**Kamil Sabiron**

**20.01.2022**

**Sprawozdanie nr 02**

**Temat: Programowanie obiektowe, aplikacja Todo**

**Przebieg zadania:**

Część praktyczna (kod, screeny, opis działania funkcji, opis przebiegu zadania)

**Kod**

**HTML**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Todo App</title>

<link rel="stylesheet" href="style.css">

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

</head>

<body>

<header>

<h1>Todo Task List</h1>

<form id="new-task-inputs">

<input type="text" id="new-task-text" placeholder="Write your task here!"/>

<input type="submit" id="new-task-submit" value="Add task"/>

</form>

</header>

<main>

<section class="task-list">

<h2>Tasks</h2>

<div id="tasks">

<!--

<div class="task">

<div class="content">

<input type="text" class="text" value="My task" readonly/>

</div>

<div class="buttons">

<button class="edit">Edit</button>

<button class="delete">Delete</button>

</div>

</div>

-->

</div>

</section>

</main>

</body>

</html>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSS**

body{

background: #303030;

display: flex;

flex-direction: column;

color: whitesmoke;

}

\*{

font-family: sans-serif, Arial;

}

header{

padding: 3em;

width: 100%;

max-width: 900px;

}

header h1{

padding-bottom: 0.5em;

font-size: 2.5rem;

font-weight: 700;

color: #606060;

}

#new-task-inputs{

display: flex;

}

#new-task-text{

flex: 1 1 0%;

padding: 1em;

border-radius: 1em;

background: #404040;

}

#new-task-submit{

color: firebrick;

border: none;

background: none;

outline: none;

appearance: none;

padding-left: 1.5em;

font-size: 15px;

font-weight: 700;

}

#new-task-submit:hover{

opacity: 0.7;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

main{

flex: 1 1 0%;

width: 100%;

max-width: 900px;

}

main h2{

color: #606060;

padding-bottom: 0.5em;

font-size: 2rem;

font-weight: 700;

}

.task-list{

padding: 3em;

}

button, input{

border: none;

background: none;

outline: none;

appearance: none;

}

#tasks .task{

display: flex;

justify-content: space-between;

background: #202020;

padding: 1em;

border-radius: 1em;

margin-bottom: 1em;

}

#tasks .task .content .text{

color: #707070;

font-size: 1.25em;

width: 100%;

display: block;

}

#tasks .task .buttons{

display: flex;

margin: 0 -0.5em;

}

.task .actions button{

cursor: pointer;

margin: 0 0.5em;

font-size: 1.25em;

}

.edit{

color: turquoise;

}

.edit:hover{

opacity: 0.7;

}

.delete{

color: rgb(255, 0, 0);

}

.delete:hover{

opacity: 0.7;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**JavaScript**

window.addEventListener('load', () => {

//Variables

const form = document.querySelector("#new-task-inputs");

const input = document.querySelector("#new-task-text");

const list = document.querySelector("#tasks");

//It works when is submit event (my add task)

form.addEventListener('submit', (test) => {

test.preventDefault(); //it stops from refreshing the page

//value assignment

const task = input.value;

//its checks value, if its empty it will print an alert

if(!task){

alert("Fill out the task!");

}

//create element and add to class "task"

const task\_element = document.createElement("div");

task\_element.classList.add("task")

//create element and add to class "content"

const task\_element\_content = document.createElement("div");

task\_element\_content.classList.add("content")

task\_element.appendChild(task\_element\_content);

//Create element - input in class "text" with value task (text in input)

const task\_element\_input = document.createElement("input");

task\_element\_input.classList.add("text");

task\_element\_input.value = task;

task\_element\_content.appendChild(task\_element\_input)

//create Buttons

const task\_element\_buttons = document.createElement("div")

task\_element\_buttons.classList.add("buttons")

//Edit button

const task\_element\_edit = document.createElement("button");

task\_element\_edit.classList.add("edit")

task\_element\_edit.innerText = "Edit";

//Delete button

const task\_element\_delete = document.createElement("button");

task\_element\_delete.classList.add("delete")

task\_element\_delete.innerText = "Delete";

task\_element\_buttons.appendChild(task\_element\_edit);

task\_element\_buttons.appendChild(task\_element\_delete);

task\_element.appendChild(task\_element\_buttons);

list.appendChild(task\_element);

//Buttons action

//Edit

task\_element\_edit.addEventListener('click', () => {

if(task\_element\_edit.innerText == "Edit"){

task\_element\_input.removeAttribute("readonly");

task\_element\_edit.innerText = "Save";

}else{

task\_element\_input.setAttribute("readonly", "readonly");

task\_element\_edit.innerText = "Edit"

}

})

//Delete

task\_element\_delete.addEventListener('click', () => {

list.removeChild(task\_element);

})

})

})

**Opis działania funkcji:**



Cały kod w JS jest zrobiony w funcji window która jest wywoływana przy ładowaniu strony i stworzone zostaly stałe zmienne.



Wszystkie funkcje od tego momentu są w funkcji submit które są wywoływane za pomocą przycisku submit (Add task), wartość input'a zostala przypisana zmiennie task i sprawdzamy czy w zmiennej task znajduje sie jakaś wartość. Jeśli nie, to wyskakuje nam alert.



Stworzenie elementów div i dopisanie ich do klas "task" i "content".



Stworzenie elementu input z klasa "text" i z wartością "task" a ona jest pobierana wczesniej z input'a



Stworzenie elementu "div" ktory zostaje przypisany klasie "buttons", następnie tworzymy element "button" któremu przypisuje klase "edit" i wartość "Edit, i to samo z przyciskiem "Delete".



Wyświetlanie elementów na ekran.



Akcje przycisków (Edit, Delete), Edit pozwala na edycje task'a a Delete na usunięcie task'a.

**Opis przebiegu zadania:**

Prace zacząłem od stworzenia repozytorium i plików. Następnie w pliku tekstowym opisałem co to jest programowanie obiektowe i przystąpiłem do HTML. Po HTML zacząłem robić cześć wizualną w CSS, która zabrała mi stosunkowo dużo czasu i w trakcie były zmiany w HTML bo postanowiłem przykładowo zmienić nie które Id na class. Gdy stwierdziłem, że strona wygląda tak jak chcę przeszedlem do JS w którym zacząłem od zablokowania wysłania pustych Task'ów (wyskakuje alert jak jest pusty). Po tym zacząłem tworzyć elementy (div, input, button) i przypisywać im klasy aby wygladaly tak jak wszystkie oraz wypisywaniem tego na ekran. Na końcu zrobiłem aktywności przycisków edit i delete, jak nazwa sama wskazuje słuzą one do edytowania i usuwania task'ów.



Strona bo załadowaniu



komunikat gdy mamy pusty Task i chcemy go dodać



Po dodaniu przykładowego taska