Name: Nikhil Bansal V

Registration Number: 23BPS1039

Web Lab 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>Digital Clock</title>
   <style>
      body{
         font-family: Arial, Helvetica, sans-serif;
         text-align: center;
         margin-top: 150px;
      }
      #clock{
         font-size: 50px;
         color: blue;
      }
   </style>
</head>
<body>
   <h1>JavaScript Digital Clock by Nikhil</h1>
   <div id="clock">00:00:00</div>

   <script>
      function updateClock(){
         const now = new Date();
         let hours = now.getHours();
```

```
        let minutes = now.getMinutes();
        let seconds = now.getSeconds();

        hours = hours < 10 ? "0" + hours : hours;
        minutes = minutes < 10 ? "0" + minutes : minutes;
        seconds = seconds < 10 ? "0" + seconds : seconds;

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

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

Output:

JavaScript Digital Clock by Nikhil

19:15:15

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;
      margin: 0;
      background: linear-gradient(135deg, #74ebd5, #acb6e5);
      font-family: 'Arial', sans-serif;
    }

    .clock {
      position: relative;
      width: 300px;
      height: 300px;
      border: 15px solid #ffffff;
      border-radius: 50%;
      background: rgba(255, 255, 255, 0.8);
      box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
    }

    .hand {
      position: absolute;
      background: #333;
```

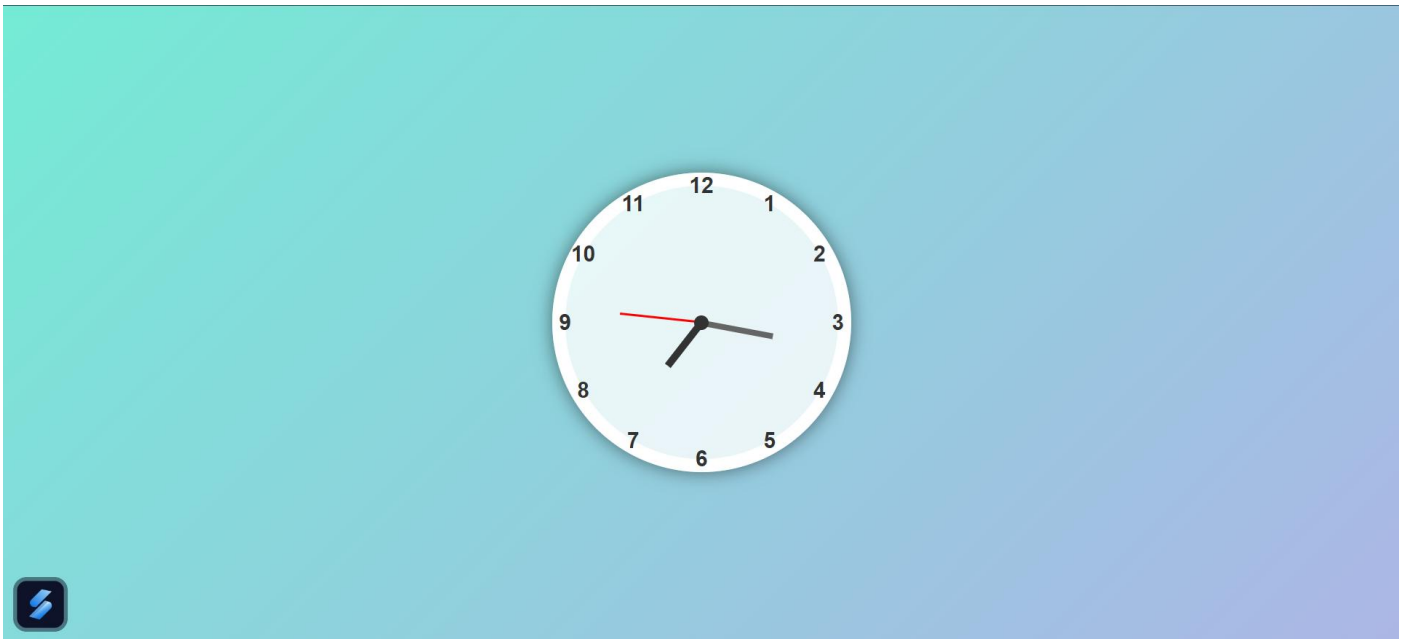
```
    transform-origin: bottom;
    bottom: 50%;
    left: 50%;
    transform: translateX(-50%);
}

.hour {
    width: 8px;
    height: 60px;
    background: #333;
    z-index: 3;
}

.minute {
    width: 6px;
    height: 80px;
    background: #666;
    z-index: 2;
}

.second {
    width: 2px;
    height: 90px;
    background: red;
    z-index: 1;
}

.center {
    position: absolute;
    width: 16px;
    height: 16px;
    background: #333;
```

```
        border-radius: 50%;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        z-index: 4;
      }

      .number {
        position: absolute;
        font-size: 24px;
        font-weight: bold;
        color: #333;
        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 class="number" id="number1">1</div>
      <div class="number" id="number2">2</div>
      <div class="number" id="number3">3</div>
      <div class="number" id="number4">4</div>
      <div class="number" id="number5">5</div>
      <div class="number" id="number6">6</div>
      <div class="number" id="number7">7</div>
      <div class="number" id="number8">8</div>
      <div class="number" id="number9">9</div>
```

```
    <div class="number" id="number10">10</div>
    <div class="number" id="number11">11</div>
    <div class="number" id="number12">12</div>
</div>

