CSI 3140 - Summer 2025

LAB 7: Interactive Shape Creator

Due Date: Monday July 14th at 11:59PM EST

Compress all your files and upload the Zip file to the Brightspace. Please name the file using the following format: `CSI3140_Lab_7_<student_id_1>_<student_id_2>.zip`

Objective:

The goal of this lab is to build a dynamic web application that combines data loading from an external XML file with interactive user input. You will use the HTML5 Canvas to render graphics, JavaScript to parse XML, and an HTML form to allow users to add new content dynamically to the canvas. This will reinforce your understanding of the Canvas API, XML, DOM manipulation, and event handling.

Data Structure (XML)

Create an XML file named `shapes.xml`. This file will serve as the initial data source for shapes to be drawn on the canvas.

- The XML must have a single `root element` named <shapes>.
- Inside the root element, define at least three different `<shape> elements`.
- Each `<shape>` element must contain the following `child elements`:
  - `<type>`: The kind of shape (e.g., "rectangle" or "circle").
  - `<color>`: The fill color for the shape (e.g., "red" or a hex code "#FFC0CB").
  - `<x>`: The starting x-coordinate.
  - `<y>`: The starting y-coordinate.
  - `<width>`: The width of the shape.
  - `<height>`: The height of the shape. For circles, this value will be used as the `radius`.

User Interface (HTML & CSS)

Create a user-friendly, two-column layout. The left column will contain the canvas and a load button, and the right column will contain a form for creating new shapes.

HTML (`index.html`) Requirements:

- A `<canvas>` element with a specific `id` (e.g., *myCanvas*).
- A `<button>` with an `id` (e.g., *loadButton*) labeled "Load Initial Shapes from XML".
- A `<form>` element with an `id` (e.g., *createShapeForm*). The form must not reload the page on submission.
- Inside the form, include the following `inputs` with appropriate `<label>`:
  - A `<select>` dropdown for the shape type ("rectangle" or "circle").
  - An `<input type="color">` for the shape color.
  - Four `<input type="number">` fields for `x`, `y,` `width`, and `height/radius`.
  - A submit `<button>` labeled "Create and Add Shape".

CSS (`style.css`) Requirements:

- Style the page to have a clean, modern look.
- Use `Flexbox` or `Grid` to create the two-column layout for the `canvas` and the `form`.
- Add a `border` to the `canvas` to make it clearly visible.
- Style the form `elements` to be well-spaced and easy to use.

Program Logic (JavaScript)

Your `script.js` file will manage the application's state and interactivity.

1. Global State:
   - You must use a single `global` variable, `shapes = []`, to store all the `shape` objects that are currently on the `canvas`. This `array` will be the "single source of truth" for what needs to be drawn.
2. Event Listeners:
   - Wrap your entire script in a `DOMContentLoaded` event listener to ensure the HTML is loaded before your script runs.
   - Attach a `click` event listener to the "Load Initial Shapes" button. This will call a function to handle loading from XML.
   - Attach a `submit` event listener to the shape creation form. This will call a function to handle adding a new shape.
3. `loadShapesFromXML()` function:
   - This function should be `asynchronous` (use async/await).
   - Use the `fetch()` API to load and read your `shapes.xml` file.
   - Use the `DOMParser` API to parse the XML text into an XML document.

- o Iterate through each `<shape>` element in the parsed XML. For each one, create a JavaScript `object` with properties (*type*, *color*, *x*, etc.) and push it into the `global` shapes `array`.
- o After the `loop` finishes, call your main drawing function (`drawAllShapes()`) to render the newly loaded `shapes` onto the `canvas`.

4. `handleCreateShape(event)` function: This function is triggered by the `form's` submit event. It should:

- o Call `event.preventDefault()` function immediately to prevent the browser from reloading the page.
- o Read the current values from all the `inputs` inside the `form`.
- o Create a new `shape` object using these values. Remember to parse the numeric values using `parseInt()`.
- o Push this new `shape` object into the `global` shapes `array`.
- o Call `drawAllShapes()` to update the `canvas` with the newly added `shape`.

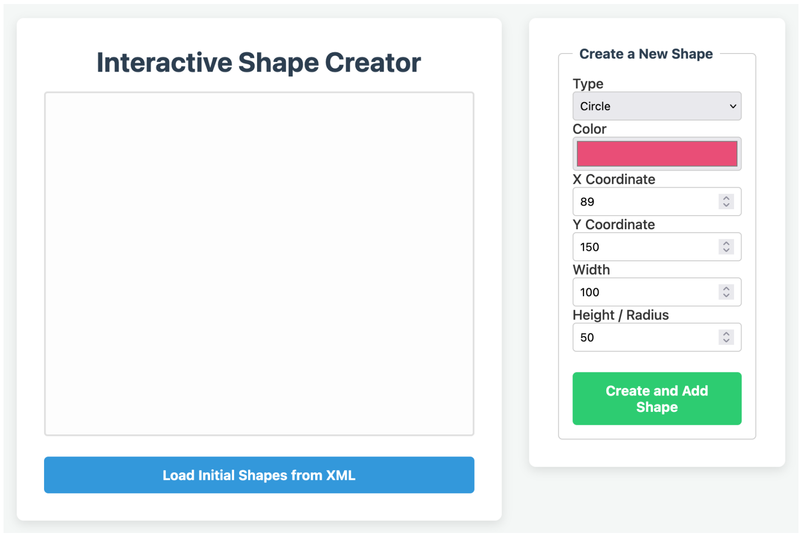
5. `drawAllShapes()` function: This is your central rendering function. It should not take any arguments.

