3/2/2025

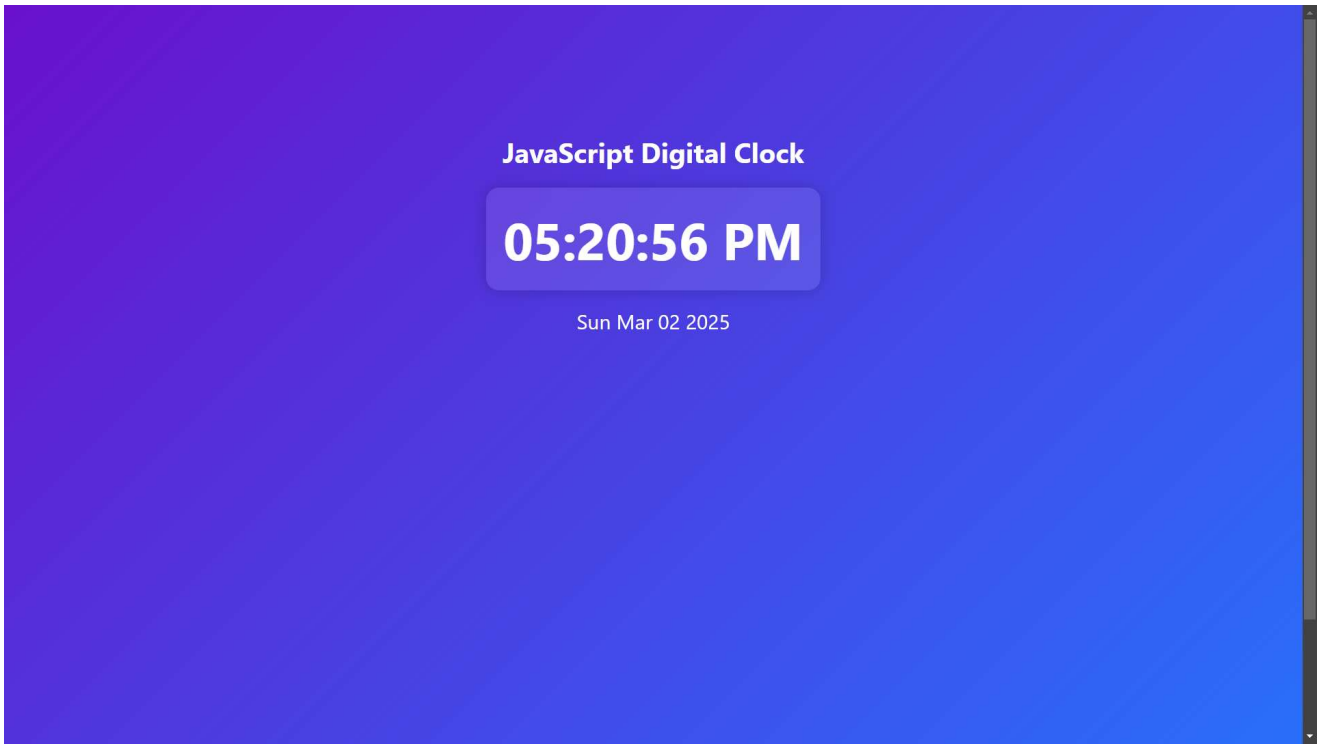
Web Programming

Aman Kumar Singh
23BPS1011

1. Digital Clock

```
<!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: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            text-align: center;
            margin-top: 150px;
            height: 100vh;
            background: linear-gradient(135deg, #6a11cb, #2575fc);
            color: white;
        }
        #clock {
            font-size: 60px;
            font-weight: bold;
            background: rgba(255, 255, 255, 0.1);
            padding: 20px;
            border-radius: 15px;
            display: inline-block;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.2);
        }
        #date {
            font-size: 24px;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1>JavaScript Digital Clock</h1>
    <div id="clock">00:00:00</div>
