Exercise 6

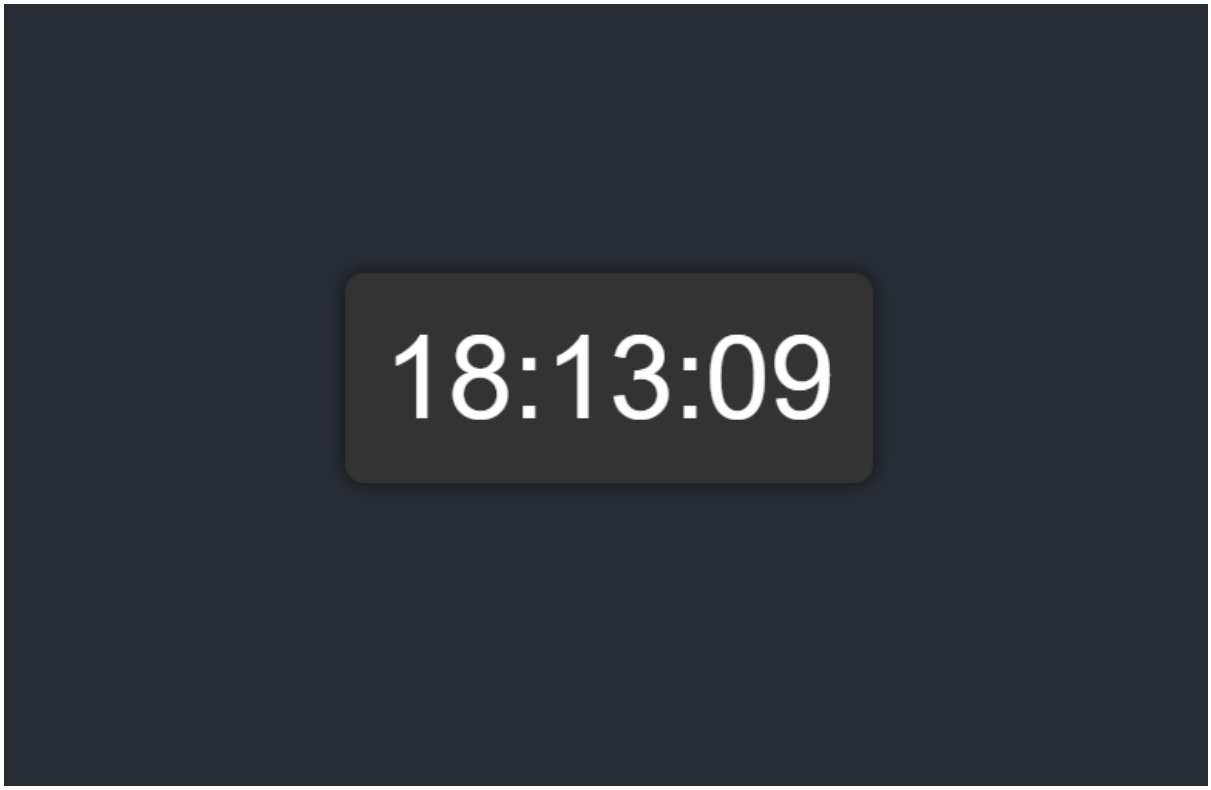
1) Digital Clock

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>EX6</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            background-color: #282c34;
            color: white;
            font-family: Arial, sans-serif;
        }

        .clock {
            font-size: 60px;
            padding: 20px;
            background-color: #333;
            border-radius: 10px;
```

```
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
        }
    </style>
</head>
<body>
    <div class="clock" id="clock"></div>

    <script>
        function updateClock() {
            const now = new Date();
            const hours = now.getHours().toString().padStart(2, '0');
            const minutes = now.getMinutes().toString().padStart(2, '0');
            const seconds = now.getSeconds().toString().padStart(2, '0');

            const timeString = `${hours}:${minutes}:${seconds}`;
            document.getElementById('clock').textContent = timeString;
        }

