

JS I DOM NA PRZYKŁADZIE LISTY TODO

SPIS TREŚCI

Spis treści.....	1
Cel zajęć.....	1
Rozpoczęcie.....	1
Uwaga.....	2
Wymagania	2
Strona HTML.....	3
Klasa Todo.....	3
Dodawanie pozycji listy	4
Usuwanie pozycji listy	5
Edycja pozycji listy	6
Odczyt / Zapis LocalStorage.....	7
Wyszukiwanie.....	9
Commit projektu do GIT.....	11
Podsumowanie.....	11

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie następujących umiejętności:

- przemieszczania się po drzewie DOM;
- dodawania, usuwania, edytowania elementów drzewa DOM.

W praktycznym wymiarze utworzona zostanie dynamiczna lista czynności do zrobienia (lista To Do).

ROZPOCZĘCIE

Rozpoczęcie zajęć. Powtórzenie metod przemieszczania się po drzewie DOM.

Wejściówka?

UWAGA

Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

WYMAGANIA

W ramach LAB B przygotowane powinny zostać:

- pojedyncza strona HTML ze skryptem ładowanym z zewnętrznego pliku JS
- lista zadań
- na dole listy pole tekstowe do dodawania nowych zadań, pole typu data/czas do określenia terminu wykonania zadania, przycisk dodawania zadania
- walidacja nowych zadań: co najmniej 3 znaki, nie więcej niż 255 znaków, data musi być pusta albo w przyszłości
- na górze listy pole wyszukiwarki
- po wpisaniu w wyszukiwarkę co najmniej 2 znaków na liście wyświetlają się wyłącznie pozycje zawierające wpisaną w wyszukiwarkę frazę
- wyszukiwana fraza zostaje wyróżniona w każdym wyniku wyszukiwania
- kliknięcie na dowolną pozycję listy zmienia ją w pole edycji; kliknięcie poza pozycję listy zapisuje zmiany
- obok każdej pozycji listy znajduje się przycisku Usuń / Śmiertnik
- wpisy na liście zapisują się do Local Storage
- po odświeżeniu strony lista wypełnia się wpisami z Local Storage

Mockupy:

The left mockup shows a simple to-do list interface. It has a search bar at the top, followed by a list of tasks. Each task is represented by a row with a checkbox, the task name, a date (e.g., 2000-01-01 or 2000-01-05), and a delete icon. At the bottom, there are three input fields: 'do zrobienia...', a date picker, and a 'Zapisz' button.

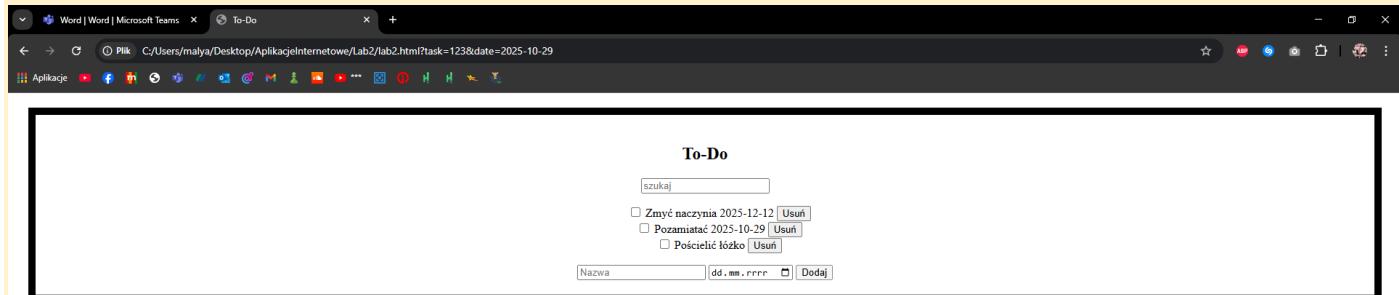
The right mockup shows the same interface but with the 'chupa chups' task selected. The task name 'chupa chups' is highlighted in yellow, and its date '2000-01-01' is displayed in an input field with a calendar icon to its left. The other tasks ('macaroon', 'candy canes', 'bon bons') are listed below it. The bottom input fields remain the same.

