index.js

import { createNote } from './notes'

import { setFilters } from './filters'

import { renderNotes } from './views'

renderNotes() // 1. wyrenderuj notatki funkcją: [renderNotes](#odn1)

document.querySelector('#create-note').addEventListener('click', (e) => { // 30. nadaj nasłuch na element create-note z html

const id = createNote() // 31. do id przypisz to, co zwróci funkcja [createNote](#odn31)

location.assign(`/edit.html#${id}`) // 36. zmień adres w przeglądarce na html.edit, a na końcu adresu dodaj id stworzone przez f. createNote

})

document.querySelector('#search-text').addEventListener('input', (e) => {

setFilters({

searchText: e.target.value

})

renderNotes()

})

document.querySelector('#filter-by').addEventListener('change', (e) => {

setFilters({

sortBy: e.target.value

})

renderNotes()

})

window.addEventListener('storage', (e) => {

if (e.key === 'notes') {

renderNotes()

}

})

views.js

import moment from 'moment'

import { getFilters } from './filters'

import { sortNotes, getNotes } from './notes'

// Generate the DOM structure for a note

const generateNoteDOM = (note) => { [// 9.](#tłum9)

const noteEl = document.createElement('a') // 10. stwórz link (zmienna noteEl)

const textEl = document.createElement('p') // 11. stwórz paragraf (zmienna textEl)

const statusEl = document.createElement('p') // 12. stwórz kolejny paragraf (zmienna statusEl)

// Setup the note title text

if (note.title.length > 0) { // 13. jeśli tytuł notatki jest większy od zera

textEl.textContent = note.title // 14. paragraf będzie miał zawaretość tekstową równą tytułowi notatki [(przykładowa tablica z notatkami)](#przykładowa_tablica_z_notatkami)

} else {

textEl.textContent = 'Unnamed note' // 15. w innym wypadku nadaj mu domyślny tytuł: 'Unnamed note'

}

textEl.classList.add('list-item\_\_title') // 16. nadaj paragrafowi klasę CSS

noteEl.appendChild(textEl) // 17. dodaj paragraf do linku

// Setup the link

noteEl.setAttribute('href', `/edit.html#${note.id}`) // 18. nadaj linkowi adres, żeby po kliknięciu otwierała się na nim przeglądarka

noteEl.classList.add('list-item') // 19. dodaj linkowi klasę w CSS

// Setup the status message

statusEl.textContent = generateLastEdited(note.updatedAt) // 20. tekst do paragrafu statusEl, stwórz funkcją: [generateLastEdited](#odn20)

statusEl.classList.add('list-item\_\_subtitle') // 22. nadaj paragrafowi klasę w CSS

noteEl.appendChild(statusEl) // 23. dodaj do linku info o statusie

return noteEl // 24. zwróć cały gotowy element – link z zawartymi w nim dwoma paragrafami. [powrót do 9](#powrótdo9)

}

// Render application notes

const renderNotes = () => { [// 1.](#tłum1)

const notesEl = document.querySelector('#notes') // 2. złap za pomocą zmiennej – notesEl, element w HTML - [notes](#odn2)

const filters = getFilters() // 3. do zmiennej filters przypisz to, co zwróci funkcja - [getFilters](#odn3)

const notes = sortNotes(filters.sortBy) // 4. do zmiennej notes przypisz to, co zwróci funkcja [sortNotes](#odn4) (domyślnie będzie to sortNotes(byEdited))

const filteredNotes = notes.filter((note) => note.title.toLowerCase().includes(filters.searchText.toLowerCase())) // 5. Wyfiltruj po szukanym słowie (literze) [tłum](#extra1).

notesEl.innerHTML = '' // 6. wyczyść cały HTML z elementu o ID: ‘notes’

if (filteredNotes.length > 0) { // 7. jeśli długość tablicy jest większa od zera:

filteredNotes.forEach((note) => { // 8. każdą notatkę z tablicy wsadź do argumentu funkcji:

const noteEl = generateNoteDOM(note) // 9. [generateNoteDOM](#odn9)

notesEl.appendChild(noteEl) // 25. każdą zwróconą notatkę dodaj do diva o ID: ‘notes’ w HTML

})

