**index.js**

import { renderTodos } from './views'

import { setFilters } from './filters'

import { createTodo, loadTodos } from './todos' // cbrewcerfcner

renderTodos() // [1. Pierwsze renderowanie](#odn1)

document.querySelector('#search-text').addEventListener('input', (e) => { // 57. chwyć w html element o id ‘[search text’](#odn57) i nadaj jego inputowi nasłuch

setFilters({

searchText: e.target.value // 58. po wpisaniu literki z nasłuchu wrzuć ją do funkcji [setFilters](#odn58)

})

renderTodos() // 59. wyrenderuj

})

document.querySelector('#new-todo').addEventListener('submit', (e) => { // 60. zrób nasłuch na element o id: [new-todo](#odn60) w htmlu

const text = e.target.elements.text.value.trim() // 61. stwórz zmienną text i przypisz jej tekst wpisany do inputa obcinając spacje z tyłu i przodu

e.preventDefault() // 62. zablokuj domyślne działanie buttona

if (text.length > 0) { // 63. Jeśli długość tekstu jest większa od zera

createTodo(text) // 64. Wykonaj funkcję [createTodo](#odn64)

renderTodos() // 65. wyrenderuj ponownie stronę

e.target.elements.text.value = '' // 70. wyczyść to co wpisałeś w inputa

}

})

document.querySelector('#hide-completed').addEventListener('change', (e) => { // 71. uchwyć [hide-completed](#odn71) w htmlu i nasłuchuj

setFilters({

hideCompleted: e.target.checked // 72. jak ktoś zahaczy kwadracik, wykonaj funkcję setFilters () wrzuci do niej albo true, albo false [tłumaczenie](#extra7)

})

renderTodos() // 73. wyrenderuj wszystko ponownie

})

window.addEventListener('storage', (e) => { // 74. nasłuchuj, czy w przeglądarce (w innej zakładce), ktoś dokonał zmian w localStorage

if (e.key === 'todos') { // 75. jeśli zmiana zaszła w tablicy o nazwie ‘todos’ to:

loadTodos() // 76. ponownie załaduj do tablicy todos zawartość ‘todos’ z localStorage ([loadTodos](#odn76))

renderTodos() // 77. wyrenderuj

}

})

**views.js**

import { getTodos, toggleTodo, removeTodo } from './todos'

import { getFilters } from './filters'

// Render application todos based on filters

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

const todoEl = document.querySelector('#todos') // [2. Uchwyć w html diva o id = „todos” i przypisz mu zmienną todoEl](#odn2)

const { searchText, hideCompleted } = getFilters() // [3. wydobądź searchText i hideCompleted z obiektu filters metdodą destructuring](#odn3)

const filteredTodos = getTodos().filter((todo) => { // [4. Zrób tablicę filteredTodos z aktualnej tablicy stąd: getTodos](#odn4) [filter tłumaczenie](#extra3)

const searchTextMatch = todo.text.toLowerCase().includes(searchText.toLowerCase()) // 5. false bądź true

const hideCompletedMatch = !hideCompleted || !todo.completed // 6. true jeśli hideComplited nie zahaczony (czyli true) ALBO complited: false

return searchTextMatch && hideCompletedMatch // 7. jeśli obydwie wartości powyżej to true, przepuść taki todo

})

const incompleteTodos = filteredTodos.filter((todo) => !todo.completed) // 7. z wartości wyfiltrowanych w filteredTodos odfiltruj te nieukończone

todoEl.innerHTML = '' // 8. wyczyść z HTMLa diva o id=’todos’

todoEl.appendChild(generateSummaryDOM(incompleteTodos)) // 9. do elementu todoEl dodaj element [generateSummaryDOM](#odn9)

if (filteredTodos.length > 0) { // 15. jeśli tablica filteredTodos jest większa od zera

filteredTodos.forEach((todo) => { // 16. dla każdego jej elementu zrób:

todoEl.appendChild(generateTodoDOM(todo)) // 17. dołącz dziecko do todoEl funkcją [generateTodoDOM](#odn17) [52.](#tłum52)

})

} else {

const messageEl = document.createElement('p') // 53. jeśli jest mniejsza od zera, stwórz nowy element html ‘p’ i przypisz do zmiennej createElement

messageEl.classList.add('empty-message') // 54. dodaj mu klasę CSS ‘empty-message’

messageEl.textContent = 'There are no to-dos to show' // 55. dodaj odpowiedni text

todoEl.appendChild(messageEl) // 56. dodaj jako dziecko do todoEl [wróć do wątku](#tłum57)

}

}

// Get the DOM elements for an individual note

const generateTodoDOM = (todo) => { // [17.](#tłum17)

const todoEl = document.createElement('label') // 18. utwórz etykietę w html i przypisz ją do zmiennej todoEl

const containerEl = document.createElement('div') // 19. utwórz diva w html i przypisz go do amiennej containerEl

const checkbox = document.createElement('input') // 20. utwórz inputa w html i przypisz go do zmiennej todoText

const todoText = document.createElement('span') // 21. utwórz span w html i przypisz go do zminnej todoText

const removeButton = document.createElement('button') // 22. utwórz button w html i przypisz go do zmiennej remoweButton

