WEB PROGRAMMING

EXERCISE-6

NAME: V I HEMASHREE

REG NO: 23BPS1178

FACULTY NAME: ASHOKA RAJAN

1.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>Snake Game</title>
    <style>
        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            margin: 0;
            background-color: #f0f0f0;
            font-family: Arial, sans-serif;
        }
```

```
#game-container {
    position: relative;
}
#game-board {
    display: grid;
    grid-template-columns: repeat(20, 20px);
    grid-template-rows: repeat(20, 20px);
    border: 2px solid #333;
    background-color: #fff;
}
.snake {
    background-color: #4CAF50;
    border-radius: 4px;
    margin: 1px;
}
.food {
    background-color: #FF5722;
    border-radius: 50%;
    margin: 2px;
}
#score {
    margin-top: 20px;
    font-size: 24px;
}
#controls {
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}
button {
    padding: 10px 20px;
    margin: 5px;
    font-size: 16px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
button:hover {
    background-color: #45a049;
}
#game-over {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: rgba(0, 0, 0, 0.8);
    color: white;
    padding: 20px;
    border-radius: 10px;
    text-align: center;
    display: none;
}
.mobile-controls {
    display: none;
    grid-template-areas:
        ". up ."
        "left . right"
        ". down .";
    gap: 10px;
```

```
            margin-top: 20px;
        }
        .mobile-controls button {
            width: 60px;
            height: 60px;
            font-size: 24px;
        }
        .up { grid-area: up; }
        .down { grid-area: down; }
        .left { grid-area: left; }
        .right { grid-area: right; }

        @media (max-width: 768px) {
            .mobile-controls {
                display: grid;
            }
        }
    </style>
</head>
<body>
    <h1>Snake Game</h1>
    <div id="game-container">
        <div id="game-board"></div>
        <div id="game-over">
            <h2>Game Over!</h2>
            <p>Your score: <span id="final-score">0</span></p>
            <button id="restart-button">Play Again</button>
        </div>
    </div>
    <div id="score">Score: 0</div>
    <div id="controls">
```

```
        <button id="start-button">Start Game</button>
        <button id="pause-button" disabled>Pause</button>
    </div>
    <div class="mobile-controls">
        <button class="up">↑</button>
        <button class="left">←</button>
        <button class="right">→</button>
        <button class="down">↓</button>
    </div>

    <script>
        document.addEventListener('DOMContentLoaded', () => {
            // Game variables
            const boardSize = 20;
            const gameBoard = document.getElementById('game-board');
            const scoreDisplay = document.getElementById('score');
            const finalScoreDisplay = document.getElementById('final-score');
            const gameOverScreen = document.getElementById('game-over');
            const startButton = document.getElementById('start-button');
            const pauseButton = document.getElementById('pause-button');
            const restartButton = document.getElementById('restart-button');

            let snake = [];
            let food = {};
            let direction = 'right';
            let nextDirection = 'right';
            let gameInterval;
            let score = 0;
            let speed = 200;
            let isPaused = false;
            let isGameOver = false;
```

```
// Initialize game board
function initBoard() {
    gameBoard.innerHTML = '';
    for (let i = 0; i < boardSize; i++) {
        for (let j = 0; j < boardSize; j++) {
            const cell = document.createElement('div');
            cell.setAttribute('data-x', j);
            cell.setAttribute('data-y', i);
            gameBoard.appendChild(cell);
        }
    }
}

// Initialize snake
function initSnake() {
    snake = [
        {x: 10, y: 10},
        {x: 9, y: 10},
        {x: 8, y: 10}
    ];
    renderSnake();
}

// Render snake on the board
function renderSnake() {
    // Clear previous snake
    document.querySelectorAll('.snake').forEach(element => {
        element.classList.remove('snake');
    });
```

```
        // Draw new snake
        snake.forEach(segment => {
            const cell = document.querySelector(`[data-x="${segment.x}"][data-y="${segment.y}"]`);
            if (cell) {
                cell.classList.add('snake');
            }
        });
    }

    // Create food at random position
    function createFood() {
        // Remove previous food
        const prevFood = document.querySelector('.food');
        if (prevFood) {
            prevFood.classList.remove('food');
        }

        // Generate random position that's not on the snake
        let validPosition = false;
        while (!validPosition) {
            food = {
                x: Math.floor(Math.random() * boardSize),
                y: Math.floor(Math.random() * boardSize)
            };

            validPosition = !snake.some(segment =>
                segment.x === food.x && segment.y === food.y
            );
        }

        // Place food
        const foodCell = document.querySelector(`[data-x="${food.x}"][data-y="${food.y}"]`);
        if (foodCell) {
            foodCell.classList.add('food');
        }
    }

    // Move the snake
    function moveSnake() {
        if (isPaused || isGameOver) return;

        direction = nextDirection;

        // Get current head position
        const head = {...snake[0]};

        // Calculate new head position
        switch (direction) {
            case 'up':
                head.y--;
                break;
            case 'down':
                head.y++;
                break;
            case 'left':
                head.x--;
                break;
            case 'right':
                head.x++;
                break;
        }
        // Check collision with walls
        if (head.x < 0 || head.x >= boardSize || head.y < 0 || head.y >= boardSize) {
          gameOver();
          return;
        }

        // Check collision with self
        if (snake.some(segment => segment.x === head.x && segment.y === head.y)) {
          gameOver();
          return;
        }

        // Add new head
        snake.unshift(head);

        // Check if snake eats food
        if (head.x === food.x && head.y === food.y) {
          // Increase score
          score += 10;
          scoreDisplay.textContent = `Score: ${score}`;

          // Create new food
          createFood();

          // Increase speed slightly
          if (speed > 50) {
            speed -= 5;
            clearInterval(gameInterval);
            gameInterval = setInterval(moveSnake, speed);
          }
        } else {
            // Remove tail
            snake.pop();
        }

        // Render snake
        renderSnake();
    }

    // Game over function
    function gameOver() {
        clearInterval(gameInterval);
        isGameOver = true;
        finalScoreDisplay.textContent = score;
        gameOverScreen.style.display = 'block';
        pauseButton.disabled = true;
        startButton.disabled = false;
    }

    // Handle keyboard input
    function handleKeydown(e) {
        switch (e.key) {
            case 'ArrowUp':
                if (direction !== 'down') {
                    nextDirection = 'up';
                }
                break;
            case 'ArrowDown':
                if (direction !== 'up') {
                    nextDirection = 'down';
                }
                break;
```

```
          case 'ArrowLeft':
            if (direction !== 'right') {
              nextDirection = 'left';
            }
            break;
          case 'ArrowRight':
            if (direction !== 'left') {
              nextDirection = 'right';
            }
            break;
          case ' ':
            togglePause();
            break;
        }
      }

      // Toggle pause state
      function togglePause() {
        if (isGameOver) return;

        isPaused = !isPaused;
        pauseButton.textContent = isPaused ? 'Resume' : 'Pause';

        if (!isPaused) {
          gameInterval = setInterval(moveSnake, speed);
        } else {
          clearInterval(gameInterval);
        }
      }

