EX-6

SHAKTHI R

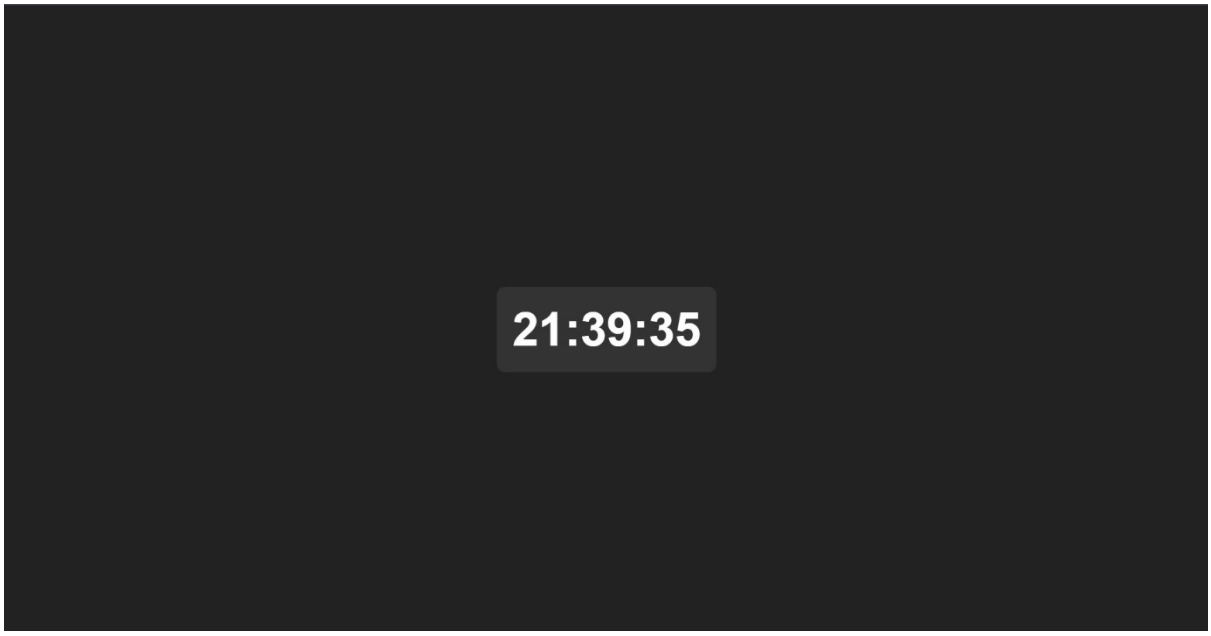
23BRS1278

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>Digital Clock</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
         background-color: #222;
         color: white;
         margin: 0;
      }
      .clock {
         font-size: 60px;
         font-weight: bold;
         background: #333;
         padding: 20px;
         border-radius: 10px;
         text-align: center;
```

```
        }
    </style>
</head>
<body>
    <div class="clock" id="clock"></div>
    <script>
        function updateClock() {
            const now = new Date();
            const hours = String(now.getHours()).padStart(2, '0');
            const minutes = String(now.getMinutes()).padStart(2, '0');
            const seconds = String(now.getSeconds()).padStart(2, '0');

