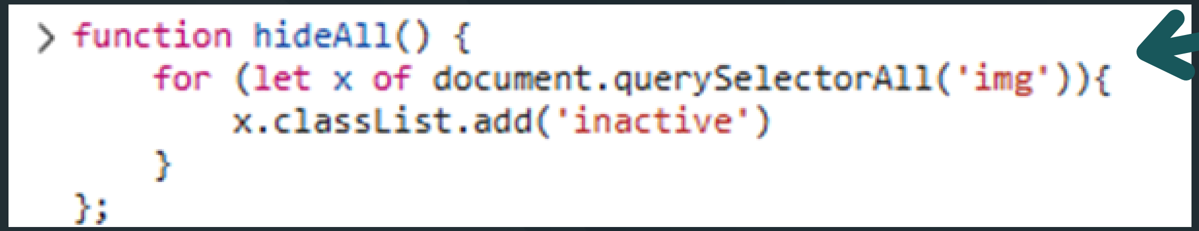
We are using an HTML class to add and remove the CSS display property here



```
.inactive {
   display: none;
}
```

DEFINING FUNCTIONS

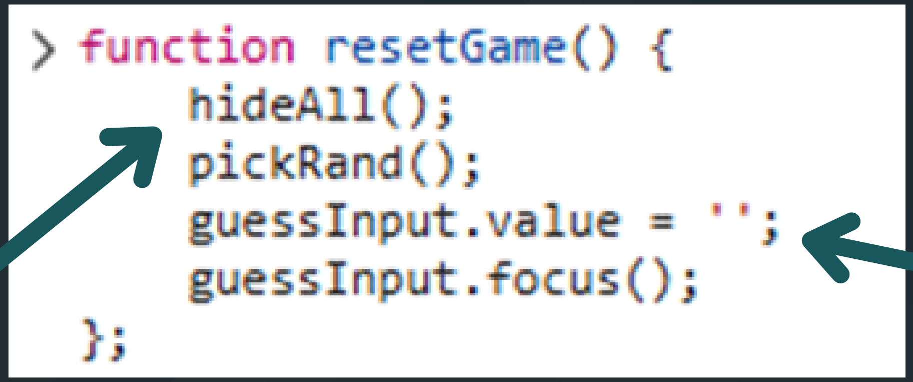
- Function to hide all images



```
> function hideAll() {
      for (let x of document.querySelectorAll('img')){
          x.classList.add('inactive')
      }
  };
```

querySelectorAll() allows us to select all the images in our HTML doc. We use what's called a 'for loop' here to look at each img element and add our 'inactive' class to each (hiding each element)

- Function to reset game



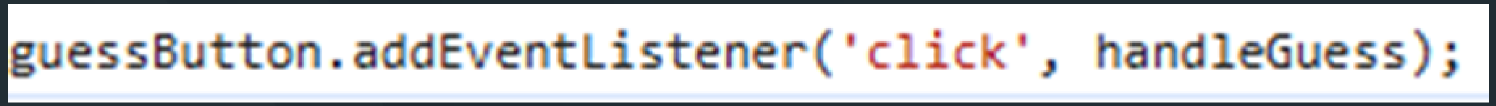
```
> function resetGame() {
      hideAll();
      pickRand();
      guessInput.value = '';
      guessInput.focus();
  };
```

Nested functions within our resetGame() function

Manipulating property of guessInput DOM element

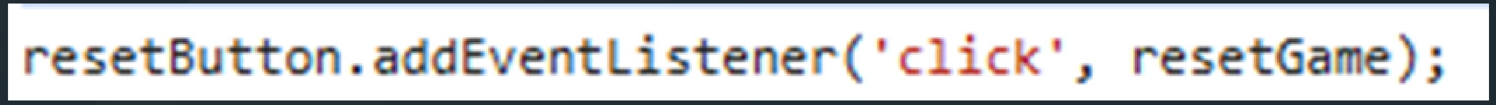
ADD EVENT LISTENERS

- Event listener for click on “guess” button



```
guessButton.addEventListener('click', handleGuess);
```

