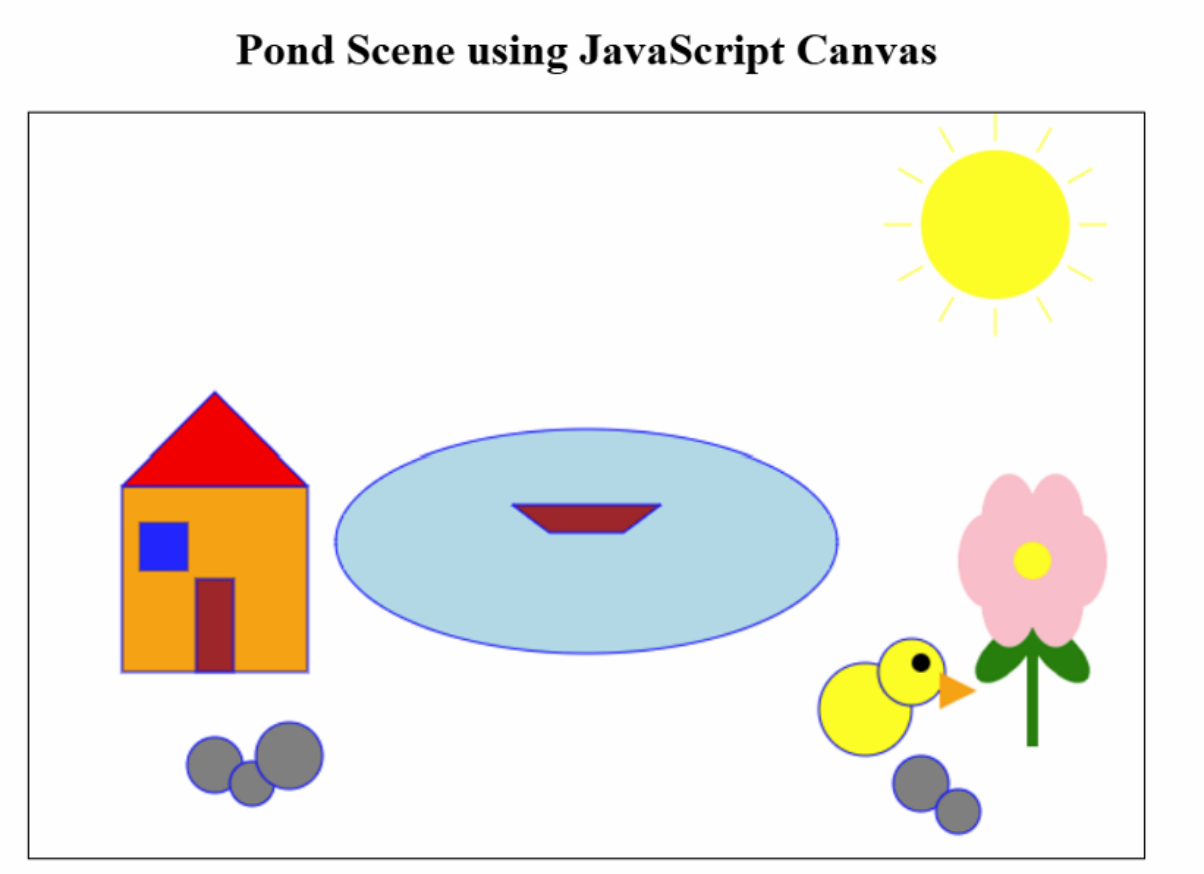
WEB PROGRAMMING ASSIGNMENT – 12

-B VENKATESH (23BCE1012)

1) Write a JavaScript program using the HTML5 Canvas API to draw a scene that consists of the following shapes and corresponding drawings: Shape Oval Polygon (Quadrilateral with curved edges) Two Circles of Different Sizes A Large Circle with Multiple Straight Lines Extending Outward A Rectangle with a Triangle on Top An Ellipse with a Vertical Line and Two Curved Shapes Multiple Small Circles Requirements: Drawing Representation Pond Boat Duck (Body & Head) Sun House Flower (Stem, Leaves, and Petals) Stones • Use the Canvas API functions such as arc(), ellipse(), fillRect(), lineTo(), moveTo(), and stroke(). • Assign different colors to each shape. • Ensure the relative positioning of the elements remains visually structured.

Sample Scene:



CODE:

<!DOCTYPE html>

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>121</title>
    <style>
      canvas {
        border: 1px solid black;
      }
      body {
        margin: 0;
      }
    </style>
  </head>
  <body>
    <canvas></canvas>
    <script>
      var canvas = document.querySelector("canvas");
      canvas.width = window.innerWidth;
      canvas.height = window.innerHeight;
      var c = canvas.getContext("2d");
      c.fillStyle = "orange";
      c.strokeStyle = "blue";
      //House
      c.fillRect(70, 300, 100, 125);
      c.strokeRect(70, 300, 100, 125);
```