} else {

const emptyMessage = document.createElement('p') // 26. albo utwórz paragraf przypisany do zmiennej emptyMessage

emptyMessage.textContent = 'No notes to show' // 27. przypisz mu tekst: 'No notes to show'

emptyMessage.classList.add('empty-message') // 28. dodaj klasę w CSS

notesEl.appendChild(emptyMessage) // 29. dodaj do diva o ID: ‘notes’ w HTML

}

}

const initializeEditPage = (noteId) => {

const titleElement = document.querySelector('#note-title')

const bodyElement = document.querySelector('#note-body')

const dateElement = document.querySelector('#last-edited')

const notes = getNotes()

const note = notes.find((note) => note.id === noteId)

if (!note) {

location.assign('/index.html')

}

titleElement.value = note.title

bodyElement.value = note.body

dateElement.textContent = generateLastEdited(note.updatedAt)

}

// Generate the last edited message

const generateLastEdited = (timestamp) => { [// 20.](#tłum20) timestamp = note.updatedAt

return `Last edited ${moment(timestamp).fromNow()}` // 21. moment weźmie liczbę z note.updatedAt i funkcją fromNow zrobi stringa

}

export { generateNoteDOM, renderNotes, generateLastEdited, initializeEditPage}

filters.js

const filters = {

searchText: '',

sortBy: 'byEdited' zwraca obiekt z filtrami

}

const getFilters = () => filters [// 3.](#tłum3)

const setFilters = (updates) => {

if (typeof updates.searchText === 'string') {

filters.searchText = updates.searchText

}

if (typeof updates.sortBy === 'string') {

filters.sortBy = updates.sortBy

}

}

export { getFilters, setFilters }

notes.js

import uuidv4 from 'uuid/v4'

import moment from 'moment'

let notes = []

// Read existing notes from localStorage

const loadNotes = () => { // funkcja ładująca notatki do tablicy z localStorage. [powrót do wątku](#extra4powrót)

const notesJSON = localStorage.getItem('notes') // do zmiennej noteJSON przypisz zassaną z localStorage tablicę ‘notes’

try { //spróbuj

return notesJSON ? JSON.parse(notesJSON) : [] // sprawdzić, czy coś tam jest, jak tak, to wyrzuć sparsowaną tablicę, jak nie, to pustą

} catch (e) { // w razie błędu

return [] // wyrzuć pustą tablicę

}

}

// Save the notes to localStorage

const saveNotes = () => {

localStorage.setItem('notes', JSON.stringify(notes)) // 35. w localStorage umieść tablicę nazywając ją ‘notes’ i wrzuć do niej zestringowaną tablicę notes

} // [wróć](#tłum31)

// Expose notes from module

const getNotes = () => notes

const createNote = () => { [// 31.](#tłum31)

const id = uuidv4() // 32. stwórz unikatowe id i przypisz je do zmiennej id

const timestamp = moment().valueOf() // 32. przy pomocy narzędzia moment stwórz dany moment i wyciągnij z niego liczbę milisekund przypisując ją do zm. timestamp

notes.push({ // 33. umieść na końcu tablicy ‘notes’ następujący obiekt:

id: id,

title: '',

body: '',

createdAt: timestamp,

updatedAt: timestamp

})

saveNotes() // 34. wykonaj funkcję saveNotes

return id

}

// Remove a note from the list

const removeNote = (id) => {

const noteIndex = notes.findIndex((note) => note.id === id)

if (noteIndex > -1) {

notes.splice(noteIndex, 1)

saveNotes()

}

}

// Sort your notes by one of three ways

const sortNotes = (sortBy) => { [// 4.](#tłum4) [tłumaczenie](#extra2)

if (sortBy === 'byEdited') {

return notes.sort((a, b) => { // układaj od lewa do prawa // tablica notes bierze się stąd – [loadNotes](#extra3)

if (a.updatedAt > b.updatedAt) { // jeśli liczba ‘a’ jest większa od liczby ‘b’,

return -1 // umieść ją wcześniej

} else if (a.updatedAt < b.updatedAt) { // jeśli liczba ‘a’ jest mniejsza od liczby ‘b’,

return 1 // umieść ją później

} else { // albo

return 0 // pozostaw ją gdzie była

}

})

} else if (sortBy === 'byCreated') {

return notes.sort((a, b) => {

if (a.createdAt > b.createdAt) {

return -1

} else if (a.createdAt < b.createdAt) {

return 1

} else {

return 0

}

})

} else if (sortBy === 'alphabetical') {

return notes.sort((a, b) => {

if (a.title.toLowerCase() < b.title.toLowerCase()) {

return -1

} else if (a.title.toLowerCase() > b.title.toLowerCase()) {

return 1

} else {

return 0

}

})

