<h1>My app 😀</h1>

<div>

<div class="controlbar">

<label for="form">Canvas form</label><br>

<select id="form">

<option value="triangle">Triangle</option>

<option value="square">Square</option>

<option value="circle">Circle</option>

</select>

<br>

</div>

<div class="controlbar">

<label for="div\_border">Canvas border color</label><br>

<input id="div\_border" type="color"><br>

</div>

<div class="controlbar">

<label for="canvas\_background">Canvas background color</label><br>

<input id="canvas\_background" type="color"><br>

</div>

<div class="controlbar">

<label for="border\_thickness">Border thickness</label><br>

<select id="border\_thickness">

<option value="2">2px</option>

<option value="4">4px</option>

<option value="6">6px</option>

<option value="8">8px</option>

<option value="10">10px</option>

<option value="20">20px</option>

</select>

<br>

</div>

<div class="controlbar">

<label for="figure\_size">Figure size</label><br>

<select id="figure\_size">

<option value="20">20px</option>

<option value="40">40px</option>

<option value="60">60px</option>

<option value="80">80px</option>

<option value="100">100px</option>

<option value="200">200px</option>

</select>

<br>

</div>

<div class="controlbar">

<label for="draw">Figure size</label><br>

<button onclick="draw()" id="draw">Draw</button><br>

</div>

<div class="controlbar">

<label for="clrStorage">Clear storage</label><br>

<button onclick="clearStorage()" id="clrStorage">Clear storage</button><br>

</div>

</div>

<canvas id="canvas">

</canvas>

\* {

text-align: center;

font-family: Arial, Helvetica, sans-serif;

}

.controlbar{

display: inline-block;

margin-right: 1rem;

}

#canvas{

border: 2px solid black;

}

const canvas = document.getElementById('canvas')

// initiating 2D context on it

const c = canvas.getContext('2d')

let storedFigures = []

addEventListener('load', () => {

canvas.width = innerWidth

canvas.height = innerHeight

var loadedFigure = JSON.parse(localStorage.getItem('Fig')) || null;

if(loadedFigure){

for(let i = 0; i < loadedFigure.length; i++){

if(loadedFigure[i].forme == 'triangle'){

drawTriangle(loadedFigure[i].figSize, loadedFigure[i].borderSize, loadedFigure[i].start, loadedFigure[i].borderColor, loadedFigure[i].backgroundColor)

}

else if(loadedFigure[i].forme == 'square'){

drawSquare(loadedFigure[i].figSize, loadedFigure[i].borderSize, loadedFigure[i].start, loadedFigure[i].borderColor, loadedFigure[i].backgroundColor)

}

else if(loadedFigure[i].forme == 'circle'){

drawCircle(loadedFigure[i].figSize, loadedFigure[i].borderSize, loadedFigure[i].start, loadedFigure[i].borderColor, loadedFigure[i].backgroundColor)

}

localStorage.setItem('Fig', JSON.stringify(storedFigures));

}

}

})

addEventListener('resize', () => {

canvas.width = innerWidth

canvas.height = innerHeight

})

function clearStorage(){

localStorage.setItem('Fig', JSON.stringify([]))

}

function draw(){

let forme = document.getElementById('form').value

if(forme == 'triangle'){

drawTriangle()

}

else if(forme == 'square'){

drawSquare()

}

else if(forme == 'circle'){

drawCircle()

}

localStorage.setItem('Fig', JSON.stringify(storedFigures));

}

function drawTriangle(figSize = parseInt(document.getElementById('figure\_size').value), borderSize = parseInt(document.getElementById('border\_thickness').value), start = getStartingPoint(figSize, borderSize), border\_color = document.getElementById('div\_border').value, background\_color = document.getElementById('canvas\_background').value){

c.beginPath()

c.moveTo(start[0], start[1])

c.lineTo(start[0], start[1]+figSize)

c.lineTo(start[0]+figSize, start[1]+figSize)

c.closePath()

c.lineWidth = borderSize

c.strokeStyle = border\_color

c.stroke()

c.fillStyle = background\_color

c.fill()

storedFigures.push({

forme: 'triangle',

figSize: figSize,

borderSize: borderSize,

start: getStartingPoint(figSize, borderSize),

borderColor: border\_color,

backgroundColor: background\_color

})

}

function drawSquare(figSize = parseInt(document.getElementById('figure\_size').value), borderSize = parseInt(document.getElementById('border\_thickness').value), start = getStartingPoint(figSize, borderSize), border\_color = document.getElementById('div\_border').value, background\_color = document.getElementById('canvas\_background').value){

c.rect(start[0], start[1], figSize, figSize)

c.lineWidth = borderSize

c.strokeStyle = border\_color

c.stroke()

c.fillStyle = background\_color

c.fill()

storedFigures.push({

forme: 'square',

figSize: figSize,

borderSize: borderSize,

start: getStartingPoint(figSize, borderSize),

borderColor: border\_color,

backgroundColor: background\_color

})

}

function drawCircle(figSize = parseInt(document.getElementById('figure\_size').value), borderSize = parseInt(document.getElementById('border\_thickness').value), start = getStartingPoint(figSize, borderSize), border\_color = document.getElementById('div\_border').value, background\_color = document.getElementById('canvas\_background').value){

c.beginPath()

c.arc(start[0], start[1], figSize/2, 0, Math.PI \* 2)

c.closePath()

c.lineWidth = borderSize

c.strokeStyle = border\_color

c.stroke()

c.fillStyle = background\_color

c.fill()

storedFigures.push({

forme: 'circle',

figSize: figSize,

borderSize: borderSize,

start: getStartingPoint(figSize, borderSize),

borderColor: border\_color,

backgroundColor: background\_color

})

}

function getStartingPoint(figSize, borderSize){

let x = (Math.random()\*(innerWidth - figSize - borderSize)) + borderSize

let y = (Math.random()\*(innerHeight - figSize - borderSize)) + borderSize

return [x,y]

}