    <div id="date">Loading date...</div>

    <script>
        function updateClock() {
            const now = new Date();
            let hours = now.getHours();
            let minutes = now.getMinutes();
            let seconds = now.getSeconds();
            const ampm = hours >= 12 ? 'PM' : 'AM';

            hours = hours % 12;
            hours = hours ? hours : 12;

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

            const timeString = hours + ":" + minutes + ":" + seconds + " " + ampm;
```

```
            document.getElementById("clock").innerText = timeString;

            const dateString = now.toDateString();
            document.getElementById("date").innerText = dateString;
        }

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



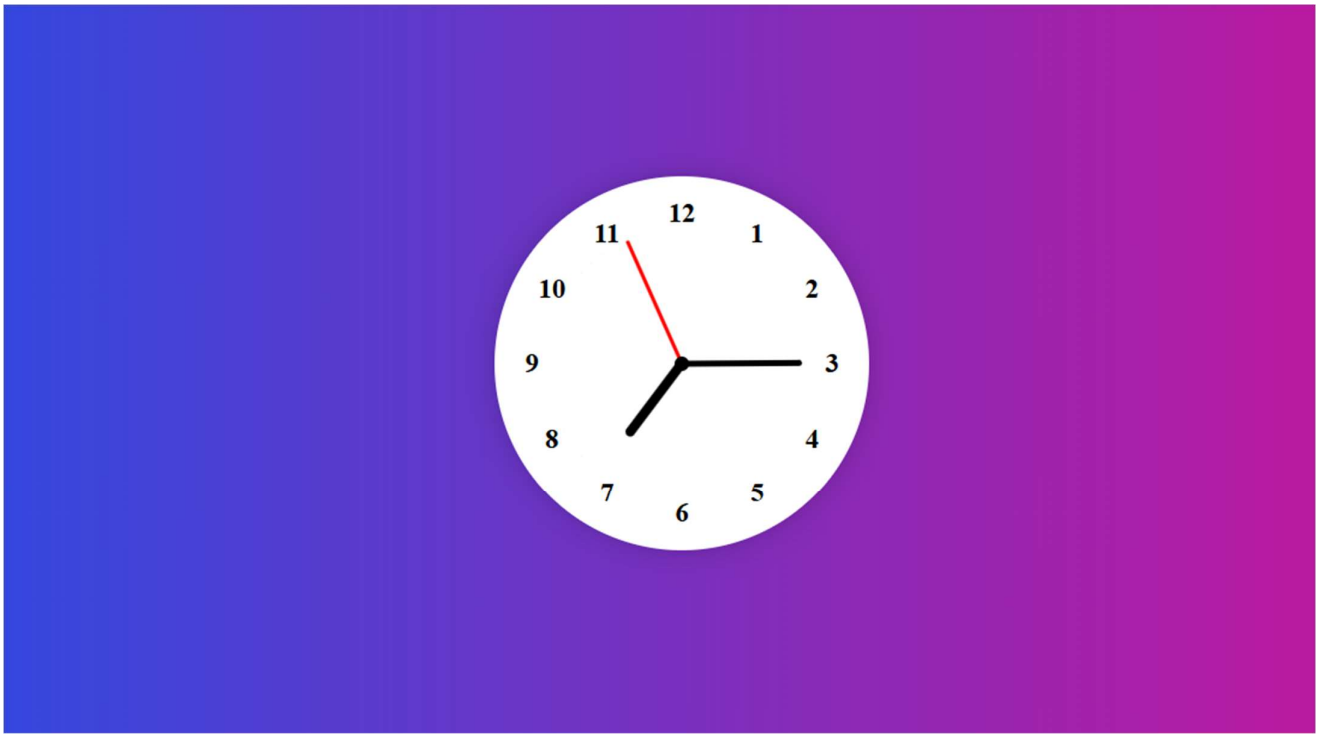
2. Analog Clock

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

