МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

**ТОКАР МАКСИМ ОЛЕКСАНДРОВИЧ**

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
доктор технічних наук, доцент
_____Т. В. Нескородєва
«____» _____ 20__ р.

# РОЗРОБКА ФРОНТЕНД ЧАСТИНИ СИСТЕМИ АВТОМАТИЧНОЇ ПЕРЕВІРКИ КОДУ НА C#

Спеціальність 122 «Комп’ютерні науки»
**Кваліфікаційна (бакалаврська) робота**

Керівник:
Антонов Ю. С., доцент кафедри
інформаційних технологій,
к.ф.-м.н., доцент

Оцінка: ___/___/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

**Токар М.О. Розробка фронтенд частини системи автоматичної перевірки коду на C#.** Спеціальність 122 «Комп’ютерні науки», освітня програма «Сучасні інформаційні технології та програмування». Донецький національний університет імені Василя Стуса, Вінниця, 2022.

Кваліфікаційна (бакалаврська) робота присвячена розробці веб-сервісу, за допомогою якого можна здійснювати компіляцію програмного коду, а також перевірки його на валідність.

Функціонал веб-додатку реалізований за допомогою мови програмування Javascript, а саме бібліотеки React

Ключові слова: веб-додаток, single page application, фронтенд, Javascript, React, компілятор, сервер.

50 сторінок, 23 рис., 40 джерел.

**Tokar M.O. Development of the frontend part of the automatic code verification system in C#.** Specialty 122 "Computer Science", educational program "Modern Information Technology and Programming". Donetsk national Vasyl Stus University, Vinnytsia, 2022.

Qualifying (bachelor's) work is devoted to the development of a web service that can be used to compile program code, as well as validate it.

The functionality of the web application is implemented using the Javascript programming language, namely the React library

Keywords: web application, single page application, frontend, Javascript, React, compiler, server.

50 pages, 23 figure, 40 sources.

**ЗМІСТ**

# ВСТУП

**Актуальність роботи:** В наш час в багатьох спеціалістів різного роду на першому місці стоїть економія часу на виконання рутинних задач, та автоматизація їх. Тому багато рішень в сфері IT спрямовані на те, щоб задовільнити потреби економії часу та енергії на виконання задач, які часто повторюються.

Будь-який програміст компілює свій код. Процес компіляції та перевірки коду відбувається регулярно, при написанні програмного коду. Звісно, дуже важливим є середовище, де відбувається ця компіляція.

Звичайно, якщо задача перевірки свого коду поставлена досвідченому спеціалісту, то компілюватись код буде в професійних IDE. Але чи доцільно використовувати таке важке програмне забезпечення суб'єкту, якому потрібно швидко перевірити невеликий обсяг програмного коду? Або ж викладачу, якому потрібно перевірити файл з написаним кодом на його коректність?

В таких випадках на заміну важким IDE приходять онлайн компілятори, за допомогою яких можна уникнути необхідність встановлення на ПК додаткового програмного забезпечення.

Цей дипломний проєкт присвячений розробці фронтенд частини системи автоматичної перевірки коду на C#.

**Мета дослідження.** Розробити веб-сервіс для зручної та швидкої перевірки коду на C#

**Завдання дослідження:**

- Проаналізувати предметну область;

- Проаналізувати існуючи рішення для автоматичної перевірки коду

- На основі отриманих результатів розробити веб-застосунок для компіляції

**Предмет дослідження дипломної роботи**. Web-технології та їх застосування для розробки сервісу перевірки коду

**Зв'язок роботи з науковими програмами, планами, темами.** Наведені у бакалаврській роботі дослідження пов'язані з фундаментальною науково-дослідною роботою «Дослідження та комп'ютерно-математичне моделювання складних систем та процесів у науці, освіті та інформаційно-комунікаційній діяльності підприємств» (№ держреєстрації 0116U002394, 2018-2022 рр.).

**Структура роботи.** Дипломна робота складається зі вступу, трьох розділів та висновку до них, списку використаних джерел та додатків.

