SPARK SUPPLEMENTARY MATERIAL

A. AI Testing Code Generation Prompt

Generate Puppeteer code to achieve the required interaction and evaluation for: [`taskDescription`]. The reference code answer is: [`referenceCode`]. Evaluation should be done by getting the element, get the requirement from the element, and return if the requirement is met. If no interaction is needed, just evaluate. Do not reply with any natural language text, only the JavaScript code. Do not include any comment. A reply example is as follows: `const selector = ''; await page.click(selector);`

B. User Study Recruitment Table

TABLE I
WE RECRUITED 16 PARTICIPANTS WITH WEB PROGRAMMING EXPERIENCE.

PID	Gender	Web Prog. Proficiency
1	Female	Intermediate
2	Female	Intermediate
3	Male	Intermediate
4	Female	Beginner
5	Female	Intermediate
6	Female	Beginner
7	Female	Intermediate
8	Female	Advanced
9	Female	Intermediate
10	Male	Beginner
11	Female	Beginner
12	Female	Beginner
13	Female	Intermediate
14	Male	Intermediate
15	Female	Beginner
16	Male	Intermediate
17	Male	Intermediate
18	Female	Beginner
19	Female	Intermediate
20	Female	Beginner
21	Male	Beginner
22	Female	Beginner

TABLE II
WE RECRUITED 16 PARTICIPANTS WITH TEACHING EXPERIENCE AND WERE FAMILIAR WITH WEB PROGRAMMING.

PID	Gender	Web Prog. Exp.	Teaching Exp.
1	Male	1 year	Teaching Assistant
2	Female	2 years	Teaching Assistant
3	Female	1 year	Teaching Assistant
4	Female	1 year	Teaching Assistant
5	Female	2 year	Teaching Assistant
7	Female	3 years	Teaching Assistant
8	Male	1 year	Instructor
9	Male	More than 6 years	Teaching Assistant
10	Male	3 years	Tutor
11	Female	1 year	Teaching Assistant
12	Female	2 years	Tutor
13	Female	3 years	Instructor
14	Female	2 years	Teaching Assistant
15	Male	3 years	None
16	Male	2 years	Teaching Assistant

C. Exercise: Create a To-Do List Webpage

In this exercise, you will create a simple to-do list webpage using HTML, CSS, and JavaScript. You will work on three files: *index.html, script.js, and styles.css*. Follow the instructions below to build and style the webpage.

1) HTML Structure:

- **Title:** Add a title `<h1>` at the top of your webpage with the text `"Todo List"` and ID `#pageTitle`. Set the font size to `25px` and make the text `bold`.
- **Input Section:**
  - Create an input box `<input type=\text">` where users can type their tasks. Give this input box the ID `input`.
  - Add a button next to the input box with the text `"Add"` and the ID `#addBtn`.
  - Place the input box and the button inside a `<div>` element with the ID `#inputContainer`.
  - Make sure the input box and the button are aligned horizontally in the same row and centered within this `<div>`.
- **Todo List Container:** Create an empty `<div>` container below the input section where the tasks will appear. Give this container the ID `#todoList`.

2) Add Button Interactivity:

- **Adding Items:** When the Add button is clicked, a new task should be added to the `#todoList` container you just created. Each new task has the class name `.todoItem`. "Adding and Removing Classes" in the Cheat Sheet for syntax guidance.
- Task Structure: Each todoItem should include two elements with class name `.itemContent` and `.deleteBtn`: .itemContent: The text of the task from `<input>`. .deleteBtn: A button with the text `"Delete"`.
- **Styling the Items:** Set the width of each `.todoItem` to `350px`. Ensure that `.itemContent`, and `.deleteBtn` are aligned in the same row, with space between the text and the Delete button.

3) Delete Button Interactivity:

- **Deleting Tasks:** When the Delete button is clicked, the entire `.todoItem` should be removed from the list.
- **Styling the Delete Button:**

- Set the background color of the Delete button to `red`.
- When you hover over the Delete button, the background color should change to `darkred`.

Follow these requirements to create your to-do list page, making sure your web page works as described.