```
//Door
c.fillStyle = "brown";
c.fillRect(110, 375, 20, 50);
c.strokeRect(110, 375, 20, 50);
//Window
c.fillStyle = "blue";
c.fillRect(80, 320, 30, 30);
//Triangle Roof
c.beginPath();
c.moveTo(70, 300);
c.lineTo(120, 250);
c.lineTo(170, 300);
c.lineTo(70, 300);
c.stroke();
c.fillStyle = "red";
c.fill();
//Stones beneath the house
c.fillStyle = "grey";
c.beginPath();
c.arc(100, 525, 20, 0, Math.PI * 2);
c.stroke();
c.fill();
c.beginPath();
c.arc(125, 540, 14, 0, Math.PI * 2);
c.stroke();
c.fill();
```

```
c.beginPath();
c.arc(150, 515, 25, 0, Math.PI * 2);
c.stroke();
c.fill();
//Stones on bottom right
c.beginPath();
c.arc(650, 550, 20, 0, Math.PI * 2);
c.stroke();
c.fill();
c.beginPath();
c.arc(680, 560, 15, 0, Math.PI * 2);
c.stroke();
c.fill();
//Duck
c.fillStyle = "yellow";
c.beginPath();
c.arc(620, 490, 30, 0, Math.PI * 2);
c.stroke();
c.fill();
c.beginPath();
c.arc(650, 460, 23, 0, Math.PI * 2);
c.stroke();
c.fill();
c.fillStyle = "black";
c.beginPath();
c.arc(655, 450, 7, 0, Math.PI * 2);
```

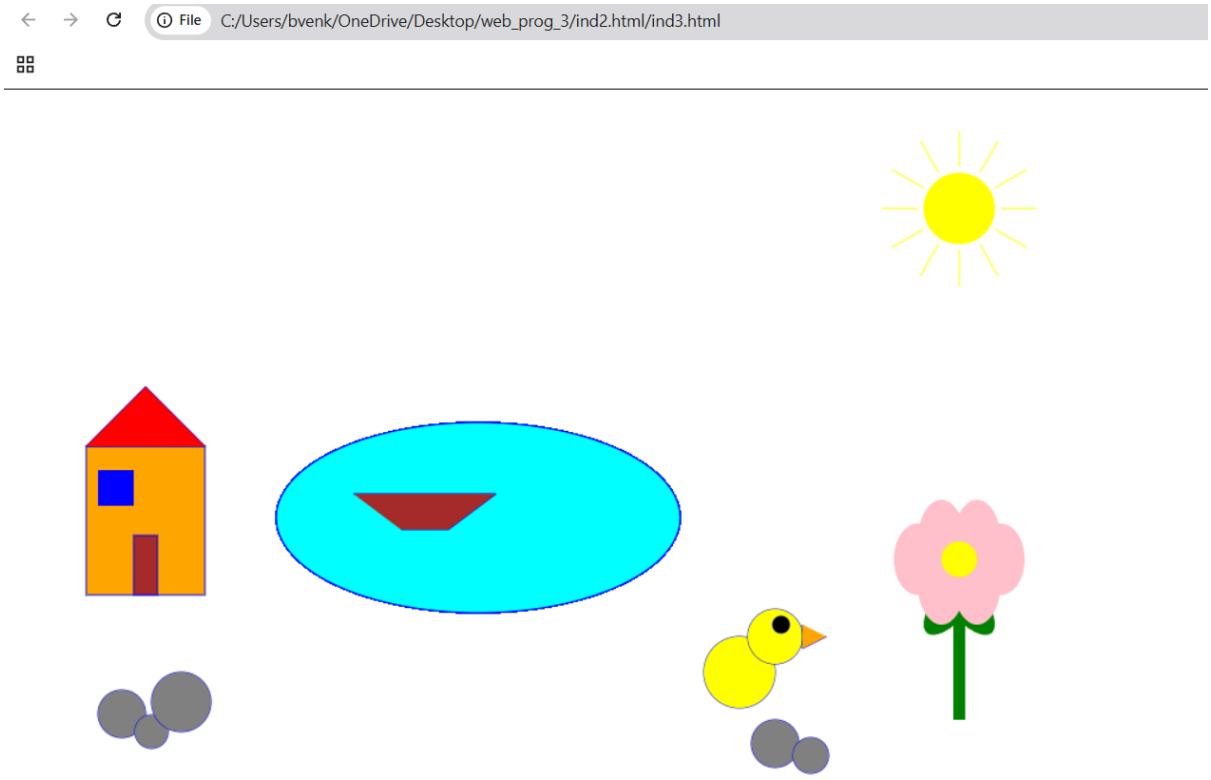
```
c.stroke();
c.fill();
c.fillStyle = "orange";
c.beginPath();
c.moveTo(673, 450);
c.lineTo(673, 470);
c.lineTo(693, 460);
c.stroke();
c.fill();
//Pond
c.fillStyle = "lightblue";
c.beginPath();
c.ellipse(400, 360, 170, 80, 0, 0, Math.PI * 2);
c.stroke();
c.fill();
//Boat
var x1 = 340;
var x2 = 460;
var x3 = 380;
var x4 = 420;
var dx = 1;
c.fillStyle = "brown";
function animate() {
  c.fillStyle = "cyan";
  c.strokeStyle = "blue";
  c.beginPath();
```