У першому розділі курсової роботи проведено огляд предметної області та існуючих програмних рішень для перевірки програмного коду.

У другому розділі наведено опис технологій, які використовувалися при розробці даного програмного забезпечення, та модель для її розробки.

У третьому розділі наводиться опис реалізації самого веб-застосунку.

Дипломна робота включає в себе 50 сторінок, 23 рисунка і список літератури з 40 джерел.

# РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Історія розвитку компіляторів

Створення перших програм компіляції слід зарахувати до початку п’ятдесятих років минулого століття. Головним завданням реалізації перших компіляторів у той час було вирішення проблеми перетворення алгебраїчних виразів у машинні коди. А фактичним роком народження теорії компіляції може вважатися 1957 рік, коли була реалізована перша програма компіляції мови Фортран. Вона видавала досить ефективні об'єктні (машинні) коди. Цей компілятор був призначений для платформ IBM 704, IBM 360 та DEC PDP-11. У вісімдесятому році минулого століття вийшов новий варіант компілятора, призначений для IBM 360 та IBM PC, який підтримував стандарт FORTRAN 77. Роком пізніше утворилася компанія Watcom, що представила у 1988 році програму компіляції Сі. Вона миттєво стала надзвичайно популярною серед фахівців із програмування, оскільки могла генерувати найшвидші коди порівняно з іншими компіляторами тих часів.

Майже всі компілятори виконують переклад програми з певної мови високого рівня програмування на машинні коди, які можуть бути виконані центральним процесором. Зазвичай дані коди призначені для виконання в конкретній операційній системі, оскільки застосовують надані системою можливості, такі як системні виклики і бібліотеки функцій.

Окремі програми компіляції, наприклад, Java, здатні переводити програму не в машинний код, а в програму на певній спеціалізованій мові низького рівня. Ця мова, або по-іншому байт-код, теж може вважатися мовою машинних кодів, оскільки вона повинна інтерпретуватися віртуальною машиною. Наприклад, для мови Java мовою віртуальної машини є JVM, або байт-код Java.

Для всіх цільових машин, наприклад, IBM, Apple і так далі, і всіх операційних систем або їх сімейств, які використовуються на цільовій машині,

необхідно написати програму компіляції. Є ще так звані програми крос-компіляції, які дозволяють на одній машині та одній операційній системі формувати коди, призначені для виконання на іншій цільовій машині чи іншій операційній системі.

Крім того, програми компіляції можна оптимізувати для різних типів процесорів з єдиного сімейства. Наприклад, коди, сформовані компілятором для процесорів сімейства i686, можуть застосовувати специфічні даних процесорів інструкції MMX, SSE, SSE2. Є також програми, що вирішують обернену задачу, а саме, вони перекладають програми з мови низького рівня на мову високого рівня. Така процедура називається декомпіляцією, а виконують її програми називають декомпіляторами.

Компілятор структурно складається із набору логічних компонентів. Лексичний аналіз здійснюється спеціальним аналізатором, який розпізнає лексеми мови та виконує їх заміну необхідними кодами. Лексемою є елементарна одиниця, яка входить до структури речень мови. Це може бути ключове слово, константа, ім'я тощо.

Синтаксичний аналізатор потрібен для з'ясування, чи задовольняють пропозиції, що входять до складу вихідної програми, граматичним правилам даної мови. Процес синтаксичного аналізу можна уявити, як формування дерева граматичного аналізу для перетворюваних речень.

Семантичний аналіз виконує контроль типів та видів кожного ідентифікатора та всіх операндів.

Процес оптимізації передбачає виконання перетворення вихідної програми на деякий проміжний формат запису. Оптимізацією проміжного коду є виділення загальних виразів та визначення підвиразів, які є константами. Оптимізаційна фаза служить для видалення надлишкових елементів програми, які викликають затримки за часом виконання і збільшують обсяг пам'яті. Ця фаза роботи з

програмою може бути виключена із загального циклу програмної обробки, якщо використовуються інші критерії проектування транслятора.

