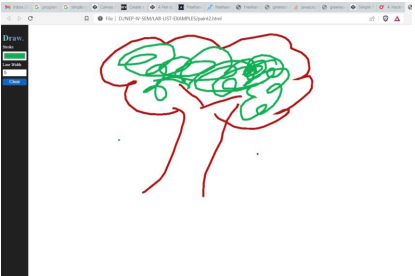
**6. Create a simple paint app which draws lines based on the selected color (chosen using color input) with selected thickness (chosen using number input) and there must be CLEAR button to clear the canvas. Sample screen shot:**

****

**Paint.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,intial-scale=1.0">

<link rel="stylesheet" href="pstyles.css">

<title>Drawing app</title>

</head>

<body>

<section class="container">

<div id="toolbar">

<h1>Draw</h1>

<label for="stroke">Stroke</label>

<input id="stroke" name='stroke' type="color">

<label for="lineWidth">Line Width</label>

<input id="lineWidth" name='lineWidth' type="number" value="5" max="10" min="1">

<button id="clear">Clear</button>

</div>

<div class="drawing-board">

<canvas id="drawing-board"></canvas>

</div>

</section>

<script src="paint1.js"></script>

</body>

</html>

**Pstyles.css:**

body{

margin:0;

padding:0;

height:100%;

overflow:hidden;

color:white;

}

h1{

background:#7f7fd5;

background:-webkit-linear-gradient(to right, #91EAE4,#86A8E7,#7F7FD5);

background:linear-gradient(to right,#91EAE4,#86A8E7,#7F7FD5);

background-clip:text;

-webkit-background-clip:text;

-webkit-text-fill-color:transparent;

}

.container{

height:100%;

display:flex;

}

#toolbar{

display:flex;

flex-direction:column;

padding:5px;

width:70px;

background-color:#202020;

}

#toolbar\*{

margin-bottom:6px;

}

#toolbar label{

font-size:12px;

}

#toolbar input{

width:100%;

}

#toolbar button{

background-color:#1565c0;

border:none;

border-radius:4px;

color:white;

padding:2px;

}

**Paint1.js:**

const canvas = document.getElementById('drawing-board');

const toolbar = document.getElementById('toolbar');

const ctx = canvas.getContext('2d');

const canvasOffsetX = canvas.offsetLeft;

const canvasOffsetY = canvas.offsetTop;

canvas.width = window.innerWidth - canvasOffsetX;

canvas.height = window.innerHeight - canvasOffsetY;

let isPainting = false;

let lineWidth = 5;

let startX;

let startY;

toolbar.addEventListener('click', e => { // Corrected arrow function parameter

if (e.target.id === 'clear') { // Fixed reference to 'e'

ctx.clearRect(0, 0, canvas.width, canvas.height);

}

});

toolbar.addEventListener('change', e => {

if (e.target.id === 'stroke') {

ctx.strokeStyle = e.target.value;

}

if (e.target.id === 'lineWidth') {

lineWidth = e.target.value;

}

});

const draw = e => {

if (!isPainting) {

return;

}

ctx.lineWidth = lineWidth;

ctx.lineCap = 'round';

ctx.lineTo(e.clientX - canvasOffsetX, e.clientY - canvasOffsetY);

ctx.stroke();

}

canvas.addEventListener('mousedown', e => {

isPainting = true;

startX = e.clientX - canvasOffsetX;

startY = e.clientY - canvasOffsetY;

ctx.beginPath();

ctx.moveTo(startX, startY);

});

canvas.addEventListener('mouseup', e => {

isPainting = false;

});

canvas.addEventListener('mousemove', draw);

**OUTPUT:**

****