```
 c.ellipse(400, 360, 170, 80, 0, 0, Math.PI * 2);
 c.fill();
 c.stroke();
 c.fillStyle = "brown";
 c.beginPath();
 c.moveTo(x3, 370);
 c.lineTo(x4, 370);
 c.lineTo(x2, 340);
 c.lineTo(x1, 340);
 c.stroke();
 c.fill();
 if (x2 == 560 || x1 == 240) {
   dx = -dx;
 }
 x1 = x1 + dx;
 x2 = x2 + dx;
 x3 = x3 + dx;
 x4 = x4 + dx;
 requestAnimationFrame(animate);
}
animate();
//Flower Base
c.fillStyle = "green";
c.strokeStyle = "green";
c.fillRect(800, 430, 10, 100);
c.beginPath();
```

```
c.ellipse(800, 430, 35, 15, -70, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(810, 430, 35, 15, 70, 0, Math.PI * 2);
c.fill();
//Petals
c.fillStyle = "pink";
c.beginPath();
c.ellipse(790, 415, 20, 35, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(820, 415, 20, 35, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(790, 380, 20, 35, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(820, 380, 20, 35, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(840, 395, 20, 30, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(770, 395, 20, 30, 0, 0, Math.PI * 2);
c.fill();
//Flower base (yellow)
```

```
    c.fillStyle = "yellow";
    c.beginPath();
    c.arc(805, 395, 15, 0, Math.PI * 2);
    c.fill();
    //Sun
    c.strokeStyle = "yellow";
    c.beginPath();
    c.arc(805, 100, 30, 0, Math.PI * 2);
    c.fill();
    for (var x = 0; x < 12; x++) {
     const angle = 30 * x;
     const movex = 805 + Math.cos(angle * (Math.PI / 180)) * 35;
     const movey = 100 + Math.sin(angle * (Math.PI / 180)) * 35;
     const endx = 805 + Math.cos(angle * (Math.PI / 180)) * 65;
     const endy = 100 + Math.sin(angle * (Math.PI / 180)) * 65;
     c.beginPath();
     c.moveTo(movex, movey);
     c.lineTo(endx, endy);
     c.stroke();
    }
   </script>
  </body>
</html>
```

OUTPUT:



2) Apply an animation effect to the boat.

CODE:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
  <title>121</title>
  <style>
   canvas {
    border: 1px solid black;
   }
```

```
  body {
    margin: 0;
  }
 </style>
</head>
<body>
 <canvas></canvas>
 <script>
  var canvas = document.querySelector("canvas");
  canvas.width = window.innerWidth;
  canvas.height = window.innerHeight;
  var c = canvas.getContext("2d");
  c.fillStyle = "orange";
  c.strokeStyle = "blue";
  //House
  c.fillRect(70, 300, 100, 125);
  c.strokeRect(70, 300, 100, 125);
  //Door
  c.fillStyle = "brown";
  c.fillRect(110, 375, 20, 50);
  c.strokeRect(110, 375, 20, 50);
  //Window
  c.fillStyle = "blue";
  c.fillRect(80, 320, 30, 30);
  //Triangle Roof
  c.beginPath();
```

```
c.moveTo(70, 300);
c.lineTo(120, 250);
c.lineTo(170, 300);
c.lineTo(70, 300);
c.stroke();
c.fillStyle = "red";
c.fill();
//Stones beneath the house
c.fillStyle = "grey";
c.beginPath();
c.arc(100, 525, 20, 0, Math.PI * 2);
c.stroke();
c.fill();
c.beginPath();
c.arc(125, 540, 14, 0, Math.PI * 2);
c.stroke();
c.fill();
c.beginPath();
c.arc(150, 515, 25, 0, Math.PI * 2);
c.stroke();
c.fill();
//Stones on bottom right
c.beginPath();
c.arc(650, 550, 20, 0, Math.PI * 2);
c.stroke();
c.fill();
```

```
c.beginPath();
c.arc(680, 560, 15, 0, Math.PI * 2);
c.stroke();
c.fill();
//Duck
c.fillStyle = "yellow";
c.beginPath();
c.arc(620, 490, 30, 0, Math.PI * 2);
c.stroke();
c.fill();
c.beginPath();
c.arc(650, 460, 23, 0, Math.PI * 2);
c.stroke();
c.fill();
c.fillStyle = "black";
c.beginPath();
c.arc(655, 450, 7, 0, Math.PI * 2);
c.stroke();
c.fill();
c.fillStyle = "orange";
c.beginPath();
c.moveTo(673, 450);
c.lineTo(673, 470);
c.lineTo(693, 460);
c.stroke();
c.fill();
```