      // Start the game
```

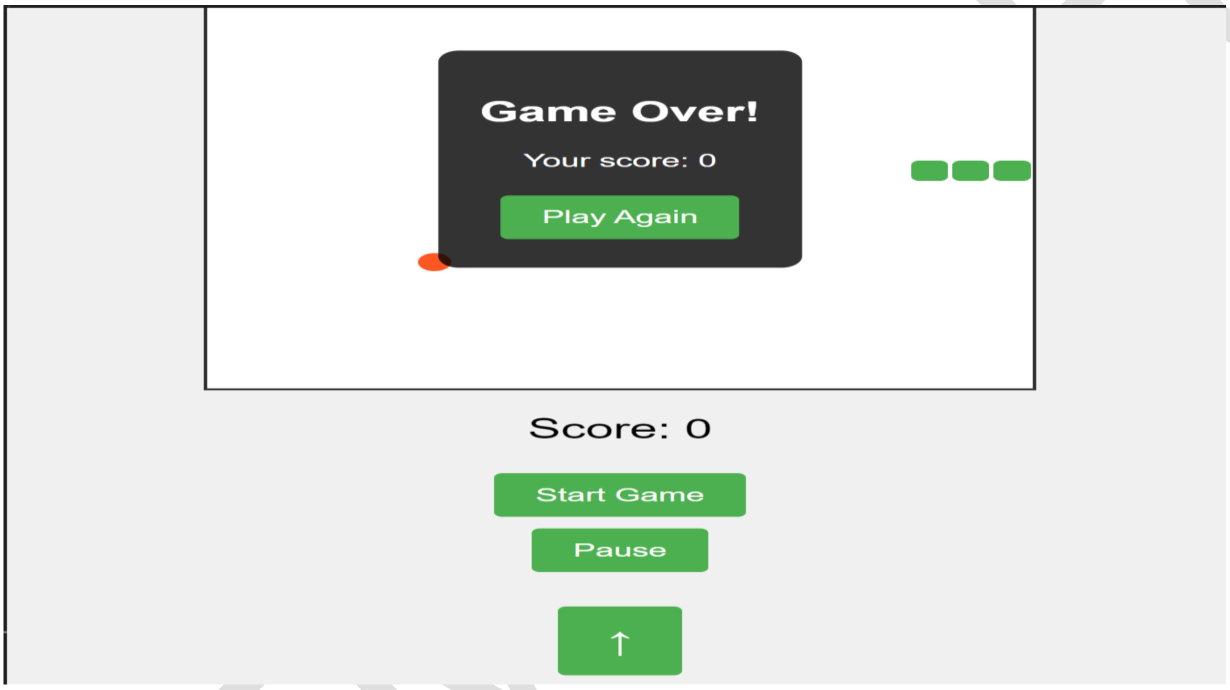
23BPS1178

```
function startGame() {
    // Reset game state
    clearInterval(gameInterval);
    score = 0;
    speed = 200;
    direction = 'right';
    nextDirection = 'right';
    isPaused = false;
    isGameOver = false;
    scoreDisplay.textContent = 'Score: 0';
    gameOverScreen.style.display = 'none';

    // Initialize game elements
    initBoard();
    initSnake();
    createFood();

    // Start game loop
    gameInterval = setInterval(moveSnake, speed);

    // Update button states
    startButton.disabled = true;
    pauseButton.disabled = false;
    pauseButton.textContent = 'Pause';
}

// Reset and restart the game
function restartGame() {
    gameOverScreen.style.display = 'none';
    startGame();
}
```

```
// Set up event listeners
startButton.addEventListener('click', startGame);
pauseButton.addEventListener('click', togglePause);
restartButton.addEventListener('click', restartGame);
document.addEventListener('keydown', handleKeydown);

// Mobile controls
document.querySelector('.up').addEventListener('click', () => {
  if (direction !== 'down') {
     nextDirection = 'up';
  }
});

document.querySelector('.down').addEventListener('click', () => {
  if (direction !== 'up') {
     nextDirection = 'down';
  }
});

document.querySelector('.left').addEventListener('click', () => {
  if (direction !== 'right') {
     nextDirection = 'left';
  }
});

document.querySelector('.right').addEventListener('click', () => {
  if (direction !== 'left') {
     nextDirection = 'right';
  }
});
```

```
            // Initialize the board
            initBoard();
        });
    </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 {
      margin: 0;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      background: linear-gradient(135deg, #ff9a9e, #fad0c4, #a1c4fd, #c2e9fb);
      overflow: hidden;
    }

    .clock {
      width: 350px;
      height: 350px;
      border-radius: 50%;
      background: radial-gradient(circle, rgba(255,255,255,0.9) 0%,
rgba(255,255,255,0.7) 70%);
      position: relative;
      box-shadow: 0 0 30px rgba(0,0,0,0.2);
      border: 12px solid white;
    }

    .center-point {
      position: absolute;
      top: 50%;
      left: 50%;
      width: 20px;
      height: 20px;
      border-radius: 50%;
      background: #333;
      transform: translate(-50%, -50%);
      z-index: 10;
    }

    .center-point::after {
      content: '';
      position: absolute;
      top: 50%;
      left: 50%;
      width: 10px;
      height: 10px;
      border-radius: 50%;
      background: #ff6b6b;
      transform: translate(-50%, -50%);
    }

    .hour-mark {
      position: absolute;
      top: 50%;
      left: 50%;
      width: 6px;
      height: 15px;
      background: #333;
      border-radius: 3px;
      transform-origin: center bottom;
    }

    .minute-mark {
      position: absolute;
      top: 50%;
      left: 50%;
      width: 2px;
      height: 8px;
      background: #777;
      border-radius: 1px;
      transform-origin: center bottom;
    }

    .hour-number {
      position: absolute;
      font-family: Arial, sans-serif;
      font-size: 24px;
      font-weight: bold;
      color: #333;
      text-align: center;
      width: 40px;
      height: 40px;
```

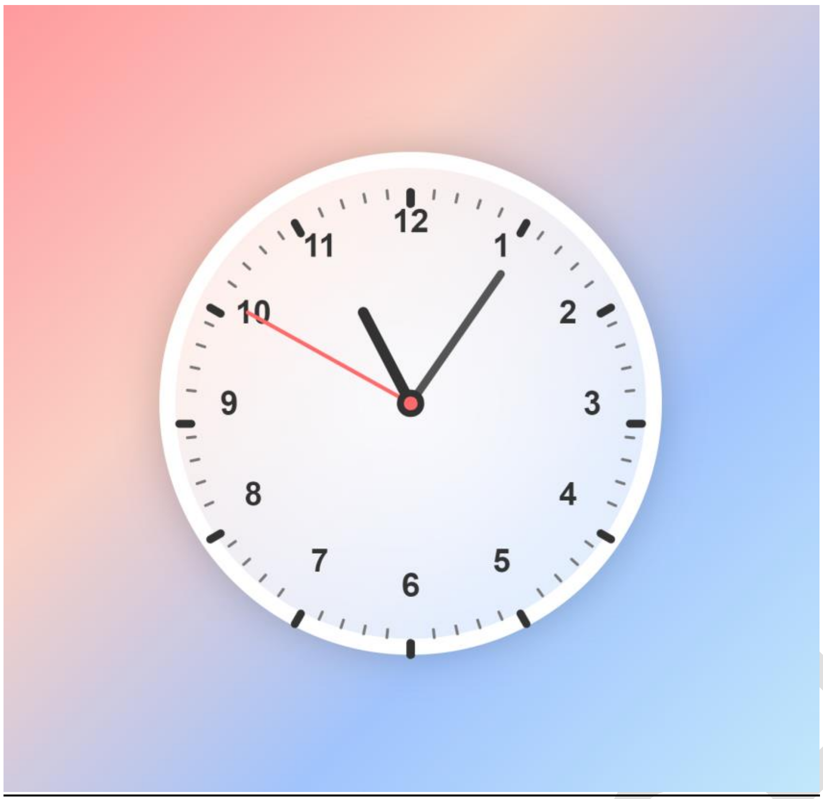
```
    line-height: 40px;
    transform: translate(-50%, -50%);
  }

  .hand {
    position: absolute;
    top: 50%;
    left: 50%;
    transform-origin: center bottom;
    border-radius: 6px 6px 3px 3px;
    z-index: 5;
  }

  .hour-hand {
    width: 8px;
    height: 80px;
    background: #333;
    transform: translate(-50%, -100%) rotate(0deg);
  }

  .minute-hand {
    width: 6px;
    height: 120px;
    background: #555;
    transform: translate(-50%, -100%) rotate(0deg);
  }
```

```
    .second-hand {
      width: 3px;
      height: 140px;
      background: #ff6b6b;
      transform: translate(-50%, -100%) rotate(0deg);
      z-index: 4;
    }
  </style>
</head>
<body>
  <div class="clock" id="clock">
    <div id="hour-marks"></div>
    <div class="hand hour-hand" id="hour-hand"></div>
    <div class="hand minute-hand" id="minute-hand"></div>
    <div class="hand second-hand" id="second-hand"></div>
    <div class="center-point"></div>
  </div>

  <script>
    // Create hour marks and numbers
    const clock = document.getElementById('clock');
    const hourMarksContainer = document.getElementById('hour-marks');
    const clockRadius = 175; // Half of clock width


    // Create hour marks and numbers
    for (let i = 0; i < 60; i++) {
      const isHourMark = i % 5 === 0;
      const mark = document.createElement('div');
      mark.className = isHourMark ? 'hour-mark' : 'minute-mark';

      const angle = i * 6;
      const angleRadians = angle * Math.PI / 180;

      // Calculate position around the clock
      const markRadius = clockRadius - 15; // Position inside the clock edge

      mark.style.transform = `translate(-50%, 0) rotate(${angle}deg) translateY(-
${markRadius}px)`;
      hourMarksContainer.appendChild(mark);

      // Add hour numbers
      if (isHourMark) {
        const hourNumber = document.createElement('div');
        hourNumber.className = 'hour-number';
        const hourNum = i / 5 === 0 ? 12 : i / 5; // Convert position to hour
number
        hourNumber.textContent = hourNum;

        // Calculate position for the number
        const numberAngle = (i * 6 - 90) * Math.PI / 180; // -90 to start at 12 o'clock
position
```