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

        const hourDeg = ((hours % 12) + minutes / 60) * 30;
        const minuteDeg = (minutes + seconds / 60) * 6;
        const secondDeg = seconds * 6;

        document.getElementById('hour').style.transform = `translateX(-50%) rotate(${hourDeg}deg)`;
        document.getElementById('minute').style.transform = `translateX(-50%) rotate(${minuteDeg}deg)`;
        document.getElementById('second').style.transform = `translateX(-50%) rotate(${secondDeg}deg)`;
    }

    function positionNumbers() {
        const clockRadius = 150;
        const numberElements = document.querySelectorAll('.number');
        const numberCount = numberElements.length;

        for (let i = 0; i < numberCount; i++) {
            const angle = (i + 1) * (360 / numberCount) * (Math.PI / 180);
            const x = clockRadius * Math.sin(angle) + clockRadius;
            const y = -clockRadius * Math.cos(angle) + clockRadius;

            numberElements[i].style.left = `${x}px`;
```

```
            numberElements[i].style.top = `${y}px`;
        }
    }

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

Output:



3) Flashlight Text

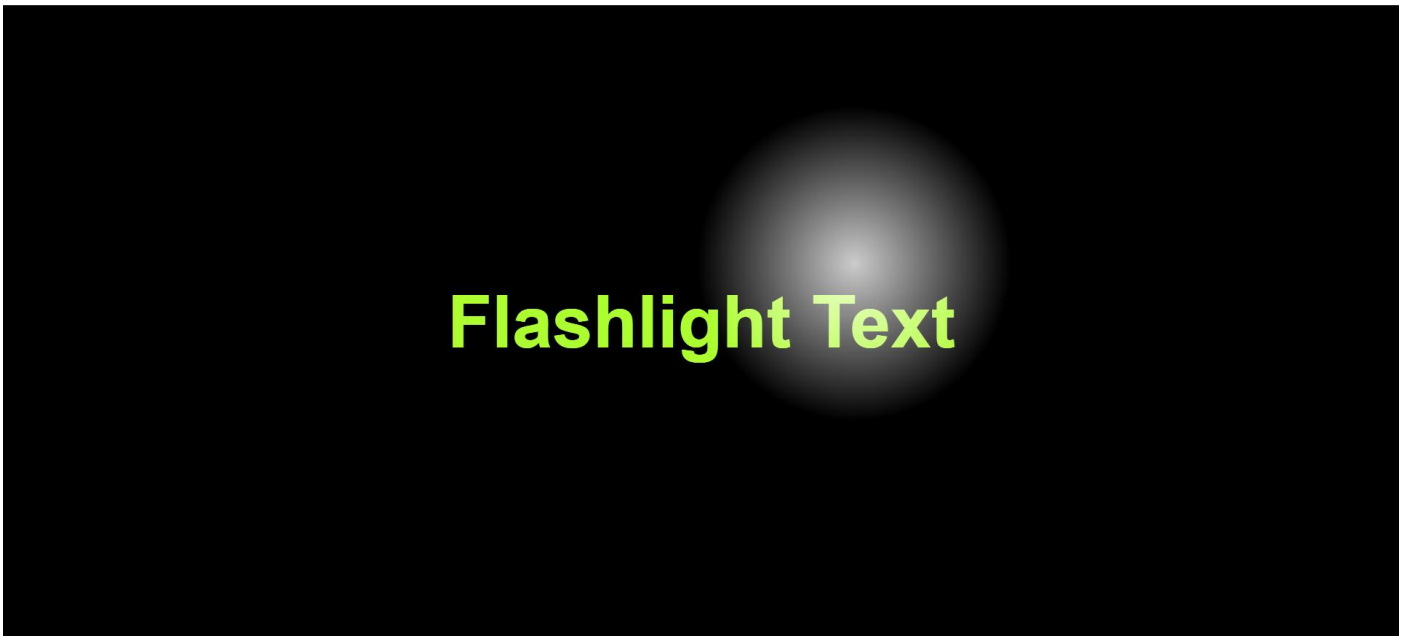
Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Flashlight Text Effect</title>
 <style>
  body {
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
   margin: 0;
   background-color: #000;
   overflow: hidden;
   font-family: 'Arial', sans-serif;
  }

  .text-container {
   position: relative;
   font-size: 5rem;
   font-weight: bold;
   color: greenyellow;
   background: linear-gradient(90deg, #ff00ff, #00ffff, #ffff00, #ff00ff);
   background-clip: text;
   -webkit-background-clip: text;
  }

  .flashlight {
   position: absolute;
```

```
    width: 350px;
    height: 350px;
    background: radial-gradient(
      circle,
      rgba(255, 255, 255, 0.8) 0%,
      rgba(255, 255, 255, 0) 70%
    );
    border-radius: 50%;
    pointer-events: none;
    transform: translate(-50%, -50%);
    mix-blend-mode: screen;
  }
 </style>
</head>
<body>
 <div class="text-container">
  Flashlight Text
 </div>
 <div class="flashlight" id="flashlight"></div>