```
            justify-content: center;
            align-items: center;
            height: 100vh;
            background: linear-gradient(to right, #0458f4, #f00888);
        }
        .clock {
            width: 250px;
            height: 250px;
            background: white;
            border-radius: 50%;
            position: relative;
            display: flex;
            justify-content: center;
            align-items: center;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.2);
        }
        .clock::before {
            content: "";
            width: 10px;
            height: 10px;
            background: black;
            border-radius: 50%;
            position: absolute;
            z-index: 10;
        }
        .hand {
            position: absolute;
            bottom: 50%;
            left: 50%;
            transform-origin: bottom;
            transform: translateX(-50%) rotate(0deg);
            border-radius: 5px;
            transition: transform 0.5s ease-in-out;
        }
        .hour {
            width: 6px;
            height: 60px;
            background: black;
        }
        .minute {
            width: 4px;
            height: 80px;
            background: black;
        }
        .second {
            width: 2px;
            height: 90px;
            background: red;
        }
        .numbers {
            position: absolute;
            width: 100%;
            height: 100%;
            font-size: 18px;
```

```
            font-weight: bold;
            color: black;
        }
        .number {
            position: absolute;
            transform: translate(-50%, -50%);
        }
    </style>
</head>
<body>
    <div class="clock">
        <div class="numbers" id="numbers"></div>
        <div class="hand hour" id="hour"></div>
        <div class="hand minute" id="minute"></div>
        <div class="hand second" id="second"></div>
    </div>

    <script>
        function createClockNumbers() {
            const clock = document.querySelector(".numbers");

            for (let i = 1; i <= 12; i++) {
                const number = document.createElement("div");
                number.classList.add("number");
                number.textContent = i;

                const angle = (i * 30) * Math.PI / 180;
                const radius = 100;
                const x = Math.sin(angle) * radius + 125;
                const y = -Math.cos(angle) * radius + 125;

                number.style.left = `${x}px`;
                number.style.top = `${y}px`;

                clock.appendChild(number);
            }
        }

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

            const hourDeg = (hours + 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)`;
```

```
        }

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



3. Flashlight Text

```
<!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;
    }
```

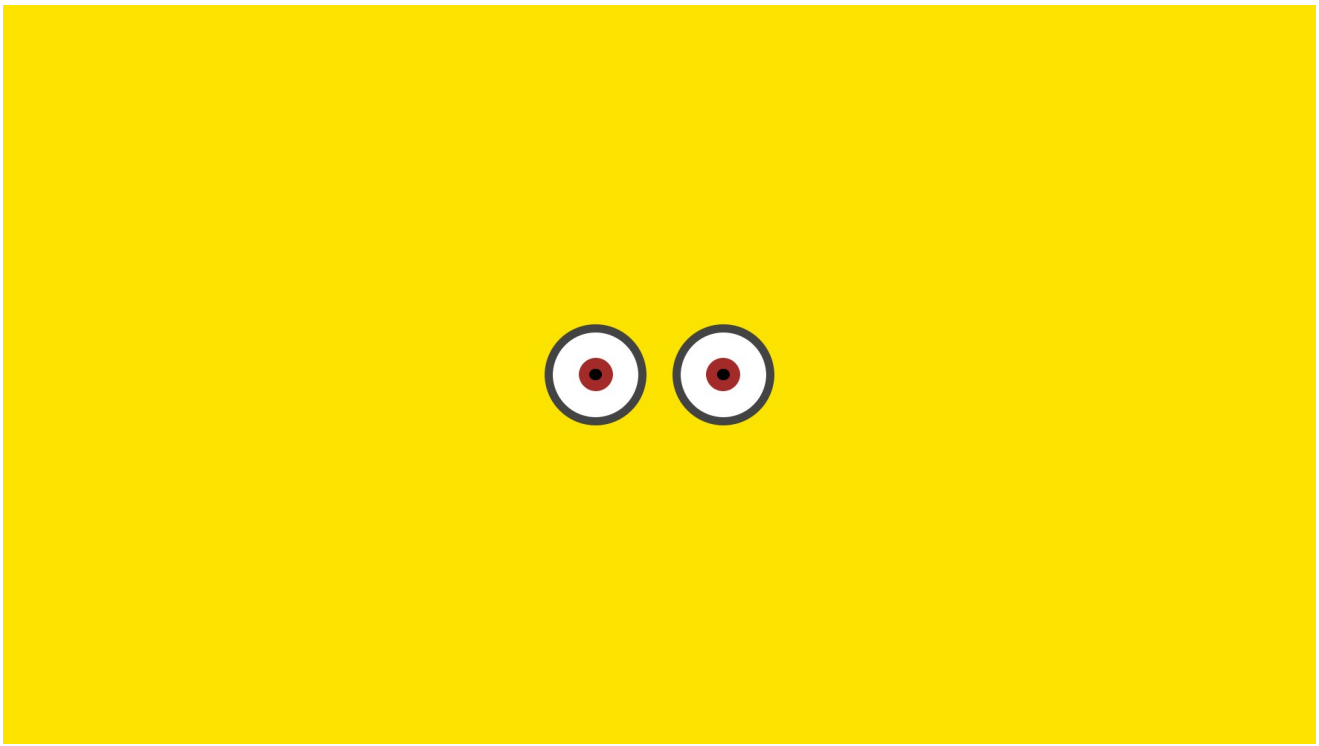
```
    .text-container {
      position: relative;
      font-size: 5rem;
      font-weight: bold;
      color: rgba(255, 255, 255, 0.1);
      background: linear-gradient(90deg, #ff00ff, #00ffff, #ffff00, #ff00ff);
      background-clip: text;
      -webkit-background-clip: text;
      -webkit-text-fill-color: transparent;
    }

    .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>
```