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

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



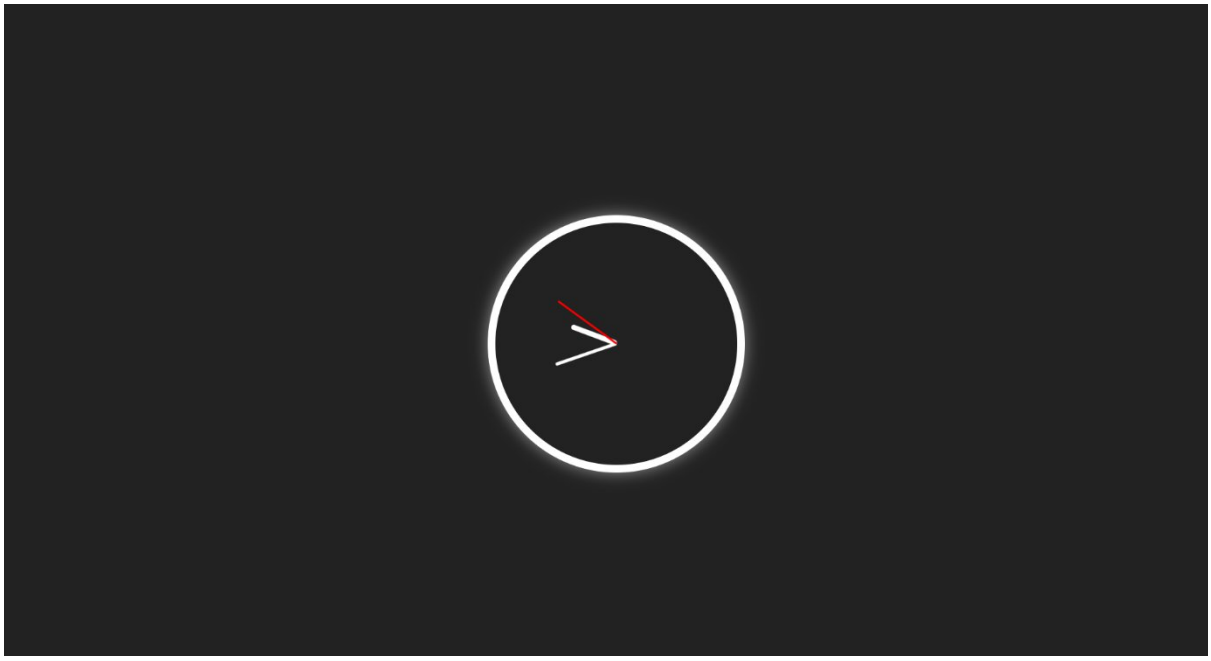
2)ANALOG CLOCK:

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 {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #222;
        }
        .clock {
            width: 300px;
            height: 300px;
            border: 10px solid white;
            border-radius: 50%;
            position: relative;
            background: url('93761e86-4af9-446e-b725-a49aa1868def.JPG') center/cover no-repeat;
            box-shadow: 0 0 20px rgba(255, 255, 255, 0.5);
        }
        .hand {
            position: absolute;
            bottom: 50%;
            left: 50%;
            transform-origin: bottom;
```

```
            transform: translateX(-50%);
            background: white;
            border-radius: 5px;
        }
        .hour {
            width: 6px;
            height: 60px;
            background: white;
        }
        .minute {
            width: 4px;
            height: 80px;
            background: white;
        }
        .second {
            width: 2px;
            height: 90px;
            background: red;
        }
    </style>
</head>
<body>
    <div class="clock">
        <div class="hand hour" id="hourHand"></div>
        <div class="hand minute" id="minuteHand"></div>
        <div class="hand second" id="secondHand"></div>
    </div>

    <script>
        function updateClock() {
            const now = new Date();
```

```
        const hours = now.getHours() % 12;
        const minutes = now.getMinutes();
        const seconds = now.getSeconds();

        const hourDeg = (hours * 30) + (minutes * 0.5);
        const minuteDeg = (minutes * 6) + (seconds * 0.1);
        const secondDeg = seconds * 6;

        document.getElementById("hourHand").style.transform = `translateX(-50%) rotate(${hourDeg}deg)`;
        document.getElementById("minuteHand").style.transform = `translateX(-50%) rotate(${minuteDeg}deg)`;
        document.getElementById("secondHand").style.transform = `translateX(-50%) rotate(${secondDeg}deg)`;
    }