```
    const numberRadius = clockRadius - 40; // Position for numbers

    const x = clockRadius + numberRadius * Math.cos(numberAngle);
    const y = clockRadius + numberRadius * Math.sin(numberAngle);

    hourNumber.style.left = x + 'px';
    hourNumber.style.top = y + 'px';

    clock.appendChild(hourNumber);
  }
}

// Clock functionality
const hourHand = document.getElementById('hour-hand');
const minuteHand = document.getElementById('minute-hand');
const secondHand = document.getElementById('second-hand');

function updateClock() {
  const now = new Date();
  const hours = now.getHours() % 12;
  const minutes = now.getMinutes();
  const seconds = now.getSeconds();
  const milliseconds = now.getMilliseconds();

  // Calculate rotation angles with smooth movement
  const secondAngle = (seconds + milliseconds / 1000) * 6;
```

```
      const minuteAngle = (minutes + seconds / 60) * 6;
      const hourAngle = (hours + minutes / 60) * 30;

      // Apply rotations
      secondHand.style.transform = `translate(-50%, -100%)
rotate(${secondAngle}deg)`;
      minuteHand.style.transform = `translate(-50%, -100%)
rotate(${minuteAngle}deg)`;
      hourHand.style.transform = `translate(-50%, -100%) rotate(${hourAngle}deg)`;

      // Call update on next animation frame for smooth second hand
      requestAnimationFrame(updateClock);
    }

    // Start the clock
    updateClock();
  </script>
</body>
</html>
```

OUTPUT:



MINION EYES:

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Minion Eye Following Cursor</title>
  <style>
    body {
      margin: 0;
      height: 100vh;
      display: flex;
```

```
   justify-content: center;
   align-items: center;
   background: #f0db4f; /* Minion yellow background */
   overflow: hidden;
   cursor: none; /* Hide default cursor for better effect */
 }

 .container {
   position: relative;
   display: flex;
   justify-content: center;
   align-items: center;
   gap: 80px;
 }

 .goggle {
   width: 200px;
   height: 200px;
   background: #333;
   border-radius: 50%;
   border: 15px solid #666;
   display: flex;
   justify-content: center;
   align-items: center;
   position: relative;
   overflow: hidden;
   box-shadow: 0 10px 20px rgba(0,0,0,0.3);
 }
```

```
.goggle::before {
  content: '';
  position: absolute;
  width: 210px;
  height: 210px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 50%;
  transform: translateY(-50%);
}

.eye {
  width: 110px;
  height: 110px;
  background: #fff;
  border-radius: 50%;
  position: relative;
  overflow: hidden;
  box-shadow: inset 0 0 20px rgba(0,0,0,0.2);
}

.iris {
  width: 50px;
  height: 50px;
  background: linear-gradient(#795548, #3e2723);
  border-radius: 50%;
  position: absolute;
  top: 50%;
```

```
    left: 50%;
    transform: translate(-50%, -50%);
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
  }

  .pupil {
    width: 25px;
    height: 25px;
    background: #000;
    border-radius: 50%;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }

  .highlight {
    width: 15px;
    height: 15px;
    background: #fff;
    border-radius: 50%;
    position: absolute;
    top: 30%;
    left: 30%;
    transform: translate(-50%, -50%);
  }

  .highlight:nth-child(2) {
    width: 8px;
    height: 8px;
    top: 60%;
    left: 70%;
  }

  .eyelid {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 50%;
    background: #f0db4f;
    transform: translateY(-100%);
    z-index: 10;
    border-bottom: 5px solid #ccc;
    transition: transform 0.3s;
  }

  .eyelid.bottom {
    top: auto;
    bottom: 0;
    transform: translateY(100%);
    border-bottom: none;
    border-top: 5px solid #ccc;
  }

  .goggle:hover .eyelid {
```

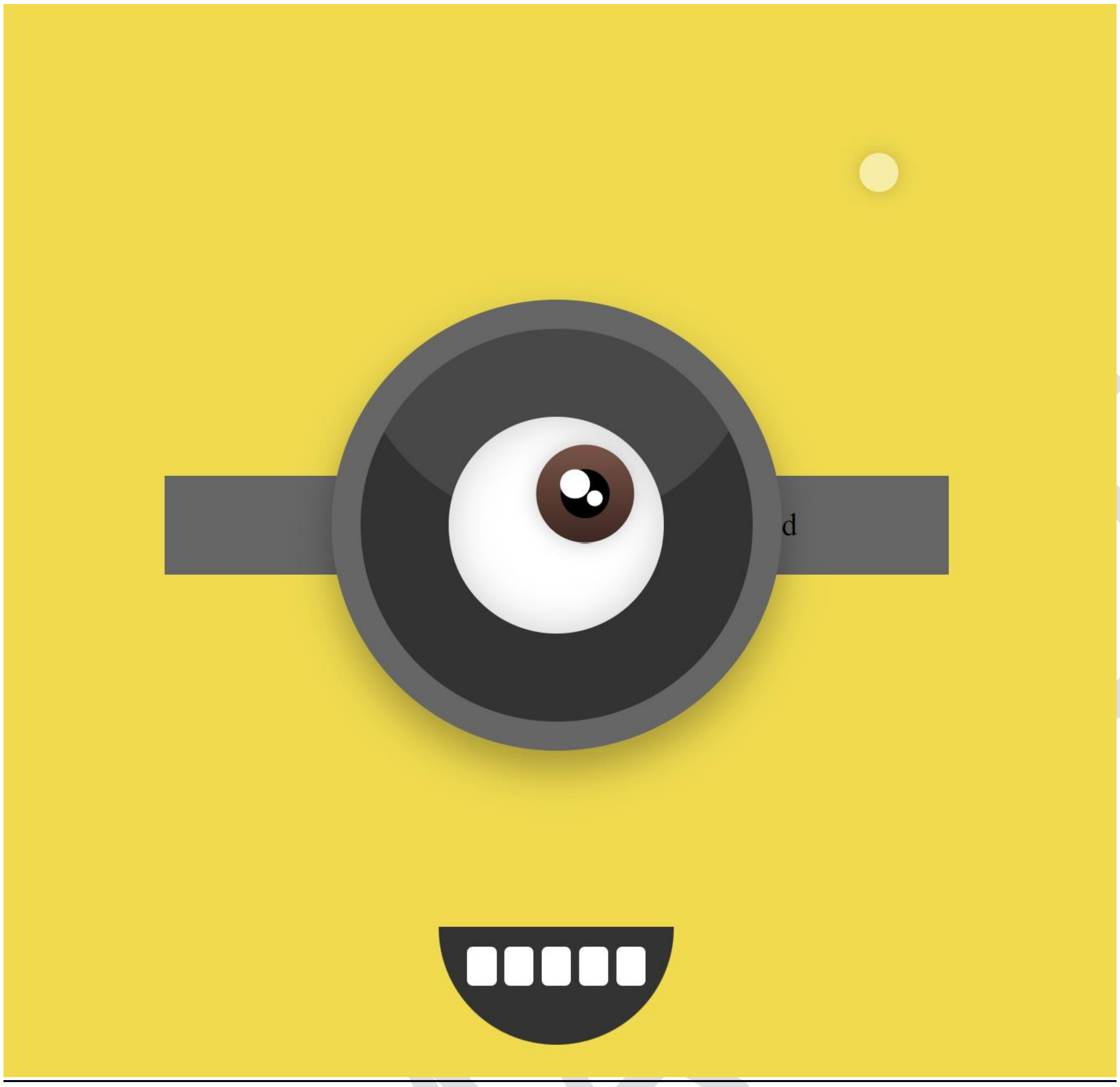
```
  transform: translateY(-50%);
}

.goggle:hover .eyelid.bottom {
  transform: translateY(50%);
}

.cursor {
  position: fixed;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  background: rgba(255, 255, 255, 0.5);
  transform: translate(-50%, -50%);
  pointer-events: none;
  z-index: 9999;
  box-shadow: 0 0 10px rgba(0,0,0,0.2);
}

/* Straps for goggles */
.strap {
  position: absolute;
  width: 400px;
  height: 50px;
  background: #666;
  z-index: -1;
}
```

```
    /* Minion mouth */
    .mouth {
      position: absolute;
      width: 120px;
      height: 60px;
      background: #333;
      border-radius: 0 0 100px 100px;
      bottom: -150px;
      display: flex;
      justify-content: center;
      overflow: hidden;
    }

    .teeth {
      display: flex;
      position: absolute;
      top: 10px;
    }

    .tooth {
      width: 15px;
      height: 20px;
      background: #fff;
      border-radius: 3px;
      margin: 0 2px;
    }
  </style>
</head>
```

```
<body>
  <div class="cursor" id="cursor"></div>


  <div class="container">
    <div class="strap"></div>


    <div class="goggle">
      <div class="eyelid"></div>
      <div class="eyelid bottom"></div>
      <div class="eye">
        <div class="iris" id="iris-left">
          <div class="pupil">
            <div class="highlight"></div>
            <div class="highlight"></div>
          </div>
        </div>
      </div>
    </div>


    <div class="mouth">
      <div class="teeth">
        <div class="tooth"></div>
        <div class="tooth"></div>
        <div class="tooth"></div>
        <div class="tooth"></div>
        <div class="tooth"></div>
      </div>
    </div>
  </div>


  <script>
    // Get DOM elements
    const cursor = document.getElementById('cursor');
    const leftIris = document.getElementById('iris-left');
    const goggle = document.querySelector('.goggle');
    const eyelids = document.querySelectorAll('.eyelid');

    // Set max movement range (in pixels)
    const maxEyeMove = 25;

    // Update cursor position
    document.addEventListener('mousemove', (e) => {
      // Move custom cursor
      cursor.style.left = e.clientX + 'px';
      cursor.style.top = e.clientY + 'px';

      // Get goggle position
      const goggleRect = goggle.getBoundingClientRect();
      const goggleCenterX = goggleRect.left + goggleRect.width / 2;
      const goggleCenterY = goggleRect.top + goggleRect.height / 2;

      // Calculate distance from cursor to center of goggle
      const distX = e.clientX - goggleCenterX;
      const distY = e.clientY - goggleCenterY;


      // Calculate distance ratio (for limiting movement)
```

```
      const distanceRatio = Math.min(1, Math.sqrt(distX * distX + distY * distY) / 300);


      // Calculate eye movement with limitations
      const moveX = (distX / goggleRect.width) * maxEyeMove * distanceRatio;
      const moveY = (distY / goggleRect.height) * maxEyeMove * distanceRatio;


      // Apply transformation to iris
      leftIris.style.transform = `translate(calc(-50% + ${moveX}px), calc(-50% +
${moveY}px))`;
    });


    // Add blinking animation
    function blink() {
      eyelids.forEach(eyelid => {
        if (eyelid.classList.contains('bottom')) {
          eyelid.style.transform = 'translateY(0%)';
        } else {
          eyelid.style.transform = 'translateY(0%)';
        }
      });


      setTimeout(() => {
        eyelids.forEach(eyelid => {
          if (eyelid.classList.contains('bottom')) {
            eyelid.style.transform = 'translateY(100%)';
          } else {
            eyelid.style.transform = 'translateY(-100%)';
          }
        });
      }, 200);

      // Random blink interval between 2 and 6 seconds
      const nextBlink = Math.random() * 4000 + 2000;
      setTimeout(blink, nextBlink);
    }

    // Start blinking
    setTimeout(blink, 1000);

    // Make eye blink when clicked
    document.addEventListener('click', () => {
      blink();
    });
  </script>
</body>
</html>
```