        setInterval(updateClock, 1000);
        updateClock();
    </script>
</body>
</html>
```

Output:



2) Analog Clock

Code:

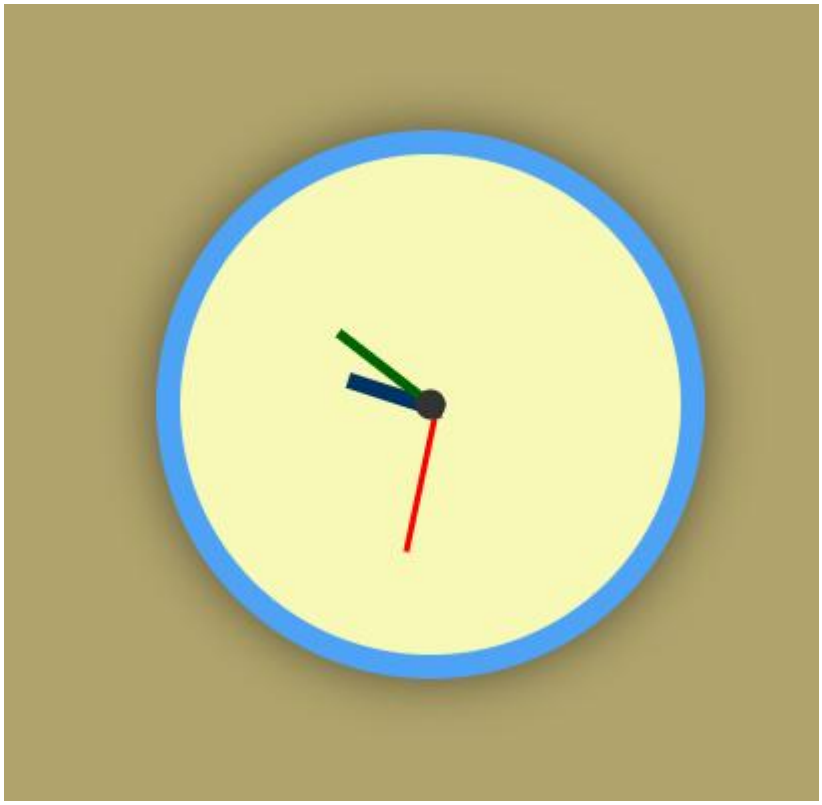
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Analog Clock</title>
    <style>
        body {
            background-color: #b0a36b;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
```

```
}
.clock {
    width: 200px;
    height: 200px;
    border: 10px solid #4fa3f7;
    border-radius: 50%;
    position: relative;
    margin: 50px auto;
    background: #f9f9b7;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
}

.hand {
    position: absolute;
    bottom: 50%;
    left: 50%;
    transform-origin: 100% 100%;
    transition: transform 0.5s ease-in-out;
}

.hour {
    width: 6px;
    height: 40px;
    background: #003366;
    transform: rotate(90deg);
}



.minute {
```

```
            width: 4px;
            height: 50px;
            background: #006400;
            transform: rotate(90deg);
        }


        .second {
            width: 2px;
            height: 60px;
            background: red;
            transform: rotate(90deg);
        }

        .center {
            width: 12px;
            height: 12px;
            background: #333;
            position: absolute;
            top: 50%;
            left: 50%;
            border-radius: 50%;
            transform: translate(-50%, -50%);
        }
    </style>
</head>
<body>
    <div class="clock">
        <div class="hand hour" id="hour"></div>
```

```
    <div class="hand minute" id="minute"></div>
    <div class="hand second" id="second"></div>
    <div class="center"></div>
  </div>

  <script>
    function updateClock() {
      const now = new Date();
      const seconds = now.getSeconds();
      const minutes = now.getMinutes();
      const hours = now.getHours();

      const secondDegrees = ((seconds / 60) * 360) + 90;
      const minuteDegrees = ((minutes / 60) * 360) + ((seconds / 60) * 6) + 90;
      const hourDegrees = ((hours / 12) * 360) + ((minutes / 60) * 30) + 90;

      document.getElementById('second').style.transform = `rotate(${secondDegrees}deg)`;
      document.getElementById('minute').style.transform = `rotate(${minuteDegrees}deg)`;
      document.getElementById('hour').style.transform = `rotate(${hourDegrees}deg)`;
    }

    setInterval(updateClock, 1000);
    updateClock();
  </script>
</body>
</html>
```

Output:



3) Flashlight text

Code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>EX 6</title>
    <style>
      * {
        margin: 0;
        box-sizing: border-box;
      }
      .image {
        background-color: rgb(255, 238, 180);
```

```
    height: 100vh;
    position: relative;
    display: flex;
    justify-content: center;
    align-items: center;
    font-family:'Courier New', Courier, monospace;
    font-size: 100px;
    font-weight: normal;
    color: #ff6f61;
   }

   .overlay {
    background-color: rgba(0, 0, 0, 0.99);
    height: 100vh;
    width: 100vw;
    position: absolute;
    top: 0;
    left: 0;
   }
  </style>
</head>
<body>
  <div class="image">
   SUPER STAR
   <div class="overlay"></div>
  </div>

