Міністерство освіти і науки України

Черкаський державний технологічний університет

Звіт

про виконання лабораторної роботи №3 з навчальної дисципліни

«Програмування WEB додатків»

Виконав: студент   
 Групи ПЗС-2044

Оратівський С.І.

Перевірив:

Метелап В.В.

Черкаси

2022

**Тема: TypeScript**

**Завдання: Переробити проект використовуючи TypeScript**

Посилання на коміт: [SerhiiOrSoratik/internet-shop-Web-Application-Programming at 13fa4f46e8c2db1d2e1ad618faab04f99da3187c (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/13fa4f46e8c2db1d2e1ad618faab04f99da3187c)

Frontend/index.ts

import React from 'react';  
import ReactDOM from 'react-dom/client';  
import './index.css';  
import App from './App';  
  
const root = ReactDOM.createRoot(***document***.getElementById('root') as HTMLInputElement);  
root.render(  
 <React.StrictMode>  
 <App />  
 </React.StrictMode>  
);

Frontend/Components/MainPage.ts

import {useMemo, useState} from "react";  
import {getEntityById, loadData, removeEntityById} from "../Api/Api";  
import {Form} from "./Form";  
import {Entity} from "../Types";  
  
type Props = {  
 filter: string,  
 showForm: boolean,  
 setShowForm: any  
}  
  
export const MainPage = (props : Props) => {  
  
 const [allEntities, setEntities] = useState([]);  
 const [editingEntity, setEditingEntity] = useState({id: '', caption: '', price: 0, description: '', availability: false});  
  
 useMemo(() => {  
 loadData(props.filter).then(result => setEntities(result));  
 }, [props.filter])  
  
 const updateData = () => {  
 loadData(props.filter).then(result => setEntities(result));  
 }  
  
 const removeById = (id : string) => {  
 removeEntityById(id).then(() => loadData('').then(result => setEntities(result)));  
 }  
  
 const editById = (entity : Entity) => {  
 setEditingEntity(entity)  
 props.setShowForm(true)  
 }  
  
 const downloadJsonFile = async (id: string) => {  
 const data = await getEntityById(id);  
 const json = ***JSON***.stringify(data);  
 let element = ***document***.createElement('a');  
 element.setAttribute('href', 'data:application/json;charset=utf-8,' + encodeURIComponent(json));  
 element.setAttribute('download', `${data.caption}.json`);  
 element.click();  
 }  
  
 const downloadScvFile = async (id: string) => {  
 const data = await getEntityById(id);  
 const csv = convertToCSV(data);  
 let element = ***document***.createElement('a');  
 element.setAttribute('href', 'data:application/json;charset=utf-8,' + encodeURIComponent(csv));  
 element.setAttribute('download', `${data.caption}.csv`);  
 element.click();  
 }  
  
 const convertToCSV = (data : Entity) => {  
 const keys = Object.keys(data);  
 const values = Object.values(data);  
 let result = keys.join(',');  
 result += '\n';  
 result += values.join(',')  
 return result;  
 }  
  
