**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) => {

setFilters({

searchText: e.target.value

})

renderTodos()

})

document.querySelector('#new-todo').addEventListener('submit', (e) => {

const text = e.target.elements.text.value.trim()

e.preventDefault()

if (text.length > 0) {

createTodo(text)

renderTodos()

e.target.elements.text.value = ''

}

})

document.querySelector('#hide-completed').addEventListener('change', (e) => {

setFilters({

hideCompleted: e.target.checked

})

renderTodos()

})

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

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

loadTodos()

renderTodos()

}

})

**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 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) {

filteredTodos.forEach((todo) => {

todoEl.appendChild(generateTodoDOM(todo))

})

} else {

const messageEl = document.createElement('p')

messageEl.classList.add('empty-message')

messageEl.textContent = 'There are no to-dos to show'

todoEl.appendChild(messageEl)

}

}

// Get the DOM elements for an individual note

const generateTodoDOM = (todo) => {

const todoEl = document.createElement('label')

const containerEl = document.createElement('div')

const checkbox = document.createElement('input')

const todoText = document.createElement('span')

const removeButton = document.createElement('button')

// Setup todo checkbox

checkbox.setAttribute('type', 'checkbox')

checkbox.checked = todo.completed

containerEl.appendChild(checkbox)

checkbox.addEventListener('change', () => {

toggleTodo(todo.id)

renderTodos()

})

// Setup the todo text

todoText.textContent = todo.text

containerEl.appendChild(todoText)

// Setup container

todoEl.classList.add('list-item')

containerEl.classList.add('list-item\_\_container')

todoEl.appendChild(containerEl)

// Setup the remove button

removeButton.textContent = 'remove'

removeButton.classList.add('button', 'button--text')

todoEl.appendChild(removeButton)

removeButton.addEventListener('click', () => {

removeTodo(todo.id)

renderTodos()

})

return todoEl

}

// Get the DOM elements for list summary

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

const summary = document.createElement('h2') // 10. utwórz element h1 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')

summary.textContent = `You have ${incompleteTodos.length} todo${plural} left`

return 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 }) => {

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)

// Fetch existing todos from localStorage

const loadTodos = () => {

const todosJSON = localStorage.getItem('todos')

try {

todos = todosJSON ? JSON.parse(todosJSON) : []

} catch (e) {

todos = []

}

}

// Save todos to localStorage

const saveTodos = () => {

localStorage.setItem('todos', JSON.stringify(todos))

}

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

const createTodo = (text) => {

todos.push({

id: uuidv4(),

text,

completed: false

})

saveTodos()

}

const removeTodo = (id) => {

const todoIndex = todos.findIndex((todo) => todo.id === id)

if (todoIndex > -1) {

todos.splice(todoIndex, 1)

saveTodos()

}

}

// Toggle the completed value for a given todo

const toggleTodo = (id) => {

const todo = todos.find((todo) => todo.id === id)

if (todo) {

todo.completed = !todo.completed

saveTodos()

}

}

loadTodos()

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" //

<label class="checkbox">

<input type="checkbox" id="hide-completed"> Hide completed

</label>

</div>

</div>

<div class="container">

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

<form id="new-todo">

<input type="text" class="input" placeholder="Something to do" name="text">

<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 ]