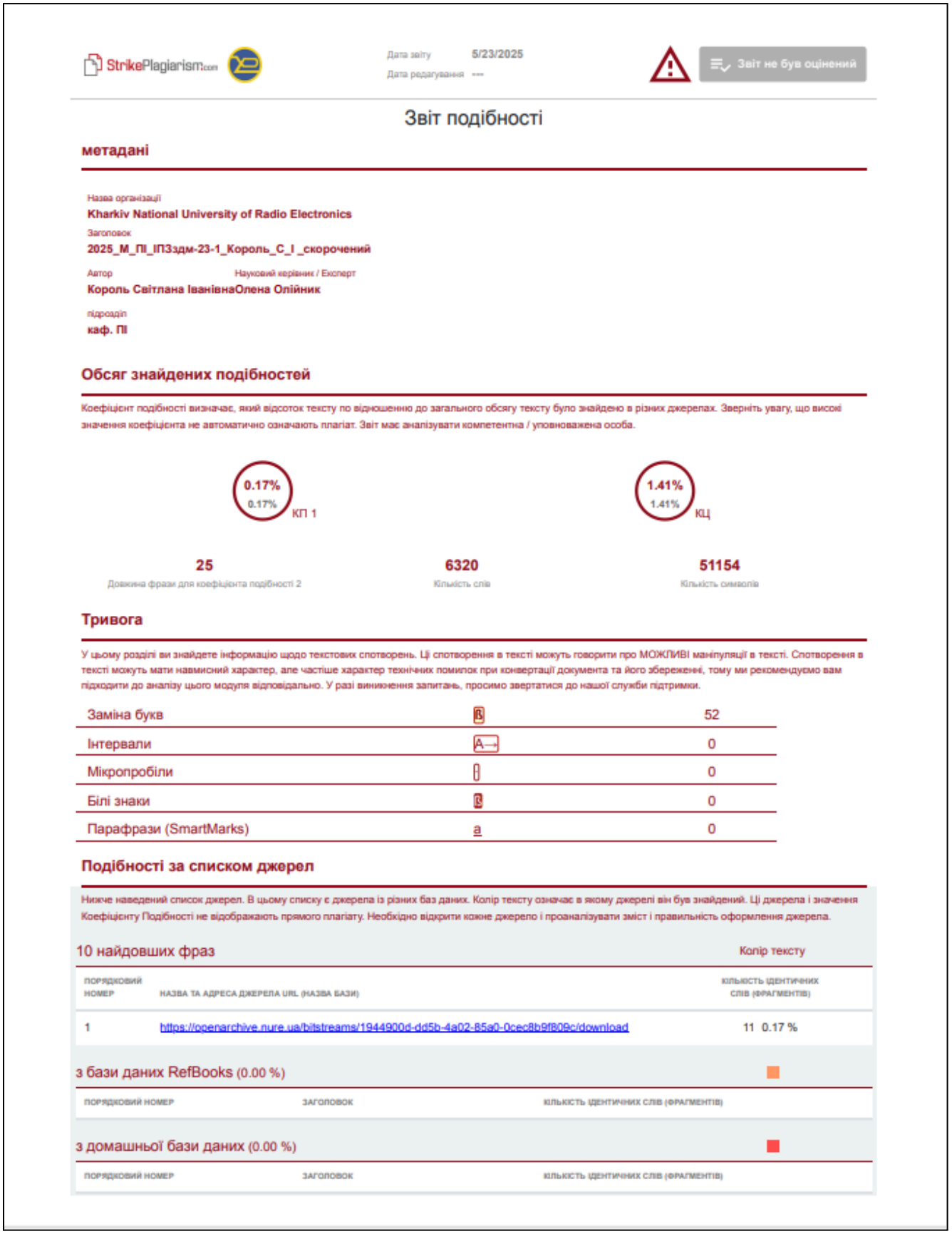
ДОДАТОК А

ЗВІТ РЕЗУЛЬТАТІВ ПЕРЕВІРКИ НА УНІКАЛЬНІСТЬ ТЕКСТУ В БАЗІ ХНУРЕ



StrikePlagiarism.com

Дата звіту 5/23/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_M_ПІ_ІПЗздм-23-1_Король_С_І _скорочений
Автор
Король Світлана Іванівна
Науковий керівник / Експерт
Олена Олійник
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.17% 0.17% КП 1

1.41% 1.41% КЦ

25
Довжина фрази для коефіцієнта подібності 2

6320
Кількість слів

51154
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		52
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		0