 <script>
  const flashlight = document.getElementById('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>
```

Output:



4) Minion Eye

Code:

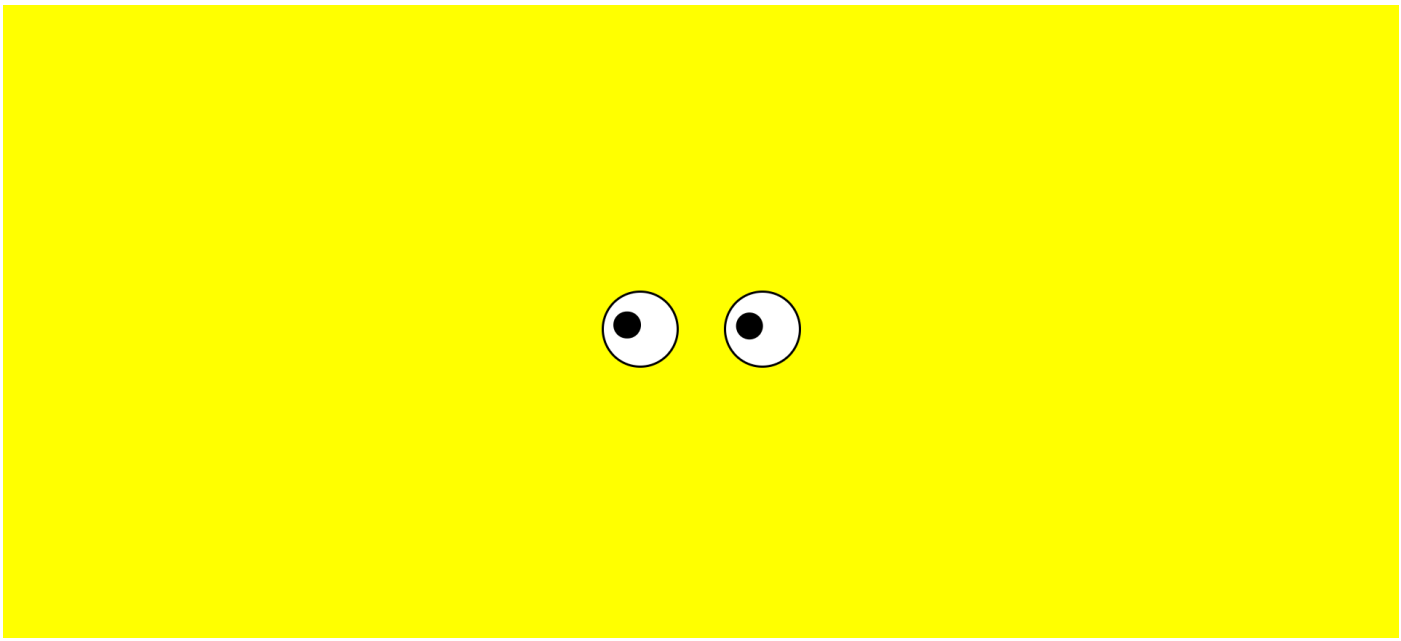
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Eye Tracking</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: yellow;
      overflow: hidden;
    }
    .eye-container {
      display: flex;
      gap: 50px;
    }
    .eye {
      width: 80px;
      height: 80px;
      background: white;
      border: 3px solid black;
      border-radius: 50%;
      display: flex;
      justify-content: center;
      align-items: center;
      position: relative;
    }
```

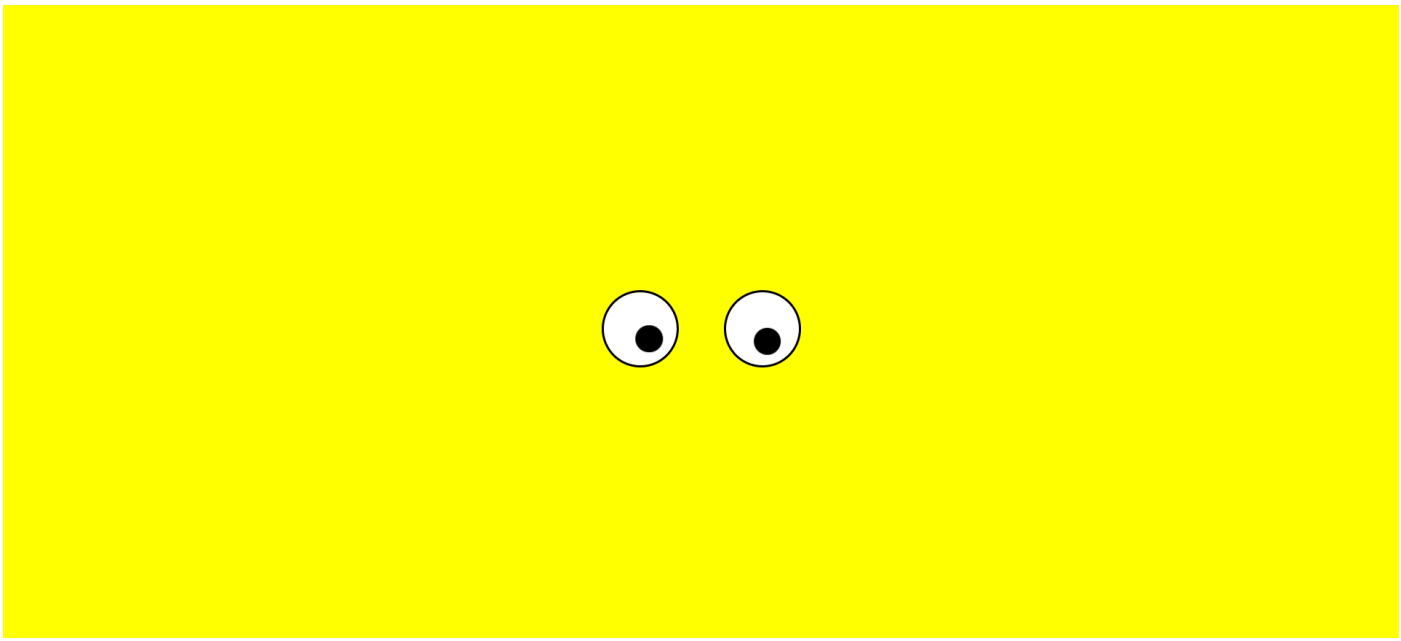
```
      .pupil {
        width: 30px;
        height: 30px;
        background: black;
        border-radius: 50%;
        position: absolute;
        transition: transform 0.05s;
      }
    </style>
</head>
<body>
    <div class="eye-container">
      <div class="eye"><div class="pupil"></div></div>
      <div class="eye"><div class="pupil"></div></div>
    </div>

    <script>
      document.addEventListener("mousemove", (event) => {
        const eyes = document.querySelectorAll(".eye");
        eyes.forEach(eye => {
          const pupil = eye.querySelector(".pupil");
          const rect = eye.getBoundingClientRect();
          const eyeX = rect.left + rect.width / 2;
          const eyeY = rect.top + rect.height / 2;
          const deltaX = event.clientX - eyeX;
          const deltaY = event.clientY - eyeY;
          const angle = Math.atan2(deltaY, deltaX);
          const distance = Math.min(15, Math.sqrt(deltaX**2 + deltaY**2) / 8);
          pupil.style.transform = `translate(${Math.cos(angle) * distance}px, ${Math.sin(angle) *
distance}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>Vertical Image Slider</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f0f0f0;
      flex-direction: column;
    }
    .slider-container {
      position: relative;
      width: 300px;
      height: 400px;
      overflow: hidden;
    }
    .slider {
      display: flex;
      flex-direction: column;
      transition: transform 0.5s ease-in-out;
    }
    .slider img {
      width: 100%;
      height: 400px;
      object-fit: cover;
```

```
        }
        .btn {
            padding: 10px 20px;
            margin: 10px;
            font-size: 18px;
            cursor: pointer;
            background-color: #3498db;
            color: white;
            border: none;
            border-radius: 5px;
            transition: background 0.3s;
        }
        .btn:hover {
            background-color: #2980b9;
        }
    </style>
</head>
<body>
    <button class="btn" onclick="prevSlide()">Previous</button>
    <div class="slider-container">
        <div class="slider" id="slider">
            <img src="cat1.png" alt="Image 1">
            <img src="cat2.png" alt="Image 2">
            <img src="cat3.png" alt="Image 3">
        </div>
    </div>
    <button class="btn" onclick="nextSlide()">Next</button>

    <script>
        let currentIndex = 0;
        const images = document.querySelectorAll(".slider img");
        const slider = document.getElementById("slider");
```

```
    function updateSlider() {
        slider.style.transform = `translateY(-${currentIndex * 400}px)`;
    }

    function nextSlide() {
        if (currentIndex < images.length - 1) {
            currentIndex++;
            updateSlider();
        }
    }

    function prevSlide() {
        if (currentIndex > 0) {
            currentIndex--;
            updateSlider();
        }
    }
  </script>
</body>
</html>
```

