Today we would do a Simon says game with JavaScript. Basically, there would be four tiles of different colors (red, blue, green and yellow) and when the game starts, a color would be picked at random by the computer then the player has to click the exact same tile. As the levels increase, the number of tiles to be clicked increases so it is a memory game i.e. you have to remember all that you have clicked and then also click on the latest tile at which ever level you are in.

The html is very simple, we have four divs corresponding to the four tiles.

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

  <meta charset="utf-8">

  <title>Simon</title>

  <link rel="stylesheet" href="styles.css">

  <link href="https://fonts.googleapis.com/css?family=Press+Start+2P" rel="stylesheet">

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

</head>

<body>

  <h1 id="level-title">Press A Key to Start</h1>

  <div class="container">

    <div class="row">

      <div type="button" id="green" class="btn green">

      </div>

      <div type="button" id="red" class="btn red">

      </div>

    </div>

    <div class="row">

      <div type="button" id="yellow" class="btn yellow">

      </div>

      <div type="button" id="blue" class="btn blue">

      </div>

    </div>

  </div>

<script src="game.js" charset="utf-8"></script>

</body>

</html>

In our JavaScript, we would create a buttons array which would basically hold the names of the four colored tiles. We would also create arrays for the game sequence and for the user click sequence. We want the game to start when the player presses any key so we would add a keydown event listener on our document. When any key is pressed, we want to call our next sequence function which would add a tile to our sequence.

var buttons = ["green", "red", "yellow", "blue"];

var gameSequence = [];

var userClickSequence = [];

var started = false;

var level = 0;

$(document).keydown(function() {

  if (started === false) {

    nextSequence();

    started = true;

  }

})

The next sequence function makes use of <code>Math.random()</code> to generate a random number and then we multiply by three (3) and round it up so that we have a random number generated between zero (0) and three (3) which would correspond to the index of the items in the buttons array.

function nextSequence() {

  level++;

  $("#level-title").text("Level " + level);

  var nextNumber = Math.round(Math.random() \* 3);

  var nextButton = buttons[nextNumber];

  gameSequence.push(nextButton);

$("#" + nextButton).fadeIn(100).fadeOut(100).fadeIn(100);

  playSound(nextButton);

}

After we get a button, we move it into the game sequence button and then we play the sound associated with the button. Our play sound function looks like this. We make use of new Audio and then we pass in the name of the audio. <br/>

Now that we have the game sequence, the next thing is for the user to click on a tile. For this, we are going to add a click event listener for all our buttons and then when a button is clicked, we add it to the user clicked sequence and then we also play the sound and finally we call the check sequence function.

$(".btn").click(function() {

  var clickedColor = $(this).attr("id");

  userClickSequence.push(clickedColor);

  buttonAnimate(clickedColor);

  playSound(clickedColor);

  checkSequence(userClickSequence.length-1);

});

Our check sequence function is basically there to check if the game sequence matched the user sequence.

function checkSequence(lastItem) {

  if (gameSequence[lastItem] === userClickSequence[lastItem]) {

    if (gameSequence.length === userClickSequence.length) {

      setTimeout(function() {

        nextSequence();

      }, 1000);

    }

  } else {

    $("body").addClass("game-over");

    setTimeout(function() {

      $("body").removeClass("game-over");

    }, 200);

    playSound("wrong");

    $("#level-title").text("Game Over, Press any Key to Restart");

    startOver();

  }

}

We simply check to see if they last item in each arrays match. If they do and the length of the arrays also match, we then call our next sequence function again which would add a new item to the game sequence array and then we can keep on clicking and checking. <br/>

If they don’t match, we apply a game over class and them remove it and play a wrong sound and finally we call start over function which would reset the game.

function startOver() {

  level = 0;

  gameSequence = [];

  started = false;

}

After the game checks if the arrays match, we would need to clear the user array and set it back to an empty array so the next sequence function would be updated to look like so.

function nextSequence() {

  level++;

  $("#level-title").text("Level " + level);

  userClickSequence = [];

  var nextNumber = Math.round(Math.random() \* 3);

  var nextButton = buttons[nextNumber];

  gameSequence.push(nextButton);

  $("#" + nextButton).fadeIn(100).fadeOut(100).fadeIn(100);

  playSound(nextButton);

}