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/1944900d-dd5b-4a02-85a0-0cec8b9f809c/download	11 0.17 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (0.17 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/1944900d-dd5b-4a02-85a0-0cec8b9f809c/download	11 (1) 0.17 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

10

ВСТУП

Сучасні тенденції у веб-розробці демонструють значне зростання популярності мови програмування TypeScript, яка виступає потужною альтернативою традиційному JavaScript. Цей тренд особливо виражений у контексті розробки односторінкових додатків (SPA), де ефективність та масштабованість є ключовими вимогами до програмного забезпечення. SPA-застосунки, що базуються на платформі React.js, користуються особливою популярністю завдяки своєму зручному підходу до створення динамічних інтерфейсів користувача. Водночас, такі додатки часто стикаються з проблемами підтримки великого обсягу коду, складності в управлінні станом та необхідністю забезпечення високої продуктивності.

Однією з основних проблем при розробці SPA є забезпечення надійності та безпеки коду, що зростає разом з його масштабом. У цьому контексті TypeScript, завдяки своїй статичній типізації, стає важливим інструментом для зменшення кількості помилок на етапі розробки, поліпшення читабельності та підтримуваності коду. TypeScript дозволяє розробникам визначати типи змінних, функцій та об'єктів, що значно знижує ймовірність помилок, пов'язаних з неправильною обробкою даних, і забезпечує кращу інтеграцію з різноманітними бібліотеками та фреймворками.

Особливо важливим є вплив TypeScript на продуктивність веб-застосунків, процес компіляції та статична перевірка типів дозволяють виявляти потенційні проблеми ще до етапу виконання програми, що, у свою чергу, покращує загальну надійність та ефективність розробки. Крім того, інструменти для автоматичного тестування, інтеграція з редакторами коду та підтримка сучасних веб-стандартів роблять процес розробки швидшим і зручнішим.

Враховуючи ці аспекти, мета даного дослідження полягає в аналізі ефективності використання TypeScript при розробці SPA-застосунків на базі React.js, а також у вивченні його переваг і недоліків порівняно з традиційним JavaScript.

11

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі

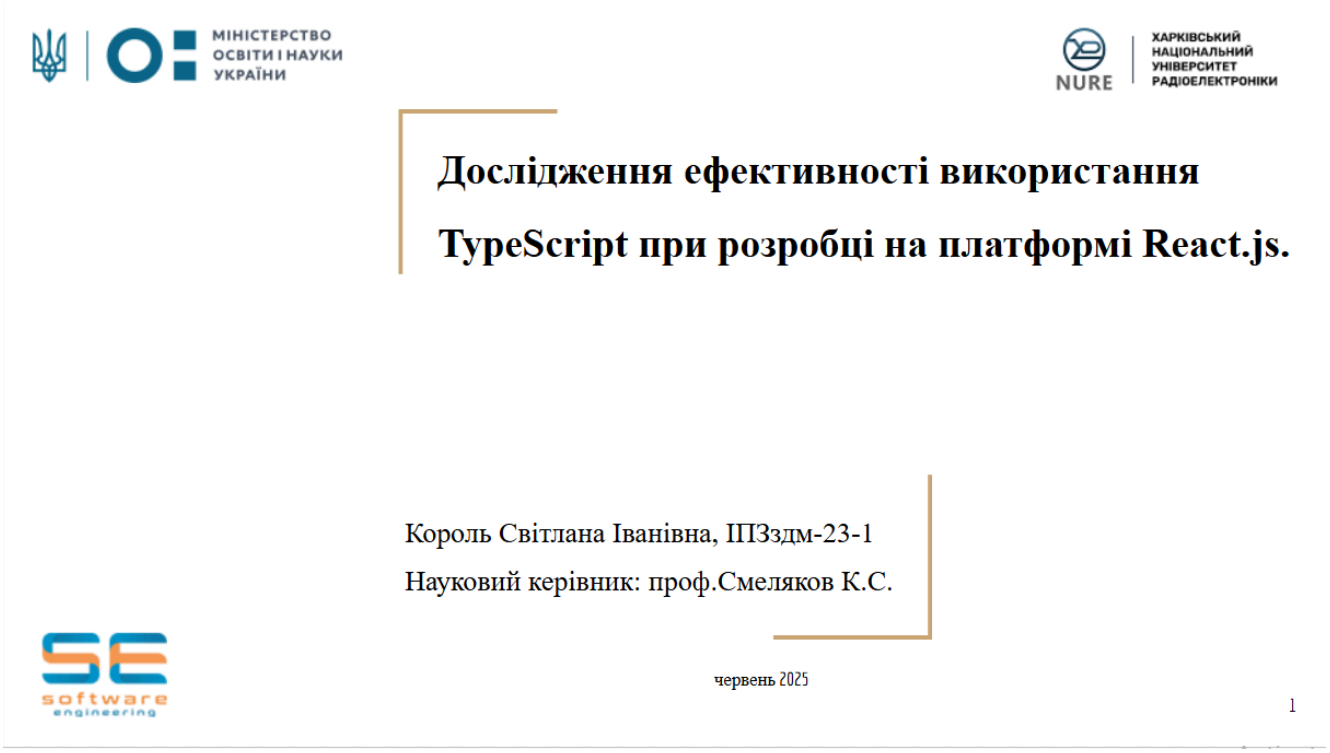
У сучасній веб-розробці спостерігається стрімке зростання складності front-end застосунків, особливо у контексті Single Page Applications (SPA). За даними State of JavaScript 2023, близько 78% розробників стикаються з проблемами підтримки великих JavaScript проектів, зокрема:

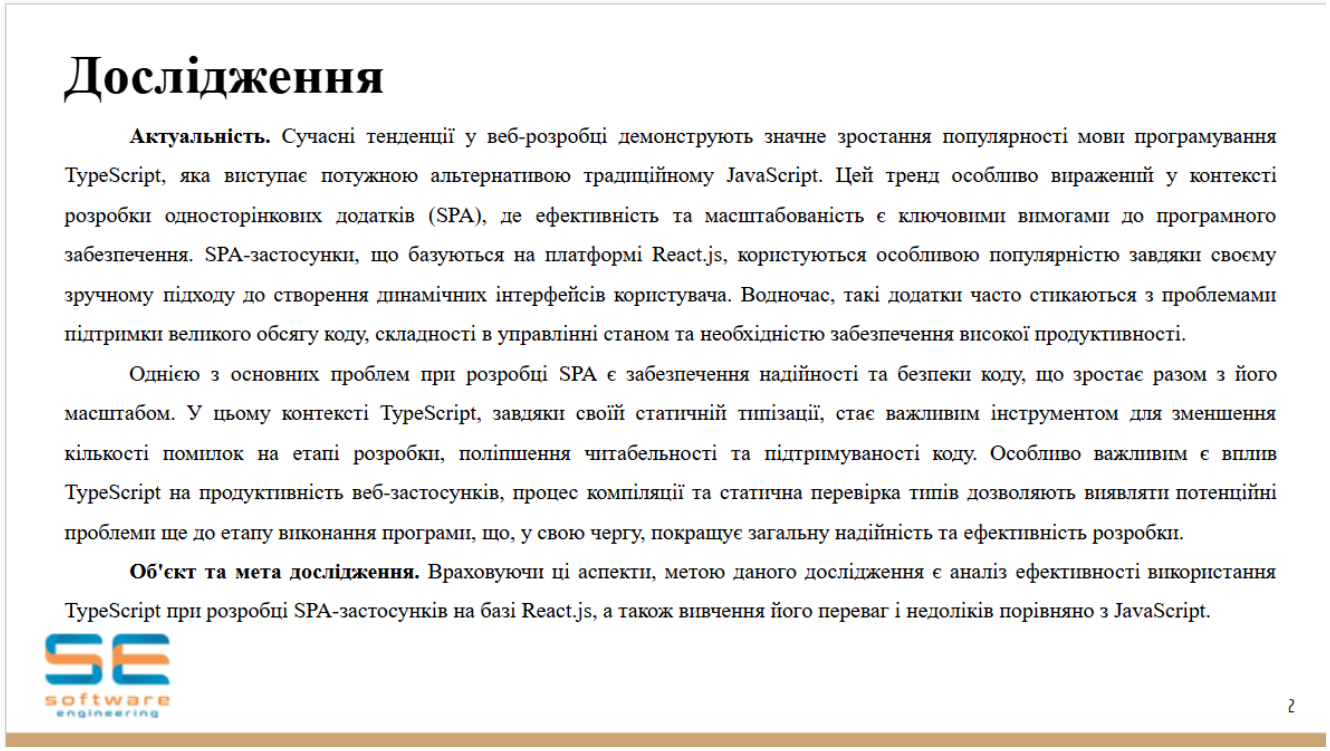
- складність відстеження типів даних у великих застосунках;
- помилки під час виконання через неправильні типи даних;
- ускладнена рефакторизація коду;
- проблеми з документацією та масштабуванням проектів.