Output:

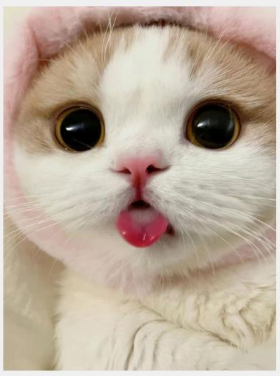
Previous

Next

Previous

Next

Previous



Next

6) Snake Game

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <style>
    /* Updated styling */
    body {
      margin: 0;
      padding: 0;
      font-family: 'Verdana', sans-serif;
      background-color: #2c3e50;
      text-align: center;
      color: #ecf0f1;
    }

    h2 {
       color: #e74c3c;
       margin-top: 20px;
    }

    #msg {
       margin-bottom: 1em;
       font-size: 1.2em;
    }

    #container {
       display: inline-block;
       padding: 15px;
```

```
        background-color: #34495e;
        border: 2px solid #e74c3c;
        border-radius: 8px;
    }

    #gameBoard {
        border: 5px dashed #ecf0f1;
        background-color: #2c3e50;
    }

    #score {
        margin-top: 1em;
        font-size: 1.5em;
    }
  </style>
  <title>Snake Game</title>
</head>
<body>
  <h2>Snake Game</h2>
  <div id="msg">Press space to pause or continue</div>
  <div id="container">
    <canvas id="gameBoard" width="500" height="500"></canvas>
    <div id="score">Score: <span id="scoreVal">0</span></div>
  </div>
  <script>
    const gameBoard = document.getElementById('gameBoard');
    const context = gameBoard.getContext('2d');
    const scoreText = document.getElementById('scoreVal');

    const WIDTH = gameBoard.width;
    const HEIGHT = gameBoard.height;
    const UNIT = 25;
```

```
let foodX;
let foodY;
let xVel = 25;
let yVel = 0;
let score = 0;
let active = true;
let started = false;
let paused = false;

let snake = [
   { x: UNIT * 3, y: 0 },
   { x: UNIT * 2, y: 0 },
   { x: UNIT, y: 0 },
   { x: 0, y: 0 }
];

window.addEventListener('keydown', keyPress);
startGame();

function startGame() {
   context.fillStyle = '#212121';
   context.fillRect(0, 0, WIDTH, HEIGHT);
   createFood();
   displayFood();
   drawSnake();
}

function clearBoard() {
   context.fillStyle = '#212121';
   context.fillRect(0, 0, WIDTH, HEIGHT);
}
```

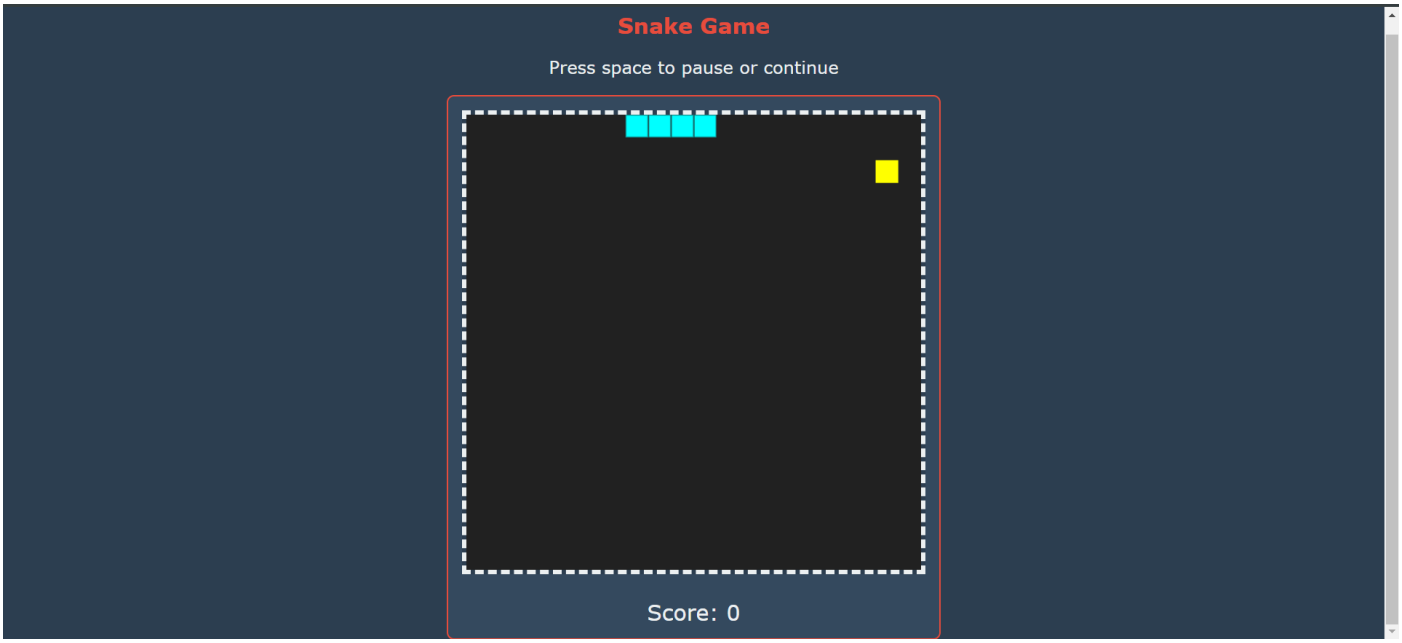
```
function createFood() {
    foodX = Math.floor(Math.random() * WIDTH / UNIT) * UNIT;
    foodY = Math.floor(Math.random() * HEIGHT / UNIT) * UNIT;
}