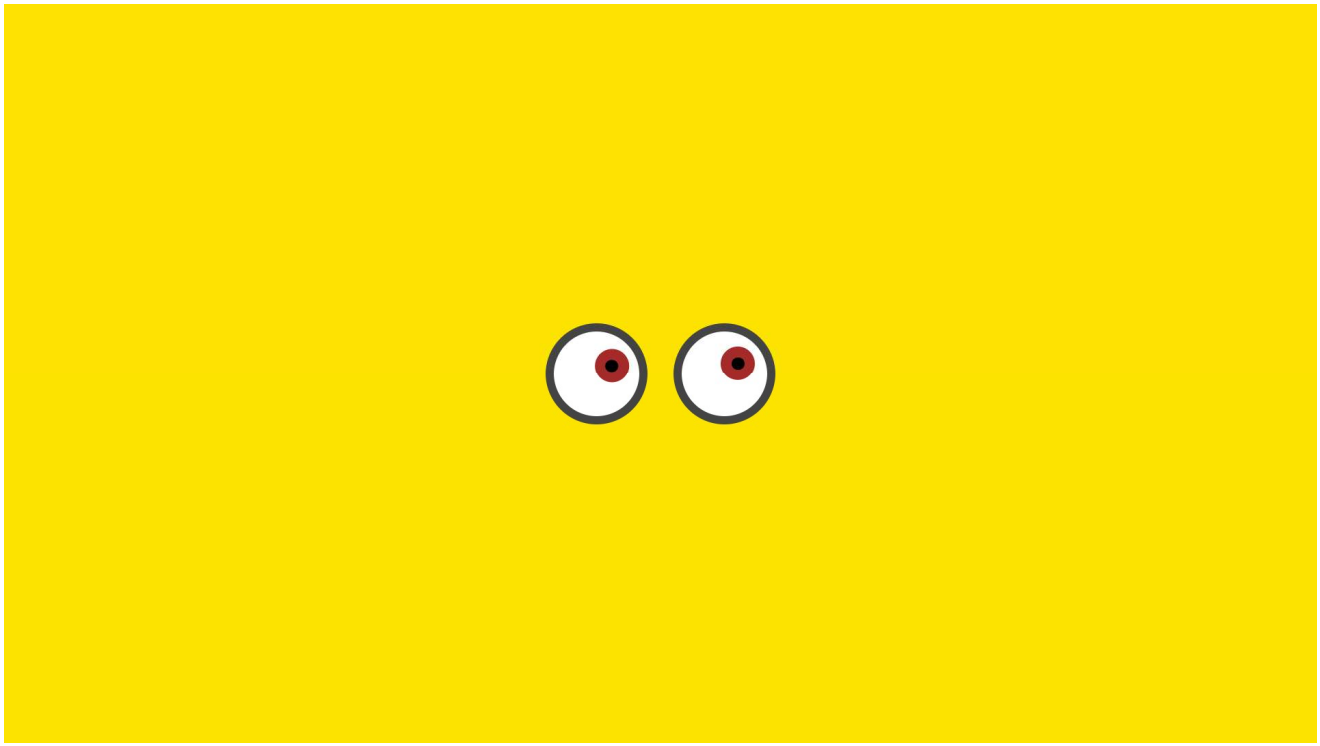
4. Minion Eyes

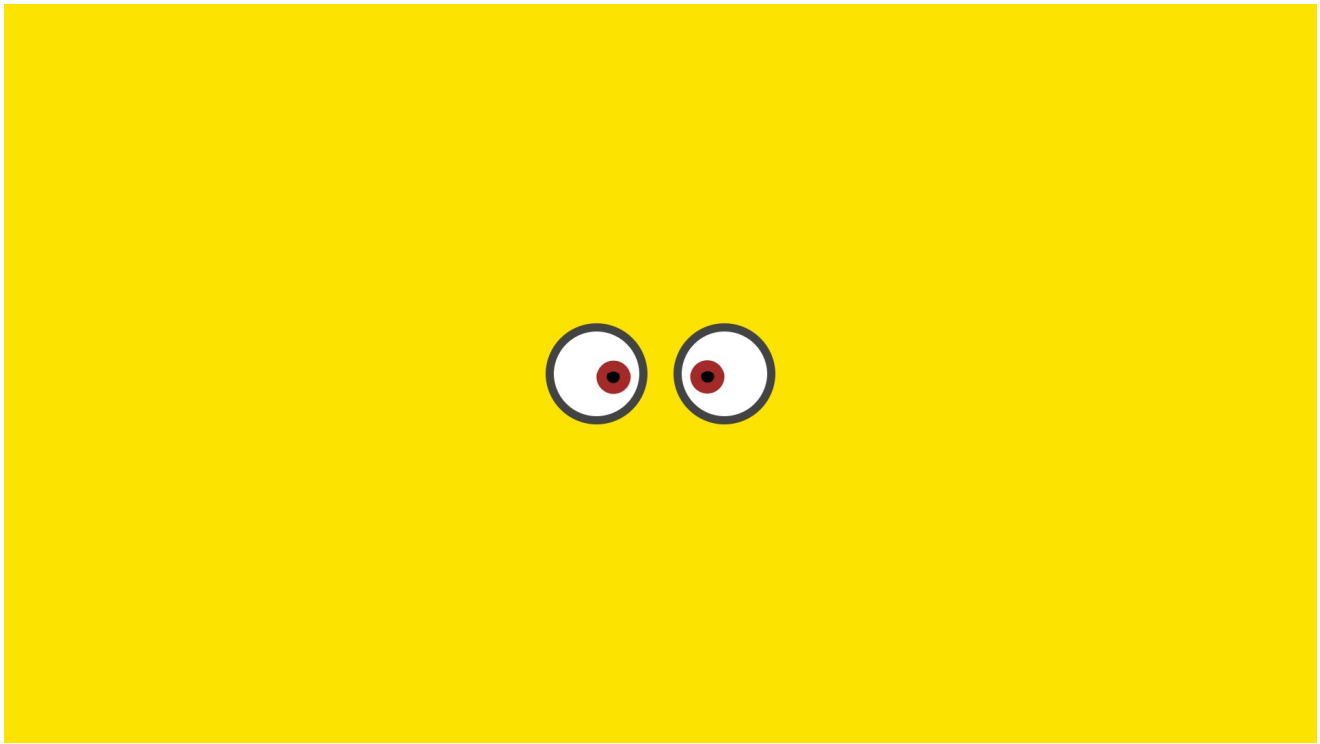
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Moving Eyes</title>
    <style>
        body {
            background-color: #FCE300;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }

        .eye-container {
            display: flex;
            gap: 30px;
        }

        .eye {
            width: 100px;
            height: 100px;
            background-color: white;
            border-radius: 50%;
```

```
            position: relative;
            display: flex;
            justify-content: center;
            align-items: center;
            border: 10px solid #444;
        }