  <script>
   document.addEventListener("mousemove", function (e) {
```

```
      const overlay = document.querySelector(".overlay");
      const x = e.clientX;
      const y = e.clientY;
      overlay.style.background = `radial-gradient(circle 200px at ${x}px ${y}px, rgba(0, 0, 0, 0) 0%, rgba(0, 0, 0, 0.99) 80%)`;
    });
  </script>
 </body>
</html>
```

Output:

4) Minion Eye

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>EX 6</title>
 <style>
```

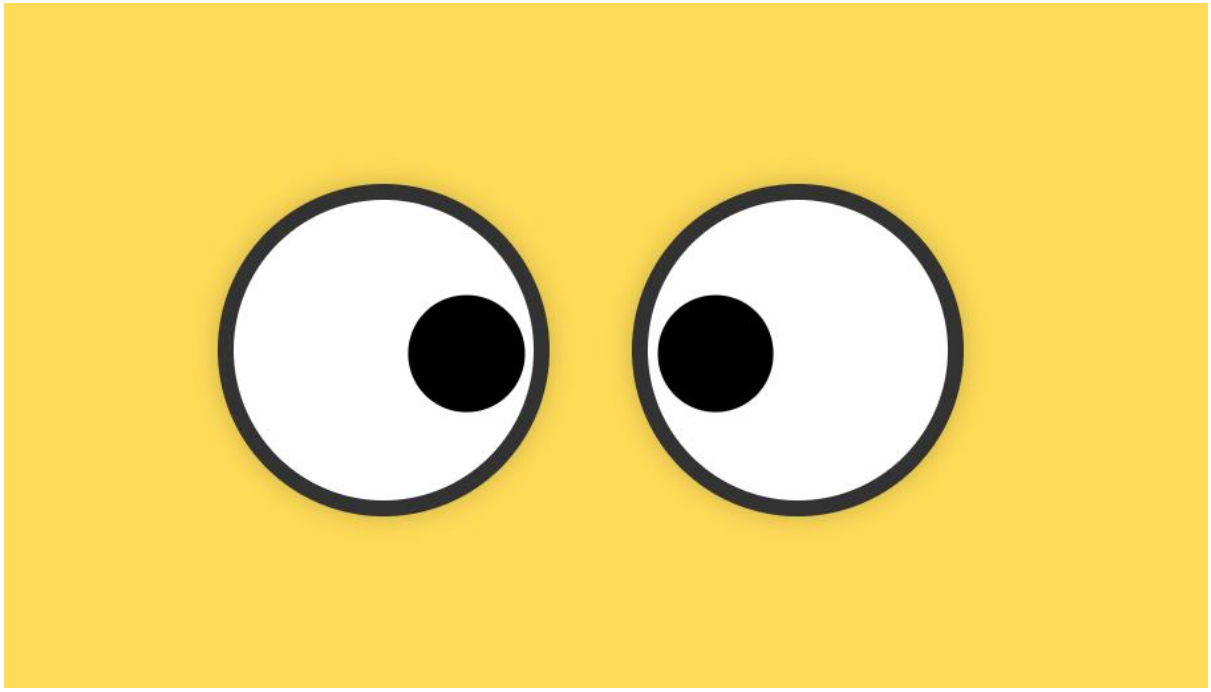
```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  background-color: #ffdb58;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.eyes-container {
  display: flex;
  justify-content: space-evenly;
  align-items: center;
  gap: 50px;
}

.eye-container {
  width: 200px;
  height: 200px;
  background-color: white;
  border-radius: 50%;
  position: relative;
  display: flex;
  justify-content: center;
```

```
    align-items: center;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);
    border: 10px solid #333;
   }

   .pupil {
    width: 70px;
    height: 70px;
    background-color: black;
    border-radius: 50%;
    position: absolute;
    transition: transform 0.1s ease-out;
   }

  </style>
</head>
<body>

  <div class="eyes-container">

   <div class="eye-container">
    <div class="pupil" id="pupil-left"></div>
   </div>

   <div class="eye-container">
    <div class="pupil" id="pupil-right"></div>
   </div>
  </div>
```

```
<script>
  const pupilLeft = document.getElementById('pupil-left');
  const pupilRight = document.getElementById('pupil-right');

  document.addEventListener('mousemove', (event) => {
    const leftEye = document.querySelectorAll('.eye-container')[0];
    const rightEye = document.querySelectorAll('.eye-container')[1];

    const leftEyeRect = leftEye.getBoundingClientRect();
    const rightEyeRect = rightEye.getBoundingClientRect();

    const leftMouseX = event.clientX;
    const leftMouseY = event.clientY;

    const leftCenterX = leftEyeRect.left + leftEyeRect.width / 2;
    const leftCenterY = leftEyeRect.top + leftEyeRect.height / 2;

    const leftAngle = Math.atan2(leftMouseY - leftCenterY, leftMouseX - leftCenterX);
    const leftDistance = Math.min(leftEyeRect.width / 4, leftEyeRect.height / 4);

    const leftPupilX = Math.cos(leftAngle) * leftDistance;
    const leftPupilY = Math.sin(leftAngle) * leftDistance;

    pupilLeft.style.transform = `translate(${leftPupilX}px, ${leftPupilY}px)`;

    const rightMouseX = event.clientX;
    const rightMouseY = event.clientY;

    const rightCenterX = rightEyeRect.left + rightEyeRect.width / 2;
```

```
    const rightCenterY = rightEyeRect.top + rightEyeRect.height / 2;

    const rightAngle = Math.atan2(rightMouseY - rightCenterY, rightMouseX - rightCenterX);
    const rightDistance = Math.min(rightEyeRect.width / 4, rightEyeRect.height / 4);

    const rightPupilX = Math.cos(rightAngle) * rightDistance;
    const rightPupilY = Math.sin(rightAngle) * rightDistance;

    pupilRight.style.transform = `translate(${rightPupilX}px, ${rightPupilY}px)`;
  });
 </script>

</body>
</html>
```