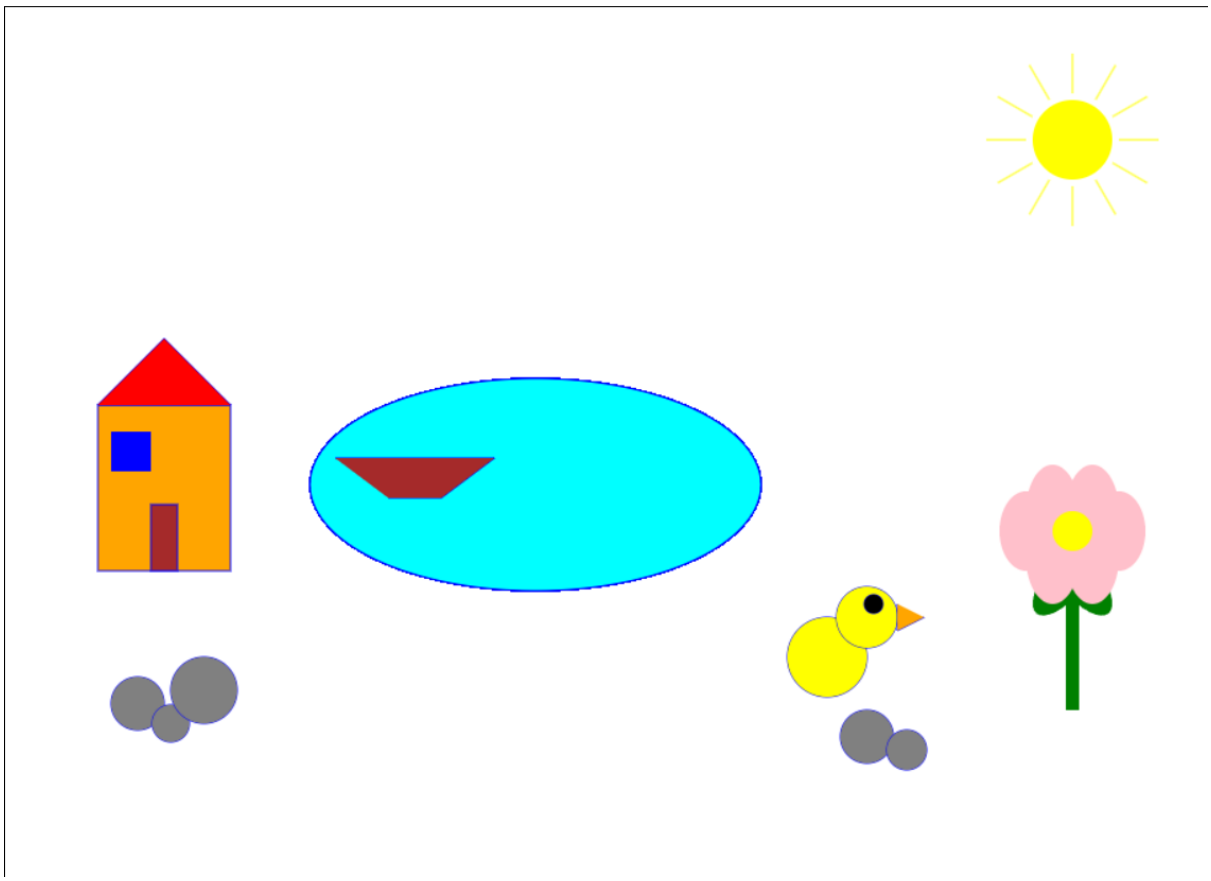
```
//Pond
c.fillStyle = "lightblue";
c.beginPath();
c.ellipse(400, 360, 170, 80, 0, 0, Math.PI * 2);
c.stroke();
c.fill();
//Boat
var x1 = 340;
var x2 = 460;
var x3 = 380;
var x4 = 420;
var dx = 1;
c.fillStyle = "brown";
function animate() {
 c.fillStyle = "cyan";
 c.strokeStyle = "blue";
 c.beginPath();
 c.ellipse(400, 360, 170, 80, 0, 0, Math.PI * 2);
 c.fill();
 c.stroke();
 c.fillStyle = "brown";
 c.beginPath();
 c.moveTo(x3, 370);
 c.lineTo(x4, 370);
 c.lineTo(x2, 340);
 c.lineTo(x1, 340);
```