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

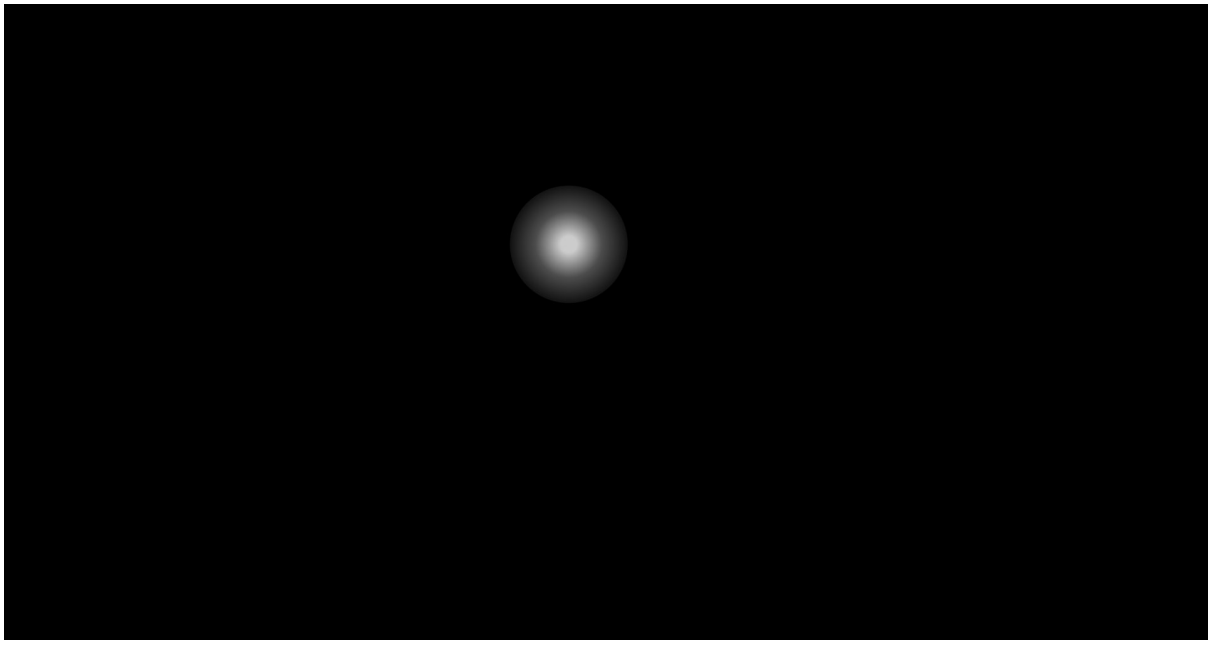


3) Flashlight text

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flashlight Effect</title>
    <style>
        body {
            margin: 0;
            width: 100vw;
            height: 100vh;
            overflow: hidden;
            background: #000;
            position: relative;
            cursor: none;
        }
        .flashlight {
            position: absolute;
            width: 150px;
            height: 150px;
            border-radius: 50%;
            background: radial-gradient(circle, rgba(255,255,255,0.8) 10%, rgba(255,255,255,0.3) 40%,
rgba(0,0,0,0.9) 80%);
            pointer-events: none;
            transform: translate(-50%, -50%);
        }
    </style>
</head>
```

```
<body>
    <div class="flashlight"></div>

    <script>
        const flashlight = document.querySelector('.flashlight');

        document.addEventListener('mousemove', (e) => {
            flashlight.style.left = `${e.clientX}px`;
            flashlight.style.top = `${e.clientY}px`;
        });