        .pupil {
            width: 40px;
            height: 40px;
            background-color: brown;
            border-radius: 50%;
            position: relative;
            display: flex;
            justify-content: center;
            align-items: center;
        }

        .inner-pupil {
            width: 15px;
            height: 15px;
            background-color: black;
            border-radius: 50%;
            position: absolute;
        }
    </style>
</head>

<body>
    <div class="eye-container">
        <div class="eye">
            <div class="pupil">
                <div class="inner-pupil"></div>
            </div>
        </div>
        <div class="eye">
            <div class="pupil">
                <div class="inner-pupil"></div>
            </div>
        </div>
    </div>
    <script>
        const eyes = document.querySelectorAll(".eye");
        const pupils = document.querySelectorAll(".pupil");

        document.addEventListener("mousemove", (event) => {
            const { clientX: mouseX, clientY: mouseY } = event;

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

                const deltaX = mouseX - eyeCenterX;
```

```
                const deltaY = mouseY - eyeCenterY;
                const angle = Math.atan2(deltaY, deltaX);

                const maxMove = 20;
                const pupilX = Math.cos(angle) * maxMove;
                const pupilY = Math.sin(angle) * maxMove;

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

</html>
```







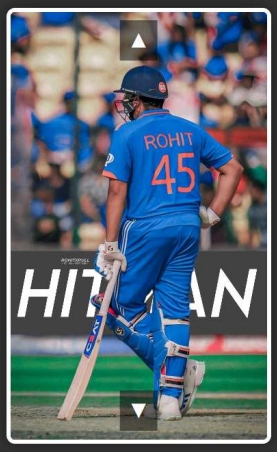
5. Vertical Image Slider

```
<!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;
    margin: 0;
    background-color: #333;
    font-family: Arial, sans-serif;
  }

  .slider-container {
    position: relative;
    width: 300px;
    height: 500px;
    overflow: hidden;
    border: 5px solid #fff;
    border-radius: 10px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
  }

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

  .slider img {
    width: 100%;
    height: 500px;
    object-fit: cover;
  }

  .nav-button {
    position: absolute;
    left: 50%;
    transform: translateX(-50%);
    background-color: rgba(0, 0, 0, 0.5);
    color: #fff;
    border: none;
    padding: 10px;
    cursor: pointer;
    font-size: 24px;
    z-index: 10;
  }

  .nav-button.prev {
    top: 10px;
  }

  .nav-button.next {
    bottom: 10px;
  }
```

```
    .nav-button:hover {
      background-color: rgba(0, 0, 0, 0.8);
    }
  </style>
</head>
<body>
  <div class="slider-container">
    <div class="slider" id="slider">
      <img src="https://wallpaperaccess.com/full/10222347.jpg" alt="Image 1">
      <img src="https://wallpaperaccess.com/full/14771823.jpg" alt="Image 2">
      <img src="https://wallpaperaccess.com/full/13548255.jpg" alt="Image 3">
      <img src="https://wallpaperaccess.com/full/13394977.jpg" alt="Image 4">
    </div>
    <button class="nav-button prev" id="prevBtn">&#9650;</button>
    <button class="nav-button next" id="nextBtn">&#9660;</button>
  </div>

  <script>
    const slider = document.getElementById('slider');
    const prevBtn = document.getElementById('prevBtn');
    const nextBtn = document.getElementById('nextBtn');
    let currentIndex = 0;

    function moveSlider(direction) {
      const totalImages = slider.children.length;
      if (direction === 'next') {
        currentIndex = (currentIndex + 1) % totalImages;
      } else if (direction === 'prev') {
        currentIndex = (currentIndex - 1 + totalImages) % totalImages;
      }
      slider.style.transform = `translateY(-${currentIndex * 500}px)`;
    }

    prevBtn.addEventListener('click', () => moveSlider('prev'));
    nextBtn.addEventListener('click', () => moveSlider('next'));

    let startY = 0;

    slider.addEventListener('touchstart', (e) => {
      startY = e.touches[0].clientY;
    });

    slider.addEventListener('touchmove', (e) => {
      e.preventDefault();
    });

    slider.addEventListener('touchend', (e) => {
      const endY = e.changedTouches[0].clientY;
      const deltaY = startY - endY;

      if (deltaY > 50) {
        moveSlider('next');
      } else if (deltaY < -50) {
```

