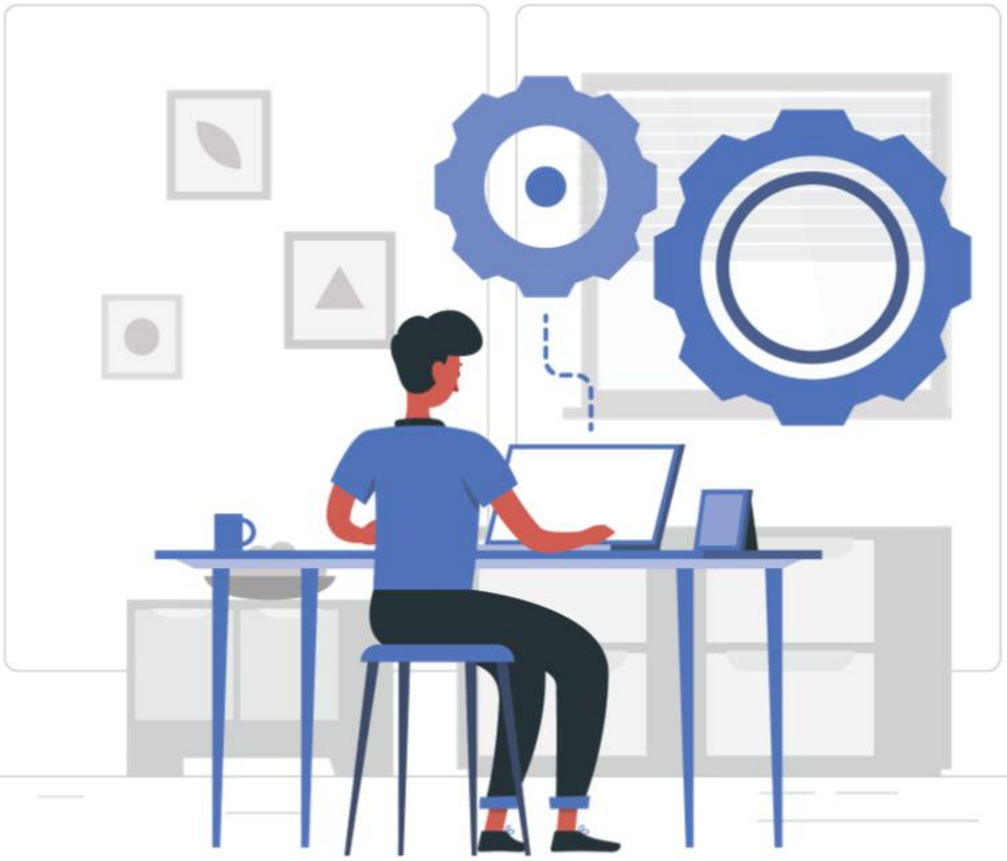


Practice

# Develop Interactive Web Pages Using DOM and DOM Events

# Practices

- Practice 1: Add a new note to the To-Do list
- Practice 2: Display all notes from the To-Do list
- Practice 3: Create a Card with Delete Button to delete a note (Optional)
- Practice 4: Toggle the Notes View (Optional)

# Context

Keep App is created by Creative Coders – a software development company. This App helps a user to maintain his daily To-Do tasks. A user can add a new note, view existing notes, and delete the selected note through this App.

For convenience, the App provides a feature that allows a user to view the notes in the form of a grid or a list.

In the current stage, the UI of the App is ready. You will be required to enable the functionalities in the App that will allow adding, displaying, and deleting operations on note data. Also, enable the toggle view functionality to switch between the grid and list views.

# Instructions for Accessing the Boilerplate

- Click here for the boilerplate.
- Please read the README.md file provided in the boilerplate for further instructions about the practice exercise.
- Read the README.md file available in the boilerplate for further instructions about the practices.
- Fork the boilerplate into your own workspace.
- Clone the boilerplate into your local system.
- Open command terminal and set the path to the folder containing the cloned boilerplate code.
- Run the command **`npm install`** to install `Mocha` and `Chai` as dependencies.
- Open the folder boilerplate code in VS Code.
- Provide the solution code in the files specified with the task details.

