1)DIGITAL CLOCK

1)html

```
<title>DIGITAL CLOCK</title>
<head>
<link rel="stylesheet" href="tutoh.css">
<script src="tubhii.js" defer></script>
</head>
<body>

   <div class="clock">
 00:00:00
</div>
</body>
</html>
```

2)css

```
*{
   margin:0px;
   padding:0px;
}

body{
   font-family:Arial,sans-serif;
   text-align:center;
   margin-top:150px;
   display:flex;
```

```
    align-items:center;
    justify-content:center;

}
.clock{

   font-size:50px;
   color:blue;
    border:2px solid black;
   border-radius:10px;
   box-shadow:10px 10px 10px 10px rgba(0,0,0,0.2);
   width:300px;
   height:100px;
   display:flex;
    align-items:center;
    justify-content:center;

}
```

3)JS

```
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.querySelector(".clock").innerText=hours+":"+minutes+":"+seconds;
}

 setInterval(updateClock,1000);
updateClock();
```

2)analog clock

```
 <!DOCTYPE html>
<title>Analog clock</title>
<head>
   <link rel="stylesheet" href="tutoh.css">
   <script src="tubhii.js" defer></script>
</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>
```

```
</body>


*{
    margin:0px;
    padding:0px;
    box-sizing:border-box;

}
body{
    display:flex;
    justify-content:center;
    align-items:center;
    height:100vh;
    background:linear-gradient(to right,#6b0012,#8f08f0);

}
.clock{
    border:2px solid black;
    width:250px;
    height:250px;
    border-radius:100%;
      position:relative;  /*here if give it absolute positioning it will be placed acc to html ..whichh is the full page so it wont matter if it is positioned absolute or relative */
    display:flex;
    align-items:center;
    justify-content:center;
```

```
}
.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:80px;
    background:red;
}

.numbers{
    position:absolute;
    width:100%;
    height:100%;
    font-size:18px;
    font-weight:bold;
    color:black;

}

.number{
    position:absolute;
```

```
    transform:translate(-50%,-50%);
}

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();
```

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 EYE

```
<!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: #ffda44; /* Yellow background */
        }

        .eye-container {
            display: flex;
            gap: 50px;
        }

        .eye {
            width: 100px;
            height: 100px;
```

```
    background: white;
    border-radius: 50%;
    display: flex;
    justify-content: center;
    align-items: center;
    position: relative;
    border: 10px solid #4d4d4d; /* Gray outer border */
}

.iris {
    width: 50px;
    height: 50px;
    background: #b22222; /* Red iris */
    border-radius: 50%;
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
}

.pupil {
    width: 20px;
    height: 20px;
    background: black;
    border-radius: 50%;
    position: absolute;
```

```
        transition: transform 0.05s linear;
      }
    </style>
</head>
<body>
    <div class="eye-container">
      <div class="eye">
        <div class="iris">
          <div class="pupil"></div>
        </div>
      </div>
      <div class="eye">
        <div class="iris">
          <div class="pupil"></div>
        </div>
      </div>
    </div>

    <script>
      const eyes = document.querySelectorAll(".pupil");
      document.addEventListener("mousemove", (event) => {
        let mouseX = event.clientX;
        let mouseY = event.clientY;
        eyes.forEach((eye) => {
          let eyeRect = eye.parentElement.getBoundingClientRect();
          let eyeX = eyeRect.left + eyeRect.width / 2;
```

```
            let eyeY = eyeRect.top + eyeRect.height / 2;
            let deltaX = mouseX - eyeX;
            let deltaY = mouseY - eyeY;
            let angle = Math.atan2(deltaY, deltaX);
            let distance = Math.min(eyeRect.width / 4, Math.hypot(deltaX, deltaY) / 10);
            let moveX = Math.cos(angle) * distance;
            let moveY = Math.sin(angle) * distance;
            eye.style.transform = translate(${moveX}px, ${moveY}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

```
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 = UNIT;
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();
            if (active) {
                nextTick();
            }
        }, 200);
    } else if (!active) {
        clearBoard();
        context.font = "bold 50px serif";
        context.fillStyle = "white";
        context.textAlign = "center";
        context.fillText("Game Over!!", WIDTH / 2, HEIGHT / 2);
        context.fillText("Press ENTER to Restart", WIDTH / 2, HEIGHT / 2 + 50);
    }
}

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

```
    nextTick();
}

if (event.keyCode === 32) { // Spacebar for pause
    paused = !paused;
    if (!paused) nextTick();
    return;
}

if (!active && event.keyCode === 13) { // Press Enter to restart
    restartGame();
    return;
}

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

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