```
        moveSlider('prev');
      }
    });
  </script>
</body>
</html>
```



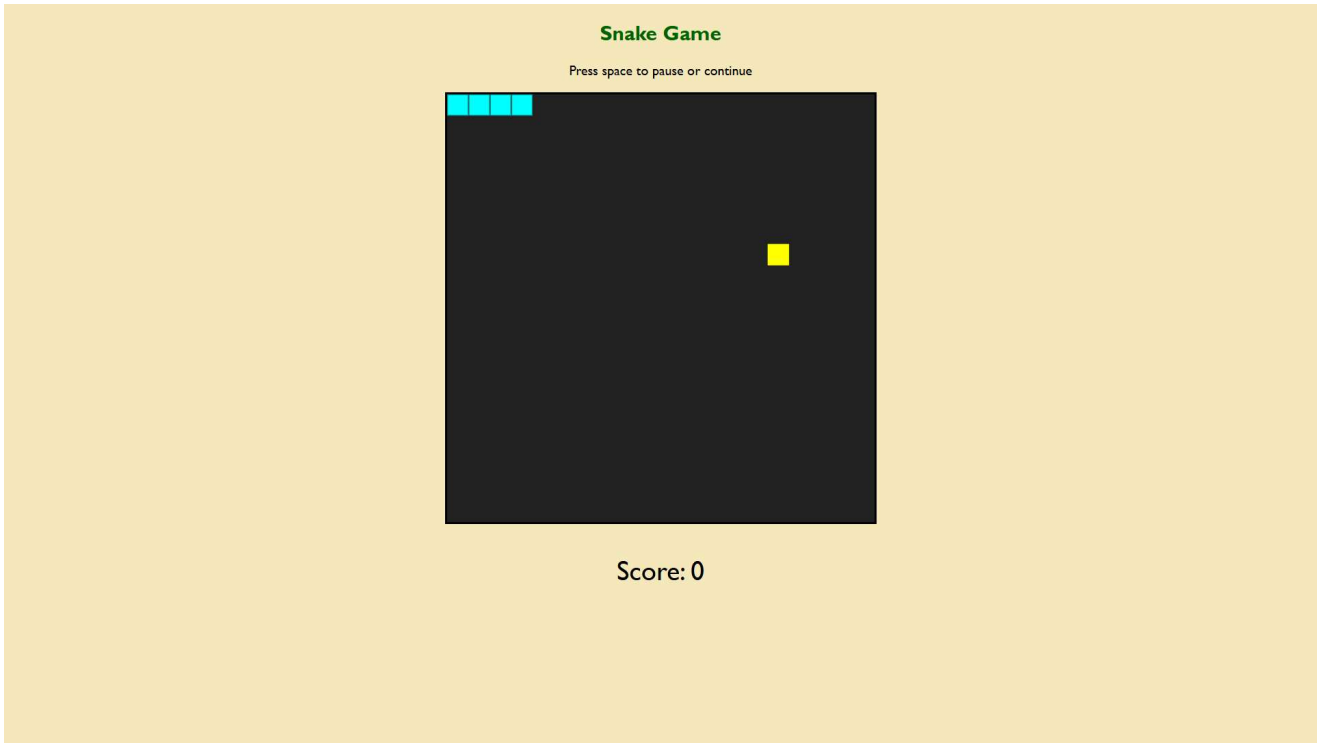
6. Snake Game