function displayFood() {
    context.fillStyle = 'yellow';
    context.fillRect(foodX, foodY, UNIT, UNIT);
}

function drawSnake() {
    context.fillStyle = 'aqua';
    context.strokeStyle = '#212121';
    snake.forEach((snakePart) => {
        context.fillRect(snakePart.x, snakePart.y, UNIT, UNIT);
        context.strokeRect(snakePart.x, snakePart.y, UNIT, UNIT);
    });
}

function moveSnake() {
    const head = { x: snake[0].x + xVel, y: snake[0].y + yVel };
    snake.unshift(head);
    if (snake[0].x == foodX && snake[0].y == foodY) {
        score += 1;
        scoreText.textContent = score;
        createFood();
    } else {
        snake.pop();
    }
}
```

```
function nextTick() {
    if (active && !paused) {
        setTimeout(() => {
            clearBoard();
            displayFood();
            moveSnake();
            drawSnake();
            checkGameOver();
            nextTick();
        }, 100);
    } else if (!active) {
        clearBoard();
        context.font = "bold 50px serif";
        context.fillStyle = "white";
        context.textAlign = "center";
        context.fillText("Game Over!!", WIDTH / 2, HEIGHT / 2);
    }
}

function keyPress(event) {
    if (!started) {
        started = true;
        nextTick();
    }

    if (event.keyCode == 32) {
        if (paused) {
            paused = false;
            nextTick();
        } else {
            paused = true;
        }
```

```
    }

    const LEFT = 37;
    const UP = 38;
    const RIGHT = 39;
    const DOWN = 40;

    switch (true) {
        case (event.keyCode == LEFT && xVel != UNIT):
            xVel = -UNIT;
            yVel = 0;
            break;
        case (event.keyCode == RIGHT && xVel != -UNIT):
            xVel = UNIT;
            yVel = 0;
            break;
        case (event.keyCode == UP && yVel != UNIT):
            xVel = 0;
            yVel = -UNIT;
            break;
        case (event.keyCode == DOWN && yVel != -UNIT):
            xVel = 0;
            yVel = UNIT;
            break;
    }
}

function checkGameOver() {
    if (snake[0].x < 0 || snake[0].x >= WIDTH || snake[0].y < 0 || snake[0].y >= HEIGHT) {
        active = false;
    }
```

```
        for (let i = 1; i < snake.length; i++) {
            if (snake[i].x === snake[0].x && snake[i].y === snake[0].y) {
                active = false;
            }
        }
    }
  </script>
</body>
</html>
```

Output:



7) Accessing web-cam with snapshot or redording

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Webcam Access</title>
 <style>
  /* New styling */
  * {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
   font-family: 'Roboto', sans-serif;
  }

  body {
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: center;
   min-height: 100vh;
   background-color: #121212;
   color: #e0e0e0;
  }

  .container {
   text-align: center;
   background-color: #1e1e1e;
   padding: 40px;
   border-radius: 12px;
```

```
  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
  max-width: 600px;
  width: 90%;
  margin: 20px;
}

h1 {
  margin-bottom: 20px;
  color: #ff9800;
}

video, canvas {
  display: block;
  margin: 20px auto;
  border: 3px solid #ff9800;
  border-radius: 10px;
  max-width: 100%;
  width: 100%;
}

button {
  padding: 10px 20px;
  margin: 10px;
  font-size: 16px;
  border: none;
  border-radius: 6px;
  background-color: #ff9800;
  color: #121212;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s;
}
```

```
    button:hover {
      background-color: #fb8c00;
      transform: scale(1.05);
    }

    button:disabled {
      background-color: #555;
      cursor: not-allowed;
    }

    #screenshot {
      display: none;
      margin: 20px auto;
      border: 3px solid #ff9800;
      border-radius: 10px;
      max-width: 100%;
    }

    #downloadLink {
      display: none;
      margin-top: 15px;
      color: #ff9800;
      text-decoration: none;
      font-size: 16px;
    }

    #downloadLink:hover {
      text-decoration: underline;
    }
  </style>
</head>
<body>
```