```
        case DOWN:
            if (yVel !== -UNIT) { xVel = 0; yVel = UNIT; }
            break;
    }
}

function checkGameOver() {
    switch (true) {
        case (snake[0].x < 0):
        case (snake[0].x >= WIDTH):
        case (snake[0].y < 0):
        case (snake[0].y >= HEIGHT):
            active = false;
            break;
    }

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

function restartGame() {
    snake = [
        { x: UNIT * 3, y: 0 },
```

```
        { x: UNIT * 2, y: 0 },
        { x: UNIT, y: 0 },
        { x: 0, y: 0 }
    ];
    xVel = UNIT;
    yVel = 0;
    score = 0;
    scoreText.textContent = score;
    active = true;
    started = false;
    paused = false;
    startGame();
}


*{
    margin:0;
    padding:0;
    font-family:'Gill Sans','Gill Sans MT','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;
}
 <!DOCTYPE html>
<title>snake_game</title>
<head>
   <link rel="stylesheet" href="tutoh.css">
   <script src="tubhii.js" defer></script>
</head>
<body>
   <h2>Snake_Game</h2>
   <div id="msg">Press space to puase 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>
</body>
</html>
```

7)Accessing web cam

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Webcam Snapshot & Recording</title>
    <style>
        body { text-align: center; font-family: Arial, sans-serif; }
        video, canvas { border: 2px solid black; margin: 10px; }
        .controls { margin-top: 10px; }
    </style>
</head>
<body>
    <h2>Webcam Access - Snapshot & Recording</h2>
    <video id="webcam" autoplay></video>
    <canvas id="snapshot" width="640" height="480"></canvas>
    <div class="controls">
        <button onclick="takeSnapshot()">Take Snapshot</button>
        <button onclick="startRecording()">Start Recording</button>
        <button onclick="stopRecording()">Stop Recording</button>
        <a id="downloadLink" style="display:none">Download Recording</a>
    </div>
    <script>
        const video = document.getElementById('webcam');
        const canvas = document.getElementById('snapshot');
```

```
const context = canvas.getContext('2d');
let mediaRecorder;
let recordedChunks = [];



navigator.mediaDevices.getUserMedia({ video: true, audio: true })
  .then(stream => {
    video.srcObject = stream;
    mediaRecorder = new MediaRecorder(stream);


    mediaRecorder.ondataavailable = event => recordedChunks.push(event.data);


    mediaRecorder.onstop = () => {
      const blob = new Blob(recordedChunks, { type: 'video/webm' });
      const url = URL.createObjectURL(blob);
      document.getElementById('downloadLink').href = url;
      document.getElementById('downloadLink').download = 'recording.webm';
      document.getElementById('downloadLink').style.display = 'block';
    };
  })
  .catch(error => console.error('Error accessing webcam:', error));


function takeSnapshot() {
  context.drawImage(video, 0, 0, canvas.width, canvas.height);
}
```

```
        function startRecording() {
            recordedChunks = [];
            mediaRecorder.start();
        }

        function stopRecording() {
            mediaRecorder.stop();
        }
    </script>
</body>
</html>
```

8)mobile flashlight

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mobile Flashlight Toggle</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 50px;
            background: #222;
            color: white;
```

```
        }
        button {
            padding: 15px 30px;
            font-size: 20px;
            border: none;
            background: #ffcc00;
            color: black;
            cursor: pointer;
            border-radius: 10px;
            margin-top: 20px;
        }
        button:active {
            background: #ff9900;
        }
    </style>
</head>
<body>
    <h1>Mobile Flashlight Control</h1>
    <button id="flashlight-toggle">Turn Flashlight ON</button>

    <script>
        let stream = null;
        let track = null;
        let flashlightOn = false;

        async function toggleFlashlight() {
```

```
        try {
            if (!flashlightOn) {
                stream = await navigator.mediaDevices.getUserMedia({ video: { facingMode: "environment" } });
                track = stream.getVideoTracks()[0];
                const capabilities = track.getCapabilities();
                if (capabilities.torch) {
                    await track.applyConstraints({ advanced: [{ torch: true }] });
                    flashlightOn = true;
                    document.getElementById("flashlight-toggle").textContent = "Turn Flashlight OFF";
                } else {
                    alert("Flashlight not supported on this device!");
                }
            } else {
                if (track) {
                    track.stop();
                }
                flashlightOn = false;
                document.getElementById("flashlight-toggle").textContent = "Turn Flashlight ON";
            }
        } catch (error) {
            console.error("Error accessing flashlight:", error);
            alert("Flashlight control is not supported on this browser/device.");
        }
    }
```

```
        document.getElementById("flashlight-toggle").addEventListener("click", toggleFlashlight);
    </script>
</body>
</html>
```