TypeScript, як надмножина JavaScript, пропонує вирішення цих проблем через систему статичної типізації. За даними дослідження GitHub's State of the Octoverse 2023, TypeScript став четвертою найпопулярнішою мовою програмування, демонструючи зростання на 78% порівняно з попереднім роком. Використання TypeScript у розробці програмного забезпечення набуває все більшої популярності завдяки низці переваг, які надає ця мова програмування. Серед ключових переваг можна виділити:

ДОДАТОК Б

Слайди презентації





Огляд літератури (аналогів)

Аналіз наукових та професійних джерел показує, що типізація стала невід'ємною частиною розробки сучасних веб-додатків. У сучасній веб-розробці спостерігається стрімке зростання складності front-end застосунків, особливо у контексті SPA. За даними State of JavaScript 2023, близько 78% розробників стикаються з проблемами підтримки великих JavaScript проектів, зокрема:

- складність відстеження типів даних у великих застосунках;
- помилки під час виконання через неправильні типи даних;
- ускладнена рефакторизація коду;
- проблеми з документацією та масштабуванням проектів.

TypeScript, як надмножина JavaScript, пропонує вирішення цих проблем через систему статичної типізації. За даними дослідження GitHub's State of the Octoverse 2023, TypeScript став четвертою найпопулярнішою мовою програмування, демонструючи зростання на 78% порівняно з попереднім роком.

Документація від офіційних джерел TypeScript Documentation та React.js Documentation) надають детальну інформацію про інтеграцію цих технологій та рекомендовані практики використання.

Результати досліджень в наукових виданнях свідчать про доцільність та ефективність застосування TypeScript у розробці SPA на базі React.js. Це формує основу для проведення подальшого експериментального дослідження у межах даної роботи.

Постановка задачі

Постановка задачі передбачає дослідження та оцінку ефективності застосування TypeScript при розробці SPA-застосунків на платформі React.js у порівнянні з використанням традиційного JavaScript (PropTypes).

Реалізація наукового дослідження складається з наступних етапів:

- аналіз переваг та недоліки застосування TypeScript у порівнянні з JavaScript на основі огляду наукових джерел;
- розробка методики порівняльного аналізу ефективності використання TypeScript та JavaScript у розробці React.js додатків;
- визначення критеріїв оцінки ефективності, таких як точність, продуктивність, зручність розробки та підтримки;
- експериментальне дослідження на основі розробки тестового SPA-застосунку, використовуючи як TypeScript, так і JavaScript;
- аналіз результатів експериментів та надати рекомендації щодо доцільності використання TypeScript у розробці React.js додатків.

Отримані результати дослідження можуть бути використані розробниками програмного забезпечення при виборі технологічного стеку для створення SPA-застосунків.

Методологія

Для проведення експериментального дослідження було підготовлено репрезентативний набір даних, а саме розроблено два ідентичні за функціональністю SPA-застосунки – один з використанням JavaScript та PropTypes , другий з використанням TypeScript. Застосунки представляють собою середньостатистичний веб-застосунок “TODO-лист” з наступним функціоналом:

- завантаження списку задач з публічного API;
- відображення задач у вигляді таблиці;
- можливість відмітки задачі як виконаної;
- кнопка старту, що запускає завантаження;
- в якості джерела даних для застосунків було використано публічне API JSONPlaceholder (обрано через його структурованість, достатній обсяг і відповідність реальним прикладам взаємодії з REST API).

Методологічні принципи, що використані в дослідженні:

- контрольовані умови, обидва підходи мають однакові умови для тестування (проект, одна структура, одна кількість компонентів);
- один тип даних, для коректного порівняння реалізовано однакова структура компонентів;
- статистика для серії вимірювань, щоб зібрати статистику та порівняти середні значення;
- візуалізація даних для висновків.