// Setup todo checkbox

checkbox.setAttribute('type', 'checkbox') // 23. ustaw inputowi atrybut: type=”checkbox”

checkbox.checked = todo.completed // 24. ma być zaznaczony ptaszek, jeśli to todo ma complited: true

containerEl.appendChild(checkbox) // 25. do diva (containerEl) wrzuć tego inputa z atrybutem checkbox

checkbox.addEventListener('change', () => { // 26. nadaj checkboxowi eventlistenera

toggleTodo(todo.id) // 27. po wykryciu zmiany wykonaj funkcję [toggleTodo](#odn27) – przełącz complited na true, bądź false

renderTodos() // [37.](#powrótDo36) wyrenderuj wszystko ponownie

})

// Setup the todo text

todoText.textContent = todo.text // 38. do ‘span’ wrzuć tekst z todo

containerEl.appendChild(todoText) // 39. do ‘div’ wrzuć powyższy span z tekstem

// Setup container

todoEl.classList.add('list-item') // 40. labelowi przypisz css

containerEl.classList.add('list-item\_\_container') // 41. divovi również przypisz css

todoEl.appendChild(containerEl) // 42. do labela dodaj diva jako dziecko

// Setup the remove button

removeButton.textContent = 'remove' // 43. ustaw tekst, jaki mam mieć klawisz

removeButton.classList.add('button', 'button--text') // 44. nadaj mu klasę w CSS

todoEl.appendChild(removeButton) // 45. dołącz klawisz jako dziecko do labela

removeButton.addEventListener('click', () => { // 46. ustaw nasłuch na klawisz

removeTodo(todo.id) // 47. wykonaj funkcję [removeTodo](#odn47)

renderTodos() // [51.](#tłum50) wyrenderuj ponownie wszystko

})

return todoEl // 52. wyrzuć cały zbudowany label z zagnieżdżonym w nim divem ( a w tym divie inputem i spanem) oraz buttonem [tutaj:](#odn52)

}

// Get the DOM elements for list summary

const generateSummaryDOM = (incompleteTodos) => { // [.9](#tłum9)

const summary = document.createElement('h2') // 10. utwórz element h2 i przypisz go do zmiennej summary

const plural = incompleteTodos.length === 1 ? '' : 's' // 11. jeśli długość tablicy incompleteTodos jest 1 przypisz zmiennej plural (po zadeklarowniu jej)

stringa:‘’,jeśli nie to ‘s’

summary.classList.add('list-title') // 12. dodanie klasy w css

summary.textContent = `You have ${incompleteTodos.length} todo${plural} left` // 13. wsadź do summary tekst

return summary // 14. zwróć summary

}

export { generateTodoDOM, renderTodos, generateSummaryDOM }

**filters.js**

const filters = {

searchText: '', // …a potem wyjmij z tego obiektu (który przechowuje filtry) co potrzeba o tak: [tłumaczenie](#extra1)

hideCompleted: false

}

const getFilters = () => filters // [3.najpierw wywołaj funkcję czyli po prostu wydobądź cały obiekt filters (aktualne filtry) stąd…](#tłum3)

const setFilters = ({ searchText, hideCompleted }) => { // [58.](#tłum58) [tłumaczenie](#extra6)

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

filters.searchText = searchText

}

if (typeof hideCompleted === 'boolean') {

filters.hideCompleted = hideCompleted

}

}

export { getFilters, setFilters }

**todos.js**

import uuidv4 from 'uuid/v4'

let todos = [] // a tu może być na przykład to: [tłumaczenie 2](#extra2) // [28.](#tłum28)

// Fetch existing todos from localStorage

const loadTodos = () => { // [32.](#tłum32) [// 76.](#tłum76)

const todosJSON = localStorage.getItem('todos') // 33. przypisz do zmiennej: todosJSON zassaną z localstorage tablicę o zmiennej ‘todos’

try { // 33. spróbuj ją sparsować:

todos = todosJSON ? JSON.parse(todosJSON) : [] // 34. jeśli todosJSON istnieje, to go sparsuj i przypisz do todos, jak nie, to przypisz pustą tablicę

} catch (e) { // 35. jeśli nastąpi error to:

todos = [] // 36. przypisz do todos pustą tablicę [powót do wątku](#tłum37)

}

}

// Save todos to localStorage

const saveTodos = () => { // [30.](#tłum30)

localStorage.setItem('todos', JSON.stringify(todos)) // 31. wejdź to localStorage i przypisz zmiennej ‘todos’ zestringowaną tablicę ‘todos’

}

const getTodos = () => todos [4.](#tłum4)

const createTodo = (text) => { [// 64.](#tłum64)

todos.push({ // 65. umieść na końcu tablicy to todo

id: uuidv4(), // 66. nadaj mu unikatowe ID

text, // 67. text = text (skrót)

completed: false // 68. domyślnie każde todo ma być niewykonane

})

saveTodos() // 69. zasejwuj to w localStorage [powróć](#tłum70)