OUTPUT:



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>Mobile Flashlight</title>
  <style>
   * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  body {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background-color: #121212;
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu,
Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
    overflow: hidden;
  }

  .scene {
    position: relative;
    width: 100%;
    height: 100vh;
    overflow: hidden;
  }

  .phone {
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
    width: 300px;
    height: 600px;
    background-color: #1a1a1a;
    border-radius: 40px;
    border: 8px solid #333;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
    overflow: hidden;
    z-index: 10;
  }

  .phone-screen {
    position: relative;
    width: 100%;
    height: 100%;
    background-color: #000;
    display: flex;
    flex-direction: column;
    color: white;
  }

  .status-bar {
    display: flex;
    justify-content: space-between;
    padding: 10px 20px;
    font-size: 14px;
    background-color: rgba(0, 0, 0, 0.8);
  }

  .time {
    font-weight: bold;
  }

  .icons span {
    margin-left: 10px;
  }

  .app-screen {
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    align-items: center;
    padding: 40px 0;
  }

  .flashlight-label {
    font-size: 24px;
    font-weight: bold;
    margin-top: 40px;
  }

  .flashlight-btn {
    width: 120px;
    height: 120px;
    border-radius: 50%;
    background-color: #333;
    display: flex;
    justify-content: center;
    align-items: center;
    cursor: pointer;
    margin: 40px 0;
    box-shadow: 0 0 10px rgba(255, 255, 255, 0.1);
    transition: all 0.3s ease;
  }

  .flashlight-btn-inner {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    background: radial-gradient(circle at center, #555, #444, #333);
    display: flex;
    justify-content: center;
    align-items: center;
    transition: all 0.3s ease;
  }

  .flashlight-icon {
    font-size: 40px;
    color: #ddd;
    transition: all 0.3s ease;
  }

  .flashlight-btn.on .flashlight-btn-inner {
    background: radial-gradient(circle at center, #fff, #eee, #ddd);
  }
```

```
  .flashlight-btn.on .flashlight-icon {
    color: #333;
  }

  .brightness-control {
    width: 80%;
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 10px;
  }

  .brightness-label {
    font-size: 16px;
    color: #ccc;
  }

  .brightness-slider {
    width: 100%;
    -webkit-appearance: none;
    height: 5px;
    border-radius: 5px;
    background: #555;
    outline: none;
    opacity: 0.7;
    transition: opacity 0.2s;
  }
```

```
.brightness-slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  background: #fff;
  cursor: pointer;
}

.light-beam {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: radial-gradient(
    circle at 50% 50%,
    rgba(255, 255, 255, 0.8) 0%,
    rgba(255, 255, 255, 0.2) 20%,
    rgba(255, 255, 255, 0) 70%
  );
  opacity: 0;
  pointer-events: none;
  transition: opacity 0.3s ease;
  mix-blend-mode: screen;
}
```

```
.bottom-bar {
  width: 100%;
  height: 5px;
  background-color: #333;
  border-radius: 3px;
  margin-bottom: 10px;
}

.objects {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  pointer-events: none;
}

.object {
  position: absolute;
  background-color: #333;
  border-radius: 10px;
}

/* Room objects */
.object:nth-child(1) {
  width: 100px;
  height: 120px;
  left: 10%;
  top: 20%;
}

.object:nth-child(2) {
  width: 150px;
  height: 80px;
  right: 15%;
  top: 30%;
}

.object:nth-child(3) {
  width: 80px;
  height: 200px;
  left: 25%;
  bottom: 10%;
}

.object:nth-child(4) {
  width: 200px;
  height: 100px;
  right: 20%;
  bottom: 20%;
}

.instructions {
  position: absolute;
  bottom: 20px;
```

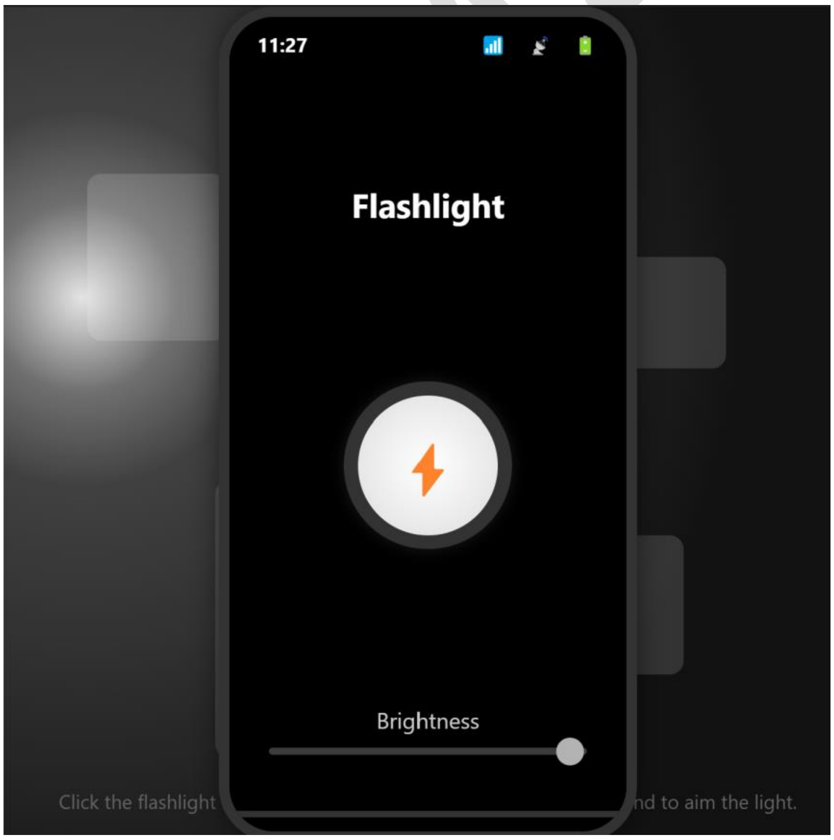
```
      text-align: center;
      color: #666;
      font-size: 14px;
      width: 100%;
      z-index: 5;
    }
  </style>
</head>
<body>
  <div class="scene">
    <div class="objects">
      <div class="object"></div>
      <div class="object"></div>
      <div class="object"></div>
      <div class="object"></div>
    </div>