Розширене порівняння досліджуваних способів типізації

Характеристика	Опис PropType	Опис TypeScript
Перевірка складних об'єктів	PropTypes.shape()для структури об'єктів.	interface та type, гнучке визначення складних типів.
Масиви та колекції	PropTypes.arrayOf()та PropTypes.oneOf()для типів елементів.	Типи масивів та колекцій через Array<T>, кортежі та Union Types.
Інтерфейси	Немає аналогів.	Можливість опису типів за допомогою interface type для забезпечення чіткої структури.
Типи для асинхронного коду	Не підтримується типізація асинхронного коду без зовнішніх бібліотек.	Підтримка типів для async/await, обіцянок (Promise).
Статична та динамічна типізація	Динамічна типізація (перевірка на етапі виконання).	Статична типізація (перевірка на етапі компіляції).
Підтримка IDE (автодоповнення)	Обмежене автодоповнення та підсвічування помилок	Повне автодоповнення, підсвічування помилок і рефакторинг коду.
Підтримка документації	Мінімальна підтримка документації.	Чітка документації через JSDoc та підтримка типів для генерації документації.
Масштабованість у великих проектах	Складно масштабувати для великих проектів.	Ідеально підходить для великих і складних проектів завдяки чіткій і статичній типізації.
Підтримка рефакторингу	Обмежена підтримка рефакторинга.	Потужна підтримка рефакторинга завдяки інтеграції типів в IDE.
Типи для внутрішніх компонентів	Можна використовувати PropTypes для внутрішніх компонентів.	Підтримка внутрішніх типів через interface, що дозволяє визначити детальну структуру компонента.
Підтримка інтернаціоналізації	Більше базової перевірки типів, без спеціальної підтримки для i18n.	Можна легко інтегрувати з бібліотеками через підтримку типів для кожної мови

Опис програмного забезпечення, що було використано у дослідженні

Для проведення експериментального дослідження ефективності застосування TypeScript при розробці на платформі React.js було обрано відповідні технології, інструменти та середовище розробки, які дозволять найбільш об'єктивно оцінити переваги та недоліки досліджуваних підходів.

В якості основного середовища розробки було обрано:

- Visual Studio Code (версія 1.84.0), редактор коду, що має широку підтримку як JavaScript, так і TypeScript.

Для розробки тестових скриптів було обрано наступний технологічний стек:

- платформа розробки React.js 18.2.0, як основна бібліотека для розробки інтерфейсу користувача;
- системи збірки та управління пакетами npm 10.2.3 для управління залежностями;
- мови програмування JavaScript (ES2022) для створення контрольного застосунку;
- мова програмування TypeScript 5.3.3 для створення експериментального застосунку;
- інструменти для тестування та вимірювання продуктивності- Lighthouse 11.4.0 - для комплексного аналізу продуктивності веб-застосунку;
- інструмент TypeScript Compiler (tsc) з використанням команди time tsc для вимірювання часу компіляції;

7

Зміст проведеного експерименту

Процедура вимірювання продуктивності з використанням інструменту Lighthouse за наступною процедурою:

- збірка проекту в продакшн режимі: npm run build;
- запуск локального сервера для роздачі статичних файлів: npx serve -s dist;
- запуск Lighthouse з наступними параметрами:

```
//bash
lighthouse http://localhost:3000 --output=json --output-path=./results.json    --chrome-flags="--headless"
--preset=desktop;
```

- повторення вимірювань 5 разів для отримання статистично значущих результатів;
- обчислення середніх значень для кожної метрики

Для оцінки впливу використання TypeScript і PropTypes на тривалість компіляції застосунку було проведено контрольовані вимірювання часу побудови (build time) кожної версії проєкту з однаковими умовами середовища та конфігурації і містила наступні кроки:

- очищення кешу компілятора: rm -rf node_modules/.cache;
- запуск компіляції з вимірюванням часу для TypeScript проекту: time npx tsc, tsc--extendedDiagnostics (що є використанням CLI інтерфейсу командного рядка);
- запуск збірки з вимірюванням часу для JavaScript : time npm run build, Measure-Command { npm run build }(за допомогою Webpack)
- повторення вимірювань 5 разів;
- обчислення середнього значення часу компіляції.

