CANVAS EXERCISE:

In an aside in the HTML toward the top of the page, add a canvas tag and make a div that will be used as a tooltip:

- Define the canvas element for drawing the chart. Add an id and width and height: `<canvas id="myChart" width="350" height="225">`
- Defines the tooltip element with initial styles (hidden, background color, text color, padding, border radius): `<div id="tooltip" style="…">`.

CODE:

```
 <canvas id="myChart" width="350" height="225"></canvas>
        <div id="tooltip" style="display: none; position: absolute; background-color: rgba(25, 25, 112, 0.8); color: white; padding: 5px; border-radius: 5px;"></div>
```

In the JavaScript:

1. **Get canvas and context:**
   - Access the canvas element: `const canvas = document.getElementById('myChart');`
   - Gets the 2D rendering context for drawing: `const ctx = canvas.getContext('2d');`

   - Create an array called frameworks. It will contain objects with name and usage properties. Names are frameworks names and usage is a number indicating popularity (it does not have to be accurate): `const frameworks = [...]`

CODE:

```
const frameworks = [
{ name: 'React', usage: 200 },
{ name: 'Angular', usage: 140 },
{ name: 'Vue', usage: 100 },
{ name: 'Svelte', usage: 70 }
];
```

3. **Set chart dimensions:**
   - `const barWidth`, `barSpacing`, and `chartHeight` variables control bar size and spacing.

CODE:

```
const barWidth = 60;
const barSpacing = 10;
```

```
const chartHeight = 200;
```

4. **Draw the bars:**
   - Iterate through frameworks: `frameworks.forEach(…)`
   - Calculate bar position (`x`, `y`, and `height`) based on framework data and chart dimensions.
   - Generate random colors for bars with `const color = '#' + Math.floor(Math.random() * 16777215).toString(16);`
   - Use `ctx.fillRect` to draw bars on the canvas.
   - Add framework names as labels with `ctx.fillText`.

CODE:

```
frameworks.forEach((framework, index) => {
const x = index * (barWidth + barSpacing);
const y = chartHeight - framework.usage;
const height = framework.usage;
// Random color for each bar
const color = '#' + Math.floor(Math.random() *
16777215).toString(16);
ctx.fillStyle = color;
ctx.fillRect(x, y, barWidth, height);
```

5. **Add chart title (part of same forEach loop):**
   - Set font, fill style, and alignment for the title.
   - Use `ctx.fillText` to draw the title "JavaScript Framework Popularity”.

CODE:

```
//add heading title
ctx.font = '16px Arial';
ctx.fillStyle = 'black';
ctx.textAlign = 'center';
ctx.fillText('JavaScript Framework Popularity',
canvas.width / 1.75, 20);
// Add labels
ctx.fillStyle = 'black';
ctx.font = '12px Arial';
ctx.fillText(framework.name, x + barWidth / 2 -
ctx.measureText(framework.name).width / 2, chartHeight +
15);
});
```

6. **Handle mouse movement and update tooltip:**

- Add a `mousemove` event listener to the canvas.
- Calculate relative mouse position within the canvas.
- Check for a hovered bar by comparing mouse coordinates with bar boundaries.
- Retrieve the tooltip element and updates its content based on the hovered bar.
- Show or hide the tooltip based on whether a bar is hovered.

CODE:

```
canvas.addEventListener('mousemove', (event) => {
    const rect = canvas.getBoundingClientRect();
    const x = event.clientX - rect.left;
    const y = event.clientY - rect.top;

    let hoveredBar = null;

    frameworks.forEach((item, index) => {
      const barX = index * (barWidth + barSpacing);
      const barY = chartHeight - item.usage;
      const barHeight = item.usage;

      if (x >= barX && x <= barX + barWidth && y >= barY &&
y <= barY + barHeight) {
        hoveredBar = item; // Store information about
hovered bar
      }
    });

    // Update tooltip content or create a new element
    const tooltipElement =
document.getElementById('tooltip'); // Add a <div> with
id="tooltip"

    if (hoveredBar) {
      tooltipElement.textContent = `Framework: $
{hoveredBar.name} - Usage: ${hoveredBar.usage}`;
      tooltipElement.style.display = 'block';
      tooltipElement.style.left = `${x + 400}px`; //
Position tooltip near the hovered bar
      tooltipElement.style.top = `${y + 200}px`;
    } else {
      tooltipElement.style.display = 'none';
    }
  });
```