# Instructions for the Practices

- The first two practices are mandatory whereas the last two practices are optional.
- The unzipped code contains **index.html** file which contains the design code of the `Keep app` web application.
- The CSS code to style the web page is available in the **styles.css** file located inside the **css** folder of the boilerplate.
- Write the code as per the requirements stated in the upcoming slides in **script.js** file. The file is located inside the **js** folder of the boilerplate.
- Use the `<script>` tag in **index.html** file to refer to the **script.js** file.
- After each practice, open the **index.html** file using Live Server and test the output.
- Submit the file for evaluation.

# Points to Remember

- It is recommended that you write JavaScript code in a script file external to index.html file.
- To access elements from the web page, use id selector instead of tag name or class selectors.
- Apply validations on fields using HTML5 built-in form validations.

# PRACTICE

## Practice 1: Add a New Note to the To-Do List

Write a JavaScript code that captures the details of the note entered by the user.

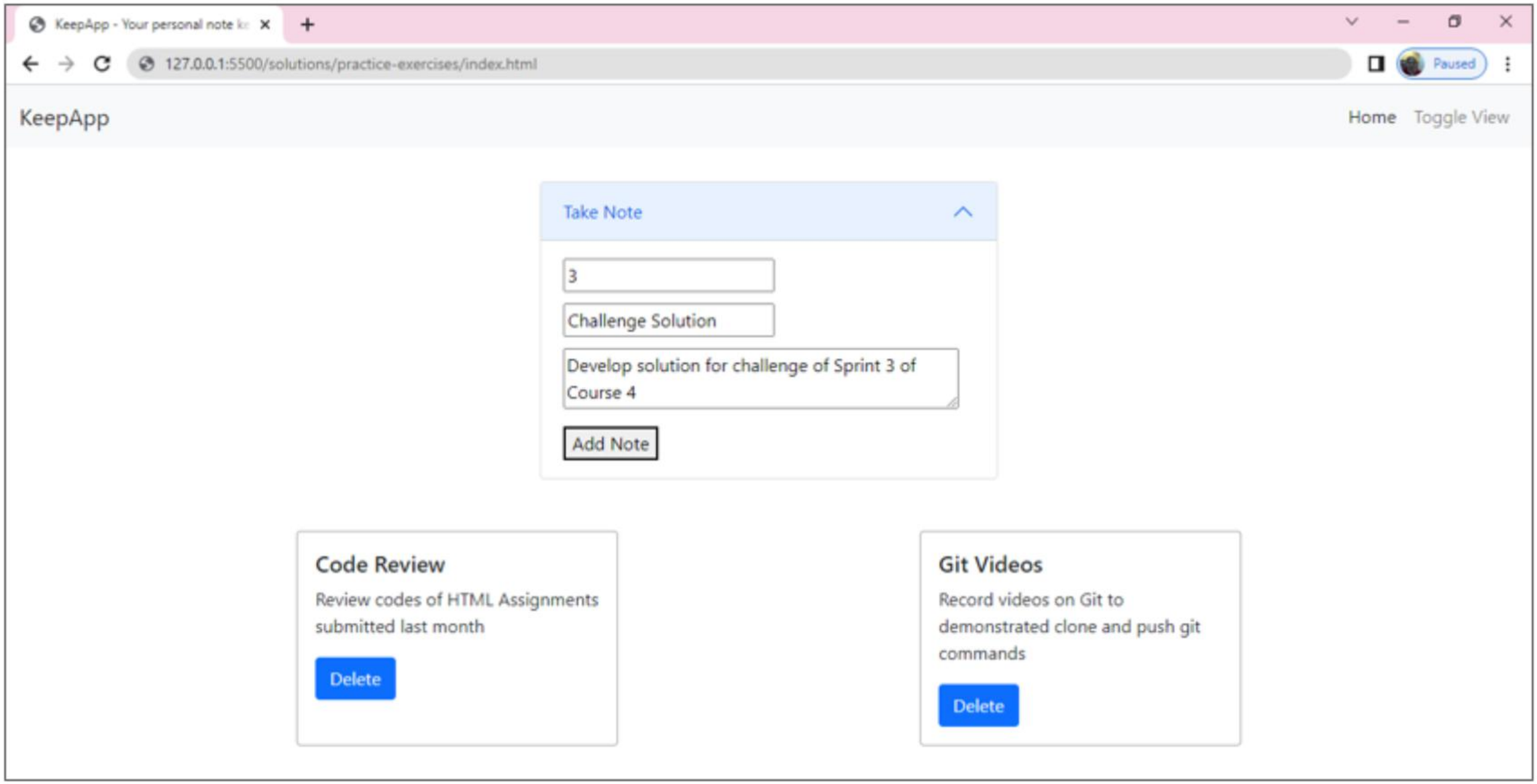
The code should save the captured details and store them as an object in the notes array.