8

Проведення експерименту

Рисунок 4.2 – Тестування продуктивності TypeScript в Lighthouse

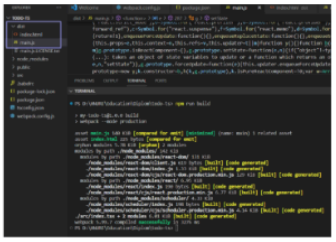


Рисунок 4.4 – Розмір бандлу тестового скрипту на TypeScript

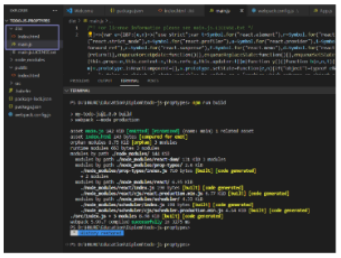


Рисунок 4.5 – Розмір бандлу тестового скрипту на PropTypes

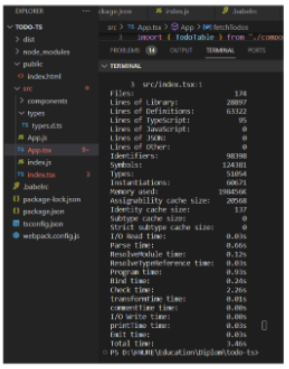


Рисунок 4.6 – Час виконання тестового скрипту, виконаного на TypeScript

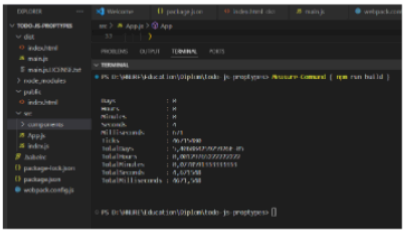


Рисунок 4.7 – Експериментальне дослідження часу збірки стріпту з використанням PropTypes за допомогою Webpack

9

Результати вимірювання продуктивності за допомогою Lighthouse

Метрика	PropTypes	TypeScript	Різниця в %
Performance Score, (бал, 0-100)	99.6	100	+0.4%
First Contentful Paint (FCP), (мс)	210	200	-4.8%
Largest Contentful Paint (LCP), (мс)	610	570	-6.6%
Speed Index, (мс)	450	420	-6.7%
Total Blocking Time (TBT), (мс)	40	10	-75%
Cumulative Layout Shift (CLS)	0	0	0%
Time to Interactive (TTI), (мс)	210	200	-4.8%
Розмір бандлу, (кБ)	142	140	-1.4%

10

Результати вимірювання часу компіляції

Вимірювання	PropTypes	TypeScript	Різниця в %
1	4.396	3.46	27%
2	4.672	3.43	36.94%
3	4.374	3.67	19.23%
4	4.350	3.45	26.1%
5	4.349	3.59	21.11%
Середнє	4.428	3.52	25.94%

Результати вимірювання часу розробки та кількості помилок

Метрика	PropTypes	TypeScript	Різниця в %
Загальний час розробки, (год)	5	5.5	+10%
Помилки на етапі розробки	14	9	-35.71%
Помилки на етапі компіляції	0	2	+100%
Помилки на етапі виконання	4	1	-75%
Загальна кількість помилок	18	16	-11%
Час на виправлення помилок, (год)	1.5	1	-33.3

Аналіз отриманих результатів

Отже на основі проведеного дослідження можна комплексно оцінити результатів трьох експериментальних блоків та зробити наступні висновки:

- TypeScript забезпечує вищу продуктивність застосунку з покращеннями за ключовими метриками Lighthouse, зокрема значним зменшенням Total Blocking Time на 75%;
- час компіляції з TypeScript на 25.94% менший, що суттєво покращує процес розробки, особливо для великих проектів з частими змінами коду;
- TypeScript знижує загальну кількість помилок на 11% та зменшує час на їх виправлення на 33.33%, хоча початковий час розробки збільшується на 10%;
- найбільша перевага TypeScript — зменшення помилок на етапі виконання на 75%, що критично важливо для забезпечення стабільності роботи застосунку для кінцевих користувачів.

Висновки

На основі дослідження теоретичних основ TypeScript та PropTypes було досліджено що:

- TypeScript є статично типізованою надбудовою над JavaScript, яка надає можливості раннього виявлення помилок, покращеного інтелектуального доповнення коду та безпечного рефакторингу.
- PropTypes є бібліотекою для JavaScript, яка дозволяє перевіряти типи у ран-таймі.

Кожен з підходів має свої переваги, але суттєво відрізняються за механізмом роботи та сферою застосування.

На основі проведеного експериментальне дослідження підтверджено:

- гіпотезу про те, що використання TypeScript має значні переваги при розробці React.js-застосунків порівняно з JavaScript та PropTypes, особливо для середніх та великих проектів з тривалим життєвим циклом.

Ці переваги проявляються як у покращеній продуктивності та надійності кінцевого продукту, так і в оптимізації процесу розробки. Незважаючи на початкові інвестиції часу у вивчення TypeScript та налаштування проекту, довгострокові вигоди від його використання перевищують недоліки, що робить TypeScript рекомендованим вибором для професійної розробки React.js-застосунків.

Ці висновки мають практичне значення для архітекторів програмного забезпечення, технічних лідерів та менеджерів проектів при прийнятті рішень щодо технологічного стеку для нових проектів або модернізації існуючих.