This mockup demonstrates the search feature. The search bar at the top contains the letter 'o'. Below it, the 'macaroon' task is listed with its name highlighted in yellow, indicating it contains the search term. The 'bon bons' task is also listed below it. The bottom input fields are identical to the previous mockups.

STRONA HTML

Prace rozpocznij od implementacji HTML z danymi wpisanymi „na sztywno”. Upewnij się, że wstawione zostały wszystkie wymagane elementy – pole wyszukiwarki, lista, pole dodawania, przycisk usuwania. To laboratorium koncentruje się na JS, więc może być ładne, ale nie musi. **Nie trać za dużo czasu na CSS** – to jest laboratorium z JS.

Wstaw zrzut ekranu przedstawiający stronę HTML z polem wyszukiwarki, listą, polem dodawania, przyciskami usuwania:



Punkty:	0	1
---------	---	---

KLASA TODO

Pierwszym instynktem może być chęć dodania zachowań bezpośrednio do elementów listy w drzewie DOM. Chociaż na krótką metę wydaje się być to najprostsze rozwiązanie, za chwilę okaże się krótkowzroczne i trudne do implementacji przy kolejnych punktach 😊

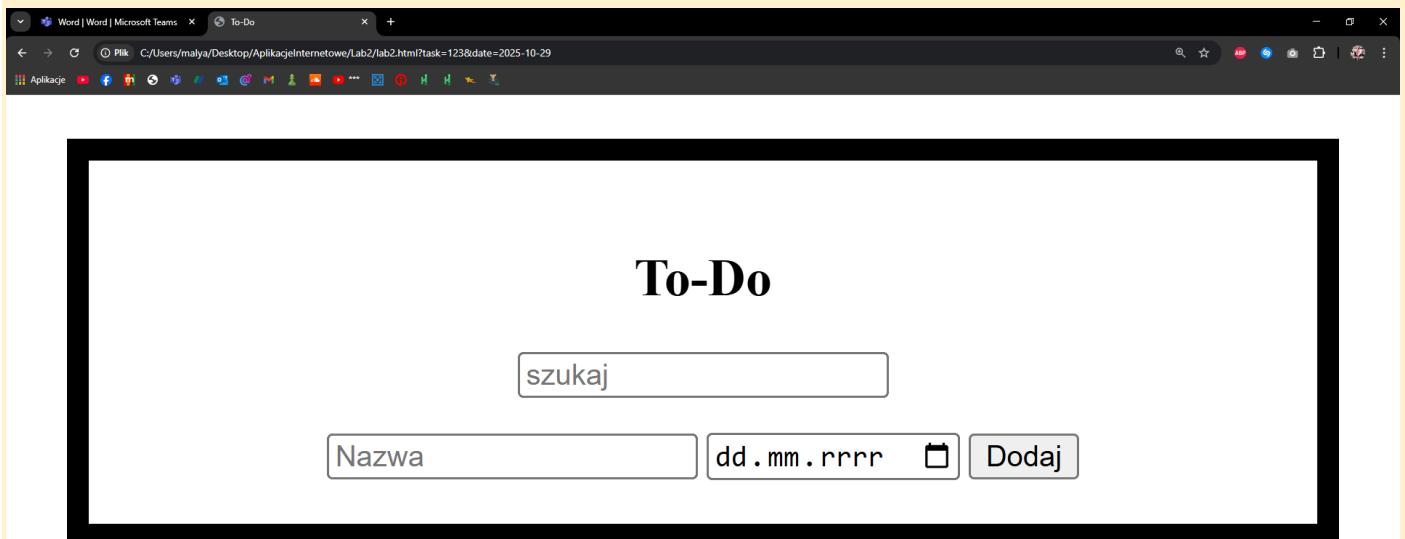
Najlepszym sposobem rozwiązania tego laboratorium jest utworzenie klasy Todo (albo po prostu obiektu z kilkoma metodami). Bez względu na przyjętą strategię, należy w tym nowoutworzonym bycie utworzyć tablicę `tasks` oraz metodę `draw()`, która wyczyści `div` z obecną wizualizacją zadań do zrobienia i wygeneruje ją na nowo na podstawie tablicy `tasks`.

W celu sprawdzenia poprawności działania, najlepiej dostać się do tablicy `tasks` i edytować jej zawartość, po czym ręcznie wywołać metodę `draw()`. Jeśli zawartość listy wyrenderuje się na nowo poprawnie – możemy iść dalej!