    <div class="light-beam" id="light-beam"></div>

    <div class="phone">
      <div class="phone-screen">
        <div class="status-bar">
          <div class="time" id="time">9:41</div>
          <div class="icons">
            <span>📶</span>
            <span>📡</span>
            <span>🔋</span>
          </div>
        </div>


        <div class="app-screen">
          <div class="flashlight-label">Flashlight</div>


          <div class="flashlight-btn" id="flashlight-btn">
            <div class="flashlight-btn-inner">
              <div class="flashlight-icon">⚡</div>
            </div>
          </div>


          <div class="brightness-control">
            <div class="brightness-label">Brightness</div>
            <input type="range" min="1" max="100" value="100" class="brightness-slider"
id="brightness-slider">
          </div>
        </div>


        <div class="bottom-bar"></div>
      </div>
    </div>


    <div class="instructions">Click the flashlight button to toggle on/off. Move your mouse
around to aim the light.</div>
  </div>


  <script>
    document.addEventListener('DOMContentLoaded', function() {
      // Get elements
```

```
const flashlightBtn = document.getElementById('flashlight-btn');
const lightBeam = document.getElementById('light-beam');
const brightnessSlider = document.getElementById('brightness-slider');
const timeDisplay = document.getElementById('time');

// Set current time
function updateTime() {
  const now = new Date();
  let hours = now.getHours();
  let minutes = now.getMinutes();

  // Format time as 12-hour with leading zeros
  hours = hours % 12 || 12;
  minutes = minutes < 10 ? '0' + minutes : minutes;

  timeDisplay.textContent = `${hours}:${minutes}`;
}

// Initial time update
updateTime();

// Update time every minute
setInterval(updateTime, 60000);

// Flashlight state
let isOn = false;

// Toggle flashlight
```

```
    flashlightBtn.addEventListener('click', function() {
     isOn = !isOn;

     if (isOn) {
       flashlightBtn.classList.add('on');
       lightBeam.style.opacity = brightnessSlider.value / 100;
     } else {
       flashlightBtn.classList.remove('on');
       lightBeam.style.opacity = 0;
     }
    });

    // Adjust brightness
    brightnessSlider.addEventListener('input', function() {
     if (isOn) {
       lightBeam.style.opacity = this.value / 100;
     }
    });

    // Move light beam with mouse
    document.addEventListener('mousemove', function(e) {
     if (isOn) {
       const x = e.clientX;
       const y = e.clientY;

       lightBeam.style.background = `radial-gradient(
         circle at ${x}px ${y}px,
         rgba(255, 255, 255, 0.9) 0%,
         rgba(255, 255, 255, 0.4) 10%,
         rgba(255, 255, 255, 0.2) 20%,
         rgba(255, 255, 255, 0) 70%
       )`;
     }
    });

    // Handle touch movement for mobile
    document.addEventListener('touchmove', function(e) {
     if (isOn && e.touches.length > 0) {
       const touch = e.touches[0];
       const x = touch.clientX;
       const y = touch.clientY;

       lightBeam.style.background = `radial-gradient(
         circle at ${x}px ${y}px,
         rgba(255, 255, 255, 0.9) 0%,
         rgba(255, 255, 255, 0.4) 10%,
         rgba(255, 255, 255, 0.2) 20%,
         rgba(255, 255, 255, 0) 70%
       )`;

       // Prevent default to avoid page scrolling
       e.preventDefault();
     }
    }, { passive: false });

    // Double-click shortcut
```

```
      document.addEventListener('dblclick', function() {
        isOn = !isOn;


        if (isOn) {
          flashlightBtn.classList.add('on');
          lightBeam.style.opacity = brightnessSlider.value / 100;
        } else {
          flashlightBtn.classList.remove('on');
          lightBeam.style.opacity = 0;
        }
      });
    });
  </script>
</body>
</html>
```

OUTPUT:

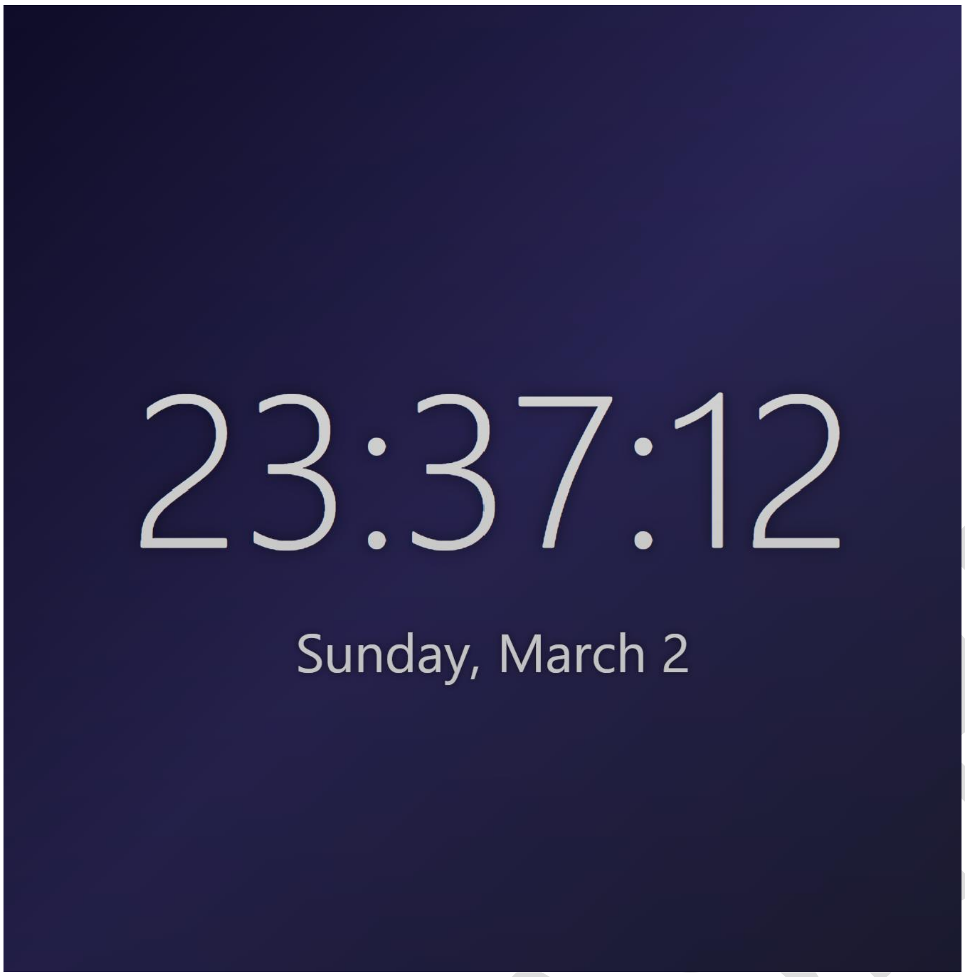


DIGITAL CLOCK:

CODE:

```
<!DOCTYPE html>
<html>
<head>
 <style>
   body, html {
    margin: 0;
    padding: 0;
    height: 100%;
    overflow: hidden;
    font-family: 'Segoe UI', sans-serif;
   }

   .lockscreen {
    position: relative;
    width: 100%;
    height: 100%;
    background: linear-gradient(135deg, #0f0c29 0%, #302b63 50%, #24243e 100%);
    background-size: cover;
    background-position: center;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
   }

   .clock {
    font-size: 8rem;
    font-weight: 300;
    color: white;
    text-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
   }

   .date {
    font-size: 2rem;
    color: white;
    text-shadow: 0 0 5px rgba(0, 0, 0, 0.5);
    margin-top: 0.5rem;
   }

   .overlay {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: linear-gradient(to bottom, rgba(0,0,0,0.1) 0%, rgba(0,0,0,0.3) 100%);
   }
 </style>
</head>
<body>
 <div class="lockscreen">
   <div class="overlay"></div>
   <div class="clock" id="clock">00:00:00</div>
   <div class="date" id="date">Sunday, March 02</div>
 </div>


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


    // Format time (hours:minutes:seconds)
    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}`;


    // Format date (weekday, month day)
    const options = { weekday: 'long', month: 'long', day: 'numeric' };
    const dateString = now.toLocaleDateString('en-US', options);
    document.getElementById('date').textContent = dateString;
   }


   // Update clock immediately and then every second
   updateClock();
   setInterval(updateClock, 1000);
 </script>
</body>
</html>
```

OUTPUT:



FLASHLIGHT TEXT:

CODE:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body, html {
      margin: 0;
      padding: 0;
      height: 100%;
      background-color: #000;
      display: flex;
      justify-content: center;
```