 return (  
 <>  
 <div id='container'>  
 {props.showForm ? <Form setShowForm={props.setShowForm} setEditingEntity={setEditingEntity}  
 editingEntity={editingEntity} updateData={updateData}/> : null}  
 <div id='catalog'>  
 {  
 allEntities ? allEntities.map((entity : Entity) =>  
 <div style={{  
 border: '1px solid black',  
 width: '20%',  
 minWidth: '300px',  
 height: 'auto',  
 marginBottom: ' 25px'  
 }}  
 key={entity.id}>  
 <button onClick={() => removeById(entity.id)}  
 style={{  
 transform: ' translate(145px, 5px)',  
 position: 'absolute',  
 height: '25px',  
 width: '25px'  
 }}>x  
 </button>  
 <button onClick={() => editById(entity)}  
 style={{  
 transform: 'translate(115px, 5px)',  
 position: 'absolute',  
 height: 25,  
 width: 25  
 }}>✎  
 </button>  
 <button onClick={() => downloadJsonFile(entity.id)}  
 style={{  
 transform: 'translate(70px, 5px)',  
 position: 'absolute',  
 height: 25,  
 width: 40  
 }}  
 >json  
 </button>  
 <button onClick={() => downloadScvFile(entity.id)}  
 style={{  
 transform: 'translate(25px, 5px)',  
 position: 'absolute',  
 height: 25,  
 width: 40  
 }}  
 >csv  
 </button>  
 <p style={{  
 color: 'gray',  
 fontSize: 12,  
 paddingTop: 10  
 }}>{entity.availability ? ' ' : 'Немає в наявності'}</p>  
 <p>{entity.caption}</p>  
 <p>{entity.price} грн</p>  
 <p style={{wordWrap: 'break-word', color: 'gray'}}>{entity.description}</p>  
 </div>  
 ) : null  
 }  
 </div>  
 </div>  
 </>  
 )  
}

Frontend/Components/Form.ts

import React, {useState} from "react";  
import {createEntity, editEntityById} from "../Api/Api";  
import {Entity} from "../Types";  
  
type Props = {  
 setShowForm: any,  
 setEditingEntity: any;  
 editingEntity: Entity;  
 updateData: any  
}  
  
export const Form = (props : Props) => {  
  
 const [caption, setCaption] = useState(props?.editingEntity?.caption ? props.editingEntity.caption : '')  
 const [price, setPrice] = useState(props?.editingEntity?.price ? props.editingEntity.price : 0)  
 const [description, setDescription] = useState(props?.editingEntity?.description ? props.editingEntity.description : '')  
  
  
 const handleSubmit = (event: React.FormEvent<HTMLFormElement>) => {  
 event.preventDefault();  
 if (props.editingEntity?.id) {  
 editEntityById({caption, price, description}, props.editingEntity.id).then(() => props.updateData())  
 } else {  
 createEntity({caption, price, description}).then(() => props.updateData())  
 }  
 clear();  
 }  
  
 const clear = () => {  
 props.setShowForm(false)  
 setDescription('')  
 setPrice(0)  
 setCaption('')  
 props.setEditingEntity({})  
 }  
  
 return (  
 <form id="crForm" encType="application/json" className="form-container" onSubmit={handleSubmit}>  
 <h3>{props.editingEntity?.id ? 'Edit book' : 'Add book'}</h3>  
 <div className="form-group">  
 <label htmlFor="cr\_caption">Caption</label>  
 <input id="cr\_caption" name="caption" placeholder="Enter caption" value={caption}  
 onChange={(e) => setCaption(e.target.value)} required/>  
 </div>  
 <div className="form-group">  
 <label htmlFor="cr\_price">Price</label>  
 <input id="cr\_price" type={"number"} name="price" placeholder="Enter price" value={price}  
 onChange={(e) => setPrice(+e.target.value)} required/>  
 </div>  
 <div className="form-group">  
 <label htmlFor="cr\_description">Description</label>  
 <input id="cr\_description" name="description" placeholder="Enter description" value={description}  
 onChange={(e) => setDescription(e.target.value)}/>  
 </div>  
 <div style={{display: 'flex', justifyContent: 'space-between'}}>  
 <button className="form\_button" value="Cancel" onClick={() => {  
 clear()  
 }}>Cancel  
 </button>  
 <button className="form\_button" type="submit">Ok</button>  
  
 </div>  
 </form>  
 )  
}

Frontend/Components/Header.ts

type Props = {  
 setFilter: any,  
 setShowForm: any  
}  
  
export const Header = (props: Props) => {  
  
 const search = () => {  
 const searchValue = (***document***.getElementById("search") as HTMLInputElement).value;  
 props.setFilter(searchValue)  
 }  
  
 return (  
 <div id="header">  
 <h1>Book shop</h1>  
 <div style={{width: '50%'}}>  
 <input style={{width: '15%', height: '88%'}} id="search"/>  
 <button className='form\_button' style={{width: '15%'}} id="searchButton" onClick={() => search()}>  
 Search  
 </button>  
 </div>  
 <button className='form\_button' id="addBook" style={{width: '15%'}} onClick={() => {  
 props.setShowForm(true)  
 }}>Add book  
 </button>  
 </div>  
 )  
}

