Checklist: Project 5

Project rejected without review

- The console logs any errors upon opening `index.html`.
- Any of the features are not implemented, e.g. initial cards are not rendered, the form can't be opened or closed, etc.
- Required corrections from a previous iteration (or iterations) haven't been completed.
- There are questions addressed to the reviewers in the project.

Project requirements

General

- The project contains:
  - A `blocks` directory with BEM blocks stored in separate CSS files (Flat BEM)
  - An `images` directory with all the images
  - A `pages` directory with an `index.css` file stored inside it
  - A `scripts` folder with `index.js` script file
  - A `vendor` directory with `normalize.css`, `fonts.css` files, and a `fonts` directory stored inside it
  - The `index.html` file
  - A `README.md` file
  - A `.prettierignore` file, which tells Prettier to ignore `normalize.css`
  - A `.gitignore` file, which tells Git to ignore `.DS_Store`
- Stylesheets are connected in a separate file.
- All the sections of the design have been coded.

- The script is moved to a separate file and included using the defer attribute or right before the closing `</body>` tag.
- The `README.md` file contains the following:
  1. The project's name
  2. A description of the project and its functionality
  3. A description of the technologies and techniques used
  4. Pictures, GIFs, or screenshots that detail the project features (highly recommended)
  5. The link to your deployed project on GitHub Pages
- There are no typos in the HTML and CSS, and the code is valid. We recommend using the W3C Markup Validator to check your code for validity.
- The code is well-formatted using the Prettier.

HTML/CSS

- `normalize.css` is imported into `index.css` before other CSS files.
- `viewport` is set correctly, `title` and `lang` are used.
- The project complies with the principles of the BEM methodology.
- The BEM file structure is used.
- Aside from the `<header>`, elements should not touch the sides of the container on any screen larger than `320px`.
- The cumulative value of the `width`, `padding`, and `margin` properties for each element is specified to be within `30px` of the design requirements at the `1440px` breakpoint.
  - It should also be within `10px` of the design requirements at `320px`.
  - In the intermediate dimensions, the layout should look similar to either the desktop or mobile view.

JavaScript + HTML/CSS

All of the features listed in the brief have been implemented and are functioning correctly:

- 6 cards have been created and rendered via JavaScript.
- The form for adding cards has been correctly coded in HTML and CSS. The form can be opened, and submitting this form adds a card.
- The like button is functional.
- The feature for removing cards has been implemented correctly.
- A card can be added by pressing Enter while a text field is active.

Modal windows:

- The popups are created with HTML and CSS. They shouldn't be created dynamically with JS.
- Popup boxes can be closed at any screen resolution.
- The popup box with the picture opens correctly, and images display properly with their aspect ratios maintained.
- The popup opens and closes smoothly using CSS.

Code is optimized:

- Any data entered by the user must not be assigned to the `innerHTML` property.
- There is no duplicate code. If a line of code has to be repeated, it must be written as a separate function.
- All numeric values are assigned to variables. Unique values that do not have their own variable are called "magic numbers," and those are considered a bad practice in programming.
- Operations on DOM elements are executed before they are inserted into the layout.
- If a variable is declared using `let`, its value must change somewhere.
- If a variable's value won't change, it is declared using `const`.
- A function performs a single task, e.g., returns the card HTML code.

The project complies with the following code style requirements:

- camelCase is used for function and variable names.
- Only nouns are used as variable names.
- Variable names clearly describe what is stored in them. If the project has several variables with similar data, then those variables have unique but descriptive names.
- Plural nouns are used for NodeList collections.
- Descriptive names are used for functions that reflect what they do.
- Function names start with a verb.
- Names must not include inappropriate or unclear abbreviations.

Interface accessibility

- All links and interactive elements have a `:hover` state.
- All `<img>` elements have an `alt` attribute containing a description in the page language.