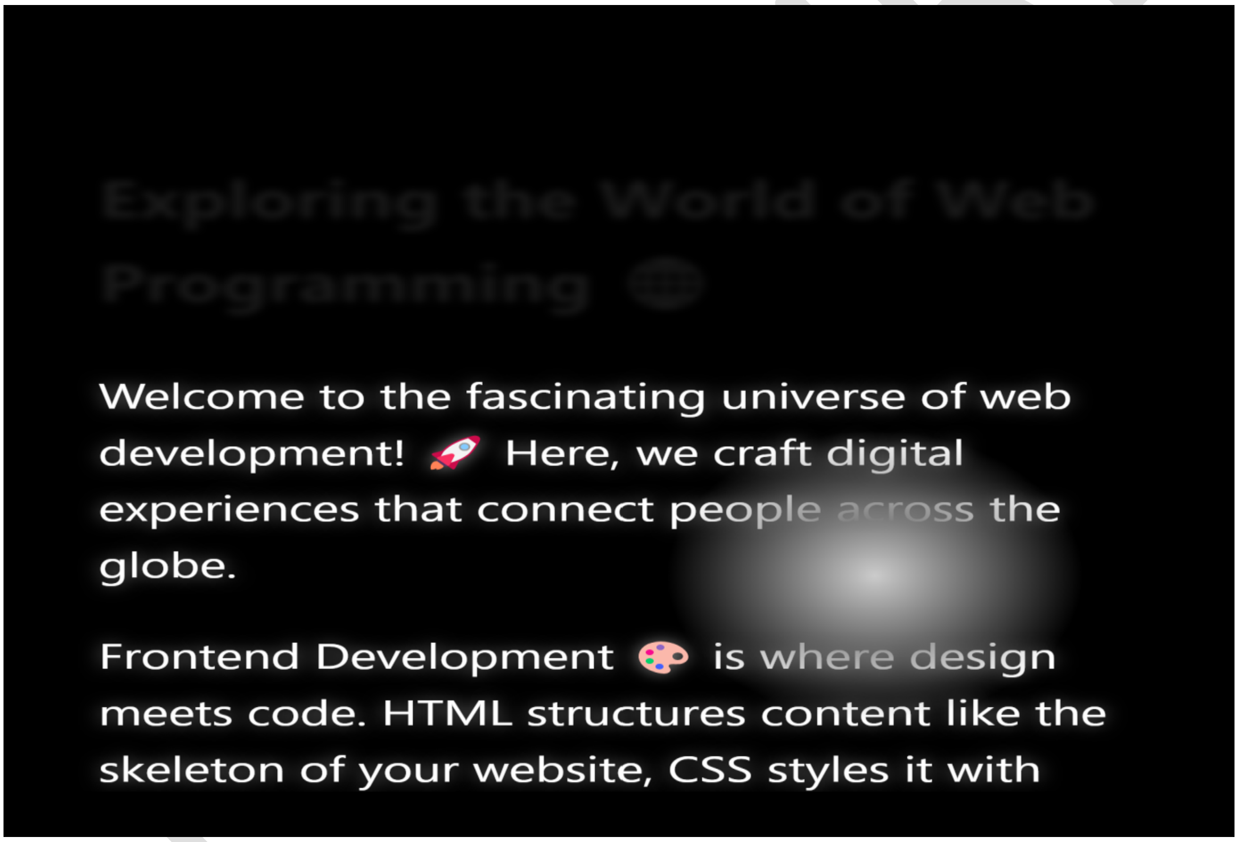
```
   align-items: center;
   overflow: hidden;
   font-family: 'Segoe UI', system-ui, sans-serif;
   cursor: none;
 }

 .container {
   width: 90%;
   max-width: 800px;
   height: 80%;
   position: relative;
   overflow: hidden;
 }

 .text-content {
   font-size: 24px;
   line-height: 1.6;
   color: transparent;
   text-shadow: 0 0 5px rgba(255, 255, 255, 0.1);
   padding: 20px;
 }

 .flashlight {
   position: fixed;
   width: 200px;
   height: 200px;
   border-radius: 50%;
   background: radial-gradient(circle, rgba(255,255,255,0.8) 0%, rgba(255,255,255,0) 70%);
```

```
      transform: translate(-50%, -50%);
      pointer-events: none;
      mix-blend-mode: difference;
      z-index: 999;
    }

    h1 {
      font-size: 36px;
      margin-bottom: 30px;
      color: transparent;
      text-shadow: 0 0 5px rgba(255, 255, 255, 0.1);
    }

    p {
      margin-bottom: 20px;
    }

    .highlight {
      color: #fff;
      text-shadow: 0 0 10px rgba(255, 255, 255, 0.5);
    }
  </style>
</head>
<body>
  <div class="flashlight" id="flashlight"></div>

  <div class="container">
    <div class="text-content">
      <h1>Exploring the World of Web Programming 🌐</h1>


      <p>Welcome to the fascinating universe of web development! 🚀 Here, we craft digital experiences that connect people across the globe.</p>


      <p>Frontend Development 🎨 is where design meets code. HTML structures content like the skeleton of your website, CSS styles it with beautiful designs, and JavaScript brings it to life with interactivity.</p>


      <p>Backend Development ⚙️ powers the invisible magic. Servers process requests, databases store information, and APIs connect different systems together seamlessly.</p>


      <p>Modern frameworks make development more efficient! React ⚛️ revolutionizes UI creation, Node.js □ brings JavaScript to the server, and Python 🐍 offers simplicity for web applications.</p>


      <p>Web security 🔒 is crucial for protecting user data. Implementing HTTPS, validating inputs, and preventing injections are essential practices every developer must know.</p>


      <p>Responsive design 📱 ensures your website looks great on all devices. Using flexible layouts and media queries allows your content to adapt to any screen size.</p>


      <p>Performance optimization ⚡ keeps users engaged. Minifying files, optimizing images, and using content delivery networks can significantly improve loading times.</p>


      <p>The web is constantly evolving! 🔄 WebAssembly, Progressive Web Apps, and GraphQL are just a few technologies shaping the future of web development.</p>
    </div>
  </div>
```

```
 <script>
  const flashlight = document.getElementById('flashlight');
  const container = document.querySelector('.container');
  const textContent = document.querySelector('.text-content');

  // Track mouse movement
  document.addEventListener('mousemove', function(e) {
    // Update flashlight position
    flashlight.style.left = e.clientX + 'px';
    flashlight.style.top = e.clientY + 'px';

    // Calculate position relative to container
    const containerRect = container.getBoundingClientRect();
    const mouseX = e.clientX;
    const mouseY = e.clientY;

    // Check if mouse is within or near text container
    const inRange =
      mouseX >= containerRect.left - 100 &&
      mouseX <= containerRect.right + 100 &&
      mouseY >= containerRect.top - 100 &&
      mouseY <= containerRect.bottom + 100;

    if (inRange) {
      // Light up text elements near the cursor
      const elements = textContent.querySelectorAll('h1, p');
      elements.forEach(element => {
        const rect = element.getBoundingClientRect();
        const centerX = rect.left + rect.width / 2;
        const centerY = rect.top + rect.height / 2;


        const distance = Math.sqrt(
          Math.pow(mouseX - centerX, 2) +
          Math.pow(mouseY - centerY, 2)
        );


        if (distance < 200) {
          element.classList.add('highlight');
        } else {
          element.classList.remove('highlight');
        }
      });
    } else {
      // Turn off all highlights when cursor is far away
      const elements = textContent.querySelectorAll('.highlight');
      elements.forEach(element => {
        element.classList.remove('highlight');
      });
    }
  });


  // Hide flashlight when cursor leaves the window
  document.addEventListener('mouseleave', function() {
    flashlight.style.display = 'none';
  });
```

```
    // Show flashlight when cursor enters the window
    document.addEventListener('mouseenter', function() {
      flashlight.style.display = 'block';
    });
  </script>
</body>
</html>
```

OUTPUT:



Exploring the World of Web Programming 🌐

Welcome to the fascinating universe of web development! 🚀 Here, we craft digital experiences that connect people across the globe.