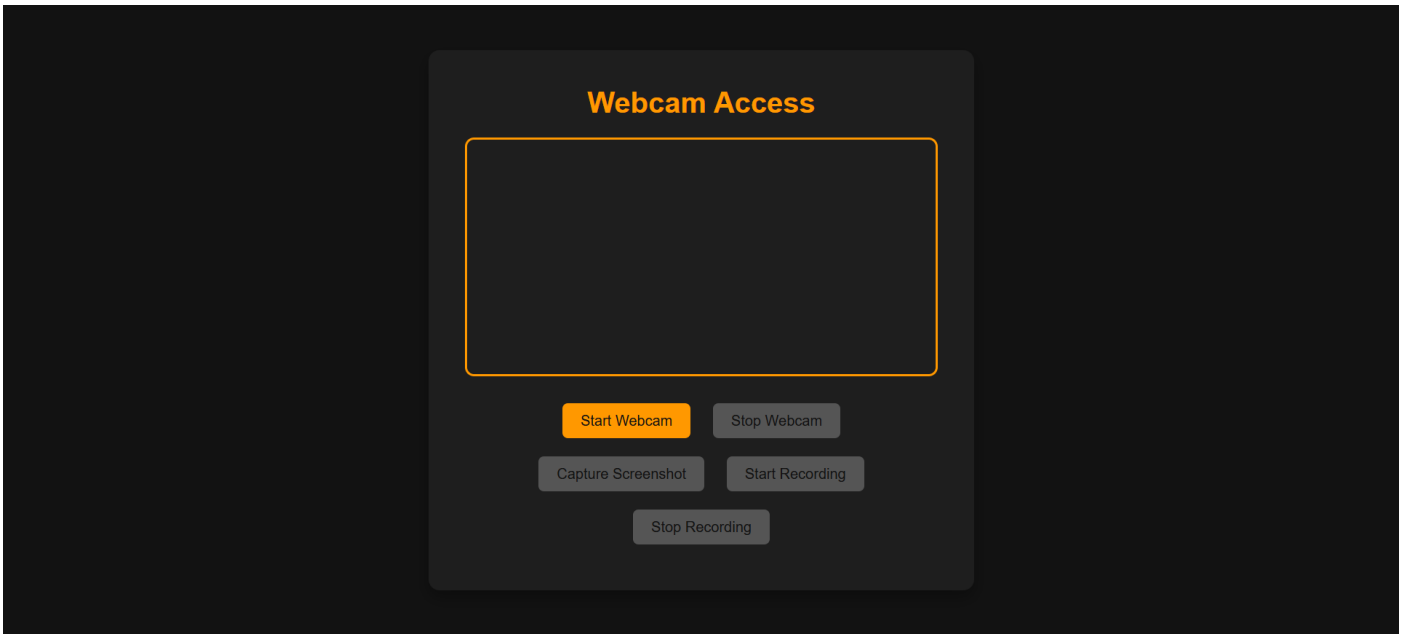
```
<div class="container">
  <h1>Webcam Access</h1>
  <video id="webcam" autoplay></video>
  <canvas id="canvas" style="display: none;"></canvas>
  <img id="screenshot" alt="Screenshot">
  <a id="downloadLink" download="recording.webm">Download Recording</a>

  <div>
    <button id="startBtn">Start Webcam</button>
    <button id="stopBtn" disabled>Stop Webcam</button>
    <button id="captureBtn" disabled>Capture Screenshot</button>
    <button id="startRecordBtn" disabled>Start Recording</button>
    <button id="stopRecordBtn" disabled>Stop Recording</button>
  </div>
</div>

<script>
  const video = document.getElementById("webcam");
  const canvas = document.getElementById("canvas");
  const screenshotImg = document.getElementById("screenshot");
  const startBtn = document.getElementById("startBtn");
  const stopBtn = document.getElementById("stopBtn");
  const captureBtn = document.getElementById("captureBtn");
  const startRecordBtn = document.getElementById("startRecordBtn");
  const stopRecordBtn = document.getElementById("stopRecordBtn");
  const downloadLink = document.getElementById("downloadLink");

  let stream = null;
  let mediaRecorder = null;
  let recordedChunks = [];

  startBtn.addEventListener("click", async () => {
```