Output:



5) Vertical Image Slider

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>EX 6</title>
 <style>
  * {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
  }

  body {
   font-family: Arial, sans-serif;
   background-color: #f7f7f7;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
  }

  .slider {
   width: 80%;
   max-width: 600px;
   overflow: hidden;
```

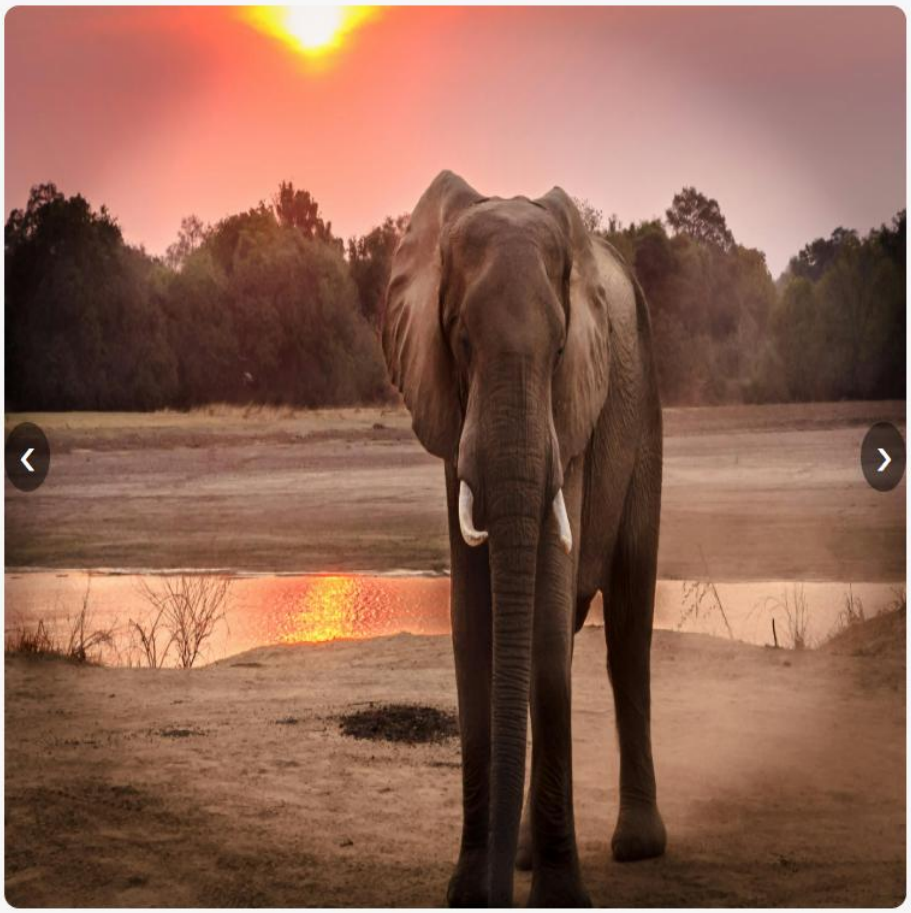
```
  border-radius: 10px;
  position: relative;
}