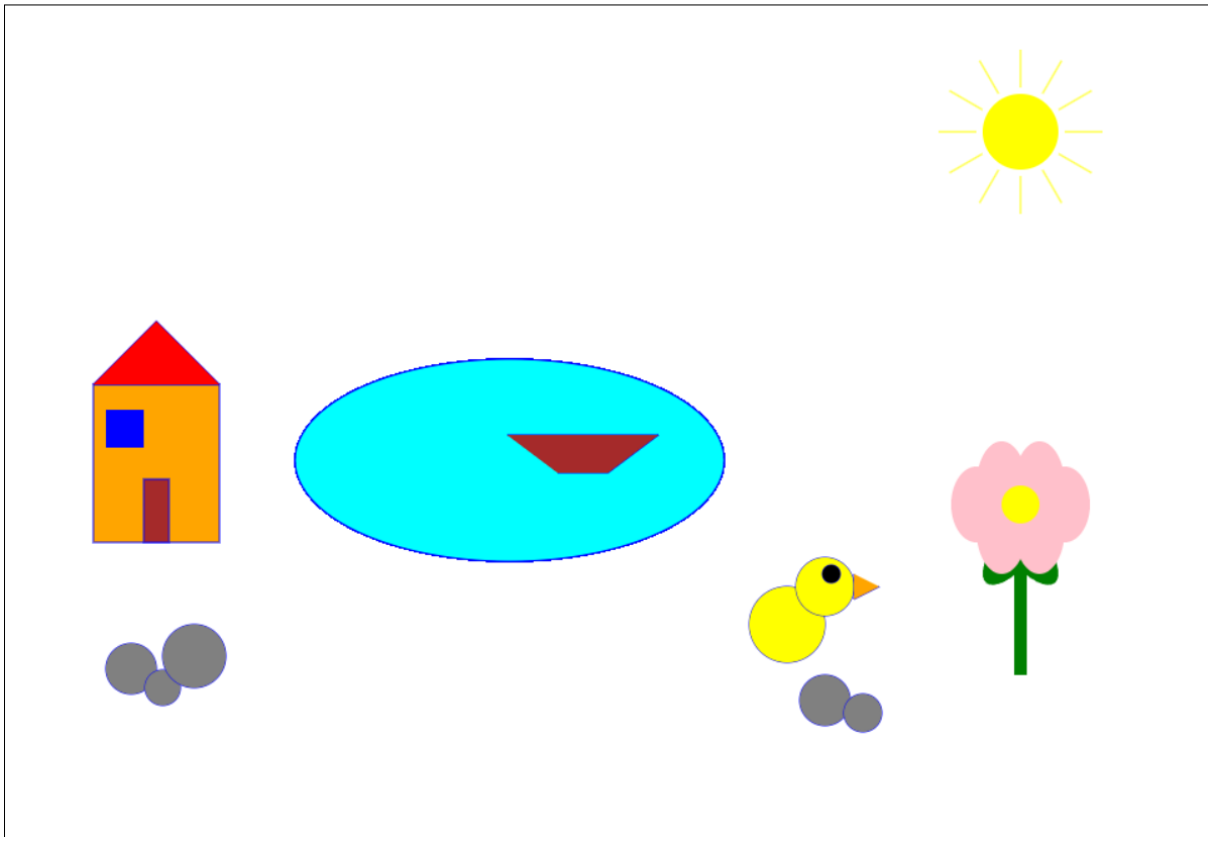
```
 c.stroke();
 c.fill();
 if (x2 == 560 || x1 == 240) {
  dx = -dx;
 }
 x1 = x1 + dx;
 x2 = x2 + dx;
 x3 = x3 + dx;
 x4 = x4 + dx;
 requestAnimationFrame(animate);
}
animate();
//Flower Base
c.fillStyle = "green";
c.strokeStyle = "green";
c.fillRect(800, 430, 10, 100);
c.beginPath();
c.ellipse(800, 430, 35, 15, -70, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(810, 430, 35, 15, 70, 0, Math.PI * 2);
c.fill();
//Petals
c.fillStyle = "pink";
c.beginPath();
c.ellipse(790, 415, 20, 35, 0, 0, Math.PI * 2);
```

```
c.fill();
c.beginPath();
c.ellipse(820, 415, 20, 35, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(790, 380, 20, 35, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(820, 380, 20, 35, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(840, 395, 20, 30, 0, 0, Math.PI * 2);
c.fill();
c.beginPath();
c.ellipse(770, 395, 20, 30, 0, 0, Math.PI * 2);
c.fill();
//Flower base (yellow)
c.fillStyle = "yellow";
c.beginPath();
c.arc(805, 395, 15, 0, Math.PI * 2);
c.fill();
//Sun
c.strokeStyle = "yellow";
c.beginPath();
c.arc(805, 100, 30, 0, Math.PI * 2);
c.fill();
```

```
    for (var x = 0; x < 12; x++) {
      const angle = 30 * x;
      const movex = 805 + Math.cos(angle * (Math.PI / 180)) * 35;
      const movey = 100 + Math.sin(angle * (Math.PI / 180)) * 35;
      const endx = 805 + Math.cos(angle * (Math.PI / 180)) * 65;
      const endy = 100 + Math.sin(angle * (Math.PI / 180)) * 65;
      c.beginPath();
      c.moveTo(movex, movey);
      c.lineTo(endx, endy);
      c.stroke();
    }
   </script>
  </body>
</html>
```

OUTPUT:





3) Write a JavaScript program that creates a working analog clock using the HTML5 Canvas API. The clock should display the current time dynamically and accurately, updating every second. Requirements: i) Use the Canvas API to draw the clock face, hands, and markings. ii) The clock must include the following elements: a. A circular clock face with a border and a filled background color. b. Hour, minute, and second hands that update dynamically based on the current time. c. Numerical or tick markings for hours (1 to 12). d. A center pivot point for the hands. iii) Ensure the hands move smoothly and update every second.

