

JS I DOM NA PRZYKŁADZIE LISTY TODO

SPIS TREŚCI

Spis treści	1
Cel zajęć	1
Rozpoczęcie	1
Uwaga	2
Wymagania	2
Strona HTML	2
Klasa Todo	3
Dodawanie pozycji listy	4
Usuwanie pozycji listy	4
Edycja pozycji listy	5
Odczyt / Zapis LocalStorage	7
Wyszukiwanie	11
Commit projektu do GIT	12
Podsumowanie	13

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 przycisk Usun / Śmietnik
- wpisy na liście zapisują się do Local Storage
- po odświeżeniu strony lista wypełnia się wpisami z Local Storage

Mockupy:

Mockup of the application interface. It features a search bar at the top. Below it is a list of tasks, each with a checkbox, a task name, a date, and a delete button (X). The tasks are: chocolate (2000-01-01), macaroon, chupa chups, candy canes (2000-01-05), and bon bons. At the bottom, there is a text input field labeled 'do zrobienia...', a date field with a calendar icon, and a 'Zapisz' button.

Mockup of the application interface showing a task being edited. The 'chupa chups' task is selected, and its details are shown in a form with a text input field, a date field, and a 'Zapisz' button. The other tasks remain in the list below.

Mockup of the application interface showing search results. The search bar contains the text 'on'. The list below shows only the tasks that contain the letter 'n': macaroon and bon bons. Each task has a checkbox, the task name, a date, and a delete button (X). At the bottom, there is a text input field labeled 'do zrobienia...', a date field with a calendar icon, and a 'Zapisz' button.

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.

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

To do List

dsadasdasd Edit Delete - Termin wykonania: 2023-11-18

daodaj1 Edit Delete - Termin wykonania: 2023-12-02

dso1ja Edit Delete - Termin wykonania: 2023-11-30

Punkty:

0

1

KLASA TODO

Pierwszym instynktem może być chęć dodania zachowań bezpośrednio do elementów listy. 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:

To do List

Add

Wstaw zrzut ekranu listy po dodaniu nowego zadania:

To do List

nowezadanie

Edit

Delete

- Termin wykonania: 2023-11-23

Add

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

USUWANIE POZYCJI LISTY

Wstaw zrzut ekranu listy przed usunięciem wybranego zadania:

To do List

nowezadanie

Edit

Delete

- Termin wykonania: 2023-11-23

usunzadanie

Edit

Delete

- Termin wykonania: 2023-11-23

Add

Wstaw zrzut ekranu listy po usunięciu zadania:

To do List

nowezadanie

Edit

Delete

- Termin wykonania: 2023-11-23

Add

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

EDYCJA POZYCJI LISTY

Wstaw zrzut ekranu listy przed edycją wybranego zadania:

To do List

nowezadanie

Edit

Delete

- Termin wykonania: 2023-11-23



Add

Wstaw zrzut ekranu listy w trakcie edytowania zadania i daty:

To do List

nowezadanie

23.11.2023



Edit

Delete

- Termin wykonania:

2023-11-23



Add

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

To do List

nowezadaniepozmianie

[Edit](#)[Delete](#)

- Termin wykonania: 2023-11-30

[Add](#)

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:

To do List

Wyszukiwarka

implementacja

Edit

Delete

- Termin wykonania: 2023-12-01

dasda

Edit

Delete

- Termin wykonania: 2023-11-26

nowezadanie3

Edit

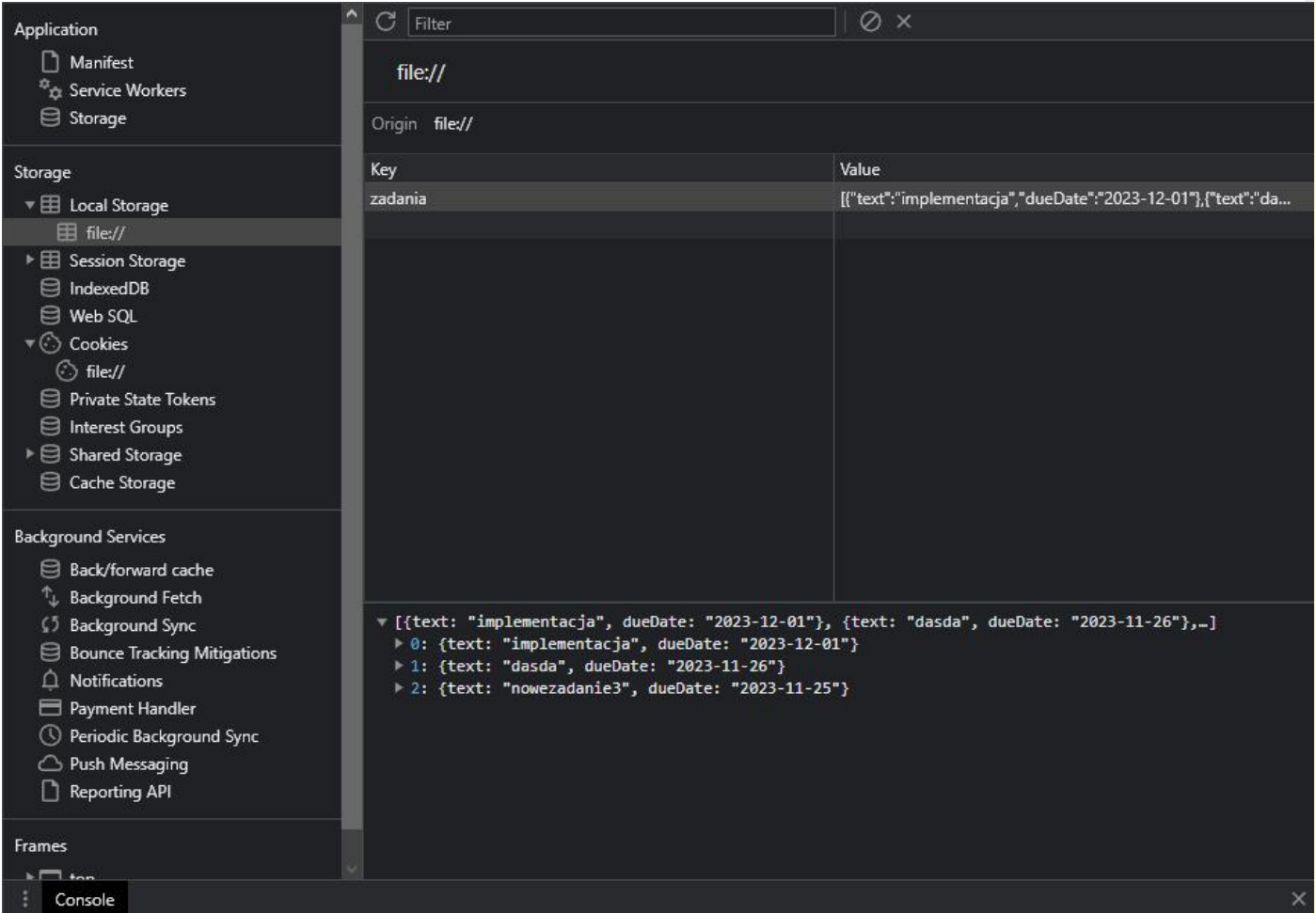
Delete

- Termin wykonania: 2023-11-25

Wpisz swoje nowe zadanie

dd.mm.rrrr

Add



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:

To do List

Wyszukiwarka

implementacja

Edit

Delete

- Termin wykonania: 2023-12-01

dasda

Edit

Delete

- Termin wykonania: 2023-11-26

nowezadanie3

Edit

Delete

- Termin wykonania: 2023-11-25

nowapozycjanaliscie

Edit

Delete

- Termin wykonania: 2023-11-18

Wpisz swoje nowe zadanie

dd.mm.yyyy

Add

Application

Manifest

Service Workers

Storage

Storage

Local Storage

file://

Session Storage

IndexedDB

Web SQL

Cookies

file://

Private State Tokens

Interest Groups

Shared Storage

Cache Storage

Background Services

Back/forward cache

Background Fetch

Background Sync

Bounce Tracking Mitigations

Notifications

Payment Handler

Periodic Background Sync

Push Messaging

Reporting API

Frames

ten

Console

Filter

file://

Origin file://

Key	Value
zadania	[{"text":"implementacja","dueDate":"2023-12-01"},{"text":"da..."]