Frontend Development 🎨 is where design meets code. HTML structures content like the skeleton of your website, CSS styles it with

VERTICAL IMAGE SLIDER:

CODE:

```
<!DOCTYPE html>
<html>
```

23BPS1178

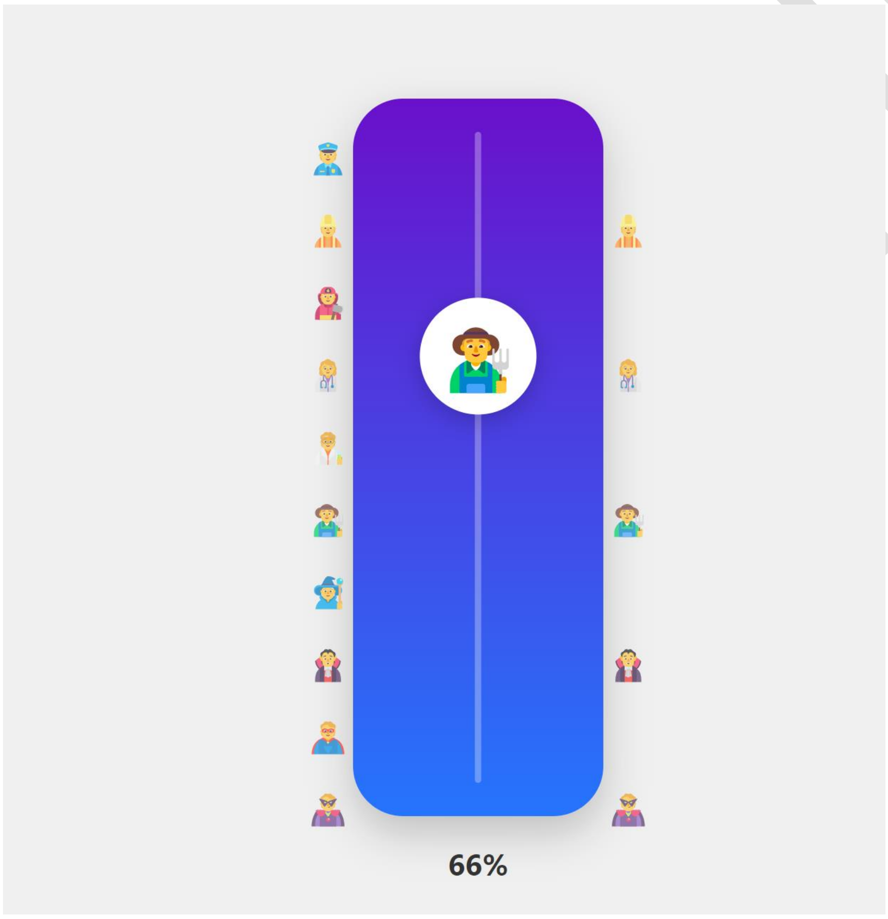
```
<head>
  <style>
    body, html {
      margin: 0;
      padding: 0;
      height: 100%;
      display: flex;
      justify-content: center;
      align-items: center;
      background-color: #f0f0f0;
      font-family: 'Segoe UI', system-ui, sans-serif;
    }

    .slider-container {
      position: relative;
      width: 120px;
      height: 400px;
      background: linear-gradient(to bottom, #6a11cb 0%, #2575fc 100%);
      border-radius: 30px;
      box-shadow: 0 10px 25px rgba(0,0,0,0.2);
      padding: 15px;
    }

    .emoji-track {
      position: absolute;
      top: 20px;
      bottom: 20px;
      left: 50%;
      transform: translateX(-50%);
      width: 4px;
      background-color: rgba(255,255,255,0.3);
      border-radius: 2px;
    }

    .slider-thumb {
      position: absolute;
      left: 50%;
      transform: translate(-50%, -50%);
      width: 70px;
      height: 70px;
      background-color: white;
      border-radius: 50%;
      display: flex;
      justify-content: center;
      align-items: center;
      font-size: 40px;
      cursor: pointer;
      box-shadow: 0 4px 15px rgba(0,0,0,0.2);
      user-select: none;
      transition: transform 0.1s ease;
    }

    .slider-thumb:active {
      transform: translate(-50%, -50%) scale(0.95);
    }
```

```
.emoji-markers {
  position: absolute;
  top: 20px;
  bottom: 20px;
  left: 0;
  right: 0;
}

.emoji-marker {
  position: absolute;
  left: -30px;
  width: 30px;
  height: 30px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 20px;
  color: rgba(255,255,255,0.7);
}

.emoji-marker.right {
  left: auto;
  right: -30px;
}

.value-display {
  position: absolute;
  bottom: -40px;
```

```
      left: 0;
      right: 0;
      text-align: center;
      font-size: 18px;
      font-weight: bold;
      color: #333;
    }
  </style>
</head>
<body>
  <div class="slider-container">
    <div class="emoji-track"></div>
    <div class="emoji-markers" id="markers"></div>
    <div class="slider-thumb" id="thumb">👮</div>
    <div class="value-display" id="value-display">50%</div>
  </div>

  <script>
    // Define emoji array (no smile-related emojis)
    const emojis = [
      '👮', // police officer
      '👷', // construction worker
      '□🚒', // firefighter
      '👩□', // health worker
      '□🔬', // scientist
      '👨🌾', // farmer
      '□', // mage
      '□', // vampire
      '□', // superhero
      '□' // supervillain
    ];

    const container = document.querySelector('.slider-container');
    const thumb = document.getElementById('thumb');
    const markers = document.getElementById('markers');
    const valueDisplay = document.getElementById('value-display');

    // Create emoji markers
    emojis.forEach((emoji, index) => {
      const marker = document.createElement('div');
      marker.className = 'emoji-marker';
      marker.textContent = emoji;
      marker.style.top = `${index * (100 / (emojis.length - 1))}%`;
      markers.appendChild(marker);

      // Create right side markers with alternating pattern
      if (index % 2 === 1) {
        const rightMarker = document.createElement('div');
        rightMarker.className = 'emoji-marker right';
        rightMarker.textContent = emoji;
        rightMarker.style.top = `${index * (100 / (emojis.length - 1))}%`;
        markers.appendChild(rightMarker);
      }
    });
```

```
  // Slider functionality
  let isDragging = false;
  const bounds = {
    min: 20, // top position (px)
    max: container.clientHeight - 20 // bottom position (px)
  };

  function updateThumbPosition(clientY) {
    const containerRect = container.getBoundingClientRect();
    let posY = clientY - containerRect.top;

    // Constrain within bounds
    posY = Math.max(bounds.min, Math.min(bounds.max, posY));

    // Update thumb position
    thumb.style.top = posY + 'px';

    // Calculate percentage
    const percentage = Math.round(((posY - bounds.min) / (bounds.max - bounds.min)) * 100);
    valueDisplay.textContent = `${100 - percentage}%`;

    // Update emoji
    const emojiIndex = Math.floor((100 - percentage) / (100 / (emojis.length - 0.99)));
    thumb.textContent = emojis[emojiIndex];
  }

  // Mouse/touch event handlers
  thumb.addEventListener('mousedown', () => {
    isDragging = true;
  });

  document.addEventListener('mousemove', (e) => {
    if (isDragging) {
      updateThumbPosition(e.clientY);
    }
  });

  document.addEventListener('mouseup', () => {
    isDragging = false;
  });

  // Touch events
  thumb.addEventListener('touchstart', (e) => {
    isDragging = true;
    e.preventDefault();
  });

  document.addEventListener('touchmove', (e) => {
    if (isDragging) {
      updateThumbPosition(e.touches[0].clientY);
      e.preventDefault();
    }
  });

  document.addEventListener('touchend', () => {
    isDragging = false;
    });

    // Set initial position (middle)
    thumb.style.top = (bounds.min + (bounds.max - bounds.min) / 2) + 'px';
  </script>
</body>
</html>
```

OUTPUT:



WEB CAMERA ACCESS:

CODE:

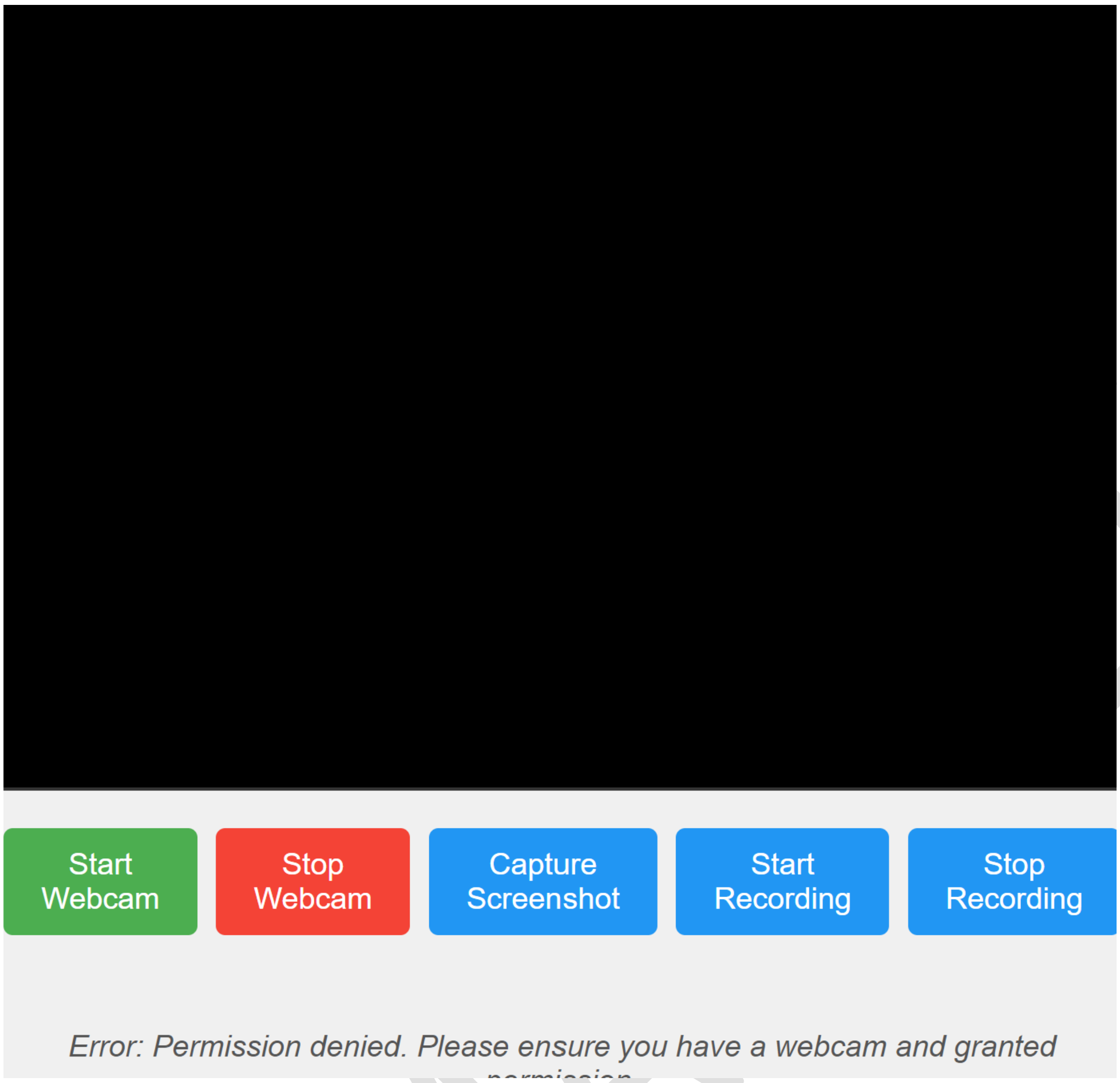
```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Webcam Video Capture</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      max-width: 900px;
      margin: 0 auto;
      padding: 20px;
      text-align: center;
      background-color: #f0f0f0;
    }
    h1 {
      color: #333;
    }
    .container {
      display: flex;
      flex-direction: column;
      align-items: center;
      gap: 20px;
    }
    .video-container {
      position: relative;
      width: 640px;
      height: 480px;
      border: 2px solid #333;
      border-radius: 8px;
      overflow: hidden;
      background-color: #000;
    }
    #video {
      width: 100%;
      height: 100%;
      object-fit: cover;
    }
    .controls {
      display: flex;
      gap: 10px;
      margin-bottom: 20px;
    }
    button {
      padding: 10px 20px;
      font-size: 16px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      transition: background-color 0.3s;
    }
    .start-btn {
      background-color: #4CAF50;
      color: white;
    }
    .start-btn:hover {
      background-color: #388E3C;
    }
    .stop-btn {
      background-color: #F44336;
      color: white;
    }
    .stop-btn:hover {
      background-color: #D32F2F;
    }
    .capture-btn, .record-btn, .stop-record-btn {
      background-color: #2196F3;
      color: white;
    }
    .capture-btn:hover, .record-btn:hover, .stop-record-btn:hover {
      background-color: #1976D2;
    }
    .status {
      margin-top: 10px;
      font-style: italic;
      color: #555;
    }
    .screenshots {
      display: flex;
      flex-wrap: wrap;
      gap: 10px;
      justify-content: center;
      margin-top: 20px;
    }
    .screenshot {
      position: relative;
      width: 200px;
      border: 1px solid #ccc;
      border-radius: 4px;
      overflow: hidden;
    }
    .screenshot img {
      width: 100%;
      display: block;
    }
    .no-webcam {
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100%;
      color: white;
      font-size: 18px;
    }
  </style>
</head>
<body>
  <h1>Webcam Video Capture</h1>

  <div class="container">
    <div class="video-container">
      <video id="video" autoplay playsinline></video>
      <div class="no-webcam" id="no-webcam">Webcam not started</div>
    </div>
    <div class="controls">
      <button id="start-btn" class="start-btn">Start Webcam</button>
      <button id="stop-btn" class="stop-btn" disabled>Stop Webcam</button>
      <button id="capture-btn" class="capture-btn" disabled>Capture Screenshot</button>
      <button id="record-btn" class="record-btn" disabled>Start Recording</button>
      <button id="stop-record-btn" class="stop-record-btn" disabled>Stop
Recording</button>
    </div>

    <p id="status" class="status">Click "Start Webcam" to begin.</p>

    <div id="screenshots" class="screenshots"></div>
    <div id="recordings" class="screenshots"></div>
  </div>

  <script>
    const video = document.getElementById('video');
    const noWebcam = document.getElementById('no-webcam');
    const startBtn = document.getElementById('start-btn');
    const stopBtn = document.getElementById('stop-btn');
    const captureBtn = document.getElementById('capture-btn');
    const recordBtn = document.getElementById('record-btn');
    const stopRecordBtn = document.getElementById('stop-record-btn');
    const status = document.getElementById('status');
    const screenshots = document.getElementById('screenshots');
    const recordingsContainer = document.getElementById('recordings');
    let stream = null;
    let mediaRecorder = null;
    let recordedChunks = [];


    // Start webcam and prompt for camera & microphone access
    startBtn.addEventListener('click', async () => {
      try {
        stream = await navigator.mediaDevices.getUserMedia({
          video: { width: { ideal: 1280 }, height: { ideal: 720 } },
          audio: true
        });
        video.srcObject = stream;
        noWebcam.style.display = 'none';
        startBtn.disabled = true;
        stopBtn.disabled = false;
        captureBtn.disabled = false;
        recordBtn.disabled = false;
        status.textContent = 'Webcam is active. You can capture screenshots or record video.';
      } catch (err) {
        console.error('Error accessing webcam:', err);
        status.textContent = `Error: ${err.message}. Please ensure you have a webcam and granted
permission.`;
      }
    });


    // Stop webcam
    stopBtn.addEventListener('click', () => {
      if (stream) {
        stream.getTracks().forEach(track => track.stop());
        video.srcObject = null;
        noWebcam.style.display = 'flex';
        startBtn.disabled = false;
        stopBtn.disabled = true;
        captureBtn.disabled = true;
        recordBtn.disabled = true;
        stopRecordBtn.disabled = true;
        status.textContent = 'Webcam stopped.';
      }
    });

    // Capture a screenshot
    captureBtn.addEventListener('click', () => {
      if (stream) {
        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        const ctx = canvas.getContext('2d');
        ctx.drawImage(video, 0, 0);
        const imgURL = canvas.toDataURL('image/png');
        const img = document.createElement('img');
        img.src = imgURL;
        const screenshotDiv = document.createElement('div');
        screenshotDiv.className = 'screenshot';
        screenshotDiv.appendChild(img);
        screenshots.appendChild(screenshotDiv);
        status.textContent = 'Screenshot captured.';
      }
    });
    // Start recording video
    recordBtn.addEventListener('click', () => {
      if (stream) {
        recordedChunks = [];
        mediaRecorder = new MediaRecorder(stream);
        mediaRecorder.ondataavailable = (e) => {
         if (e.data.size > 0) {
          recordedChunks.push(e.data);
         }
        };
        mediaRecorder.onstop = () => {
         const blob = new Blob(recordedChunks, { type: 'video/webm' });
         const url = URL.createObjectURL(blob);
         const videoElem = document.createElement('video');
         videoElem.src = url;
         videoElem.controls = true;
         videoElem.style.width = '300px';
         const recordingDiv = document.createElement('div');
         recordingDiv.className = 'screenshot';
         recordingDiv.appendChild(videoElem);
         recordingsContainer.appendChild(recordingDiv);
         status.textContent = 'Recording stopped and saved.';
        };
        mediaRecorder.start();
        recordBtn.disabled = true;
        stopRecordBtn.disabled = false;
        status.textContent = 'Recording...';
      }
```

```
    });

    // Stop video recording
    stopRecordBtn.addEventListener('click', () => {
      if (mediaRecorder && mediaRecorder.state !== 'inactive') {
        mediaRecorder.stop();
        recordBtn.disabled = false;
        stopRecordBtn.disabled = true;
      }
    });
  </script>
</body>
</html>
```

OUTPUT:

23BPS1178



Error: Permission denied. Please ensure you have a webcam and granted