Note: Steps to do this practice are given in the upcoming slide.

# Steps to Add a New Note to the To-Do List

- Step 1: Define JavaScript function **`saveNote()`** in the **script.js** file. The function **`saveNote()`** should:
  - Read the note details inputted by the user on the web page.
  - Implement validation to check for missing note details.
  - Display confirmation message if the note details are provided and the note gets stored in the notes array.
- Step 2: The function **`saveNote()`** should get invoked when the user clicks the `Add Note` button.

# Expected Output – Entering Note Details



PRACTICE

## Practice 2: Display All Notes From the To-Do List

Write a JavaScript code that reads notes data from the notes array and displays them on the UI.

Each of the notes' data should be displayed in a card layout.

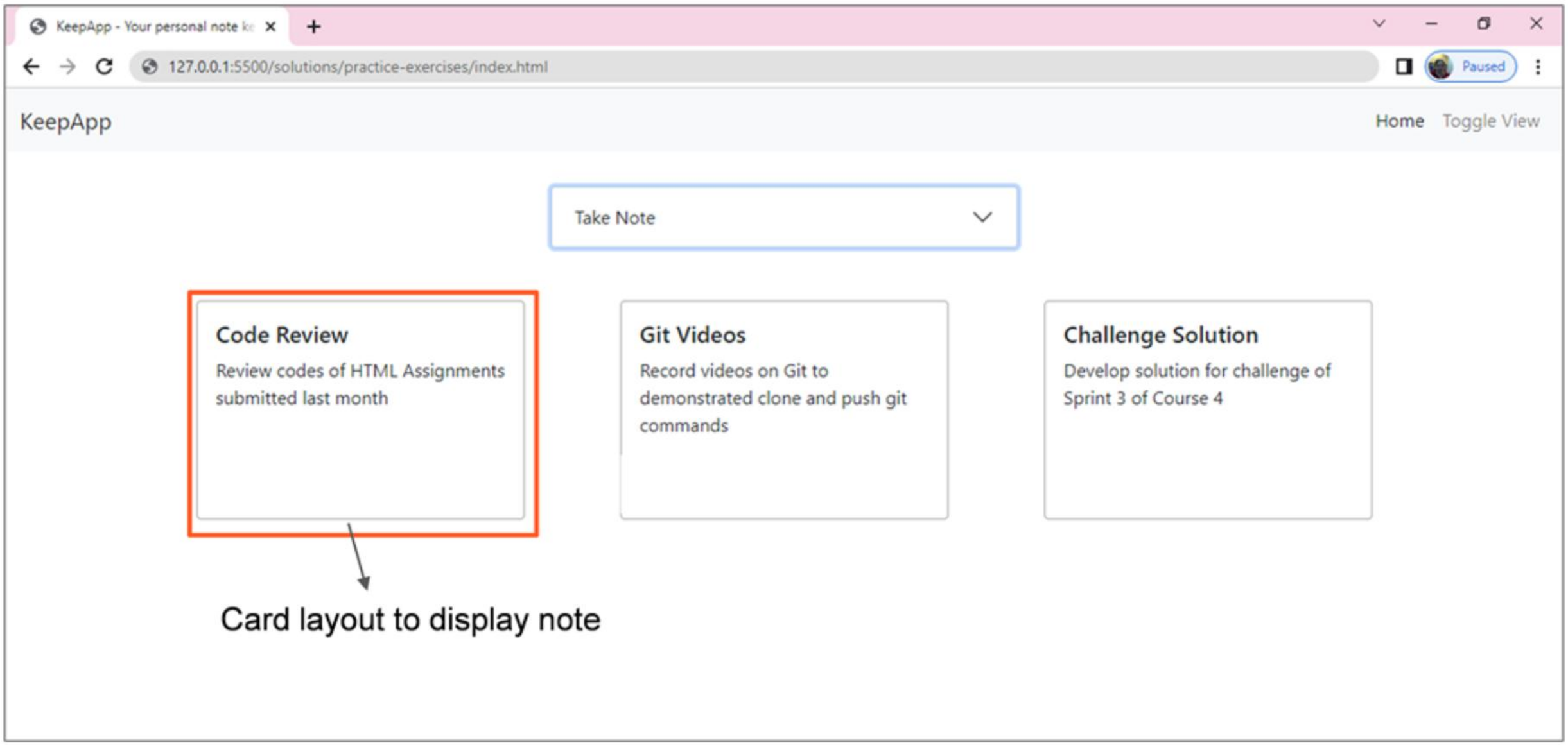
Note: Steps to do this practice are given in the upcoming slide.

