Practical 2 Instructions

📋 Assessment Rules

This is an OPEN BOOK assessment - NO AI tools allowed.

- You MAY use: documentation, course notes, previous assignments, and web searches with Qwant NO GOOGLE DUE TO AI ANSWERS
- You MAY NOT use: ChatGPT, Claude, Copilot, or any other AI assistants
- AI-free search engine: Qwant.com

⚠️ Quick Tip

Partial credit is available for each section. You can receive credit for properly setting up webpack and babel configuration files even if the bundled version doesn't run perfectly. Attempt all parts of the assignment and include all your work in the submission.

🔧 Setup

1. Download `practical-2-start.zip` from the Assignment Dropbox.
2. Unzip the file. The folder should contain:

```
practical-2-start/
├── index.html
├── styles/
│   └── styles.css
└── src/
    ├── app.js
    ├── drawUtils.js
    └── mathUtils.js
```

3. Be aware that this zip includes a mac-specific sub-folder called _MACOSX. Feel free to delete it and make sure that you are in the “deeper” practical-2-start folder as your root folder in VS Code.

🧰 `index.html`

Make the following enhancements:

- Add Bulma via CDN in the `<head>` section via a `<link>` tag: https://cdn.jsdelivr.net/npm/bulma@1.0.3/css/bulma.min.css
- Use Bulma's components and utility classes to improve the visual structure and appearance. Most of the necessary elements are already present in the HTML—you just need to apply the right classes to the parent container. Here are the required style updates with links to examples in the relevant Bulma docs:

- A visible app title of **“Shapinator”** (this requires an additional element to be added where indicated)
  Bulma title class
- The shape selection `<select>` element
  Bulma select class
- The "Rotate" checkbox
  Bulma checkbox class
- The "Draw" button
  Bulma button class (primary)

📝 Other elements may already be styled—only apply what's missing.

🎨 `styles.css`

This file is currently empty. Add at least the following:

- A rule to apply a **monospaced font** to all text on the page.
- 💡 **Hint:** Consider the CSS cascade—make sure your monospace font rule isn't overridden by Bulma's default styles. Think about selector specificity and the order of your `<link>` tags in the HTML.
- Optional: spacing or padding rules if needed to improve layout.

🧮 `drawUtils.js`

Implement and export the function:

```
drawShape(ctx, shape, size, rotation)
```

Parameters:

- `ctx`: canvas drawing context
- `shape`: one of `"line"`, `"square"`, or `"triangle"`
- `size`: numeric value (in pixels)
- `rotation`: in radians (rotate shapes around their center)

What's provided:

- `"triangle"` and `"square"` are already implemented as examples
- Helpful comments in the code guide you on what to add

You'll add code to:

- save and restore the previous transformation matrix

- transform the drawing matrix (translate to center, then rotate)
- begin and end the path
- draw the `"line"` case (use the provided triangle and square as reference)
- export the function

Key concept: After translating to the canvas center, you're drawing at origin (0,0). To make a shape rotate around its center, draw it centered at the origin.

➗ `mathUtils.js`

This file will be an additional module containing useful math functions.

- Create and export this function:

  ```
  getRandomInt(min, max)
  ```

  which will return the resulting integer between min and max (inclusive).

- Use this expression to generate the requested random integer:

  ```
  Math.floor(Math.random() * (max - min + 1)) + min
  ```

  No default parameters are required.

🔌 `app.js`

This is the main script that connects everything together.

- Import `drawShape` from drawUtils.js and `getRandomInt` from mathUtils.js.
- Select the canvas element from the DOM and get its 2D drawing context. (Hint: you'll need both `canvas` and `ctx` variables - check how they're used in the existing code.)
- Review the rest of the code to understand its flow.

Additions:

- Randomly (using `getRandomInt`) update the shape size to a value from 50 to 200 every time the “Draw” button is clicked.
- Ensure `drawShape()` is called (the comments tell you where) with the correct parameters.
- ***Bonus Option:*** On load, dynamically set the app title (the one you styled with Bulma) to:
  `Shapinator X000`
  where **X** is a random digit from **2 to 9** (use `getRandomInt` and a **template string**).

✅ How You Know It Works

- Clicking **Draw** renders the correct shape on the canvas.
- If **Rotate** is checked, the shape spins smoothly (after clicking Draw).
- If **Rotate** is not checked, the shape stops rotating (after clicking Draw) or stays still.

⚙️ Transpiling (Webpack + Babel)

After your code works:

1. In your project folder, initialize npm:

   ```
   npm init -y
   ```

2. Install the required development dependencies:

   ```
   npm install --save-dev webpack webpack-cli @babel/core babel-loader @babel/preset-env
   ```

3. Configure Webpack and Babel:

   - Create `webpack.config.js` and configure the entry key for this project.

     HINT: See the abridged bundling and transpiling guide: https://github.com/rit-igm-web/igme-330-shared/blob/main/notes/bund-transp.md

   - Modify `package.json` so that you have scripts to run it.
   - Add Babel configuration so that it both bundles AND transpiles to ES5.

4. Build your project:

   ```
   npm run start
   ```

5. Add a `<script>` tag *where it needs to go* linking to `bundle.js` in your `index.html`.

📤 What to Submit

1. Assignment Dropbox Version

- Rename your folder:
  `lastname-firstinitial-practical-2`
- Delete `node_modules`
- Zip the remaining folder
- Upload the `.zip` to the Assignment Dropbox

⚠️ Make sure to delete `node_modules` before zipping. Keep everything else including your `src/` folder, configuration files, and the bundled output.