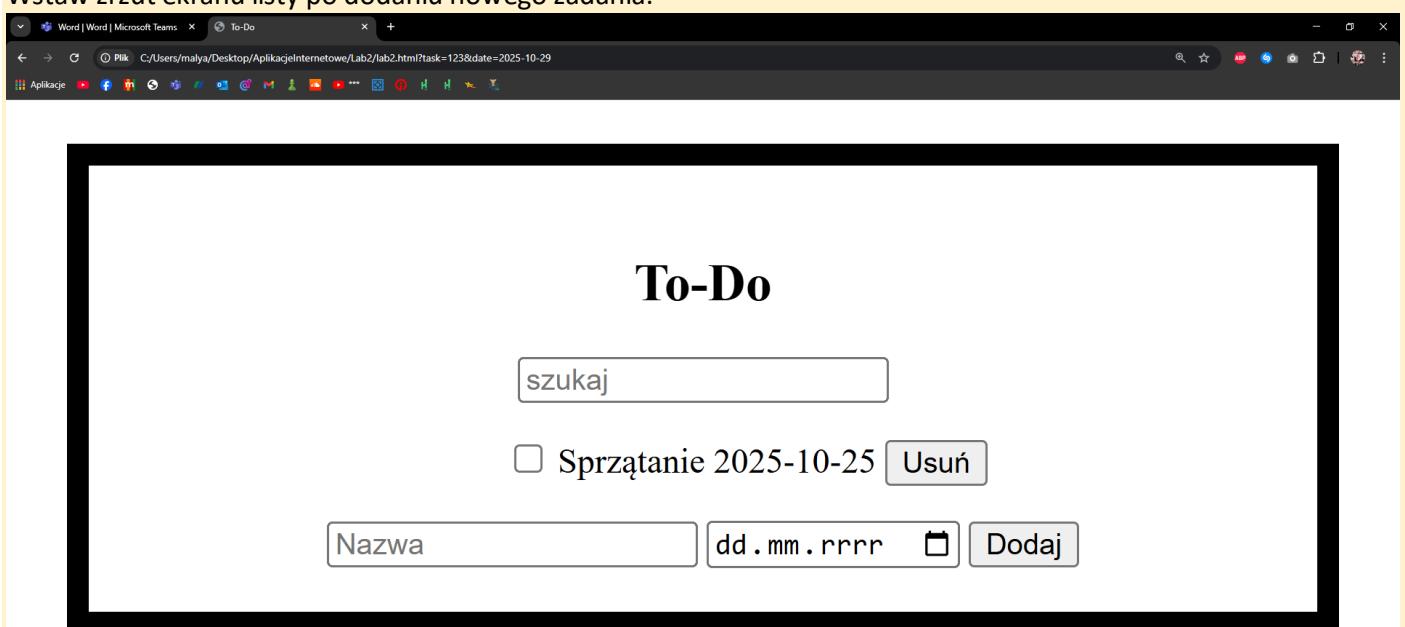
Zaimplementuj dodawanie, usuwanie, edycję pozycji listy – wszystko modyfikujące tablicę `tasks` i wywołujące na koniec metodę `draw()`.

DODAWANIE POZYCJI LISTY

Wstaw zrzut ekranu listy przed dodaniem nowego zadania:



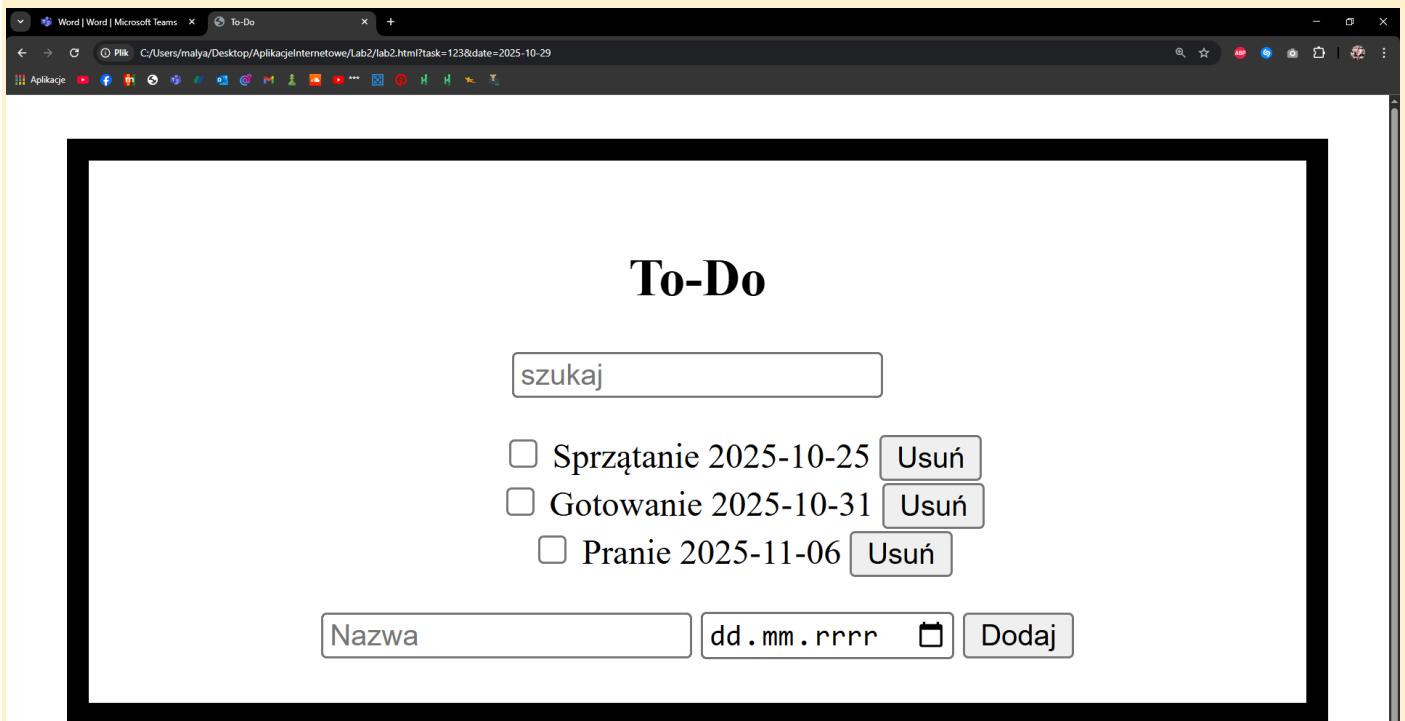
Wstaw zrzut ekranu listy po dodaniu nowego zadania:



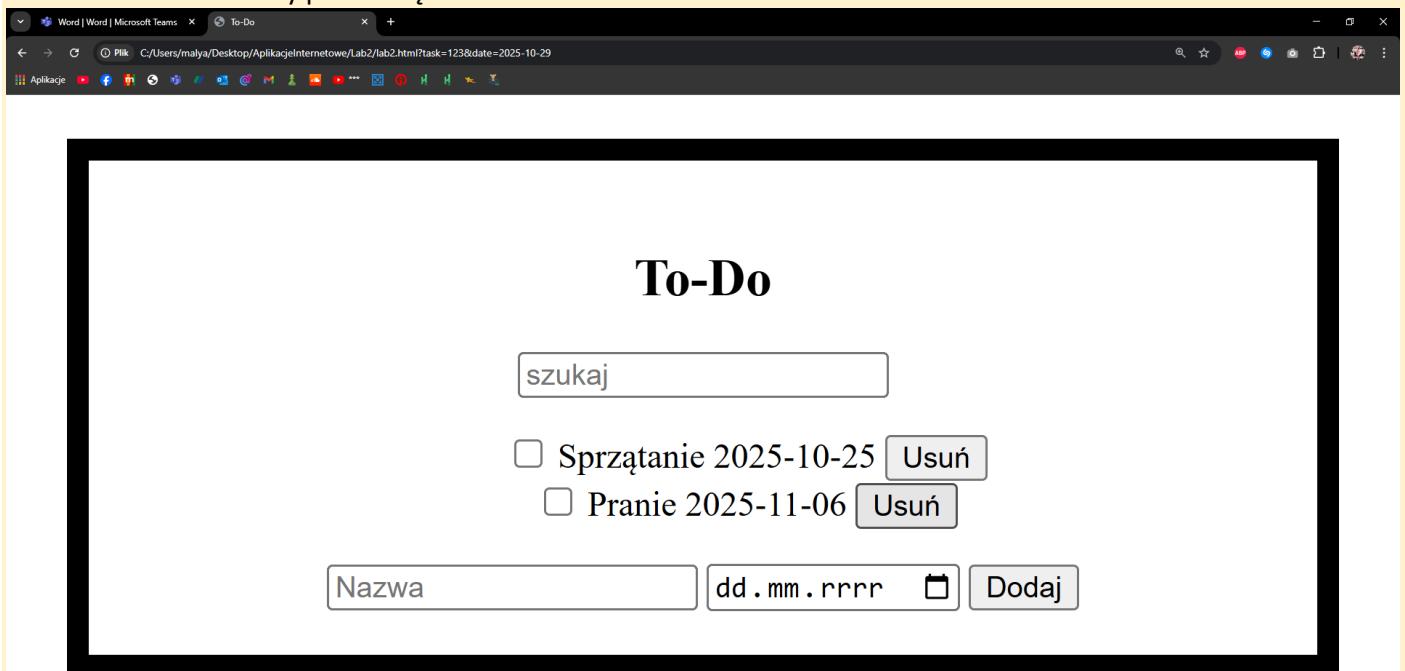
Punkty:	0	1
---------	---	---

USUWANIE POZYCJI LISTY

Wstaw zrzut ekranu listy przed usunięciem wybranego zadania:



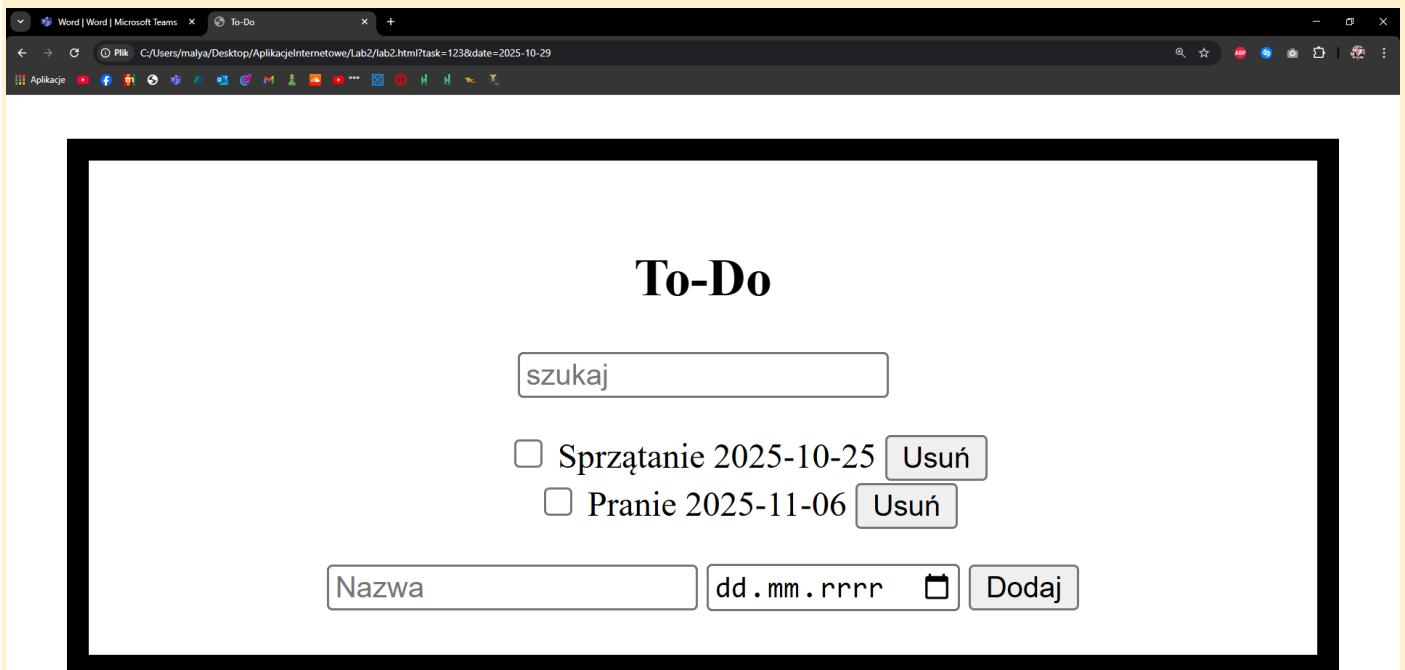
Wstaw zrzut ekranu listy po usunięciu zadania:



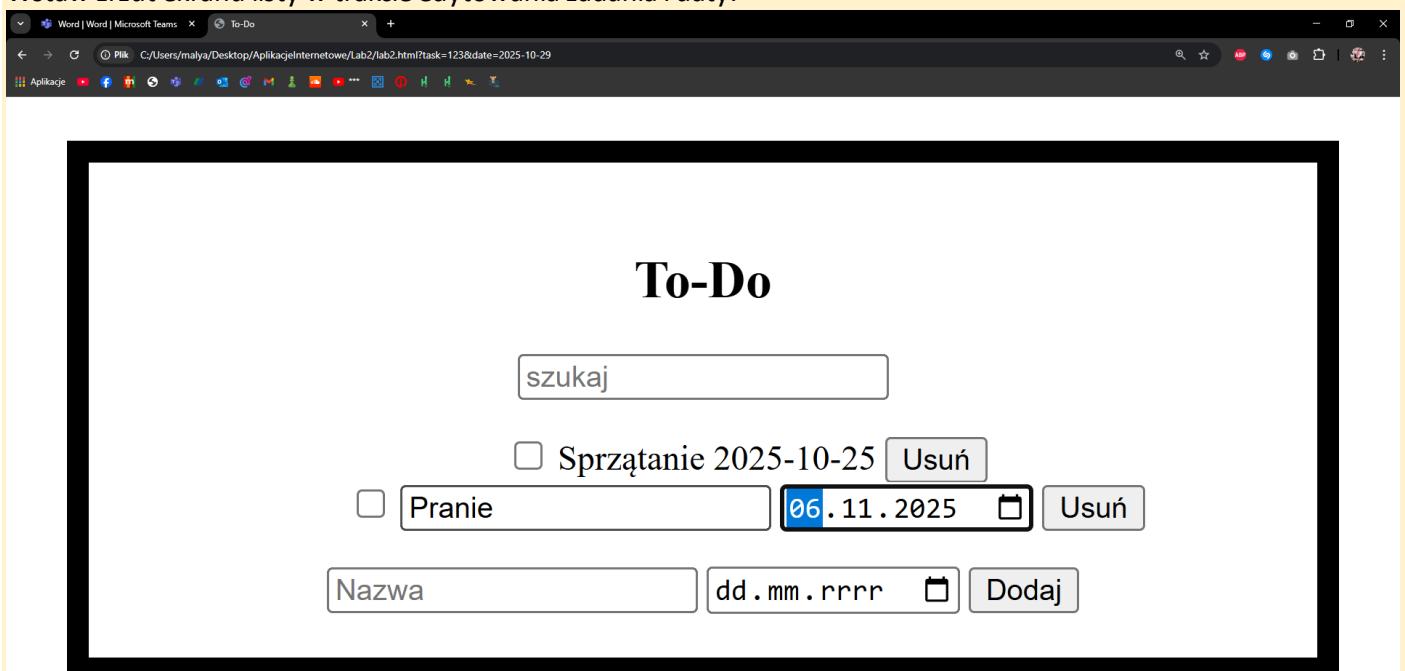
Punkty:	0	1
---------	---	---

EDYCJA POZYCJI LISTY

Wstaw zrzut ekranu listy przed edycją wybranego zadania:



Wstaw zrzut ekranu listy w trakcie edytowania zadania i daty:



Wstaw zrzut ekranu listy po edycji zadania i daty. Upewnij się, że dane się zapisały i zadanie jest zmienione:

The screenshot shows a web-based To-Do application. At the top, there's a navigation bar with tabs for 'Word | Word | Microsoft Teams' and 'Ta-Do'. Below the tab bar, the URL is 'C:/Users/malya/Desktop/AplikacjeInternetowe/Lab2/lab2.html?task=123&date=2025-10-29'. The main content area has a large black rectangular redaction box. Inside this box, the word 'To-Do' is centered in a bold, black font. Below it is a search bar containing the word 'szukaj'. Underneath the search bar are two task items, each with a checkbox, a due date, and a delete button labeled 'Usuń':

- Sprzątanie 2025-10-25 Usuń
- Wyrzucanie śmieci 2026-05-22 Usuń

At the bottom of the redacted area are three input fields: 'Nazwa' (Name), 'dd . mm . rrrr' (Date format), and a calendar icon. To the right of these fields is a 'Dodaj' (Add) button.

