Simon Says Game

BIG PICTURE (Before Code)

This game works in **4 simple ideas**:

1️⃣ Computer shows a color pattern
2️⃣ User repeats the pattern by clicking
3️⃣ We compare **step-by-step**
4️⃣ Wrong click → **Game Over + Score**

HTML EXPLANATION (Structure of the Game)

```
<h1> Simon Says Game</h1>
<h2>press any key to start the game</h2>
```

- `<h1>` → Game title
- `<h2>` → Game messages (level, game over, score)

```
<div class="btn-container">
```

- This is the **main box** holding all buttons

```
<div class="btn red">1</div>
```

- Each `.btn` is a **game button**
- `red / yellow / green / purple` classes decide **color**
- Numbers are only for display, game logic uses **colors**

CSS EXPLANATION (How the Game Looks)

Button shape

```
.btn{
    height: 200px;
    width: 200px;
    border-radius: 20%;
}
```

- Makes square buttons with rounded corners (Simon style)

Color classes

```
.red { background-color: #d95980; }
```

- Each button gets its own color

Flash effects

```
.flash { background-color: white; }
.userflash { background-color: black; }
```

- **White flash** → Game shows sequence
- **Black flash** → User clicks button

This helps user visually understand:

“Computer did this” vs “I clicked this”

JAVASCRIPT EXPLANATION (GAME BRAIN)

Variables (Memory of the Game)

```
let gameseq = [];
let userseq = [];
```

- `gameseq` → Computer's pattern
- `userseq` → What user clicks

```
let btns = ["red", "yellow", "green", "purple"];
```

- All possible button colors

```
let started = false;
let level = 0;
```

- `started` → Prevents restarting mid-game
- `level` → Score + difficulty

```
let h2 = document.querySelector("h2");
```

- Used to update game messages dynamically

Game Start Logic

```
document.addEventListener("keypress", function () {
```

- Game starts when **any key is pressed**

```
if (!started) {
    started = true;
```

- Prevents restarting while game is running

```
level = 0;
gameseq = [];
```

- Fresh start → old data cleared

```
levelup();
```

- Starts Level 1

Flash Functions (Visual Feedback)

Game flash

```
function gameFlash(btn) {
```

- Used when **computer shows pattern**

User flash

```
function userFlash(btn) {
```

- Used when **user clicks button**

Separate functions = **clean & readable code**

Level Up Logic (MOST IMPORTANT)

```
function levelup() {
```

This is the **heart of the game**

```
userseq = [];
```

- User must start fresh every level

```
level++;
h2.innerText = `Level ${level}`;
```

- Level increases
- Message updated

```
let randIndex = Math.floor(Math.random() * 4);
```

- Random number between `0-3`

```
let randColor = btns[randIndex];
```

- Picks random color

```
gameseq.push(randColor);
```

- Saves computer's pattern

```
gameFlash(randButton);
```

- Shows pattern visually

Button Click Handling

```
function btnPress() {
```

Runs **every time user clicks a button**

```
if (!started) return;
```

- Clicking before game starts does nothing

```
userFlash(btn);
```

- Visual feedback for user click

```
userseq.push(userColor);
```

- Saves user's move

```
checkAnswer(userseq.length - 1);
```

- Checks **only the latest move**
  Efficient & smart

Matching Logic (Brain Comparison)

```
function checkAnswer(idx) {
```

- Compares **user click** with **game pattern**

```
if (userseq[idx] === gameseq[idx])
```

- Correct click → continue

```
if (userseq.length === gameseq.length)
```

- Full sequence matched → next level

```
else {
    gameOver();
}
```

- One mistake → Game Over

Game Over + Score

```
h2.innerHTML = `❌ Game Over! <br> Score: ${level}`;
```

- Shows score = **highest level reached**

```
resetGame();
```

- Game resets, waits for key press

Reset Game

```
started = false;
gameseq = [];
userseq = [];
level = 0;
```

- Everything cleared → clean restart

Event Listeners

```
btn.addEventListener("click", btnPress);
```

- Makes buttons interactive