Frontend/Api/Api.ts

import {NewOrUpdatedEntity} from "../Types";  
  
export const loadData = async (searchValue : string) => {  
 if (searchValue) {  
 return await fetch('http://localhost:3001/product/search/' + searchValue,  
 {  
 method: 'GET',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
 } else {  
 return await fetch('http://localhost:3001/product',  
 {  
 method: 'GET',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
 }  
}  
  
export const removeEntityById = async (id : string) => {  
 return await fetch('http://localhost:3001/product/' + id,  
 {  
 method: 'DELETE',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data)  
 .catch(error => ***console***.log(error))  
}  
  
export const getEntityById = async (id : string) => {  
 return await fetch('http://localhost:3001/product/' + id,  
 {  
 method: 'GET',  
 headers: {'Accept': 'application/json'}  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
}  
  
export const createEntity = async (entity : NewOrUpdatedEntity) => {  
 return await fetch('http://localhost:3001/product',  
 {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: ***JSON***.stringify(entity)  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
}  
  
export const editEntityById = async (entity : NewOrUpdatedEntity, id : string) => {  
 return await fetch('http://localhost:3001/product/' + id,  
 {  
 method: 'PUT',  
 headers: {'Content-Type': 'application/json'},  
 body: ***JSON***.stringify(entity)  
 }).then(data => data.json())  
 .then(result => result)  
 .catch(error => ***console***.log(error))  
}

Frontend/Types.ts

export type NewOrUpdatedEntity = {  
 caption: string,  
 price: number,  
 description: string,  
}  
  
export type Entity = {  
 id: string,  
 caption: string,  
 price: number,  
 description: string,  
 availability: boolean  
}

**Питання:**

1) Як зводити до типу(cast) у TypeScript?

Слід використовувати затвердження типу(type assertion)

let array: any = [1,2,3,4,5];

let arrayLength: number = (<Array[]>array).length;

//або

let array: any = [1,2,3,4,5];

let strLength: number = (array as Array[]).length;

Обидва варіанти еквівалентні, однак, якщо ви використовуєте typescript в JSX ключове слово як зручніше.

2) Як перевірити тип об'єкта?

Використовуючи JavaScript оператор instanceof:

class Animal{

private name:string;

}

let bird = New Animal ();

console.log(bird instanceof Animal) // true

3) Як встановити TypeScript?

Typescript доступний у менеджерах пакетів NPM та Yarn

npm install -g typescript

4) Як скомпілювати один файл?

//helo.ts

console.log('Hello world!')

$tsc hello.ts

5) Як оголосити масив?

Масив є одним із базових типів, щоб оголосити його потрібно використовувати такий синтаксис:

let array: number[] = [1, 2, 3];

interface Person{

name:string;

}

let employees:Array<Person>=[{name:'John Doe'}] //declaring array of objects

6) Як перевірити, чи є змінна undefined та/або null?

Найпростіший спосіб – використовувати оператор (something == null). Повертає true, якщо щось дорівнює null або undefined. Також можна використовувати оператор заперечення.

let obj:any={};

let nullValue:any=null;

let undefinedValue=obj.someProp;

console.log(nullValue==null)//true

console.log(undefinedValue==null)//true

if(!nullValue || !undefinedValue){

console.log(true)

}

7) Як створити тип перерахування в TypeScript?

Використовуючи ключове слово enum. Тіло перерахування може складатися з нуля чи багатьох елементів. Кожен елемент має ім'я та значення. Значення може бути оголошено як константа або обчислено.

enum Status {

New = 1,

Pending,

InProgress,

Done

}

console.log(Status.Pending)//2

8) Як перезавантажити метод базового класу?