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 using Canvas</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f0f0f0;
    }
    canvas {
      background-color: white;
```

```
        border-radius: 50%;
      }
    </style>
</head>
<body>

    <canvas id="clockCanvas" width="400" height="400"></canvas>

    <script>
      const canvas = document.getElementById("clockCanvas");
      const ctx = canvas.getContext("2d");
      const centerX = canvas.width / 2;
      const centerY = canvas.height / 2;
      const radius = 150;

      function drawClockFace() {
        ctx.fillStyle = "#ffffff"; // White background
        ctx.strokeStyle = "#000000"; // Black border
        ctx.lineWidth = 8;

        // Draw the outer circle
        ctx.beginPath();
        ctx.arc(centerX, centerY, radius, 0, Math.PI * 2);
        ctx.fill();
        ctx.stroke();
      }
```

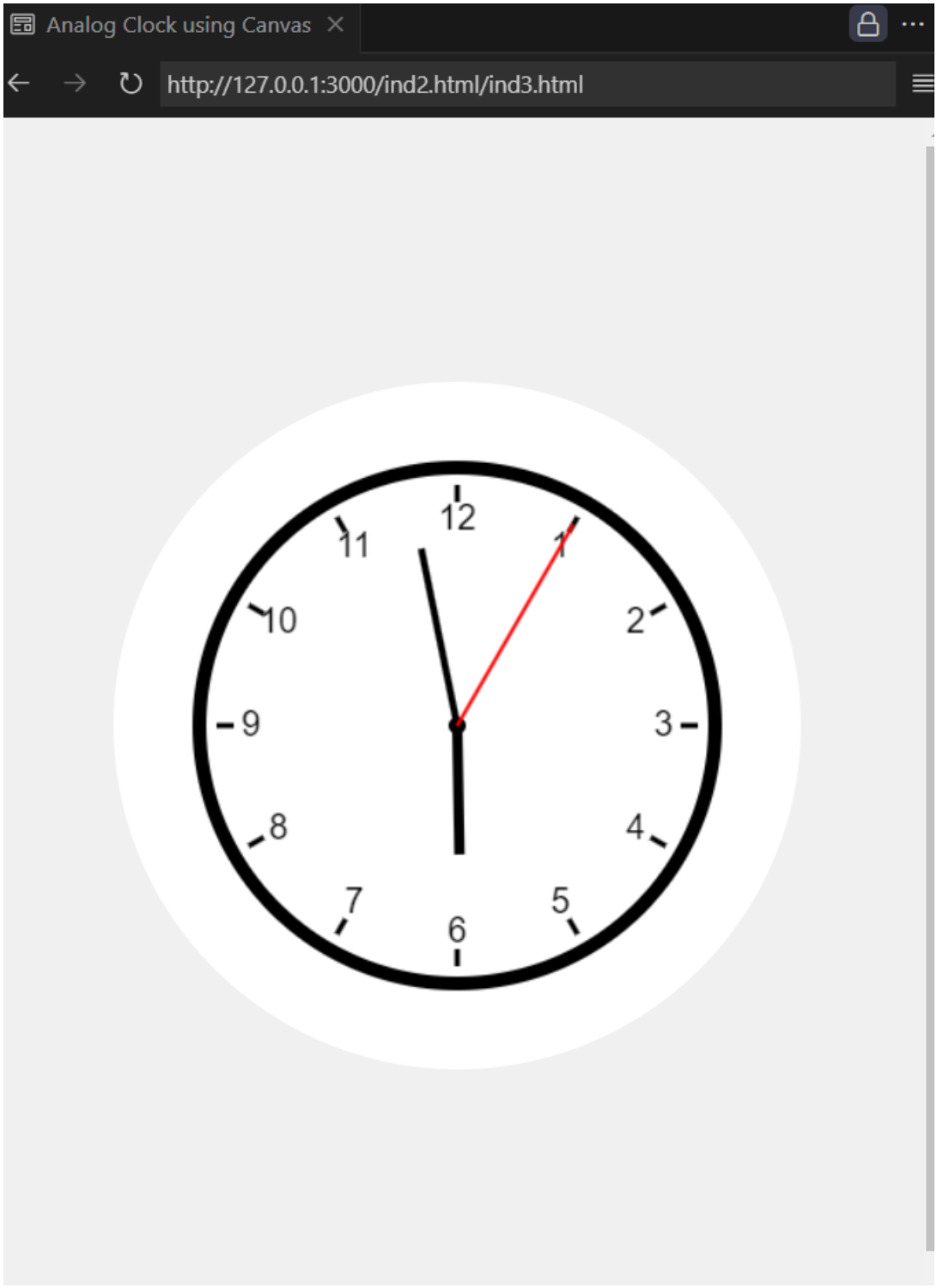
```
function drawNumbers() {
    ctx.font = "20px Arial";
    ctx.fillStyle = "black";
    ctx.textAlign = "center";
    ctx.textBaseline = "middle";

    for (let num = 1; num <= 12; num++) {
        let angle = ((num - 3) * Math.PI) / 6;
        let x = centerX + Math.cos(angle) * (radius - 30);
        let y = centerY + Math.sin(angle) * (radius - 30);
        ctx.fillText(num, x, y);
    }
}

function drawTicks() {
    for (let i = 0; i < 60; i++) {
        let angle = (i * Math.PI) / 30;
        let x1 = centerX + Math.cos(angle) * (radius - 10);
        let y1 = centerY + Math.sin(angle) * (radius - 10);
        let x2 = centerX + Math.cos(angle) * (radius - (i % 5 === 0 ? 20 : 10));
        let y2 = centerY + Math.sin(angle) * (radius - (i % 5 === 0 ? 20 : 10));

        ctx.lineWidth = i % 5 === 0 ? 3 : 1;
        ctx.strokeStyle = "#000000";
        ctx.beginPath();
```