▼ [{"text": "implementacja", dueDate: "2023-12-01"}, {"text": "dasda", dueDate: "2023-11-26"},...]

▶ 0: {text: "implementacja", dueDate: "2023-12-01"}

▶ 1: {text: "dasda", dueDate: "2023-11-26"}

▶ 2: {text: "nowezadanie3", dueDate: "2023-11-25"}

▶ 3: {text: "nowapozycjanaliscie", dueDate: "2023-11-18"}

Punkty:

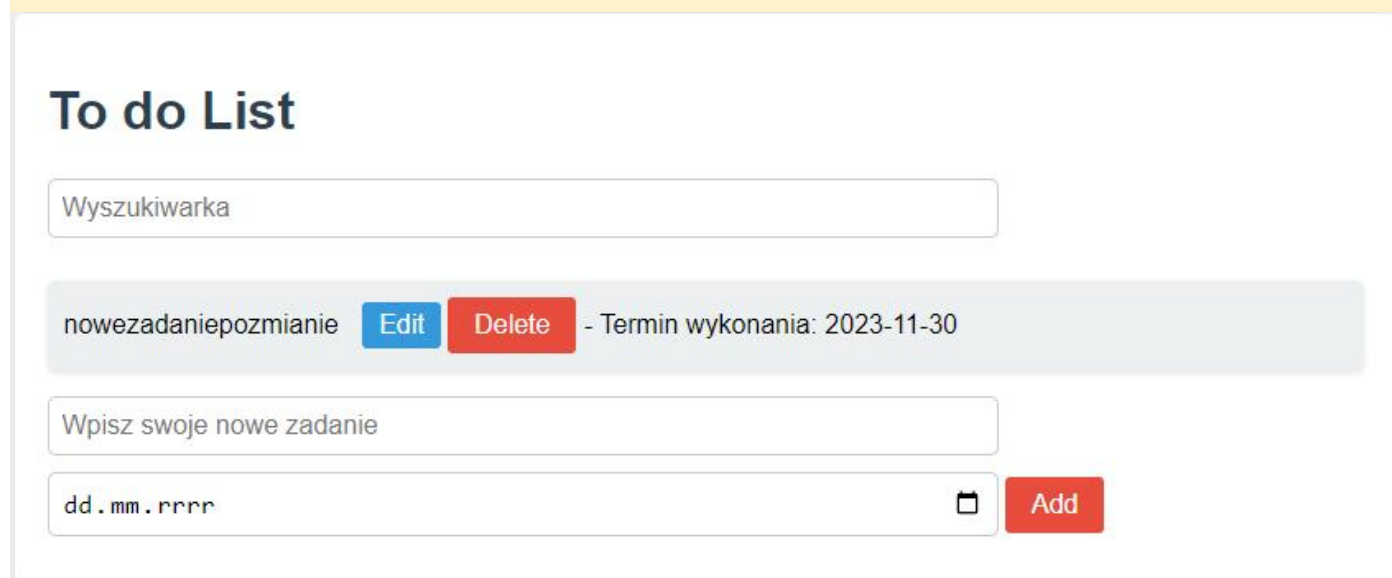
0

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:

To do List

nowezadaniepozmianie

Edit

Delete

- Termin wykonania: 2023-11-30



Add

Punkty:

0

1

Wstaw zrzut ekranu przedstawiający podświetlenie szukanej frazy w wynikach wyszukiwania, przykładowo dla frazy `imp` i zadania `implementacja` otrzymujemy: `implementacja`:

To do List

implementacja

Edit

Delete

- Termin wykonania: 2023-12-01



Add

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:

...link, np. <https://github.com/inazwisko/ai1-lab/tree/lab-b...>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Nabyłem umiejętność tworzenia interaktywnych list zadań przy użyciu HTML, CSS i JavaScript. Opanowałem obsługę Local Storage do przechowywania danych w przeglądarce. Nauczyłem się obsługi zdarzeń i manipulowania nimi, co umożliwiło mi edycję zadań na liście. Rozwinięte umiejętności tworzenia interfejsu użytkownika pomogły mi w dodawaniu, edycji i usuwaniu zadań, a także w dynamicznym filtrowaniu wyników. Dodatkowo, zdobyłem umiejętność stosowania różnych stylów CSS do kształtowania wyglądu i interakcji elementów strony internetowej.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.