Просто реалізуйте необхідний метод у похідному класі. Метод у базовому класі повинен мати public або protected. модифікатор

class Shape{

public paint(){

throw new Error("Abstract method execution.")

}

}

class Square extends Shape{

private height:number;

constructor (height:number){

super();

this.height=height;

}

paint(){

console.log('|Square|')

}

}

let obj: Shape = new Square (5);

obj.paint(); // | Square |

9) Як вести налагодження (debug) TypeScript?

Немає рекомендованого способу зробити це. Найбільш популярним є використання бібліотеки ts-node, яка дає вам можливість завантажувати \*.ts файли безпосередньо у JavaScript або запустити файл ts. Також за допомогою протоколу V8 Inspector ми можемо налагоджувати Node.js програму безпосередньо в Chrome, використовуючи потужні Chrome devtools.

require('ts-node').register();

require('./MyAwesomeClass');

$ts-node script.ts

10) Як імпортувати модуль?

Будь-який файл, який містить ключове слово import/export, вважається модулем. Існує два типи експорту: default та named. Для імпорту експорту за промовчанням напишіть: import \* as Module from './SomeModule'. Для іменованого імпорту export-import {SomeFunc} from './SomeModule'.

export default class MyClass{

someProp:any;

someMethod(){

}

}

//Module B

import MyClass from './ModuleA'

import \* as moment from "moment";

11) Як називати інтерфейси?

Згідно з офіційними рекомендаціями щодо кодування, для імен типів слід використовувати PascalCase. Також не використовуйте префікс "I" в іменах інтерфейсів. Для реалізації інтерфейсу можна додати суфікс Impl(щось на зразок WritableImpl)

12) Що таке union type?

Змінна повинна бути оголошена з типом union, якщо вона може бути встановлена ​​значення різних типів. Це може бути корисно, коли вам потрібно повернути правильний результат або нульове значення, наприклад, функції find

13) Що таке декоратори та як їх можна використовувати?

Декоратор - це функція, яка дозволяє додавати анотації та синтаксис мета програмування в оголошення класу та функцій-членів. Декоратор може бути приєднаний до оголошення класу, оголошення властивості, методу, методів доступу get/set або параметру. Синтаксис дуже простий, схожий на інструкції Java.

@Entity

export default class MyClass{

@Prop("primaryKey")

private id:number;

}

14) Як отримати доступ до методів і властивостей суперкласу?

Щоб викликати конструктора базового класу, просто запустіть super (). Для виклику методу базового класу використовуйте наступний синтаксис:

class Abstract{

public method(){

}

}

class Concrete extends Abstract{

paint(){

super.method();

}

}

15) Як використовувати forEach і for.. у TypeScript?

Немає різниці між тим, як використовувати JavaScript, за винятком того, що ви можете вказати тип для елемента масиву . Також буде помилка компілятора при спробі запустити метод forEach для об'єкта або рядка.

let notArray='dsd';

notArray.forEach(e=>{}) // Compilation error - Unresolved function forEach

let array:number[]=[1,2,3];

array.forEach(e=>e.toExponential(2)) // Compiler knows that has number type

16) Які модифікатори доступу існують у TypeScript?

Існує три типи: public,private,protected. За замовчуванням, всі властивості та методи класу є відкритими, але краще додати це явно, щоб зробити код більш читаним.

Використовуйте protected модифікатор, якщо ви хочете надати доступ до всіх класів нащадків, це корисно у складній конструкції ООП.

Нарешті, використовуйте private для інкапсуляції деталей для реалізації, це допомагає захистити клієнта класу від його внутрішніх змін структури. Зверніть увагу, що всі ці перевірки доступу працюватимуть лише під час компіляції, тому що JS не підтримує модифікатори доступу.

17) Що таке качина типізація (duck typing)?

