

# 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.....	7
Odczyt / Zapis LocalStorage .....	10
Wyszukiwanie.....	12
Commit projektu do GIT.....	15
Podsumowanie.....	16

## 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 first mockup shows a simple list of tasks:

checkbox	task	date	trash icon
<input type="checkbox"/>	chocolate	2000-01-01	
<input type="checkbox"/>	macaroon		
<input type="checkbox"/>	chupa chups		
<input type="checkbox"/>	candy canes	2000-01-05	
<input type="checkbox"/>	bon bons		

The second mockup shows the 'chupa chups' task selected:

checkbox	task	date	trash icon
<input type="checkbox"/>	chupa chups		
<input type="checkbox"/>	macaroon		
<input type="checkbox"/>	candy canes	2000-01-05	
<input type="checkbox"/>	bon bons		

The search bar shows 'on'.

checkbox	task	date	trash icon
<input type="checkbox"/>	macaroon		
<input type="checkbox"/>	bon bons		

## 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:

### Formularz- LAB B

Wyszukaj zadania

- |  |            |  |
|--|------------|--|
| <input type="checkbox"/> umyć okna                 | 2025-10-18 |  |
| <input checked="" type="checkbox"/> wrócić do domu | 2025-10-17 |  |
| <input type="checkbox"/> pobawić się z kotem       | 2025-10-19 |  |

do zrobienia...

dd.mm.rrrr

Zapisz

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:

### Formularz- LAB B

Wyszukaj zadania

do zrobienia...

dd.mm.yyyy

**Zapisz**

Wstaw zrzut ekranu listy po dodaniu nowego zadania:

## Formularz- LAB B

Wyszukaj zadania

zabawa z kotem

2025-10-18 

do zrobienia...

dd.mm.yyyy 

Zapisz

Punkty:

0

1

### USUWANIE POZYCJI LISTY

Wstaw zrzut ekranu listy przed usunięciem wybranego zadania:

## Formularz- LAB B

Wyszukaj zadania

zabawa z kotem

2025-10-18



powrót do Szczecina

2025-10-19



do zrobienia...

dd.mm.yyyy

Zapisz

Wstaw zrzut ekranu listy po usunięciu zadania:

## Formularz- LAB B

Wyszukaj zadania

powrót do Szczecina

2025-10-19 

do zrobienia...

dd.mm.yyyy

Zapisz

Punkty:

0

1

### EDYCJA POZYCJI LISTY

Wstaw zrzut ekranu listy przed edycją wybranego zadania:

## Formularz- LAB B

Wyszukaj zadania

powrót do Szczecina

2025-10-19 

do zrobienia...

dd.mm.yyyy

Zapisz

Wstaw zrzut ekranu listy w trakcie edytowania zadania i daty:

## Formularz- LAB B

Wyszukaj zadania

powrót do Szczecina

19.10.2025

do zrobienia...

dd.mm.yyyy

Zapisz

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

## Formularz- LAB B

Wyszukaj zadania

powrót do Szczecina z domu

2025-10-20 

do zrobienia...

dd.mm.yyyy

Zapisz

Punkty:

0

1

### ODCZYT / ZAPIS LOCALSTORAGE

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:

# Formularz- LAB B

Wyszukaj zadania

powrót do Szczecina z domu

2025-10-20



do zrobienia...

dd.mm.yyyy

Zapisz

```
▼ [{id: 1760687439660, text: "powrót do Szczecina z domu", date: "2025-10-20", done: false}]\n  ↳ 0: {id: 1760687439660, text: "powrót do Szczecina z domu", date: "2025-10-20", 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:

# Formularz- LAB B

Wyszukaj zadania

- |   |            |  |
|---|------------|--|
| <input type="checkbox"/> powrót do Szczecina z domu | 2025-10-20 |  |
| <input type="checkbox"/> pojechać na zakupy         | 2025-10-18 |  |

do zrobienia...

dd.mm.yyyy

Zapisz

```
▼ [{id: 1760687439660, text: "powrót do Szczecina z domu", date: "2025-10-20", done: false},...]
  ↳ 0: {id: 1760687439660, text: "powrót do Szczecina z domu", date: "2025-10-20", done: false}
  ↳ 1: {id: 1760687703491, text: "pojechać na zakupy", date: "2025-10-18", 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:

## Formularz- LAB B

Wyszukaj zadania

- |   |            |  |
|---|------------|--|
| <input type="checkbox"/> powrót do Szczecina z domu | 2025-10-20 |  |
| <input type="checkbox"/> pojechać na zakupy         | 2025-10-18 |  |
| <input type="checkbox"/> umyć samochód              | 2025-10-19 |  |
| <input type="checkbox"/> oddać samochód do naprawy  | 2025-10-21 |  |

do zrobienia...

dd.mm.yyyy

Zapisz

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

## Formularz- LAB B

samochód

umyć samochód

2025-10-19



oddać samochód do naprawy

2025-10-21



do zrobienia...

dd.mm.rrrr

Zapisz

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:

## Formularz- LAB B

ko

Ala ma **kota**



do zrobienia...

dd.mm.rrrr

Zapisz

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/kuba200333/aplikacje-internetowe1/tree/lab-b/lab-b>

## PODSUMOWANIE

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

Podobało mi się to laboratorium, bo poznałem możliwości javascript i obsługi formularza. Przydatna umiejętność.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.