- o The first thing this function must do is clear the entire canvas using the `ctx.clearRect(0,0,canvas.width,canvas.height)` command. This prevents old drawings from remaining on screen.
- o `Loop` through the `global` shapes `array`.
  - ▪ For each `shape` object in the `array`, use a `switch` or `if/else` statement to check its `type`.
  - ▪ Based on the `type`, use the correct `canvas` 2D context methods to draw the `shape` with its specified color, position, and dimensions (`fillRect()` for rectangles; `beginPath(), arc(), and fill()` for circles).

Application Flow

1. When the page first loads, the user sees a blank `canvas` and the `form`.
2. The user clicks the "Load Initial Shapes from XML" button. The `canvas` is cleared, and the `shapes` defined in `shapes.xml` are drawn.
3. The user fills out the `form` to define a new `shape` (e.g., a red circle).
4. The user clicks the "Create and Add Shape" button. The new shape (e.g., red circle) appears on the `canvas` in addition to existing `shapes`. The `form` fields remain unchanged.
5. The user can add more `shapes`, and they will continue to be added to the `canvas`.
6. If the user clicks "Load Initial Shapes from XML" again, the `canvas` will be cleared and reset to show only the original `shapes` from the XML file.

Example:



Create a New Shape

Type: Circle

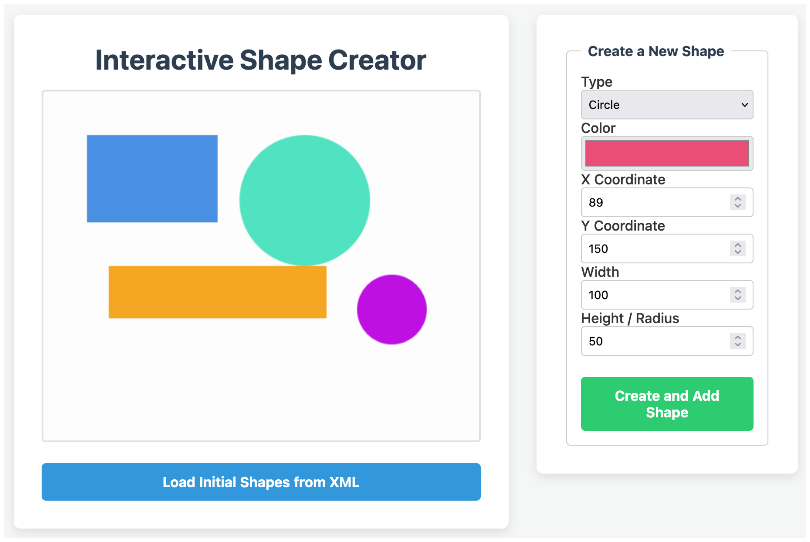
Color

X Coordinate: 89

Y Coordinate: 150

Width: 100

Height / Radius: 50



Create a New Shape

Type: Circle

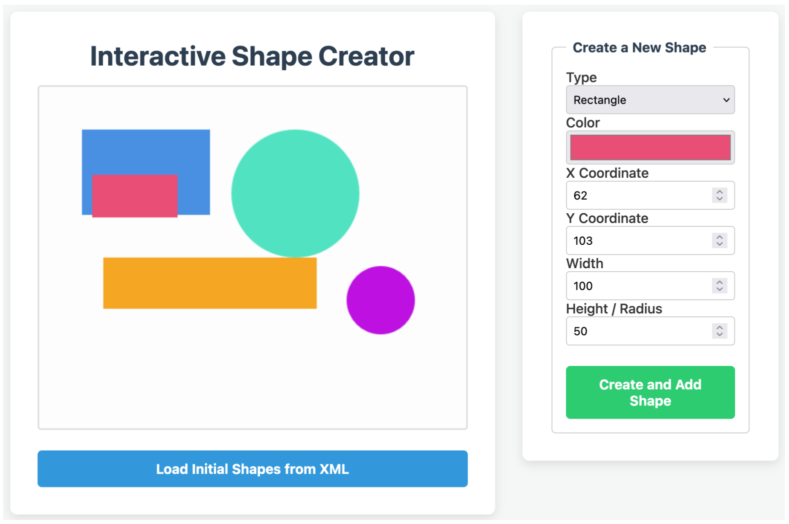
Color

X Coordinate: 89

Y Coordinate: 150

Width: 100

Height / Radius: 50



Create a New Shape

Type: Rectangle

Color

X Coordinate: 62

Y Coordinate: 103

Width: 100

Height / Radius: 50