.slider-images {
  display: flex;
  transition: transform 0.5s ease-in-out;
}

.slider-images img {
  width: 100%;
  height: auto;
  border-radius: 10px;
}

.navigation {
  position: absolute;
  top: 50%;
  width: 100%;
  display: flex;
  justify-content: space-between;
  transform: translateY(-50%);
}

.prev, .next {
  background-color: rgba(0, 0, 0, 0.5);
  color: white;
  padding: 10px;
  border: none;
```

```
    font-size: 20px;
    cursor: pointer;
    border-radius: 50%;
  }

  .prev:hover, .next:hover {
    background-color: rgba(0, 0, 0, 0.7);
  }

 </style>
</head>
<body>

 <div class="slider">
  <div class="slider-images">
   <img src="https://images.pexels.com/photos/1054655/pexels-photo-1054655.jpeg?cs=srgb&dl=pexels-hsapir-1054655.jpg&fm=jpg" alt="Image 1">
   <img src="https://st.depositphotos.com/2001755/3622/i/450/depositphotos_36220949-stock-photo-beautiful-landscape.jpg" alt="Image 2">
   <img src="https://media.istockphoto.com/id/1419410282/photo/silent-forest-in-spring-with-beautiful-bright-sun-rays.jpg?s=612x612&w=0&k=20&c=UHeb1pGOw6ozr6utsenXHhV19vW6oiPIxDqhKCS2Llk=" alt="Image 3">
  </div>

  <div class="navigation">
   <button class="prev">&#10094;</button>
   <button class="next">&#10095;</button>
  </div>
 </div>
```

```
<script>
  let currentIndex = 0;
  const images = document.querySelectorAll('.slider-images img');
  const totalImages = images.length;

  const prevButton = document.querySelector('.prev');
  const nextButton = document.querySelector('.next');
  const sliderImages = document.querySelector('.slider-images');

  function showImage(index) {
    if (index >= totalImages) {
      currentIndex = 0;
    } else if (index < 0) {
      currentIndex = totalImages - 1;
    } else {
      currentIndex = index;
    }
    sliderImages.style.transform = `translateX(-${currentIndex * 100}%)`;
  }

  prevButton.addEventListener('click', () => {
    showImage(currentIndex - 1);
  });

  nextButton.addEventListener('click', () => {
    showImage(currentIndex + 1);
  });

  setInterval(() => {
```

```
    showImage(currentIndex + 1);
  }, 3000);
</script>

</body>
</html>
```

Output:



6) Snake

Code:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>EX 6</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
      background-color: #f1f1f1;
      font-family: 'Arial', sans-serif;
      flex-direction: column;
    }

    .game-container {
      text-align: center;
    }

    canvas {
      border: 3px solid #333;
      background-color: #2d2d2d;
```

```
  box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);
}

.score {
  color: #0033ff;
  font-size: 24px;
  margin-bottom: 20px;
  font-weight: bold;
}

.game-over {
  color: #fff;
  font-size: 36px;
  font-weight: bold;
  background-color: rgba(0, 0, 0, 0.7);
  padding: 20px;
  border-radius: 10px;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  display: none;
}

.reset-btn {
  background-color: #4CAF50;
  color: white;
  padding: 15px 25px;
  border: none;
```

```
      font-size: 20px;
      cursor: pointer;
      border-radius: 5px;
      margin-top: 20px;
      display: none;
    }

    .reset-btn:hover {
      background-color: #45a049;
    }

  </style>
</head>
<body>

  <div class="game-container">
    <div class="score">Score: 0</div>
    <canvas id="gameCanvas" width="400" height="400"></canvas>
    <div class="game-over">Game Over! Final Score: 0</div>
    <button class="reset-btn" onclick="resetGame()">Play Again</button>
  </div>

  <script>
    const canvas = document.getElementById('gameCanvas');
    const ctx = canvas.getContext('2d');

    const gridSize = 20;
    const canvasSize = 400;
```