```
        ctx.moveTo(x1, y1);
        ctx.lineTo(x2, y2);
        ctx.stroke();
      }
    }

    function drawHand(angle, length, width, color) {
      ctx.lineWidth = width;
      ctx.strokeStyle = color;
      ctx.beginPath();
      ctx.moveTo(centerX, centerY);
      ctx.lineTo(centerX + Math.cos(angle) * length, centerY +
Math.sin(angle) * length);
      ctx.stroke();
    }

    function drawCenterPivot() {
      ctx.fillStyle = "#000000";
      ctx.beginPath();
      ctx.arc(centerX, centerY, 5, 0, Math.PI * 2);
      ctx.fill();
    }

    function drawClock() {
      ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear canvas

      drawClockFace(); // Draw the clock face
```

```
    drawNumbers(); // Draw numbers
    drawTicks(); // Draw tick marks
    drawCenterPivot(); // Draw center pivot

    let now = new Date();
    let hours = now.getHours() % 12;
    let minutes = now.getMinutes();
    let seconds = now.getSeconds();

    // Calculate angles
    let hourAngle = ((hours + minutes / 60) * Math.PI) / 6 - Math.PI / 2;
    let minuteAngle = ((minutes + seconds / 60) * Math.PI) / 30 - Math.PI /
2;
    let secondAngle = (seconds * Math.PI) / 30 - Math.PI / 2;

    // Draw hands
    drawHand(hourAngle, radius * 0.5, 6, "black"); // Hour hand
    drawHand(minuteAngle, radius * 0.7, 4, "black"); // Minute hand
    drawHand(secondAngle, radius * 0.9, 2, "red"); // Second hand
  }

  function updateClock() {
    drawClock();
    requestAnimationFrame(updateClock); // Smooth animation
  }

  updateClock(); // Start the clock
```

```
    </script>

</body>
</html>
```

OUTPUT:



4) Write a JavaScript program that dynamically generates the charts (bar chart, line chart, pie chart and a donut chart) using Plotly.js. Each chart must include: a. Labeled X and Y axes (for bar and line charts). b. Title for each chart. c. Different colors for data points. d. Legend (for the pie chart and donut) showing categories. ii) The chart should be scaled properly to fit within the display area.

CODE:

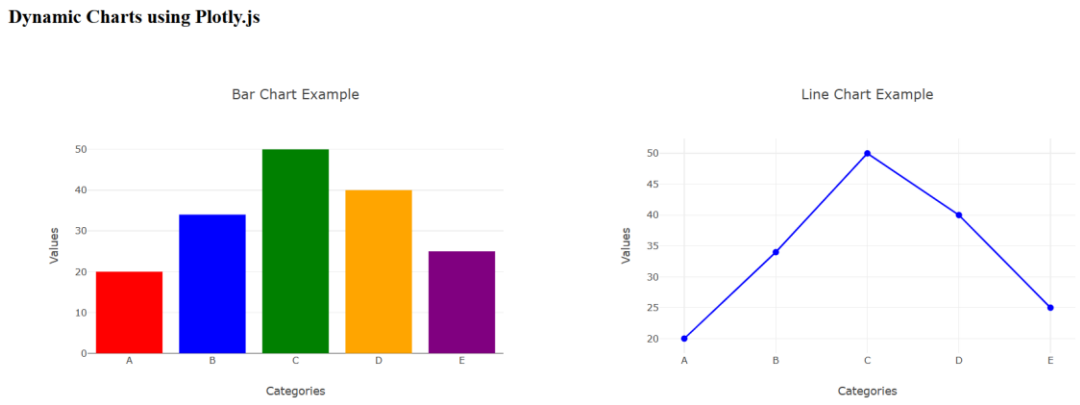
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dynamic Charts with Plotly.js</title>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  <style>
    .chart-container {
      width: 45%;
      display: inline-block;
      margin: 20px;
    }
  </style>
</head>
<body>

  <h2>Dynamic Charts using Plotly.js</h2>

  <div id="barChart" class="chart-container"></div>
  <div id="lineChart" class="chart-container"></div>
  <div id="pieChart" class="chart-container"></div>
  <div id="donutChart" class="chart-container"></div>

  <script>
```

