🎮 Web-Based Game Programming Mini Workshop

🧭 Session 1: Drawing and Movement Basics

🧱 Design Overview

- Teach students how to set up and draw on an HTML5 `<canvas>`.
- Introduce keyboard-controlled movement of a simple object (a rectangle).

🧠 Core Functions

Canvas Setup

```
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');
```

Game Loop

```
function gameLoop() {
  update();
  draw();
  requestAnimationFrame(gameLoop);
}
```

Movement Handling

```javascript
document.addEventListener('keydown', (e) => keys[e.key] = true);
document.addEventListener('keyup', (e) => keys[e.key] = false);
```

⚠ Things to Notice

- Use `requestAnimationFrame()` for smooth, efficient animation.
- Always clear the canvas each frame with `ctx.clearRect( … )`.
- Organize logic clearly into update and draw phases.

🎯 Session 2: Collision Detection and Interactivity

🎮 Design Overview

- Add interactivity with a falling object that the player can catch.
- Score increases with each successful catch.

🧠 Core Functions

Collision Detection (AABB)

```
function isColliding(a, b) {
  return (
    a.x < b.x + b.width &&
    a.x + a.width > b.x &&
    a.y < b.y + b.height &&
    a.y + a.height > b.y
  );
}
```

Score Handling

```
let score = 0;
// On collision:
score++;
```

Reset Falling Object

```
falling.x = Math.random() * (canvas.width - falling.width);
```

⚠ Things to Notice

- Call collision logic inside `update()` every frame.
- Ensure canvas boundaries are respected using proper math.
- Draw score text after everything else using `ctx.fillText(...)`.

🚀 Session 3: Mini Game - Dodge the Blocks

🧱 Game Design

- Player dodges falling blocks.
- The game ends upon collision.
- Score increases over time.

🧠 Core Functions

Spawning Blocks

```
function spawnBlock() {
  blocks.push({ x: Math.random() * 360, y: 0, width: 30, height: 30, speed: 3 })
}
setInterval(spawnBlock, 1000);
```

Game State and Collision

```
let gameOver = false;

if (isColliding(player, block)) gameOver = true;
```

Game Over Message

```
if (gameOver) ctx.fillText('Game Over!', 160, 200);
```

Memory Management

```
blocks = blocks.filter(block ⇒ block.y < canvas.height);
```

⚠ Things to Notice

- Skip game loop updates if `gameOver` is `true`.
- Prevent memory bloat by removing off-screen objects.
- Maintain modularity with update/draw structure.

🛠️ Extensions

- Add restart button or key listener.
- Use images and sounds.
- Increase difficulty with time (e.g., block speed).