        document.addEventListener('touchmove', (e) => {
            flashlight.style.left = `${e.touches[0].clientX}px`;
            flashlight.style.top = `${e.touches[0].clientY}px`;
        });
    </script>
</body>
</html>
```



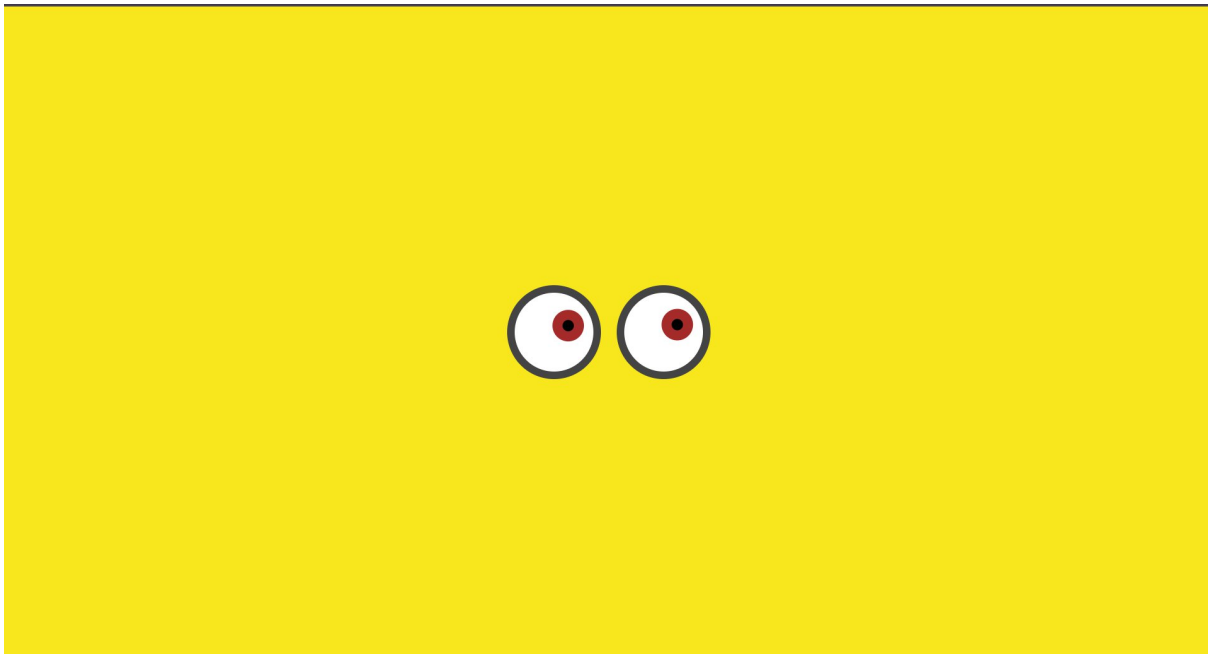
4)MINION EYE:

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Moving Eyes</title>
    <style>
        body {
            background-color: #f8e71c;
            margin: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .eyes {
            display: flex;
            gap: 20px;
        }
        .eye {
            width: 100px;
            height: 100px;
            background: white;
            border-radius: 50%;
            display: flex;
            justify-content: center;
            align-items: center;
            border: 10px solid #444;
```

```
            position: relative;
        }
        .pupil {
            width: 40px;
            height: 40px;
            background: brown;
            border-radius: 50%;
            position: absolute;
            display: flex;
            justify-content: center;
            align-items: center;
            transition: transform 0.1s ease-out;
        }
        .inner {
            width: 15px;
            height: 15px;
            background: black;
            border-radius: 50%;
        }
    </style>
</head>
<body>
    <div class="eyes">
        <div class="eye"><div class="pupil"><div class="inner"></div></div></div>
        <div class="eye"><div class="pupil"><div class="inner"></div></div></div>
    </div>
    <script>
        const eyes = document.querySelectorAll(".eye");
        const pupils = document.querySelectorAll(".pupil");

        document.addEventListener("mousemove", (event) => {
```

```
        let { clientX: mouseX, clientY: mouseY } = event;

        eyes.forEach((eye, index) => {
            let eyeRect = eye.getBoundingClientRect();
            let eyeCenterX = eyeRect.left + eyeRect.width / 2;
            let eyeCenterY = eyeRect.top + eyeRect.height / 2;

            let deltaX = mouseX - eyeCenterX;
            let deltaY = mouseY - eyeCenterY;
            let angle = Math.atan2(deltaY, deltaX);

            let maxMove = 20; // Max distance pupil can move
            let moveX = Math.cos(angle) * maxMove;
            let moveY = Math.sin(angle) * maxMove;

            pupils[index].style.transform = `translate(${moveX}px, ${moveY}px)`;
        });
    });
  </script>
</body>
</html>
```



5)Vertical Image Slider

Code:

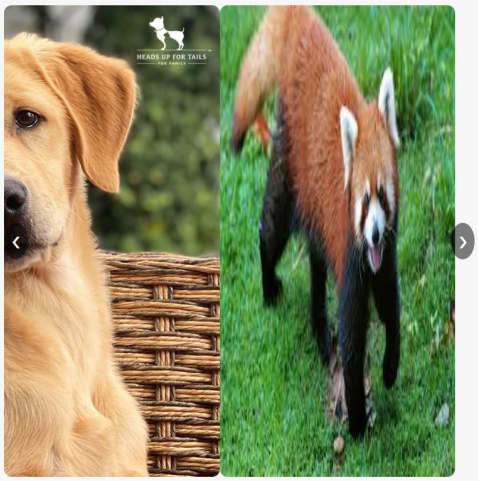
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>EX 6</title>
 <style>
  * {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
  }

  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>
```



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');
```

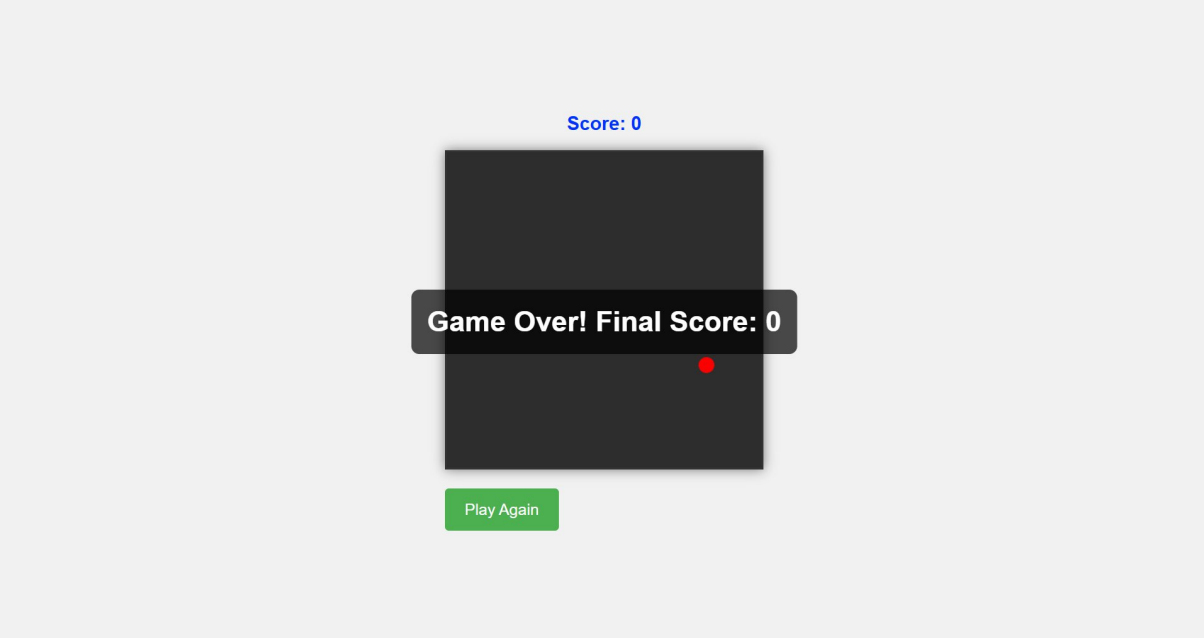
```
function startGame() {
  gameInterval = setInterval(() => {
    moveSnake();
    checkCollisions();
    draw();
  }, 100);
}

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>



7) Accessing Web-cam with snapshot, recording

Code:

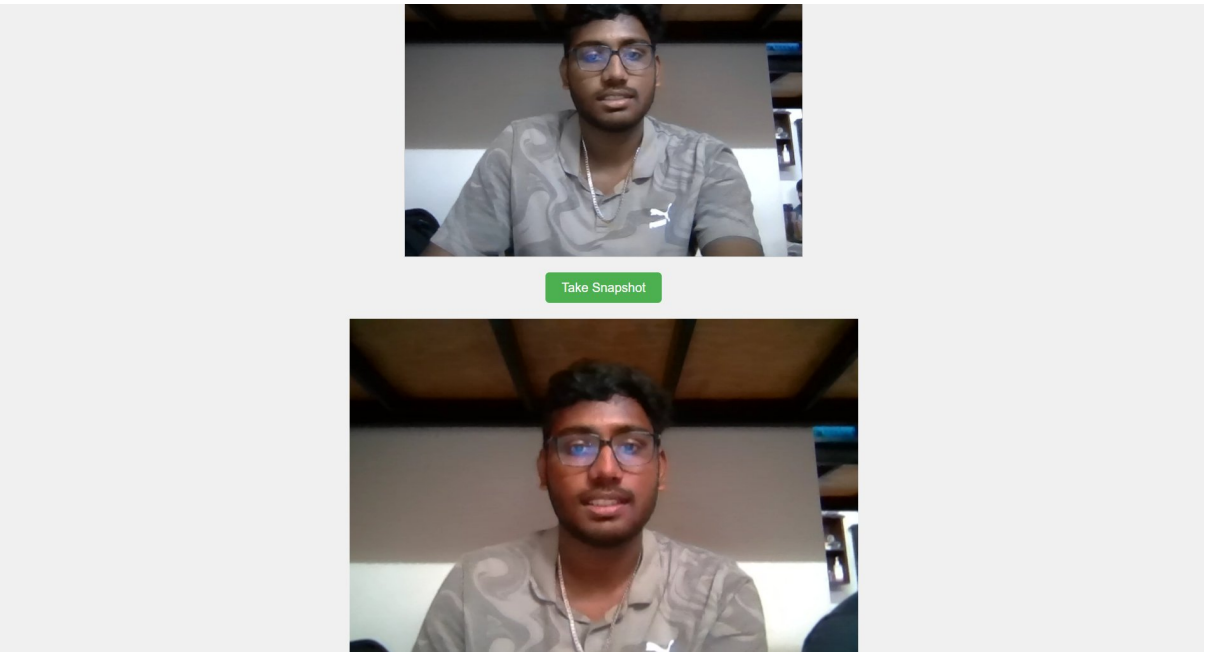
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>EX 6</title>
  <style>
    body {
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      height: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

Game Over! Final Score: 0



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>
```