ДОДАТОК В

Апробація у вигляді тез у конференції "Сучасні Інформаційні Технології та Системи Штучного Інтелекту MIT@AIS-2025

Дослідження Ефективності Застосування Typescript при Розробці на Платформі React.js

Світлана Король[a], Анастасія Чуприна[b]

Харківський національний університет радіоелектроніки, пр. Науки 14, м. Харків, 61166, Україна
Харківський національний університет радіоелектроніки, пр. Науки 14, м. Харків, 61166, Україна

Анотація

У статті розглянуто переваги та недоліки використання мови програмування TypeScript при розробці односторінкових додатків (SPA) на платформі React.js. Проведено аналіз наукових джерел та досліджень з даної тематики, визначено ключові критерії оцінки ефективності застосування TypeScript у порівнянні з традиційним JavaScript. Результати дослідження демонструють значні переваги TypeScript щодо забезпечення типобезпеки, підтримуваності коду та продуктивності розробки.

Ключові слова 1

TypeScript, React.js, статична типізація, WEB, SPA, JavaScript, PropTypes, масштабованість коду, продуктивність розробки, рефакторинг, якість програмного забезпечення

Сучасна розробка веб-додатків характеризується стрімким зростанням складності, що зумовлює необхідність пошуку ефективних інструментів для підтримки якості коду та зменшення кількості помилок. TypeScript, як надбудова над JavaScript із системою статичної типізації, пропонує вирішення багатьох проблем, з якими стикаються розробники при створенні масштабних SPA-застосунків. Особливої актуальності це набуває в контексті розробки на React.js, де ефективність роботи з компонентами, станом та життєвими циклами компонентів є критично важливою.

Аналіз наукових та професійних джерел демонструє, що типізація стала невід'ємною частиною розробки сучасних веб-додатків [1] [8]. У праці "Learning TypeScript: Enhance Your Web Development Skills" (2022) Черни розглядає еволюцію типізації у JavaScript-екосистемі та визначає TypeScript як найпоширеніше рішення для статичної типізації [3]. Автор демонструє, як TypeScript допомагає запобігати помилкам на етапі компіляції, що значно підвищує надійність коду.

Файн та Мойсеєнко у книзі "TypeScript Quickly" (2020) досліджують аспекти інтеграції TypeScript із сучасними фреймворками, включаючи React.js [5]. Вони підкреслюють, що TypeScript забезпечує кращу документацію коду, покращує процес рефакторингу та сприяє створенню більш стійких до помилок додатків.

Spinks у дослідженні "React and TypeScript: Develop Maintainable and Scalable Web Applications" (2021) зосереджується на взаємодії TypeScript із React.js, аналізуючи переваги використання статичної типізації для компонентного підходу [6]. Автор демонструє, як типізація покращує розуміння структури компонентів, їх пропсів та стану.

Основні переваги використання TypeScript у порівнянні з PropTypes включають:

- Статична типізація на рівні всього проекту - TypeScript забезпечує типізацію не лише для пропсів компонентів, але й для всіх частин коду, включаючи хуки, сервіси та утиліти [1][3].
- Перевірка типів на етапі компіляції - на відміну від PropTypes, який виконує перевірку типів лише під час виконання (runtime), TypeScript виявляє помилки типів на етапі компіляції [5].

MIT@AIS'2025: 1st International Scientific and Practical Conference "Modern Information Technologies and Artificial Intelligence Systems", May 19-22, 2025, Kharkiv-Yaremche, Ukraine
EMAIL: svitlana.korol.cpe@nure.ua (A. 1); anastasiya.chuprina@nure.ua (A. 2)
ORCID: 0009-0005-7096-5072 (A. 1); 0000-0003-0364-9900 (A. 2)

- Автодоповнення та підказки в IDE - забезпечує кращу інтеграцію з IDE, включаючи автодоповнення, підказки типів та навігацію по коду, що підвищує продуктивність розробників [6].
- Типізація складних структур даних - дозволяє визначати та перевіряти складні типи даних, включаючи дженерики, інтерфейси, типи об'єднання, перетину [7].
- Кращий рефакторинг коду - завдяки статичній типізації, зміни в одній частині коду автоматично відображаються у всіх пов'язаних місцях [5].
- Розширена документація коду - TypeScript діє як вбудована документація, надаючи чітке розуміння очікуваних типів даних [6].

Однак слід враховувати і недоліки TypeScript:

- Складніша крива навчання - особливо для розробників, які не знайомі зі статично типізованими мовами [3].
- Додатковий етап компіляції - що може збільшити час збірки проекту [5].
- Конфігурація та налаштування - вимагає додаткової конфігурації (tsconfig.json) [7].
- Обмеження при роботі з динамічними даними - іноді TypeScript може бути занадто суворим при роботі з непередбачуваними структурами даних [4].
- Додаткові залежності - необхідність встановлення пакетів з типами (@types/*) [6].