}

const removeTodo = (id) => { [// 47.](#tłum47) [tłumaczenie findIndex](#extra5)

const todoIndex = todos.findIndex((todo) => todo.id === id) // 48. przeleć po wszystkich elementach tablicy i znajdź nr indeksu todo o takim samym ID

if (todoIndex > -1) { // 49. jeśli todoIndex jest większ niż -1 (czyli od zera w górę to):

todos.splice(todoIndex, 1) // 49. wytnij z tablicy todos element o takim indexie i niech to będzie tylko ten jeden element

saveTodos() // 50. zaktualizuj tablicę todos w taki sposób [powrót do wątku](#tłum51)

}

}

// Toggle the completed value for a given todo

const toggleTodo = (id) => { [// 27.](#tłum27)

const todo = todos.find((todo) => todo.id === id) // 28. weź tablicę todos stąd: [todos](#odn28) i odnajdź w niej to jedno konkretne todo po id metodą: [find](#extra4)

if (todo) {

todo.completed = !todo.completed // 29. jeśli todo ma complited: true, to przerób na false i na odwrót

saveTodos() // 30. [zaktualizuj tablicę todos](#odn30)

}

}

loadTodos() // 32. [załaduj z localstorage tablicę ‘todos’ – tę funkcję zassie index.js i ją wykona z automatu](#odn32)

export { loadTodos, getTodos, createTodo, removeTodo, toggleTodo }

**Index.html**

<!DOCTYPE html>

<html>

<head>

<title>To-do App</title>

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

<link href="/images/favicon.png?v=1" rel="icon">

</head>

<body>

<div class="header">

<div class="container">

<h1 class="header\_\_title">To-do App</h1>

<h2 class="header\_\_subtitle">What do you need to do?</h2>

</div>

</div>

<div class="actions">

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

<input id="search-text" class="input" type="text" placeholder="Filter todos"> // [57.](#tłum57bis)

<label class="checkbox">

<input type="checkbox" id="hide-completed"> Hide completed [// 71.](#tłum71)

</label>

</div>

</div>

<div class="container">

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

<form id="new-todo"> [// 60.](#tłum60)

<input type="text" class="input" placeholder="Something to do" name="text"> // to co wpiszemy do inputa będzie stanowiło text.value

<button class="button">Add Todo</button>

</form>

</div>

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

</body>

</html>

**Wytłumaczenia:**

**[Tłumaczenie nr 1:](#extra1origin)**

const filters = {

searchText: '',

hideCompleted: false

}

const getFilters = () => filters

const { searchText, hideCompleted } = getFilters()

console.log(searchText) // pokaże: ''

console.log(hideCompleted) //pokaże: false

**[tłumaczenie nr:2](#extra2origin)**

[{"id":"8a52709b-a146-4460-9ca2-55e19bd49671","text":"aaa","completed":false},

{"id":"ebe138ca-fec4-4fec-9d6e-dc74c5c4b428","text":"bbb","completed":false},

{"id":"f4b50bdf-4edf-4285-82c3-624c1015a7eb","text":"ccc","completed":false},

{"id":"8e4d75c3-5190-4359-9545-f0a856da209b","text":"ddd","completed":false},

{"id":"26b128e2-d579-4086-a290-eb3a47cd408d","text":"eee","completed":false},

{"id":"586e9219-94f8-4e93-88c0-05034fcd4a22","text":"hjklnm","completed":false}]

[**tłumaczenie nr: 3**](#extra3origin)

**metoda filter:**

**let digits = [123, 87, 90, 37, 22, 213]**

**let newDigits = digits.filter(function(digit) {**

**return (digit \* 2) < 300**

**})**

**console.log(newDigits) // pokaże:** [ 123, 87, 90, 37, 22 ]

[tłumacznie nr: 4](#extra4origin)

let vegetables = ['tomato', 'carrot', 'potato', 'cucumber', 'pumpkin', 'carrot']

let firstMatchingVegetable = vegetables.find(function(vegetable) {

return vegetable === 'carrot'

})

console.log(firstMatchingVegetable) // pokaże "carrot"

[tłumaczenie: 5](#extra5origin)

let numbers = [4238, 4738, 4553, 22, 980, 1098, 111]

let indexOfFirstMatchingNumber = numbers.findIndex(function (number) {

return (number - 1000) > 3000

})

console.log(indexOfFirstMatchingNumber) // pokaże 0

[tłumaczenie: 6](#extra6origin)

const filters = {

searchText: '',

hideCompleted: false

}

const setFilters = ({ searchText, hideCompleted }) => {

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

filters.searchText = searchText

}

if (typeof hideCompleted === 'boolean') {

filters.hideCompleted = hideCompleted

}

}

setFilters({

searchText: 'a'

})

console.log(filters) // pokaże: { hideCompleted: false, searchText: "a" }

[tłumaczenie: 7](#extra7origin)

const filters = {

searchText: '',

hideCompleted: false

}

const setFilters = ({ searchText, hideCompleted }) => {

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

filters.searchText = searchText

}

if (typeof hideCompleted === 'boolean') {

filters.hideCompleted = hideCompleted

}

}

setFilters({

hideCompleted: true

})

console.log(filters)