## 1.2 Мета онлайн компіляторів

Онлайн-компілятори - інструменти для перетворення коду програміста в машинний, зрозумілий для комп'ютера. В цьому випадку йдеться, зокрема, й про написання й виконання програми. Зазвичай для цього використовують IDE - інтегроване середовище розробки, спеціальні застосунки. Але IDE дуже комплексні й потребують багато пам'яті. Утім, для деяких завдань є можливість зберегти час і ресурси й упоратися за допомогою браузера. Саме для цього і створюється велика кількість подібних онлайн сервісів

## 1.3 Огляд подібних сервісів

На просторах інтернету можна знайти дуже велику кількість сервісів, де можна перевірити на правильність та скомпілювати програмний код. Всі вони мають якісь свої характерні особливості, та додаткові можливості. Загалом, подібного роду рішень – багато. Але, всі вони мають суттєві недоліки. По-перше, наявні в інтернеті онлайн компілятори написані за допомогою вже застарілих технологій. А саме тому швидкодійність таких процесів, як перевірка коду виявляється не такою вже й швидкою. По-друге, наявні сервіси з автоматичної перевірки коду мають здебільшого дизайн, не дуже дружній до клієнта.

### 1.3.1 Веб-сервіс ideone

Онлайн-компілятор IdeOne, з широким вибором мов програмування, набором інструментів, що дозволяє безпосередньо в браузері редагувати та компілювати код, написаний будь-якою з шістдесяти доступних мов

програмування. Часто розробники називають такі середовища – online compiler, використовуючи англійську назву.

IdeOne, програма, що працює в режимі online C, має переваги в можливості налаштування:

- ліміт часу;
- стандарт потоків уведення;
- додавання позначок до написаного коду, описів та коментарів;
- можливість демонстрації прикладу друзям, іншим членам команди-розробників.

Одне те, що це онлайн компілятор C, з можливістю групової роботи над одним проектом, робить його по-справжньому затребуваним. Є можливість зібрати в групу друзів чи колег і працювати навіть перебуваючи далеко один від одного. Всі члени групи можуть робити файли форк.

Форк – відгалуження частини коду, з можливістю продовжити роботу над проєктом, використовуючи сторонню мову програмування. При цьому вихідний код зберігається.

Полегшує роботу розробників корисна фішка - можливість вставити частину тексту за допомогою спеціалізованого віджету.

Характерна риса — на відміну від багатьох інших компіляторів цей підтримує не лише найпопулярніші мови, а й Асемблер, Фортран тощо. Тут є три рівні доступу до коду — публічний (він відображатиметься на окремій сторінці), секретний (доступний лише за посиланням), приватний (зможете переглядати лише ви за умови реєстрації). IDEONE має ще деякі обмеження для незареєстрованих користувачів — час виконання програми. За наявності акаунту він становить 15 секунд, без — 5.