- Event listener for click on “reset” button



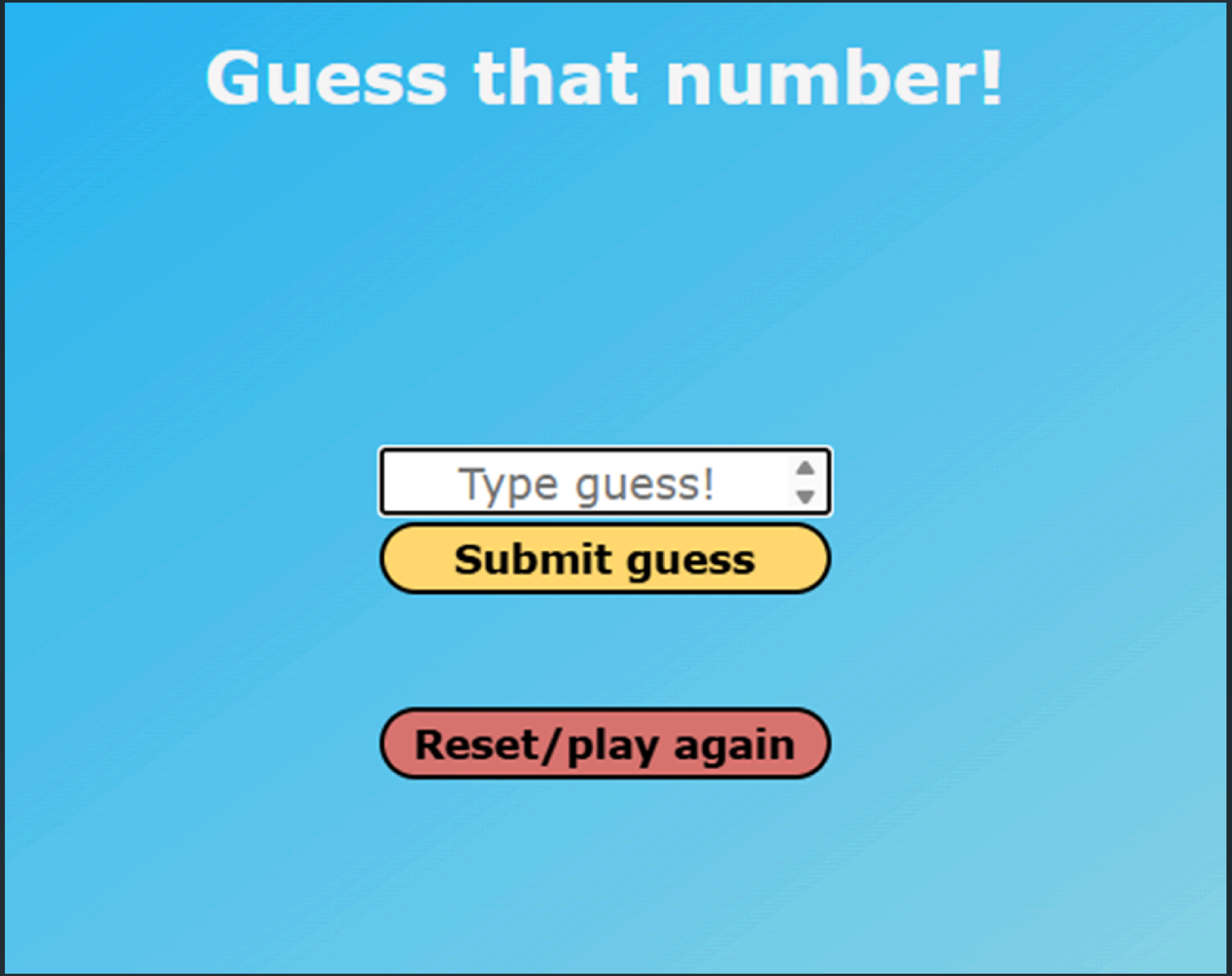
```
resetButton.addEventListener('click', resetGame);
```

COMPLETED!

Now we can play our game!

IMPORTANT NOTE:
Refreshing the browser will remove all the code that we just wrote.

DO NOT refresh the browser!



THAT'S IT! YOU'VE BUILT A JS PROJECT!

- See links on this page for:
  - This presentation in PDF form
  - All the code from this project
- Reach out to me with any questions or feedback on the presentation!
  - My email address: mike@mikeschober.dev

MIKESCHOBER.DEV