# Steps to Display All Notes From the To-Do List

- Step 1: Define JavaScript function **`displayNotes()`** in the **script.js** file. The function **`displayNotes()`** should:
  - Read the details of notes from the notes array.
  - Dynamically add a `div` element for each note in the array.
    - The `div` element should contain:
      - A `heading` element to display the title of the note.
      - A `para` element to display content of the note.
    - The `div` element should be applied with style properties to display the note in a card layout.
- Step 2: The function **`displayNotes()`** should get invoked when the page loads.

Note: The Expected Output is shown in the upcoming slide.

# Expected Output



# PRACTICE

## Practice 3: Create a Card with Delete Button to Delete a Note

Modify JavaScript code that adds a new note to the existing note list. The code should now add a Delete button to each card displaying the notes data.

Upon clicking the button, the corresponding note should get deleted.

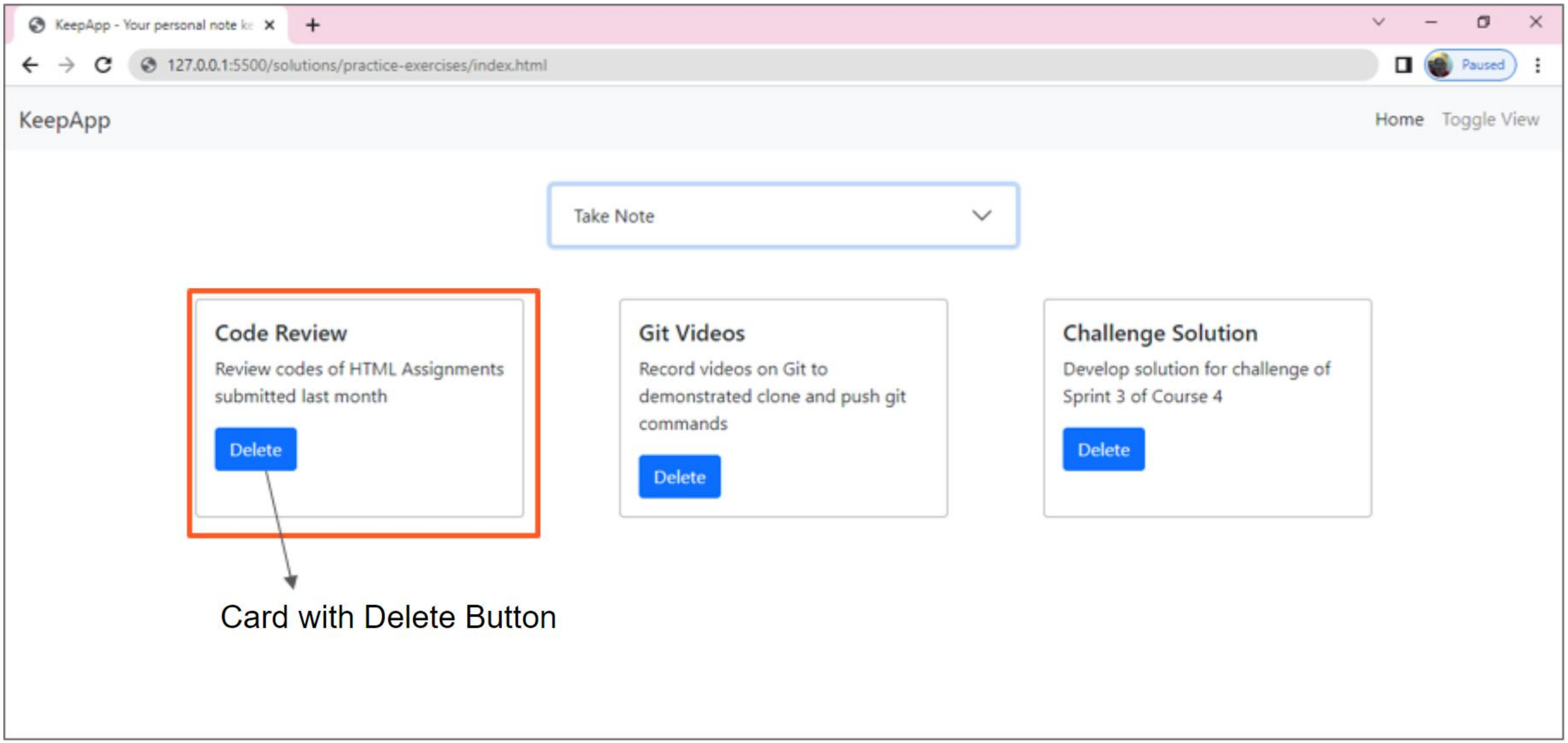
Note: Steps to do this practice are given in the upcoming slide.

# Steps to Create a Card With Delete Button to Delete a Note

- Step 1: In the script.js file, modify the **`displayNotes()`** function code to dynamically add a button to delete the note.
  - Style the button to make it look raised.
- Step 2: Define JavaScript function **`deleteNote()`** to delete the selected note and remove it from the notes array.
  - The function should:
    - Accept id of note as the parameter
    - Find the note by the id in the array and remove the note
    - Upon deletion, display the confirmation message.
- Step 3: The function **`deleteNote()`** should be used as the event handler method for the delete button.

Note: The Expected Output is shown in the upcoming slide.

# Expected Output



PRACTICE

## Practice 4: Toggle the Notes View

The notes on the UI should by default be displayed in the grid view.

Provide an option on the UI that will allow users to toggle between the grid view and the list view.