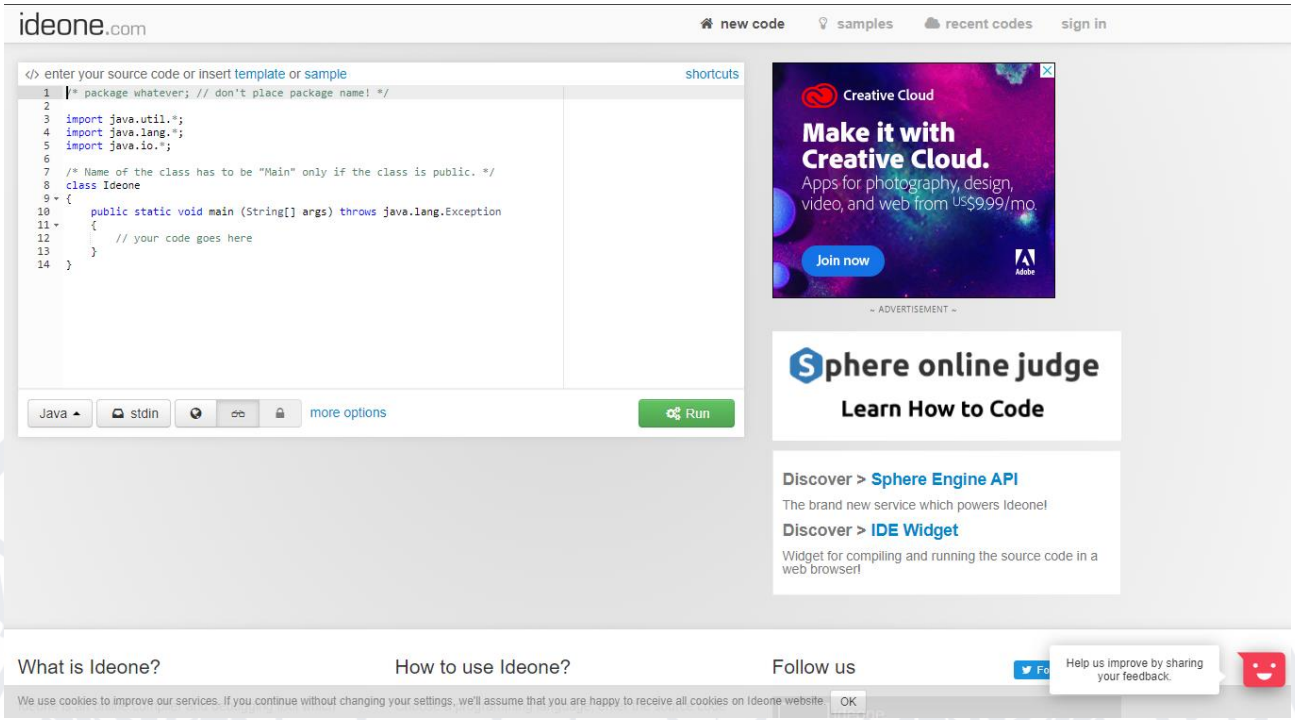


*Рис. 1.1 Веб-сервіс ideone*

### 1.3.2 Веб-сервіс CodePad

Мінімалістичний онлайн-сервіс для зберігання готового коду чи різних поетапних версій CodePad. Тут можна зберігати код, запускати його для виконання, вивести попередні результати. У нього є одна дуже потрібна перевага - аскетизм дає можливість стабільно працювати в умовах повільного інтернет-з'єднання.

Має великий мінус - відсутнє підсвічування синтаксису, при введенні шматків тексту у форму. Але код підсвічується після збереження і при перегляді.

На перший погляд може здатись, що є замало функцій, які підтримує цей редактор. Однак такий лаконічність та простота дозволяє швидко і легко завантажити редактор навіть за "повільного" інтернету. Взагалі для будь-якого розробника важлива зручність при роботі з файлами та шматками коду. Тому розробники цінують  онлайн компілятори, адже це скорочує час написання програм та іншого програмного забезпечення.