Качина типізація вважає, що два об'єкти мають однаковий тип, якщо вони мають однаковий набір властивостей. Оголошення об'єкта літералом одно створення об'єкта за допомогою оператора new.

18) Що таке дженерики (generics)? Як їх використати?

Generic є деяким узагальненим типом, який може бути замінений будь-яким конкретним типом, таким як string або number. Універсальні шаблони дозволяють писати універсальні функції, із збереженням типу, без використання "будь-якого" типу. Існують універсальні функції, класи або інтерфейси, наприклад, можна написати клас Container<T>, який реалізує базові операції збору, незалежно від типу елементів, що зберігаються.

function loggingObject<T>(arg: T): T {

console.log(arg.toString());

return arg;

}

19) Як використовувати простори імен із модулями TypeScript?

Простір імен допомагає організувати код шляхом угруповання логічно пов'язаних типів, класів, функцій, а також з їх допомогою запобігає конфліктам імен у різних модулях. Так як модулі вже існують у FS, тому ми повинні завантажити їх на шляху та імені файлу, тому для нас є логічна схема організації. Конфлікти імен можна вирішити за допомогою команди "імпортувати як"

20) Як передати параметри компілятору TypeScript?

Через аргументи командного рядка або файл tsconfig.json. Якщо файл tsconfig.json, розташований у певному каталозі, це означає, що це корінь проекту TypeScript. Є такі варіанти: компілятор параметри,файли,typeRoots, включати, виключати.

21) Є якийсь спосіб автоматично перекомпілювати проект, якщо файл змінився?

Так, виконайте команду tsc з аргументом -w, також передайте опцію --no-initial-compiler, якщо ви хочете компілювати тільки після змін, але без початкової компіляції.

22) Як використовувати JSX у проекті TS?

Щоб використовувати JSX, збережіть код у файлах з розширенням \*.jsx та увімкніть опцію jsx у tsconfig.json. Існує три режими JSX react, preserve, react-native.

23). Навіщо потрібен тип «Omit»?

Відповідь: Це новий тип, в якому можна вказати властивості, які виключатимуться з вихідного типу.

Приклад:

type Person = {name: string; age: number; location: string; };

type QuantumPerson = Omit<Person, 'location'>; // Аналогічно наступному рядку

QuantumPerson = {name: string; age: number; };

24). Коли потрібно використати ключове слово “declare”?

Відповідь: При використанні бібліотеки JavaScript, яка не оголошена у вашому TypeScript-проекті.

Приклад:

declare const libraryName;

25). Як автоматично отримати файли declaration?

Відповідь: Встановити компілятор на true у файлі tsconfig.json

Приклад:

{

"compilerOptions": {

...

"declaration": true,

}

}

26). Як перевантажити функцію?

Відповідь: Потрібно використовувати те саме ім'я функції над оригінальною функцією без дужок {} і змінити число та типи аргументів та/або тип значення, що повертається.

Приклад:

function add(x:string, y:string): string;

function add(x: number, y: number): number {

return x + y;

}

27). Як зробити всі характеристики інтерфейсу необов'язковими?

Відповідь: Використовуйте тип Partial

Приклад:

interface Person {

name: string;

age: number;

}

type PartialPerson = Partial<Person>; // Аналогічно до наступного коду

linesinterface PartialPerson {

name?: string;

age?: number;

}

28). До чого можна використовувати декоратори?

Відповідь: Класи, властивості, методи та аргументи методу.

Приклад:

@MyClassDecorator

export class Person {

…

@MyPropertyDecorator myProperty: string;

}

29). Навіщо потрібен тип «Record»?

Відповідь: Він дозволяє створювати типізовану карту

Приклад:

let Person = Record<string, number> = {};

Person.age = 25;

30) Як можна отримати доступ до класів поза модулем, у якому вони визначені?

Відповідь: Використовуйте ключове слово export перед назвою класу.

Приклад:

export class Person {}