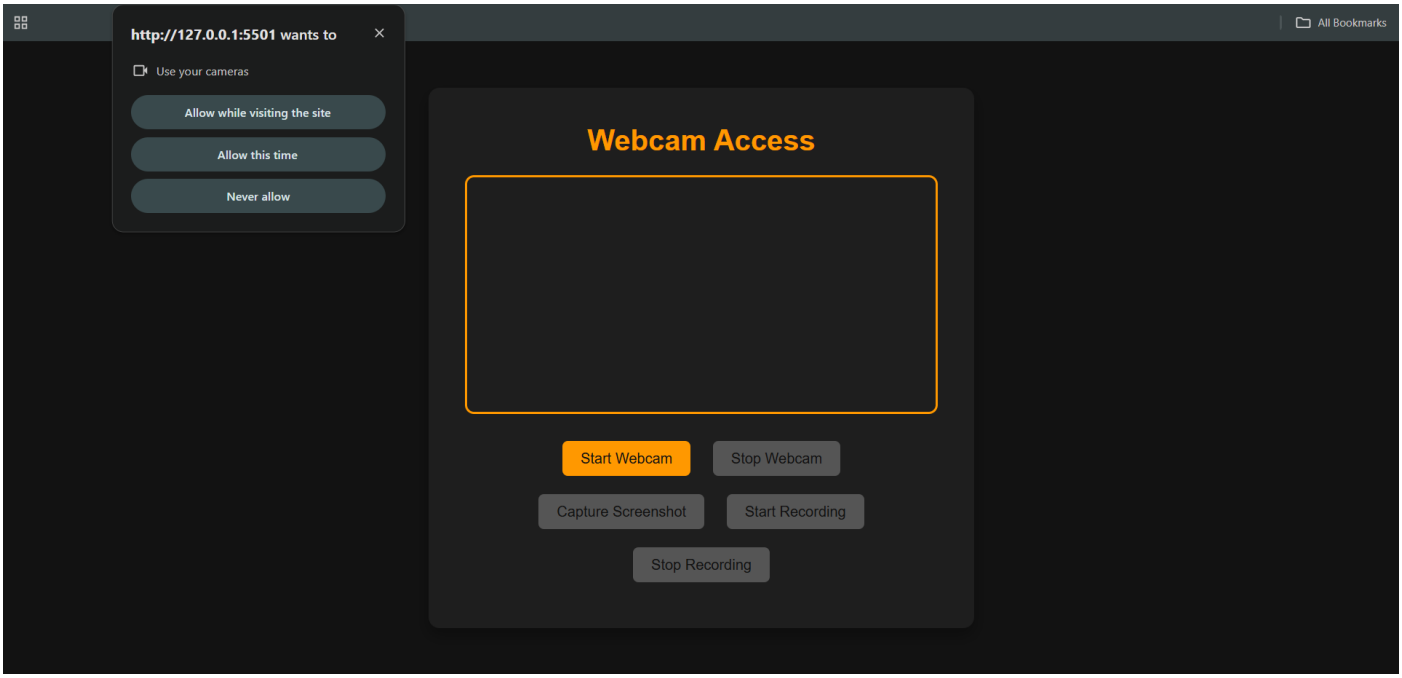
```
  try {
    stream = await navigator.mediaDevices.getUserMedia({ video: true });
    video.srcObject = stream;

    startBtn.disabled = true;
    stopBtn.disabled = false;
    captureBtn.disabled = false;
    startRecordBtn.disabled = false;
  } catch (error) {
    console.error("Error accessing webcam:", error);
    alert("Could not access webcam. Please check browser settings.");
  }
});

stopBtn.addEventListener("click", () => {
  if (stream) {
    let tracks = stream.getTracks();
    tracks.forEach(track => track.stop());
    video.srcObject = null;
  }

  startBtn.disabled = false;
  stopBtn.disabled = true;
  captureBtn.disabled = true;
  startRecordBtn.disabled = true;
  stopRecordBtn.disabled = true;
});

captureBtn.addEventListener("click", () => {
  const context = canvas.getContext("2d");
  canvas.width = video.videoWidth;
  canvas.height = video.videoHeight;
```

```
  context.drawImage(video, 0, 0, canvas.width, canvas.height);

  screenshotImg.src = canvas.toDataURL("image/png");
  screenshotImg.style.display = "block";

  const link = document.createElement("a");
  link.href = screenshotImg.src;
  link.download = "screenshot.png";
  link.click();
});

startRecordBtn.addEventListener("click", () => {
  recordedChunks = [];
  mediaRecorder = new MediaRecorder(stream);

  mediaRecorder.ondataavailable = (event) => {
    if (event.data.size > 0) {
      recordedChunks.push(event.data);
    }
  };

  mediaRecorder.onstop = () => {
    const blob = new Blob(recordedChunks, { type: "video/webm" });
    const videoURL = URL.createObjectURL(blob);

    downloadLink.href = videoURL;
    downloadLink.style.display = "block";
  };

  mediaRecorder.start();
  startRecordBtn.disabled = true;
  stopRecordBtn.disabled = false;
```

```
  });

  stopRecordBtn.addEventListener("click", () => {
    if (mediaRecorder && mediaRecorder.state !== "inactive") {
      mediaRecorder.stop();
    }
  });
 </script>
</body>
</html>
```

Output:




http://127.0.0.1:5501 wants to
Use your cameras
Allow while visiting the site
Allow this time
Never allow
All Bookmarks
Webcam Access
Start Webcam
Stop Webcam
Capture Screenshot
Start Recording
Stop Recording

8) Mobile Flashlight

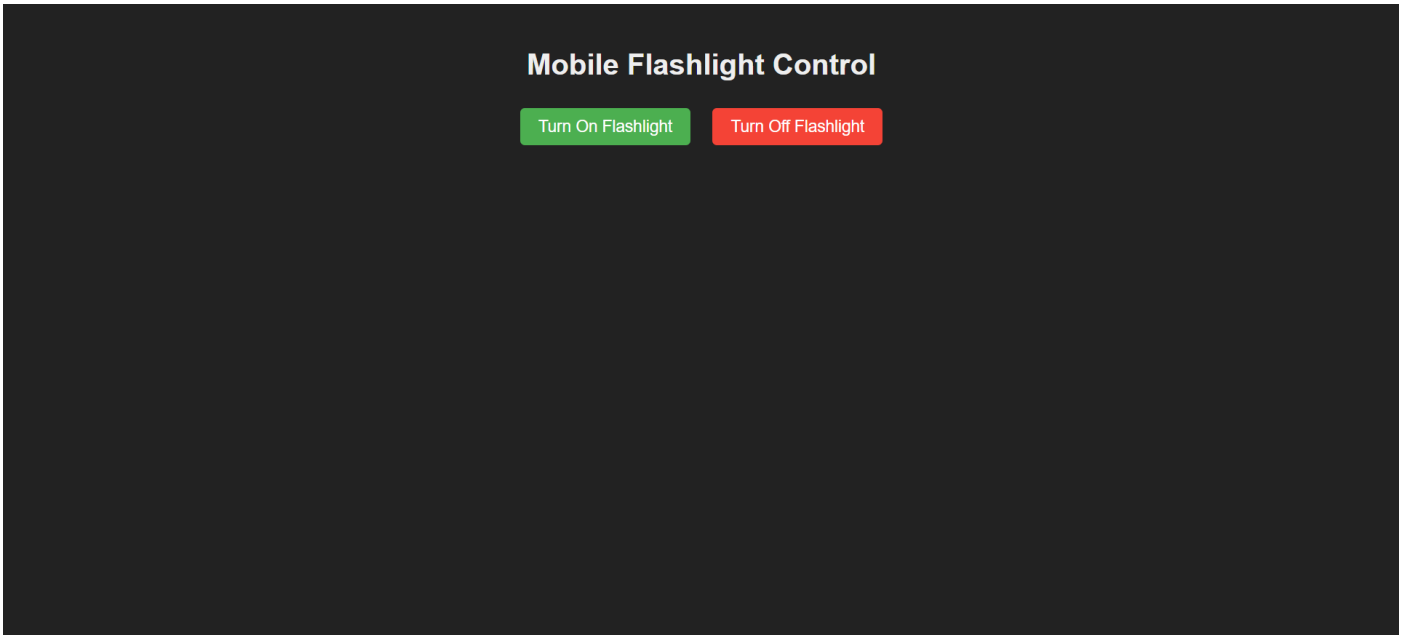
Code:

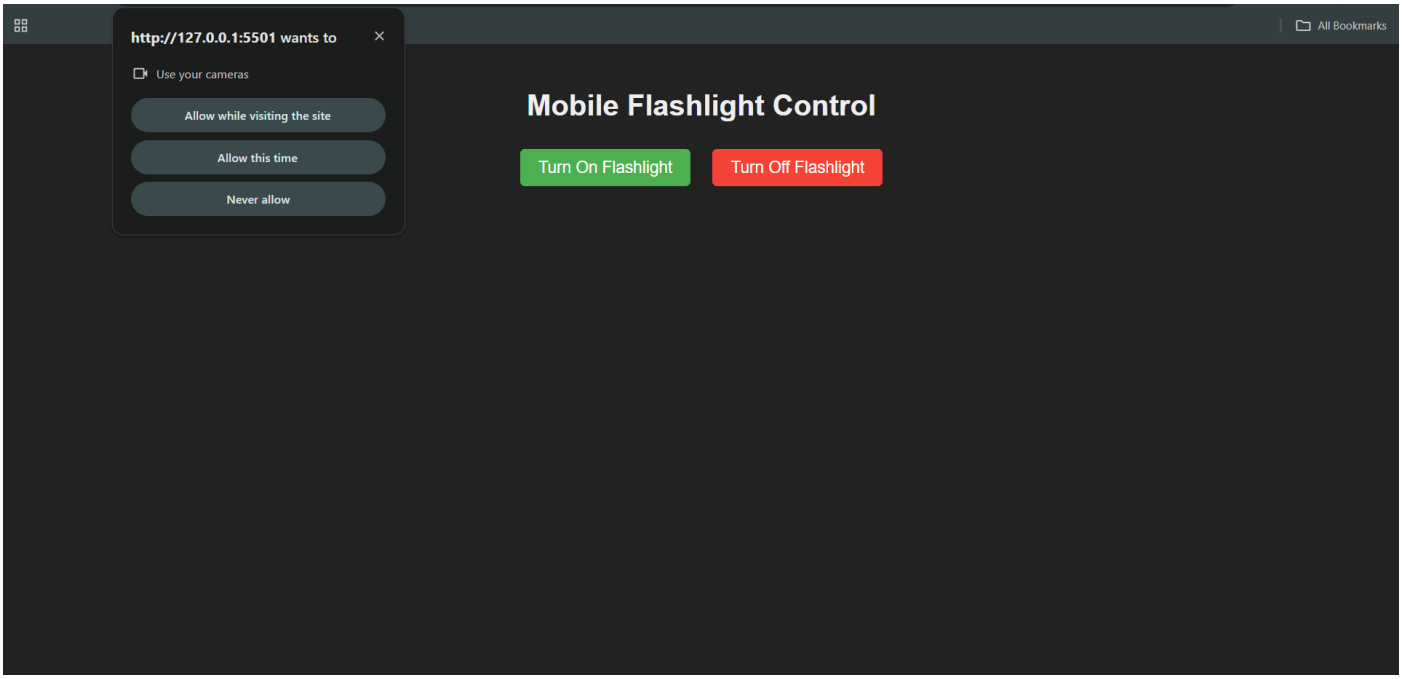
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <title>Mobile Flashlight</title>
 <style>
  body {
   font-family: sans-serif;
   text-align: center;
   background-color: #222;
   color: #eee;
   padding: 20px;
  }
  h1 {
   margin-bottom: 20px;
  }
  button {
   padding: 10px 20px;
   font-size: 18px;
   margin: 10px;
   border: none;
   border-radius: 5px;
   cursor: pointer;
  }
  #turnOn {
   background-color: #4CAF50;
   color: white;
  }
  #turnOff {
```

```
    background-color: #f44336;
    color: white;
   }
  </style>
</head>
<body>
  <h1>Mobile Flashlight Control</h1>
  <button id="turnOn">Turn On Flashlight</button>
  <button id="turnOff" disabled>Turn Off Flashlight</button>

  <script>
   let stream;
   let track;

   const turnOnBtn = document.getElementById('turnOn');
   const turnOffBtn = document.getElementById('turnOff');

   turnOnBtn.addEventListener('click', async () => {
    try {
     stream = await navigator.mediaDevices.getUserMedia({
      video: { facingMode: { ideal: 'environment' } }
     });
     track = stream.getVideoTracks()[0];
     const capabilities = track.getCapabilities();
     if ('torch' in capabilities) {
      // Turn on the torch
      await track.applyConstraints({
       advanced: [{ torch: true }]
      });
      turnOnBtn.disabled = true;
      turnOffBtn.disabled = false;
     } else {
```

```
        alert('Torch not supported on this device.');
        stream.getTracks().forEach(track => track.stop());
      }
    } catch (error) {
      console.error("Error accessing camera or turning on torch:", error);
      alert("Could not access camera or enable torch. Please check permissions and compatibility.");
    }
  });

  turnOffBtn.addEventListener('click', async () => {
    try {
      if (track) {
        await track.applyConstraints({
          advanced: [{ torch: false }]
        });
      }
    } catch (error) {
      console.error("Error turning off torch:", error);
    }
    if (stream) {
      stream.getTracks().forEach(track => track.stop());
    }
    turnOnBtn.disabled = false;
    turnOffBtn.disabled = true;
  });
 </script>
</body>
</html>
```

Output:


Mobile Flashlight Control
Turn On Flashlight
Turn Off Flashlight


http://127.0.0.1:5501 wants to
Use your cameras
Allow while visiting the site
Allow this time
Never allow
All Bookmarks
Mobile Flashlight Control
Turn On Flashlight
Turn Off Flashlight