Вандеркам у "Effective TypeScript" наводить результати аналізу 62 проектів, де застосування TypeScript дозволило зменшити кількість помилок під час розробки на 15-40% [7]. Спінкс демонструє, що час на виявлення та виправлення помилок скоротився на 33%, загальний час розробки для нових функцій зменшився на 15%, а швидкість рефакторингу зросла на 47% [6].

Файн та Мойсеєнко виявили, що для проектів з понад 100,000 рядків коду TypeScript забезпечує зниження вартості підтримки коду на 26% та зменшення кількості регресійних помилок на 38% [5]. Черни показав, що після початкового періоду навчання продуктивність команди зростала на 12-20% [3].

Frill аналізував вплив TypeScript на продуктивність React 18 додатків і встановив, що TypeScript-проекти демонструють на 7% вищу продуктивність рендерингу компонентів, на 12% менші розміри бандлів після оптимізації та на 23% менше помилок при використанні нових API React 18 [4].

Таблиця 1

Порівняння показників ефективності TypeScript та JavaScript

Показник	Typescript	Javascript	Різниця
Зменшення кількості помилок під час розробки	Висока	Середня	15-40%
Час на виявлення та виправлення помилок	Низький	Високий	-33%
Швидкість рефакторингу	Висока	Середня	+47%
Вартість підтримки коду	Низька	Висока	-26%
Кількість регресійних помилок	Низька	Висока	-38%
Продуктивність команди (після навчання)	Висока	Середня	+12-20%

На основі теоретичної оцінки цих параметрів в роботі розроблено методологію порівняльного аналізу, яка включає експериментальну розробку тестового SPA-застосунку з використанням двох підходів: чистого JavaScript з PropTypes для валідації властивостей компонентів та TypeScript з його системою статичної типізації. Кожен підхід оцінювався за кількома критеріями

В результаті проведеного експерименту отримано результати, які можуть бути корисними для розробників програмного забезпечення при виборі технологічного стеку для створення

SPA-застосунків [6] [8]. Аналіз показав, що TypeScript дозволяє знижувати кількість помилок завдяки статичній типізації, що полегшує підтримку та масштабування коду, в той час як використання чистого JavaScript з PropTypes є більш гнучким, але менш ефективним у великих проектах з точки зору надійності та підтримки.

Висновки

Результати аналізу наукових та професійних джерел свідчать про значні переваги використання TypeScript у розробці React.js додатків, особливо для середніх та великих проектів. Статична типізація, покращена інтеграція з IDE, зменшення кількості помилок на етапі компіляції та кращі можливості для рефакторингу коду роблять TypeScript привабливим вибором для розробників, які прагнуть підвищити якість та підтримуваність своїх додатків.

Водночас, для малих проектів з коротким життєвим циклом або для прототипування, додаткові витрати на впровадження TypeScript можуть не завжди бути виправданими. Ефективність використання TypeScript також значною мірою залежить від правильного налаштування, дотримання рекомендованих практик та рівня знань команди.

Література

[1] TypeScript Documentation. URL: https://www.typescriptlang.org/docs/
[2] React.js Documentation. URL: https://reactjs.org/docs/
[3] Черни Б. Learning TypeScript: Enhance Your Web Development Skills. O'Reilly Media, 2022. 324 с.
[4] Frill A. Pro React 18. Apress, 2023. 456 с.
[5] Фэйн Я., Мойсеенко А. TypeScript Quickly. Manning Publications, 2020. 350 с.
[6] Spinks S. React and TypeScript: Develop Maintainable and Scalable Web Applications. Packt Publishing, 2021. 280 с.
[7] Vanderkam D. Effective TypeScript: 62 Specific Ways to Improve Your TypeScript. O'Reilly Media, 2020. 264 с.
[8] Smelyakov K., Karachevtsev D, Kulemza D, Samoilenko Y, Patlan O, Chupryna A : 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020 - Proceedings, Pages 187 - 191, Effectiveness of Preprocessing Algorithms for Natural Language Processing Applications

Науковий керівник: Чуприна Анастасія Сергіївна, доцент, кафедра Програмної інженерії Харківського національного університету радіоелектроніки

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

1

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-1 (група)

Король Світлана Іванівна
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні позначки	
	7.15 Додатки	

Зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

26.05.2025