**Simon Says Code Walkthrough**

**HTML (index.html):**

* The HTML file provides the structure of the web page and includes the necessary links to CSS and JavaScript files.

**HTML Structure:**

* The <!DOCTYPE html> declaration defines the document type and version.
* Inside the <head> section, the character encoding is specified with <meta charset="UTF-8">.
* The page title is set using <title>Simon Says</title>.
* The CSS stylesheet is linked with <link rel="stylesheet" href="styles.css">.
* The main content of the game is placed within the <body> element.

**JavaScript and jQuery Script Tags:**

* Script tags are included at the end of the <body>:
  + <script src="game.js" defer></script>: This script tag links to the game's JavaScript file (game.js) and specifies the defer attribute to ensure that the script is executed after the HTML content is parsed.

**CSS (styles.css):**

* The CSS file defines the styles and visual layout for the game elements.

**CSS Styles:**

* The styles include formatting for various game elements, such as buttons, the game board, and the game-over message.
* Colors, fonts, margins, and padding are customized to create an appealing visual design.
* CSS classes like .btn, .game-over, and others are used to select and style specific elements in the HTML.

**JavaScript (game.js):**

* The JavaScript file contains the game logic and interactivity for the Simon Says game.

**Variable Declarations:**

* Several variables are declared at the beginning of the file, including:
* buttonColours: An array that holds the colors used in the game.
* gamePattern: An array that stores the randomly generated sequence by Simon.
* userClickedPattern: An array that records the user's input.
* started: A boolean variable to track whether the game has started.
* level: An integer to keep track of the current level.

**Event Listeners:**

* The code sets up two event listeners:
  + $(document).keypress(): Listens for a keypress to start the game when any key is pressed.
  + $(".btn").click(): Listens for clicks on game buttons to record user input and trigger actions.

**Game Initialization (nextSequence):**

* nextSequence(): A function that initializes a new game level by generating a random color and adding it to gamePattern. It also animates the sequence by fading in and out the buttons.

**User Input Handling:**

* When a user clicks on a button, its color is recorded in userClickedPattern.
* The game checks the user's input against the correct sequence using the checkAnswer(currentLevel) function.

**Checking Answers (checkAnswer):**

* checkAnswer(currentLevel): A function that compares the user's input to Simon's sequence. If the input is correct, the game proceeds to the next level; otherwise, it triggers a game over scenario.

**Game Over Handling:**

* When a game over occurs, the code plays a "wrong" sound, adds a "game-over" class to the body for a red overlay, and displays a game over message with an option to restart.

**Additional Functions:**

* playSound(name): Plays the sound associated with a color or "wrong."
* animatePress(currentColor): Adds a visual effect when a button is clicked.
* startOver(): Resets the game variables to start a new game.

**Customization and Expansion:**

* The code allows for customization of colors, fonts, and sounds, making it possible to personalize the game's appearance and experience.