D. Exercise: Create an Image Carousel

In this exercise, you will create a simple image carousel webpage using HTML, CSS, and JavaScript. You will work on three files: *index.html, script.js, and styles.css*. Follow the instructions below to build and style the webpage.

1) HTML Structure:

- **Title:** Add a title `<h1>` at the top of your webpage with the text `"Gallery"` and the ID #pageTitle. Set the font size to `25px` and make the text `bold`.
- **Thumbnails Section:** Create a `<div>` container below the title for the thumbnails. Give this container the ID `#thumbnails`. This container will eventually display a series of images.
- **Featured Image Section:**
  - Below the thumbnails section, add a `<div>` container with ID `#featured_container`.
  - Add an `img` element with the ID `#featured` in the `#featured_container`. This image will display a larger version of the selected thumbnail.
  - Also, include a `div` below the `img` element with the ID `#current_description` in the `#featured_container` to show the description of the currently selected image.

2) Thumbnails Generation:

- **Images Gallery:**
  - We have provided an array called images in your script.js in your starter file. Each element in this array should be an object containing three properties: url, alt, id, and description. Write a script that loops through the images array and creates a new `<img>` element for each image.
  - Each `<img>` element should be appended to the thumbnails container. And it should has
    - Its `src` attribute is set to the `url` from images
    - Its `alt` attribute is set to the `alt` from images
    - Its `id` attribute is set to the `id` from images
- **Styling the Thumbnails:**
  - Ensure all thumbnail images are displayed in a horizontal row and centered in the thumbnails container.
  - Each thumbnail image should have a width of `100px`.

3) Making Images Clickable:

- **Click an image:**
  - Add interactivity so that when a user clicks on a thumbnail, the `src` and `alt` attributes of the image `#featured` element are updated to show the selected image.
  - Also, update the `#current_description` element to display the description from images associated with the clicked image.
- **Highlight the selected image:**
  - Ensure that the currently selected thumbnail image is highlighted with a `1px solid red` border.
  - Make sure that only one image is highlighted at a time. When a new thumbnail is clicked, remove the highlighted class from any previously highlighted image.
- **Styling the Featured Image:** The featured image should be centered and have a width of `500px`.

Follow these requirements to create your to-do list page, making sure your web page works as described.

E. Quiz Questions Example

1) How many students are currently enrolled in the class?
2) Students were expected to achieve the following learning outcomes before the in-class exercises. Based on their performance, is there a learning objective you would like to revisit at the beginning of your next lecture? Why? *(You may consider this question throughout the entire observation.)*
   - Getting references to DOM nodes, for example with `querySelector()` and `getElementById()`.
   - Creating new nodes, for example with `innerHTML()` and `createElement()`.
   - Understanding how to modify the layout using flexbox.
   - Adding and removing nodes to the DOM with `appendChild()` and `removeChild()`.
   - Setting attributes to elements, e.g., `font-size`.
   - Manipulating styles with `Element.style.*` and `Element.classList.*`.
   - I am not sure.
3) After the third timestamp, for students who didn't set the `pageTitle` font size to 25px, what font size did they use instead?
   - 30px
   - 32px
   - 24px
   - 16px
   - I am not sure.
4) After the 4th timestamp, please enter at least 1 student number for those who have set the font size to 25px.
5) In the sixth timestamp, how many students have correctly implemented all the requirements in the Basic Setup checkpoint according to the rubric?
   - 0–6 students
   - 6–11 students
   - 11–16 students
   - More than 17 students
   - I am not sure.

6) From the students who have not yet completed the programming problem, select one who appears to be struggling. Review this student's programming progress history and provide feedback.

7) After the 8th timestamp, explore freely with the tool to review students' implementation of the Add Button Interactivity and explain any issue you observed.

8) In the 15th timestamp, among these students, who is furthest from correctly finishing this programming problem?
   - Student 3
   - Student 5
   - Student 6
   - Student 7
   - I am not sure.

9) After the time alert, what error did you observe for the added todo item functionality that failed to meet the `space-between` requirement for `.itemContent` and `.deleteBtn` in `.todoItem`, causing them to appear on two separate rows? What do you think might be the reason for this?

   - 0 student
   - 1 student
   - 2 students
   - 3 students
   - 4 students
   - I am not sure.