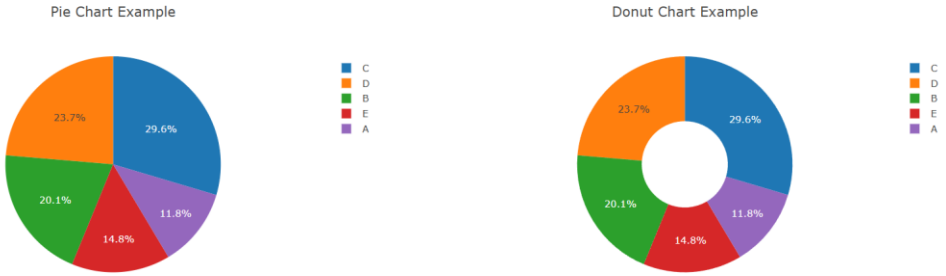
```
// Data for the charts
const categories = ['A', 'B', 'C', 'D', 'E'];
const values = [20, 34, 50, 40, 25];

// ==== BAR CHART ====
let barData = [{
  x: categories,
  y: values,
  type: 'bar',
  marker: { color: ['red', 'blue', 'green', 'orange', 'purple'] }
}];

let barLayout = {
  title: "Bar Chart Example",
  xaxis: { title: "Categories" },
  yaxis: { title: "Values" }
};

Plotly.newPlot('barChart', barData, barLayout);

// ==== LINE CHART ====
let lineData = [{
  x: categories,
  y: values,
  type: 'scatter',
  mode: 'lines+markers',
```

```
  marker: { color: 'blue', size: 8 }
}];

let lineLayout = {
  title: "Line Chart Example",
  xaxis: { title: "Categories" },
  yaxis: { title: "Values" }
};

Plotly.newPlot('lineChart', lineData, lineLayout);

// ==== PIE CHART ====
let pieData = [{
  labels: categories,
  values: values,
  type: 'pie'
}];

let pieLayout = {
  title: "Pie Chart Example",
  showlegend: true
};

Plotly.newPlot('pieChart', pieData, pieLayout);

// ==== DONUT CHART ====
```

```
    let donutData = [{
      labels: categories,
      values: values,
      type: 'pie',
      hole: 0.4  // Makes it a donut chart
    }];

    let donutLayout = {
      title: "Donut Chart Example",
      showlegend: true
    };

    Plotly.newPlot('donutChart', donutData, donutLayout);
  </script>

</body>
</html>
```

OUTPUT:





5) Write a JavaScript program that dynamically creates and manipulates overlapping elements using CSS z-index. The program should allow the user to change the stacking order of elements by adjusting their z-index values. • Create at least three overlapping elements (e.g., div boxes or images). • Use CSS z-index to control the layering order of these elements. • Provide buttons or user input to dynamically adjust the z-index values using JavaScript. • Display the current z-index value of each element.

CODE:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Z-Index Manipulation</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      text-align: center;

    }
```

```
    .container {
      position: relative;
      width: 400px;
      height: 300px;
      margin: 50px auto;
      border: 2px solid black;
    }
    .box {
      position: absolute;
      width: 150px;
      height: 150px;
      color: white;
      font-size: 18px;
      font-weight: bold;
      display: flex;
      align-items: center;
      justify-content: center;
      border-radius: 10px;
    }
    #box1 { background: red; top: 50px; left: 50px; z-index: 1; }
    #box2 { background: blue; top: 80px; left: 100px; z-index: 2; }
    #box3 { background: green; top: 110px; left: 150px; z-index: 3;
}
```

```
      .controls {
         margin-top: 20px;
      }
   </style>
</head>
<body>

   <h2>Change Z-Index Dynamically</h2>

   <div class="container">
      <div id="box1" class="box">Box 1 (Z: 1)</div>
      <div id="box2" class="box">Box 2 (Z: 2)</div>
      <div id="box3" class="box">Box 3 (Z: 3)</div>
   </div>

   <div class="controls">
      <h3>Adjust Z-Index:</h3>
      <div>
         <button onclick="changeZIndex('box1', 1)">Box 1
↑</button>
         <button onclick="changeZIndex('box1', -1)">Box 1
↓</button>
      </div>
      <div>
```

```
        <button onclick="changeZIndex('box2', 1)">Box 2 ↑</button>
        <button onclick="changeZIndex('box2', -1)">Box 2 ↓</button>
      </div>
      <div>
        <button onclick="changeZIndex('box3', 1)">Box 3 ↑</button>
        <button onclick="changeZIndex('box3', -1)">Box 3 ↓</button>
      </div>
    </div>

    <script>
      function changeZIndex(boxId, change) {
        let box = document.getElementById(boxId);
        let currentZIndex = parseInt(window.getComputedStyle(box).zIndex);
        let newZIndex = currentZIndex + change;

        // Ensure z-index stays within reasonable bounds (e.g., 1 to 10)
        if (newZIndex >= 1 && newZIndex <= 10) {
          box.style.zIndex = newZIndex;
          box.innerText = `${boxId.charAt(3)} (Z: ${newZIndex})`;
```

```
        }
    }
  </script>

</body>
</html>
```

OUTPUT: