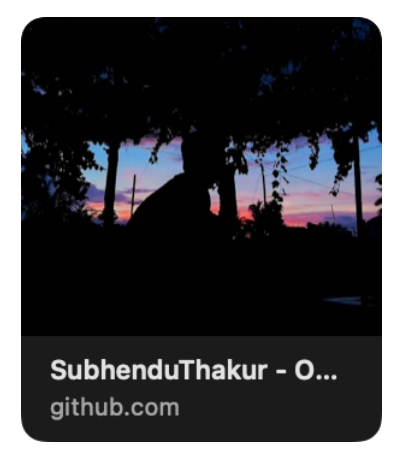
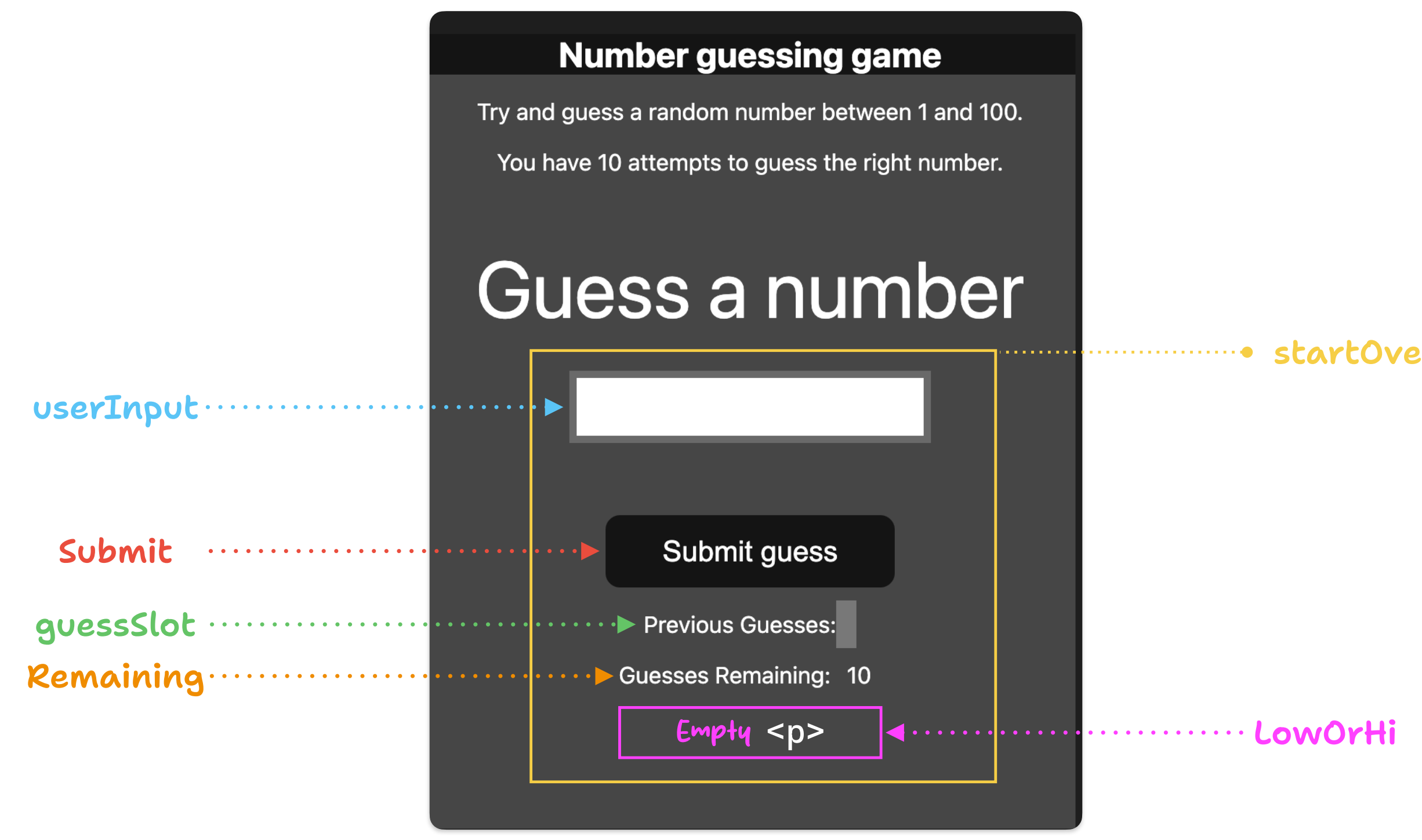
Chai aur Code : Project 4

EDITED BY





Number guessing game

Try and guess a random number between 1 and 100.

You have 10 attempts to guess the right number.

Guess a number

Submit guess

Previous Guesses:

Guesses Remaining: 10

- **submit** → Submit button
- **userInput** → Input field where the user types their guess
- **guessSlot** → Displays previous guesses
- **remaining** → Shows the number of remaining attempts
- **lowOrHi** → Displays feedback (Too High/Too Low)
- **startOver** → Stores game results and restart button

1️⃣ Generating a Random Number

Generates a number between 1-100

Multiplying by 100 scales it to a number between 0 and 99.999.

Adding 1 makes it between 1 and 100.

```
let randomNumber = parseInt(Math.random() * 100 + 1);
```

2️⃣ Selecting Elements from the HTML

```
const submit = document.querySelector('#subt');
const userInput = document.querySelector('#guessField');
const guessSlot = document.querySelector('.guesses');
const remaining = document.querySelector('.lastResult');
const lowOrHi = document.querySelector('.lowOrHi');
const startOver = document.querySelector('.resultParas');
```

```
<form class="form">
<label2 for="guessField" id="guess">Guess a number</label>
<input type="text" id="guessField" class="guessField">
<input type="submit" id="subt" value="Submit guess" class="guessSubmit">
</form>

<div class="resultParas">
<p>Previous Guesses: <span class="guesses"></span></p>
<p>Guesses Remaining: <span class="lastResult">10</span></p>
<p class="lowOrHi"></p>
</div>
```

3️⃣ Setting Up Initial Game Variables

✅ Purpose: Initializes game variables.

```
const p = document.createElement('p'); p creates a new <p> element (used later for the "Start New Game" button).
let prevGuess = []; prevGuess stores all the user's guesses.
let numGuess = 1; numGuess keeps track of how many guesses have been made.
let playGame = true; playGame is a boolean flag to track if the game is running.
```

4️⃣ Handling the Submit Button Click

Submit guess

✅ Purpose: Gets the user's guess and sends it for validation.

1. Event Listener waits for the user to click the Submit Guess button.
2. e.preventDefault(); prevents form submission (so the page doesn't reload).

69 (value inside userInput)

```
if (playGame) {
  submit.addEventListener('click', function (e) {
    e.preventDefault();
    const guess = parseInt(userInput.value); 3. parseInt(userInput.value) converts the user's input into an integer (number), and stores into a variable "guess".
    console.log(guess);
    validateGuess(guess); 4. The number is passed to validateGuess() for further checks.
  });
}
```

5️⃣ Validating the Guess

✅ Purpose: Validates the guess and decides what happens next.

The guess parameter takes the value passed from validateGuess(guess)
- If the user enters 25, guess = 25.
- If the user enters 80, guess = 80.

```
function validateGuess(guess) {
  if (isNaN(guess)) { 1. isNaN(guess) : If guess is not a number, show an alert.
    alert('Please enter a valid number');
  } else if (guess < 1) { 2. guess < 1 : If guess is too small, show an alert.
    alert('Please enter a number more than 1');
  } else if (guess > 100) { 3. guess > 100 : If guess is too big, show an alert.
    alert('Please enter a  number less than 100');
  } else {
    prevGuess.push(guess); 4. If guess is valid, it's added to prevGuess and processed.
    if (numGuess === 11) { 5. If the user has guessed 10 times, the game ends. How will be the game end?
      displayGuess(guess); 6. We will call displayGuess(guess); to update the UI. (Previous Guesses and Guesses Remaining)
      displayMessage(`Game Over. Random number was ${randomNumber}`); 7. Calls displayMessage() to show the game-over message.
      endGame(); 8. Calls endGame(); to stop the game.
    } else { If attempts remains
      displayGuess(guess); 9. else keep updating the "Previous Guesses" and "Guesses Remaining".
      checkGuess(guess); 10. Calls checkGuess(guess); to compare the guess with randomNumber.
    }
  }
}
```

6️⃣ Checking if the Guess is Correct

✅ Purpose: Compares the guess and provides feedback.

```
function checkGuess(guess) {
  if (guess === randomNumber) { 1. If guess is equal to randomNumber
    displayMessage(`You guessed it right`); 2. Then displayMessage 'You guessed it right'
    endGame(); 3. After that end the game
  } else if (guess < randomNumber) { 4. else if guess number is less than randomNumber
    displayMessage(`Number is TOOO low`); 5. displayMessage Number is TOOO low
  } else if (guess > randomNumber) { 6. else if guess number is greater than randomNumber
    displayMessage(`Number is TOOO High`); 7. displayMessage Number is TOOO High
  }
}
```

7️⃣ Displaying the Guess and Remaining Attempts

✅ Purpose: Updates the "Previous Guess" and "Guesses Remaining"

```
function displayGuess(guess) {
  userInput.value = ''; 1. Clears the input field (userInput.value = '';).
  guessSlot.innerHTML += `${guess}, `; 2. Adds the guess to the Previous Guesses section.
  numGuess++; 3. Increments numGuess (number of attempts).
  remaining.innerHTML = `${11 - numGuess} `; 4. Updates the remaining attempts count.
}
```

```
guessSlot.innerHTML = guessSlot.innerHTML + ${guess}
guessSlot.innerHTML = guessSlot.innerHTML + "69, "
guessSlot.innerHTML = "69, "
```

```
guessSlot.innerHTML = guessSlot.innerHTML + ${guess}
guessSlot.innerHTML = guessSlot.innerHTML + "96, "
guessSlot.innerHTML = "69, " + "96, "
```

```
numGuess++;
numGuess + 1;
1 + 1
// 2 + 1
// 3 + 1
```

```
11 - numGuess
11 - 1
// 11 - 2
// 11 - 3
```

8️⃣ Displaying Messages

✅ Purpose: Shows feedback messages (too high, too low, or correct guess).

```
function displayMessage(message) {
  lowOrHi.innerHTML = `<h2>${message}</h2>`; 1. Takes a message and displays it inside the lowOrHi <div>.
}
```

9️⃣ Ending the Game

✅ Purpose: Stops the game and creates a "Start New Game" button.

```
function endGame() {
  userInput.value = ''; 1. Clears the input field.
  userInput.setAttribute('disabled', ''); 2. Disables the input field so no more guesses can be made.
  p.classList.add('button'); • Adds the class button to the <p> element for styling (though it's not a real button yet), and i feel it's useless.
  p.innerHTML = `<h2 id="newGame">Start new Game</h2>`; 3. Creates a new game button (<h2 id="newGame">Start new Game</h2>).
  startOver.appendChild(p); 4. Adds this button to the startOver <div>.
  playGame = false; 5. Sets playGame = false to stop the game.
  newGame(); 6. Calls newGame(); to set up a new game.
}
```

```
<form class="form">
<label2 for="guessField" id="guess">Guess a number</label>
<input type="text" id="guessField" class="guessField">
<input type="submit" id="subt" value="Submit guess"
class="guessSubmit">
</form>

<div class="resultParas">
<p >Previous Guesses: <span class="guesses"></span></p>
<p >Guesses Remaining: <span class="lastResult">10</span></p>
<p class="lowOrHi"></p>
<h2 id="newGame">Start new Game</h2>
</div>
```

🔟 Restarting the Game

✅ Purpose: Resets everything and starts a new game.

```
function newGame() {
  const newGameButton = document.querySelector('#newGame'); 1. Selects the "Start New Game" button.
  newGameButton.addEventListener('click', function (e) { 2. Adds an event listener to detect when the button is clicked.
    randomNumber = parseInt(Math.random() * 100 + 1); 3. Generates a new random number.
    prevGuess = []; 4. Resets prevGuess
    numGuess = 1; 5. Resets numGuess (attempts count).
    guessSlot.innerHTML = ''; 6. Resets guessSlot (Previous Guesses)
    remaining.innerHTML = `${11 - numGuess} `; 7. Resets remaining (Guesses Remaining)
    userInput.removeAttribute('disabled'); 8. Removes disable attribute which makes userInput enable
    startOver.removeChild(p); 9. Removes the "Start New Game" button.

    playGame = true; 10. Sets playGame = true to start the game again.
  });
}
```