} else {

return notes

}

}

const updateNote = (id, updates) => {

const note = notes.find((note) => note.id === id)

if (!note) {

return

}

if (typeof updates.title === 'string') {

note.title = updates.title

note.updatedAt = moment().valueOf()

}

if (typeof updates.body === 'string') {

note.body = updates.body

note.updatedAt = moment().valueOf()

}

saveNotes()

return note

}

notes = loadNotes() // wywołuje funkcję zdefiniowaną [tutaj](#extra4)

export { getNotes, createNote, removeNote, sortNotes, updateNote }

edit.js

import { initializeEditPage, generateLastEdited } from './views'

import { updateNote, removeNote } from './notes'

const titleElement = document.querySelector('#note-title')

const bodyElement = document.querySelector('#note-body')

const removeElement = document.querySelector('#remove-note')

const dateElement = document.querySelector('#last-edited')

const noteId = location.hash.substring(1)

initializeEditPage(noteId)

titleElement.addEventListener('input', (e) => {

const note = updateNote(noteId, {

title: e.target.value

})

dateElement.textContent = generateLastEdited(note.updatedAt)

})

bodyElement.addEventListener('input', (e) => {

const note = updateNote(noteId, {

body: e.target.value

})

dateElement.textContent = generateLastEdited(note.updatedAt)

})

removeElement.addEventListener('click', (e) => {

removeNote(noteId)

location.assign('/index.html')

})

window.addEventListener('storage', (e) => {

if (e.key === 'notes') {

initializeEditPage(noteId)

}

})

index.html

<!DOCTYPE html>

<html>

<head>

<title>Notes App</title>

<link href="/images/favicon.png" rel="icon">

<link href="/styles/styles.css" rel="stylesheet">

</head>

<body>

<header class="header">

<div class="container">

<h1 class="header\_\_title">Notes App</h1>

<h2 class="header\_\_subtitle">Take notes and never forget</h2>

</div>

</header>

<div class="actions">

<div class="actions\_\_container">

<input id="search-text" type="text" class="input" placeholder="Filter notes">

<select id="filter-by" class="dropdown">

<option value="byEdited">Sort by last edited</option>

<option value="byCreated">Sort by recently created</option>

<option value="alphabetical">Sort alphabetically</option>

</select>

</div>

</div>

<div class="container">

<div id="notes"></div> [// 2.](#tłum2)

<button id="create-note" class="button">Create Note</button> //

</div>

<script src="/scripts/index-bundle.js"></script>

</body>

</html>

edit.html

<!DOCTYPE html>

<html>

<head>

<title>Notes App</title>

<link href="/images/favicon.png" rel="icon">

<link href="/styles/styles.css" rel="stylesheet">

</head>

<body>

<div class="header">

<div class="container">

<h1 class="header\_\_title">Notes App</h1>

<h2 class="header\_\_subtitle">Take notes and never forget</h2>

</div>

</div>

<div class="actions">

<div class="actions\_\_container actions\_\_container--spaced">

<a href="/index.html">Home</a>

<span id="last-edited"></span>

</div>

</div>

<div class="container">

<input id="note-title" class="title-input" placeholder="Note title">

<textarea id="note-body" class="body-input" placeholder="Enter note text"></textarea>

<button id="remove-note" class="button button--secondary">Remove note</button>

</div>

<script src="/scripts/edit-bundle.js"></script>

</body>

</html>

[tłumaczenie: 1](#extra1powrót)

let notes = [{"id":"1c978620-e126-4f61-b0cf-fc4f1839afd9","title":"Dentysta","body":"zaplombować ząb","createdAt":1543668003089,"updatedAt":1543668014591}, [(przykładowa tabliza z notatkami)](#przykładowa_tablica_z_notatkami_powrót)

{"id":"96131a35-2c66-4d8d-b60d-ad88b662cfd7","title":"Obejrzeć serial","body":"Ozark","createdAt":1543667982273,"updatedAt":1543667997199},

{"id":"034c7ff8-a657-4f29-84dd-32fae2a27642","title":"Iść do fryzjera","body":"obciąć się na krótko","createdAt":1543667915066,"updatedAt":1543667926786},

{"id":"3be9f65a-367d-46db-b2e9-d5b6b7c67ee0","title":"Zrobić zakupy","body":"masło, jaja, ser","createdAt":1543667881965,"updatedAt":1543667907699},