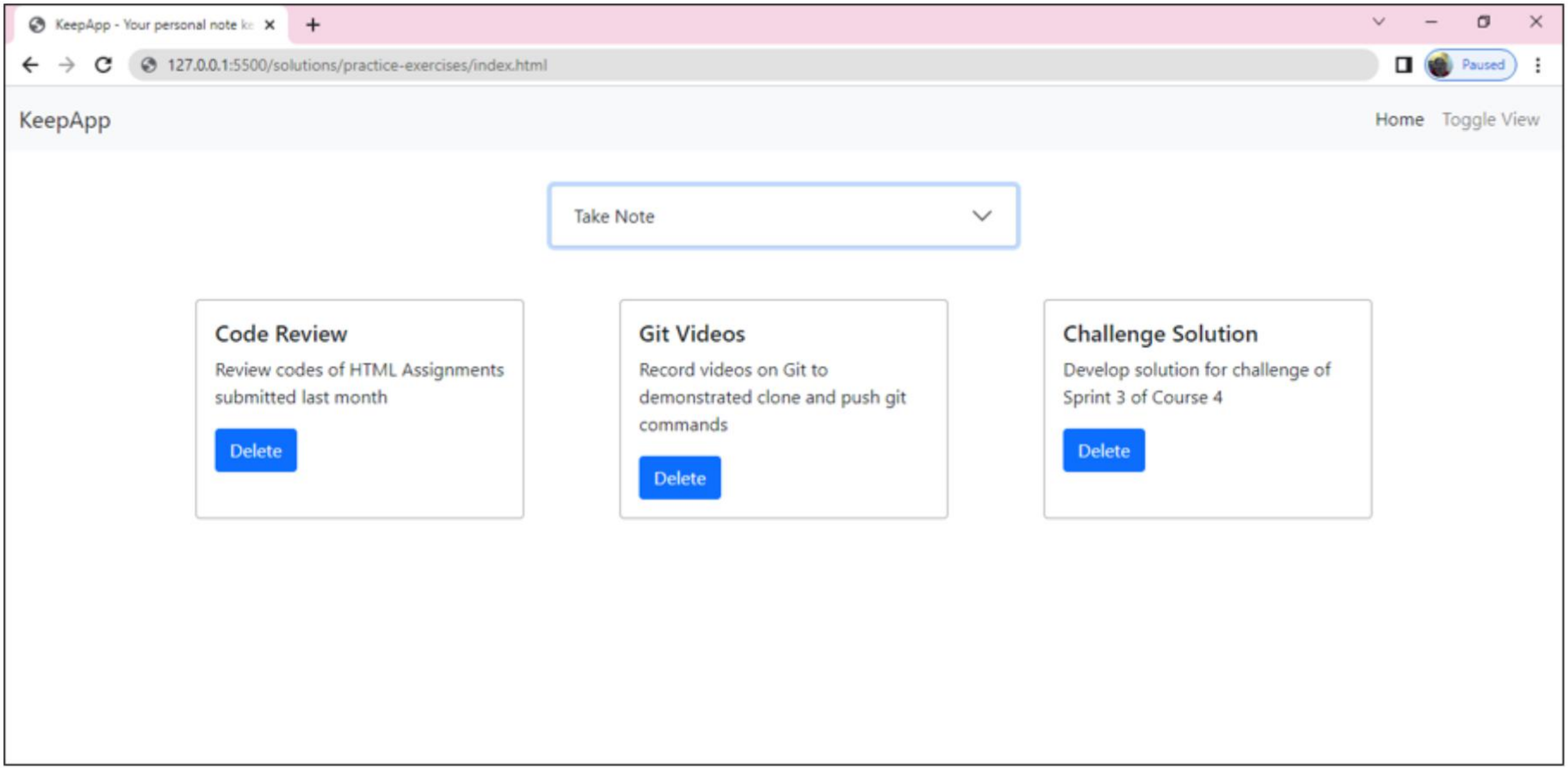
Note: Steps to do this practice are given in the upcoming slide.

# Steps to Toggle the Notes View

- Step 1: In the **script.js** file, define function **`toggleView().`** The **`toggleView()`** function should dynamically add styles/CSS class that toggles the view between grid view and list view.
- Step 2: The function **`toggleView()`** should get invoked when the user clicks on the link with the text "Toggle View" .

Note: The Expected Output is shown in the upcoming slides.

# Expected Output – Grid View



# Expected Output – List View