```
<!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>
      body {
        margin: 0;
        padding: 0;
        font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
        background-color: #f5e8ba;
        text-align: center;
      }

      h2 {
          color: darkgreen;
      }

      #msg {
          margin-bottom: 1em;
      }

      #gameBoard {
          border: 3px solid;
      }
```

```
      #score {
          margin-top: 1em;
          font-size: 2em;
      }
    </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>
```



7. Accessing webcam with snapshot, recording

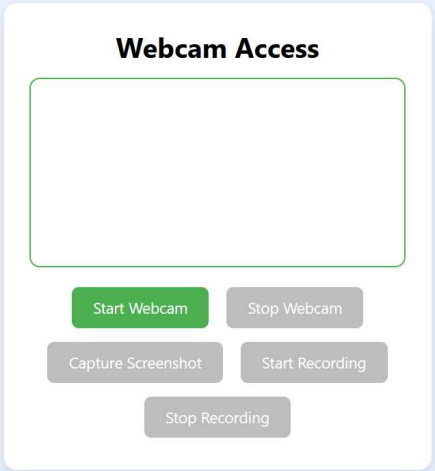
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

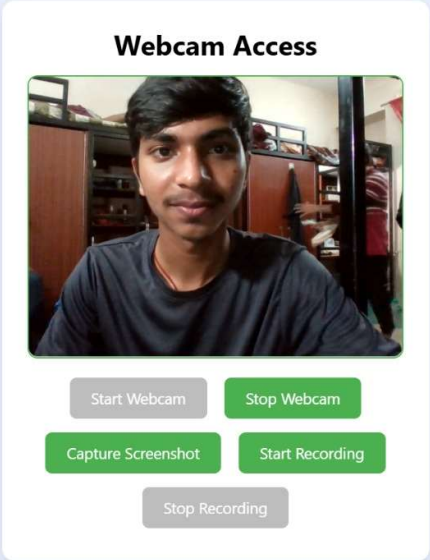
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Webcam Access</title>
<style>
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  }

  body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-color: #e8f0fe;
  }

  .container {
    text-align: center;
    background-color: white;
    padding: 30px;
    border-radius: 15px;
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
    max-width: 500px;
    width: 100%;
  }

  video, canvas {
    display: block;
    margin: 15px auto;
    border: 2px solid #4caf50;
    border-radius: 12px;
    max-width: 100%;
    width: 100%;
  }

  button {
    padding: 12px 25px;
    margin: 8px;
    font-size: 18px;
    border: none;
    border-radius: 8px;
    background-color: #4caf50;
    color: white;
    cursor: pointer;
    transition: background-color 0.3s ease;
  }

  button:hover {
    background-color: #45a049;
  }
```

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

    #screenshot {
      display: none;
      margin: 15px auto;
      border: 2px solid #4caf50;
      border-radius: 12px;
      max-width: 100%;
    }

    #downloadLink {
      display: none;
      margin-top: 10px;
      color: #4caf50;
      text-decoration: none;
    }

    #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>
```



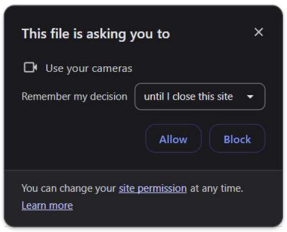


8. Flashlight

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flashlight Control</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        button {
            padding: 10px 20px;
            font-size: 16px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <button id="toggleFlashlight">Turn On Flashlight</button>
    <script>
      let stream;
      let track;
      let isFlashlightOn = false;
```

```
      async function toggleFlashlight() {
          if (!isFlashlightOn) {
              try {
                  stream = await navigator.mediaDevices.getUserMedia({ video: {
facingMode: "environment" } });
                  track = stream.getVideoTracks()[0];

                  const capabilities = track.getCapabilities();
                  if ("torch" in capabilities) {
                      await track.applyConstraints({ advanced: [{ torch: true }] });
                      document.getElementById("toggleFlashlight").textContent = "Turn Off
Flashlight";
                      isFlashlightOn = true;
                  } else {
                      alert("Your device does not support flashlight control.");
                  }
              } catch (error) {
                  console.error("Error accessing flashlight:", error);
                  alert("Failed to access flashlight.");
              }
          } else {
              track.stop();
              document.getElementById("toggleFlashlight").textContent = "Turn On
Flashlight";
              isFlashlightOn = false;
          }
      }
      document.getElementById("toggleFlashlight").addEventListener("click",
toggleFlashlight);
    </script>
</body>
</html>
```

Turn On Flashlight



Turn On Flashlight