{"id":"c2e14173-ebe4-44aa-a7ca-71817d3d4ff1","title":"Obiad","body":"ugotować zupę","createdAt":1543668026447,"updatedAt":1543668036000}]

const filters = {

searchText: 'Ob',

sortBy: 'byEdited'

}

const filteredNotes = notes.filter((note) => note.title.toLowerCase().includes(filters.searchText.toLowerCase()))

console.log(filteredNotes) // zwróci: [[object Object] {

body: "Ozark",

createdAt: 1543667982273,

id: "96131a35-2c66-4d8d-b60d-ad88b662cfd7",

title: "Obejrzeć serial",

updatedAt: 1543667997199

}, [object Object] {

body: "masło, jaja, ser",

createdAt: 1543667881965,

id: "3be9f65a-367d-46db-b2e9-d5b6b7c67ee0",

title: "Zrobić zakupy",

updatedAt: 1543667907699

}, [object Object] {

body: "ugotować zupę",

createdAt: 1543668026447,

id: "c2e14173-ebe4-44aa-a7ca-71817d3d4ff1",

title: "Obiad",

updatedAt: 1543668036000

}]

[tłumaczenie: 2](#extra2powrót)

let notes = [{"id":"1c978620-e126-4f61-b0cf-fc4f1839afd9","title":"Dentysta","body":"zaplombować ząb","createdAt":1543668003089,"updatedAt":1543668014591},

{"id":"96131a35-2c66-4d8d-b60d-ad88b662cfd7","title":"Obejrzeć serial","body":"Ozark","createdAt":1543667982273,"updatedAt":1543667997199},

{"id":"034c7ff8-a657-4f29-84dd-32fae2a27642","title":"Iść do fryzjera","body":"obciąć się na krótko","createdAt":1543667915066,"updatedAt":1543667926786},

{"id":"3be9f65a-367d-46db-b2e9-d5b6b7c67ee0","title":"Zrobić zakupy","body":"masło, jaja, ser","createdAt":1543667881965,"updatedAt":1543667907699},

{"id":"c2e14173-ebe4-44aa-a7ca-71817d3d4ff1","title":"Obiad","body":"ugotować zupę","createdAt":1543668026447,"updatedAt":1543668036000}]

const sortNotes = (sortBy) => {

if (sortBy === 'byEdited') {

return notes.sort((a, b) => {

if (a.updatedAt > b.updatedAt) {

return -1

} else if (a.updatedAt < b.updatedAt) {

return 1

} else {

return 0

}

})

} else if (sortBy === 'byCreated') {

return notes.sort((a, b) => {

if (a.createdAt > b.createdAt) {

return -1

} else if (a.createdAt < b.createdAt) {

return 1

} else {

return 0

}

})

} else if (sortBy === 'alphabetical') {

return notes.sort((a, b) => {

if (a.title.toLowerCase() < b.title.toLowerCase()) {

return -1

} else if (a.title.toLowerCase() > b.title.toLowerCase()) {

return 1

} else {

return 0

}

})

} else {

return notes

}

}

sortNotes('alphabetical')

console.log(notes) // wyświetli posortowane alfabetycznie: [[object Object] {

body: "zaplombować ząb",

createdAt: 1543668003089,

id: "1c978620-e126-4f61-b0cf-fc4f1839afd9",

title: "Dentysta",

updatedAt: 1543668014591

}, [object Object] {

body: "obciąć się na krótko",

createdAt: 1543667915066,

id: "034c7ff8-a657-4f29-84dd-32fae2a27642",

title: "Iść do fryzjera",

updatedAt: 1543667926786

}, [object Object] {

body: "Ozark",

createdAt: 1543667982273,

id: "96131a35-2c66-4d8d-b60d-ad88b662cfd7",

title: "Obejrzeć serial",

updatedAt: 1543667997199

}, [object Object] {

body: "ugotować zupę",

createdAt: 1543668026447,

id: "c2e14173-ebe4-44aa-a7ca-71817d3d4ff1",

title: "Obiad",

updatedAt: 1543668036000

}, [object Object] {

body: "masło, jaja, ser",

createdAt: 1543667881965,

id: "3be9f65a-367d-46db-b2e9-d5b6b7c67ee0",

title: "Zrobić zakupy",

updatedAt: 1543667907699

}]