Punkty:	0	1
---------	---	---

ODCZYT / ZAPIS LOCAL STORAGE

Zastosowanie klasy Todo w realizacji tego laboratorium pozwala w bardzo łatwy sposób odczytywać i zapisywać stan listy do pamięci przeglądarki. Wystarczy serializacja / deserializacja za pomocą `JSON.parse()` i `JSON.stringify()`.

Wstaw zrzuty ekranu przedstawiające wygląd listy i zawartość local storage gdy na liście są pewne zadania:

The screenshot shows a browser window with a 'To-Do' application open. The application has a search bar ('szukaj') and a list of tasks:

- Sprzątanie 2025-10-23 [Usuń]
- Wyrzucanie śmieci 2026-03-12 [Usuń]

Below the list are input fields ('Nazwa', 'dd.mm.yyyy', 'Dodaj') and a 'Dodaj' button.

In the background, the DevTools Application tab is visible, showing the contents of the local storage under the key 'tasks':

```
[[{"text": "Sprzątanie", "date": "2025-10-23", "done": false}, {"text": "Wyrzucanie śmieci", "date": "2026-03-12", "done": false}]]
```

Wstaw zrzuty ekranu przedstawiające wygląd listy i zawartość local storage po dodaniu nowej pozycji listy. Upewnij się, że widoczne w local storage są dane dotyczące nowego zadania:

The screenshot shows the same browser setup as the previous one, but with a new task added to the list:

- Sprzątanie 2025-10-23 [Usuń]
- Wyrzucanie śmieci 2026-03-12 [Usuń]
- Zadanie domowe 2025-10-26 [Usuń]

The DevTools Application tab shows the updated local storage content for the 'tasks' key:

```
[[{"text": "Sprzątanie", "date": "2025-10-23", "done": false}, {"text": "Wyrzucanie śmieci", "date": "2026-03-12", "done": false}, {"text": "Zadanie domowe", "date": "2025-10-26", "done": false}]]
```

Punkty:

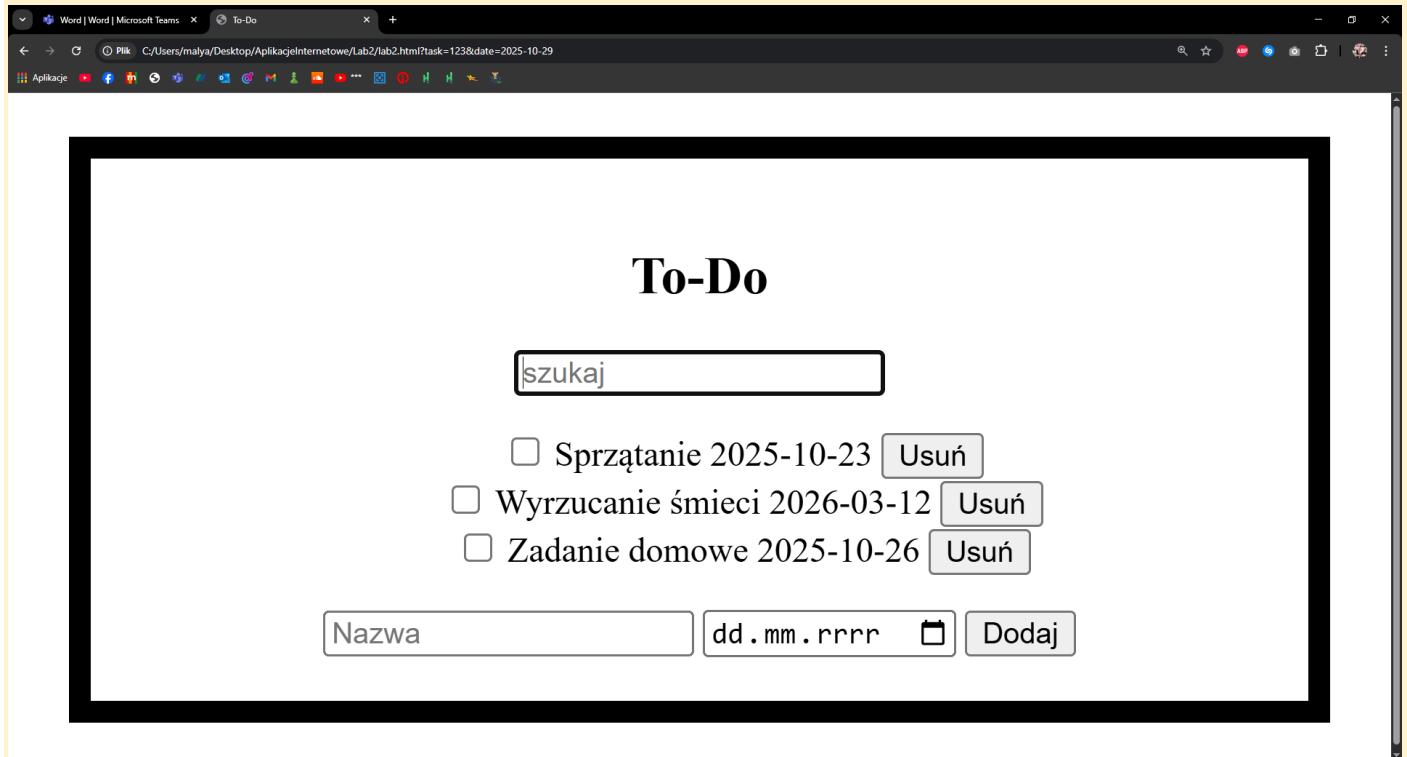
0

1

WYSZUKIWANIE

Na koniec zostało filtrowanie wyników. Proponowanym podejściem do tego tematu jest umieszczenie w klasie Todo właściwości `term` – frazy wyszukiwanej przez użytkownika. Następnie można utworzyć metodę `getFilteredTasks`, albo getter `filteredTasks`, która zwracać będzie te elementy tablicy `tasks`, które odpowiadają zapytaniu. Można użyć funkcji wyższego rzędu `filter()`.

Wstaw zrzut ekranu listy, gdy pole wyszukiwania jest puste:



Wstaw zrzut ekranu listy, gdy w polu wyszukiwania wpisano wystarczająco dużo znaków, by zadziałało filtrowanie. Upewnij się, że chociaż 2 wyniki będą wciąż widoczne:

The screenshot shows a web-based To-Do application. At the top, there's a search bar containing the letters "rz". Below it, two tasks are listed, each with a checkbox, a due date, and a "Usuń" button:

- Sprzątanie 2025-10-23 Usuń
- Wyrzucanie śmieci 2026-03-12 Usuń

At the bottom, there are input fields for "Nazwa" and "dd.mm.rrrr", a calendar icon, and a "Dodaj" button.

Punkty:	0	1
---------	---	---

Wstaw zrzut ekranu przedstawiający podświetlenie szukanej frazy w wynikach wyszukiwania, przykładowo dla frazy ko i zadania Ala ma kota otrzymujemy: Ala ma **kota**:

The screenshot shows a web-based To-Do application. At the top, there's a search bar containing the letters "dom". Below it, one task is listed, with a checkbox, a due date, and a "Usuń" button:

- Zadanie **domowe** 2025-10-26 Usuń

At the bottom, there are input fields for "Nazwa" and "dd.mm.rrrr", a calendar icon, and a "Dodaj" button.

Punkty:	0	1
---------	---	---

COMMIT PROJEKTU DO GIT

Zacommituj i pushnij swoje rozwiązanie do repozytorium GIT.

Upewnij się, czy wszystko dobrze się wysłało. Jeśli tak, to z poziomu przeglądarki utwórz branch o nazwie lab-b na podstawie głównej gałęzi kodu.

Podaj link do brancha lab-b w swoim repozytorium:

<https://github.com/anticzektriczek/AplikacjeInternetowe/tree/main/Lab2>

PODSUMOWANIE

W kilku słowach/zdaniach napisz swoje przemyślenia odnośnie tego laboratorium. Nie używaj LLM.

Z początku zadanie wydawało się trudne, ponieważ nie znałem js tak dobrze. Gdy zacząłem zadanie i dodawałem kolejne funkcje dawało mi to satysfakcję kiedy widziałem efekty mojej pracy.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.