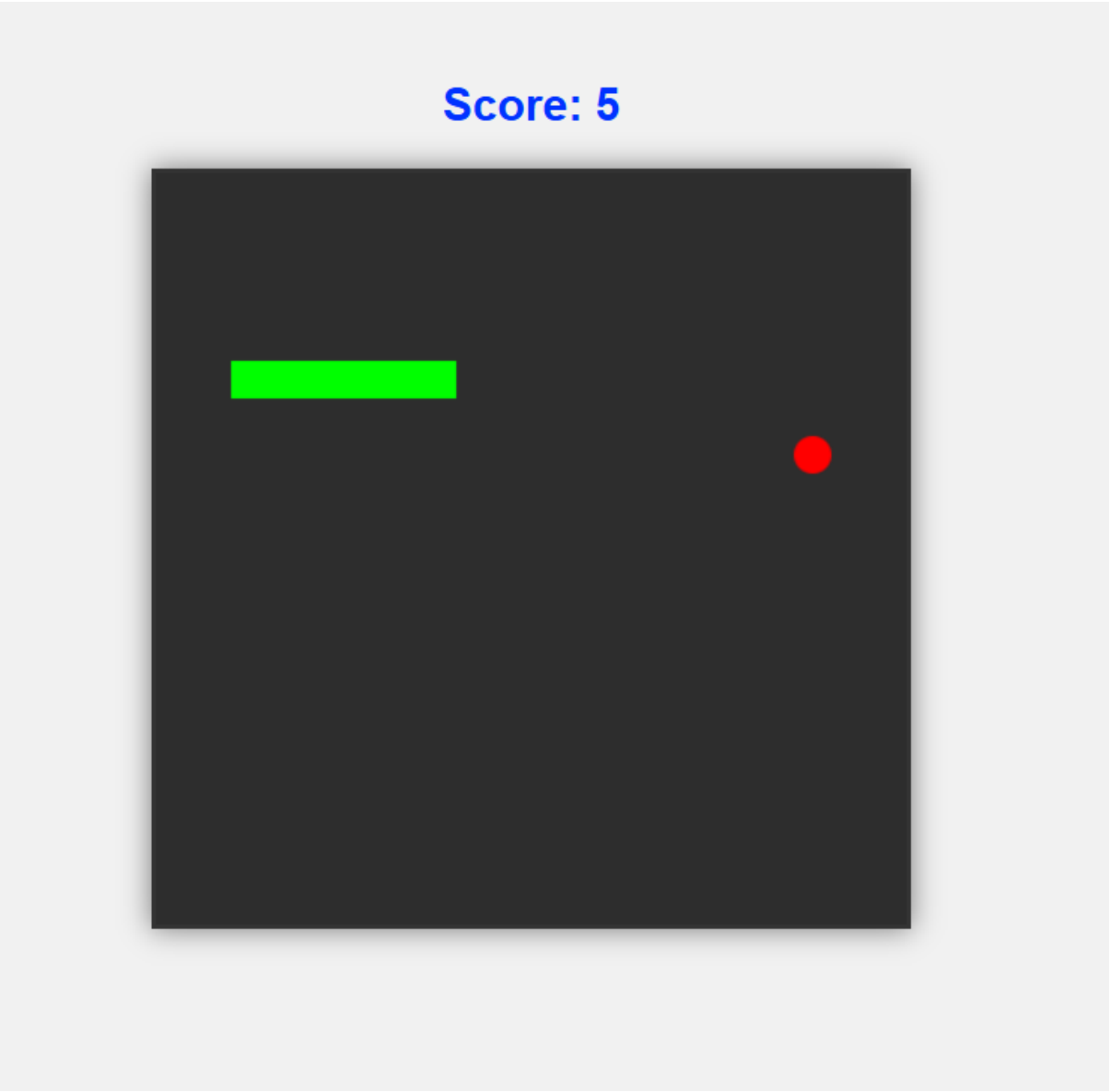
```
let snake = [{ x: 200, y: 200 }];
let food = spawnFood();
let direction = 'RIGHT';
let score = 0;
let gameInterval;

const scoreElement = document.querySelector('.score');
const gameOverElement = document.querySelector('.game-over');
const resetButton = document.querySelector('.reset-btn');

function startGame() {
  gameInterval = setInterval(() => {
    moveSnake();
    checkCollisions();
    draw();
  }, 100);
}

function moveSnake() {
  const head = { ...snake[0] };

  if (direction === 'LEFT') head.x -= gridSize;
  if (direction === 'RIGHT') head.x += gridSize;
  if (direction === 'UP') head.y -= gridSize;
  if (direction === 'DOWN') head.y += gridSize;

  snake.unshift(head);

  if (head.x === food.x && head.y === food.y) {
```

```
    score++;
    food = spawnFood();
    scoreElement.textContent = `Score: ${score}`;
  } else {
    snake.pop();
  }
}

function checkCollisions() {
  const head = snake[0];

  if (head.x < 0 || head.x >= canvasSize || head.y < 0 || head.y >= canvasSize) {
    gameOver();
  }

  for (let i = 1; i < snake.length; i++) {
    if (head.x === snake[i].x && head.y === snake[i].y) {
      gameOver();
    }
  }
}

function spawnFood() {
  const x = Math.floor(Math.random() * (canvasSize / gridSize)) * gridSize;
  const y = Math.floor(Math.random() * (canvasSize / gridSize)) * gridSize;
  return { x, y };
}

function draw() {
```

```
  ctx.clearRect(0, 0, canvasSize, canvasSize);

  ctx.fillStyle = 'lime';
  for (let i = 0; i < snake.length; i++) {
    ctx.fillRect(snake[i].x, snake[i].y, gridSize, gridSize);
  }

  ctx.fillStyle = 'red';
  ctx.beginPath();
  ctx.arc(food.x + gridSize / 2, food.y + gridSize / 2, gridSize / 2, 0, 2 * Math.PI);
  ctx.fill();
}

function gameOver() {
  clearInterval(gameInterval);
  gameOverElement.style.display = 'block';
  resetButton.style.display = 'block';
}

function resetGame() {
  snake = [{ x: 200, y: 200 }];
  direction = 'RIGHT';
  score = 0;
  scoreElement.textContent = `Score: ${score}`;
  food = spawnFood();
  gameOverElement.style.display = 'none';
  resetButton.style.display = 'none';
  startGame();
}
```

```
  document.addEventListener('keydown', (event) => {
    if (event.key === 'ArrowLeft' && direction !== 'RIGHT') {
      direction = 'LEFT';
    } else if (event.key === 'ArrowRight' && direction !== 'LEFT') {
      direction = 'RIGHT';
    } else if (event.key === 'ArrowUp' && direction !== 'DOWN') {
      direction = 'UP';
    } else if (event.key === 'ArrowDown' && direction !== 'UP') {
      direction = 'DOWN';
    }
  });

  startGame();
 </script>

</body>
</html>
```

Output:



7) Accessing Web-cam with snapshot, recording

Code:

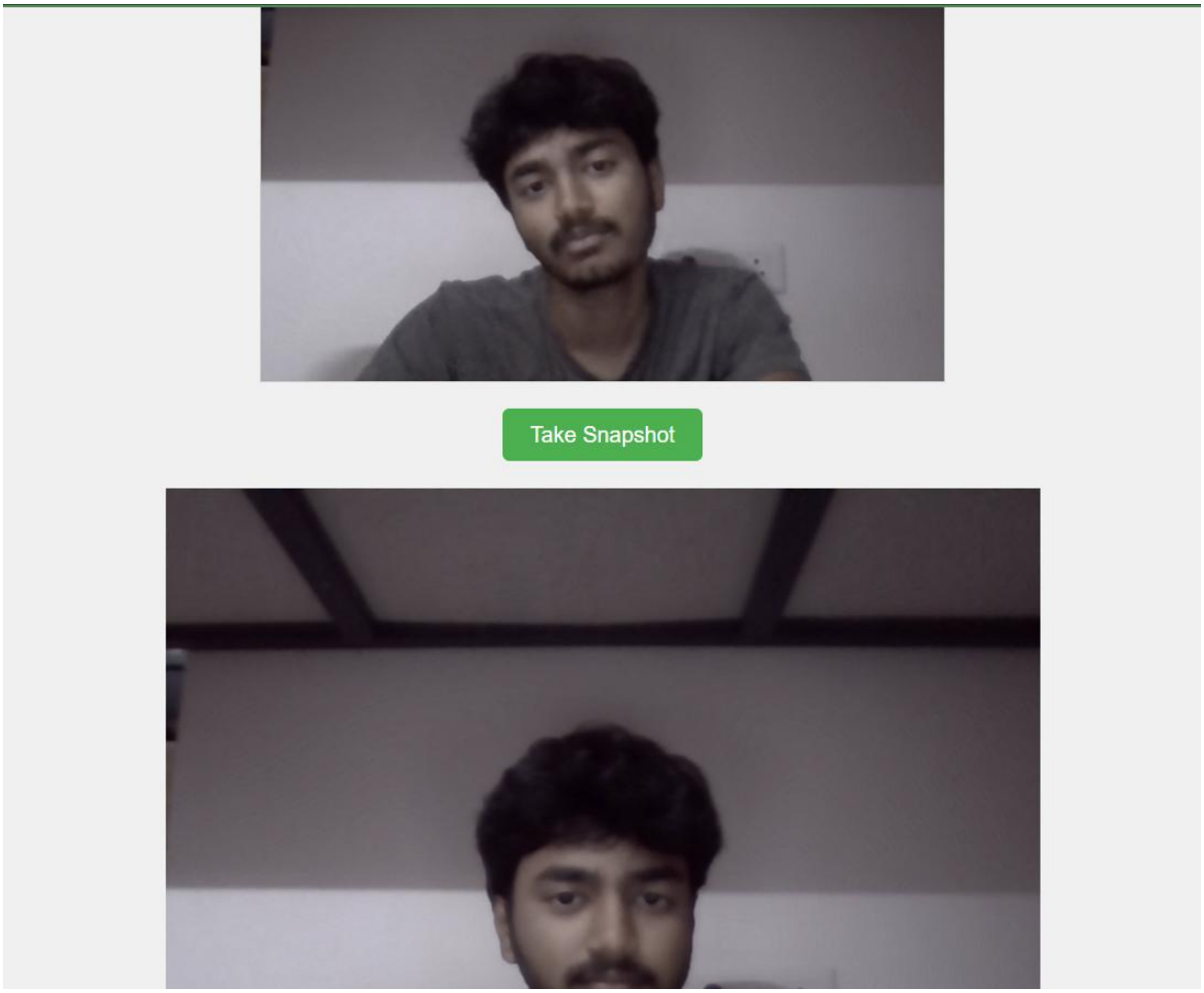
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>EX 6</title>
```

