// Отправляем данные на сервер

async function postClientsServer(obj) {

let btn = document.querySelector(".modal-btn");

btn.append(createPreLoader("loader-btn"));

try {

const response = await fetch("http://localhost:3000/api/clients", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify(obj),

});

if (response.ok) {

// Если запрос успешный удаляем прелоадер, закрывает модальное окно и отправляем данные

document.querySelector(".loader-btn").remove();

createDeleteModal(".modal-wrapper", ".modal-inner");

const data = await response.json();

return data;

}

} catch (error) {

console.error(`Ошибка от сервера - ${error}`);

createErrorServer(errorMessage);

}

}

// Изменяем данные на сервере

async function changeClientInfo(client) {

let btn = document.querySelector(".modal-btn");

btn.append(createPreLoader("loader-btn"));

try {

const response = await fetch(

`http://localhost:3000/api/clients/${client.id}`,

{

method: "PATCH",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

name: client.name,

surname: client.surname,

lastName: client.lastName,

updatedAt: client.updatedAt,

contacts: client.contacts,

}),

}

);

if (response.ok) {

// Если запрос успешный удаляем прелоадер, закрывает модальное окно и получаем данные

document.querySelector(".loader-btn").remove();

createDeleteModal(".modal-wrapper", ".modal-inner");

const data = await response.json();

return data;

}

} catch (error) {

document.querySelector(".loader-btn").remove();

createErrorServer(errorMessage);

console.error(`Ошибка от сервера - ${error}`);

}

}

// Получаем данные с сервера

async function getClientsServer() {

table.$tableBody.append(createPreLoader("loader-table"));

const response = await fetch("http://localhost:3000/api/clients", {

method: "GET",

headers: {

"Content-Type": "application/json",

},

});

const data = await response.json();

if (response.status === 200) {

createPreLoader("loader-table").remove();

return data;

}

}

let servreData = await getClientsServer();

if (servreData != null) {

studentList = servreData;

}

// Удаляем данные с сервера

async function deleteClientServer(id) {

const response = await fetch(`http://localhost:3000/api/clients/${id}`, {

method: "DELETE",

});

const date = await response.json();

return date;

}

// Поиск на сервере

async function getSearchServer(client) {

table.$tableBody.append(createPreLoader("loader-table"));

try {

const response = await fetch(

`http://localhost:3000/api/clients/?search=${client.trim()}`

);

const data = response.json();

if (response.status === 200) {

createPreLoader("loader-table").remove();

return data;

}

} catch (error) {

console.error(error);

}

}

// Создание списка контактов

function createContacts(student) {

let type;

switch (student.type) {

case "Телефон":

type = "tel";

break;

case "Facebook":

type = "fb";

break;

case "Email":

type = "email";

break;

case "Вконтакте":

type = "Вконтакте";

break;

}

let $item = document.createElement("span");

$item.classList.add("contacts-item");

$item.innerHTML = icon[type];

$item.setAttribute("data-tooltip", `${student.type}: ${student.value}`);

//Создание тултипа

let tooltipElem;

$item.addEventListener("mouseover", function (event) {

let target = event.currentTarget;

let tooltipDOM = target.dataset.tooltip;

if (!tooltipDOM) return;

tooltipElem = document.createElement("div");

tooltipElem.classList.add("tooltip");

tooltipElem.innerHTML = tooltipDOM;

document.body.append(tooltipElem);

let coords = target.getBoundingClientRect();

let left =

coords.left + (target.offsetWidth - tooltipElem.offsetWidth) / 2;

if (left < 0) left = 5;

let top = coords.top - tooltipElem.offsetHeight - 5;

if (top < 0) left = coords.top + target.offsetHeight + 5;

tooltipElem.style.left = left + "px";

tooltipElem.style.top = top + "px";

});

$item.addEventListener("mouseout", function (event) {

if (tooltipElem) {

tooltipElem.remove();

tooltipElem = null;

}

});

return $item;

}

function inputMask() {

const contacts = document.querySelectorAll(".wrapper-contacts");

contacts.forEach((contact) => {

const select = contact.querySelector(".select-contacts");

const input = contact.querySelector(".input-contacts");

let mask;

// Функция для создания маски

function updateMask(type) {

if (mask) {

mask.destroy(); // Удаление старой маски

input.value = ""; // Очищаем значение input

}

switch (type) {

case "Телефон":

mask = IMask(input, { mask: "+{7}(000)000-00-00" });

break;

case "Вконтакте":

mask = IMask(input, {

mask: "{vk.com/}{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}",

});

break;

case "Facebook":

mask = IMask(input, {

mask: "{f\\acebook.com/}\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",

});

break;

case "Email":

// для Email маска не нужна, просто чистим маску, если она была

break;

default:

// no default case

}

}

// Инициализация маски при загрузке страницы, если select уже имеет значение

updateMask(select.value);

select.addEventListener("change", () => {

updateMask(select.value);

});

});

}

//Валидация почты

function validationEmail() {

const form = document.querySelector(".form");

function isEmailValid(value) {

const EMAIL\_REGEXP = /^[\w-\.]+@[\w-]+\.[a-z]{2,4}$/i;

return EMAIL\_REGEXP.test(value);

}

// Определение обработчика input для валидации email

function onInput(input) {

const handler = () => {

if (isEmailValid(input.value)) {

input.classList.remove("invalid", "icon-invalid");

input.classList.add("valid", "icon-valid");

document.querySelector(".modal-btn").style.pointerEvents = "auto";

} else {

input.classList.remove("valid", "icon-valid");

input.classList.add("invalid", "icon-invalid");

document.querySelector(".modal-btn").style.pointerEvents = "none";

}

};

input.addEventListener("input", handler);

return handler;

}

const contacts = form.querySelector(".contacts-container").children;

Array.from(contacts).forEach((contact) => {

const select = contact.querySelector(".select-contacts");

const input = contact.querySelector(".input-contacts");

let inputHandler = null;

select.addEventListener("change", () => {

// Удаление предыдущего обработчика, если он был установлен

if (inputHandler) {

input.removeEventListener("input", inputHandler);

input.classList.remove("valid", "icon-valid");

input.classList.remove("invalid", "icon-invalid");

inputHandler = null;

}

if (select.value === "Email") {

inputHandler = onInput(input); // Присваиваем новый обработчик

}

});

// Удаление стилей при потере фокуса

input.addEventListener("blur", () => {

input.classList.remove("valid", "icon-valid");

input.classList.remove("invalid", "icon-invalid");

});

});

}

//Сортировка

function sortClients(clients, prop, dir) {

return clients.sort((userOne, userTwo) => {

if (

!dir == false

? userOne[prop] < userTwo[prop]

: userOne[prop] > userTwo[prop]

)

return -1;

});

}