```
<style>
  body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-color: #f0f0f0;
    margin: 0;
    font-family: Arial, sans-serif;
  }
  video {
    width: 100%;
    max-width: 500px;
    border: 1px solid #ccc;
    margin-bottom: 20px;
  }
  button {
    padding: 10px 20px;
    font-size: 16px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
  }
  button:hover {
    background-color: #45a049;
  }
```

```
    canvas {
      display: none;
    }
    .snapshot {
      margin-top: 20px;
    }
  </style>
</head>
<body>

  <video id="video" autoplay></video>
  <button id="snapshotBtn">Take Snapshot</button>
  <div class="snapshot">
    <canvas id="canvas"></canvas>
    <img id="snapshot" alt="Snapshot">
  </div>

  <script>
    const video = document.getElementById('video');
    const snapshotBtn = document.getElementById('snapshotBtn');
    const canvas = document.getElementById('canvas');
    const snapshot = document.getElementById('snapshot');

    navigator.mediaDevices.getUserMedia({ video: true })
      .then((stream) => {
        video.srcObject = stream;
      })
      .catch((err) => {
        console.error("Error accessing webcam: ", err);
```

```
    });

  snapshotBtn.addEventListener('click', () => {

    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;

    const ctx = canvas.getContext('2d');
    ctx.drawImage(video, 0, 0, canvas.width, canvas.height);

    const imageUrl = canvas.toDataURL('image/png');
    snapshot.src = imageUrl;
    snapshot.style.display = 'block';
  });
 </script>

</body>
</html>
```

Output:



8) Mobile Flashlight

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>EX 6</title>
 <style>
  body {
   display: flex;
   justify-content: center;
```

```
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
    margin: 0;
    font-family: Arial, sans-serif;
   }
   button {
    padding: 15px 30px;
    font-size: 20px;
    color: #fff;
    background-color: #4CAF50;
    border: none;
    border-radius: 5px;
    cursor: pointer;
   }
   button:active {
    background-color: #45a049;
   }
  </style>
</head>
<body>

  <button id="flashlightBtn">Toggle Flashlight</button>

  <script>
   let flashlightEnabled = false;
   let stream = null;
   let videoTrack = null;
```

```
  async function toggleFlashlight() {
    try {
      if (!flashlightEnabled) {

        stream = await navigator.mediaDevices.getUserMedia({ video: { facingMode:
'environment' } });
        videoTrack = stream.getVideoTracks()[0];



        const capabilities = videoTrack.getCapabilities();
        if (capabilities.torch) {
          videoTrack.applyConstraints({ advanced: [{ torch: true }] });
          flashlightEnabled = true;
          console.log('Flashlight ON');
        }
      } else {

        if (videoTrack) {
          videoTrack.applyConstraints({ advanced: [{ torch: false }] });
          flashlightEnabled = false;
          console.log('Flashlight OFF');
        }
      }
    } catch (err) {
      console.error('Error accessing camera:', err);
      alert('Flashlight is not supported on this device or browser.');
    }
  }

  document.getElementById('flashlightBtn').addEventListener('click', toggleFlashlight);
```

```
 </script>


</body>
</html>
```

Output:

Done By: Yagav Akhilesh S R (23BRS1408)