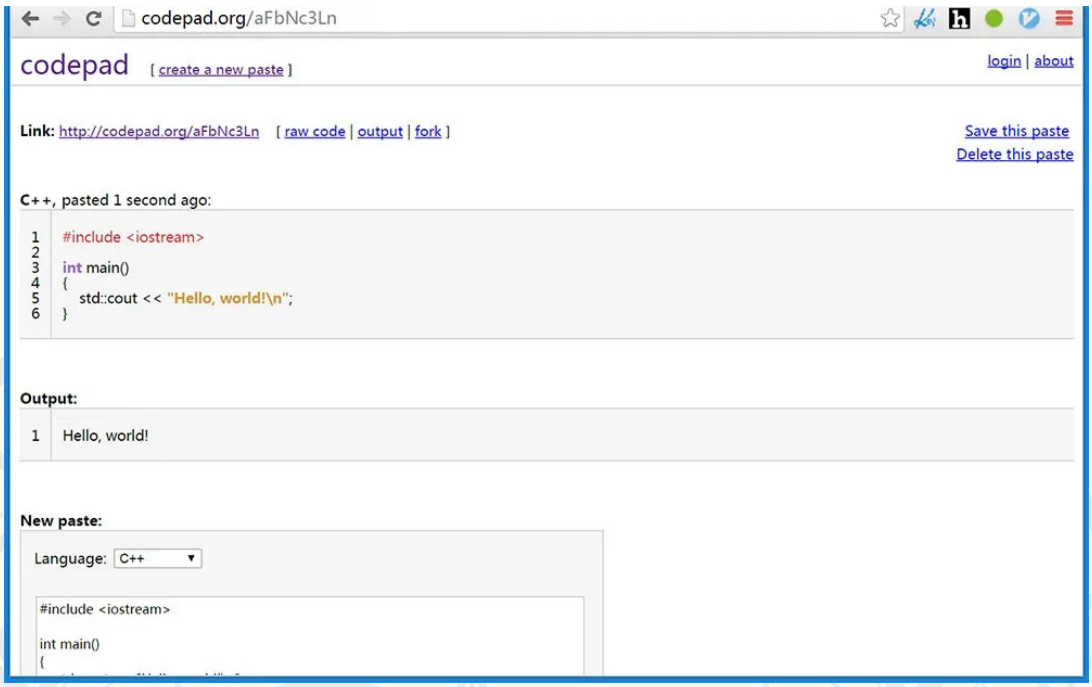


*Рис. 1.2 Веб-сервіс CodePad*

### 1.3.3 Веб-сервіс DotNetFiddle

Цей редактор підтримує C#, F# і VB.NET. Він дозволяє ділитися кодом, як для перегляду, так і для спільної роботи. Також є різні режими роботи — для консольного застосунку, скрипту, за шаблоном MVC і з фреймворком Nancy. А ще наявна опція «tidy up» — якщо ручне розставляння відступів забирає багато часу.

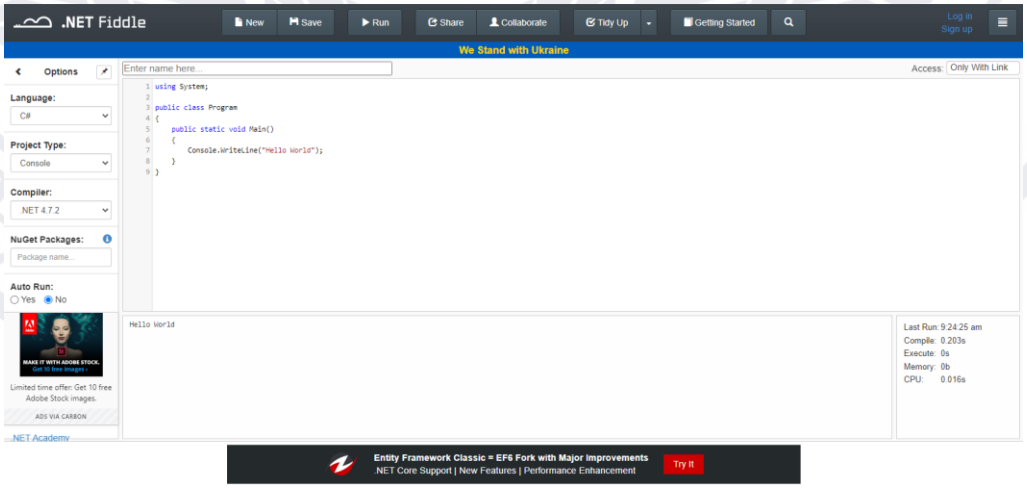


*Рис. 1.3 Веб-сервіс DotNetFiddle*

### 1.3.4 Веб-сервіс Codewars

Codewars – це сервіс, за допомогою якого можна вирішувати алгоритмічні задачі різного рівня складності. Тут доступні різні рейтинги, щоб вирішення задач було своєрідним змаганням. Складність задач є багаторівневою, чим більше задач вирішено – тим складніші завдання стають доступними

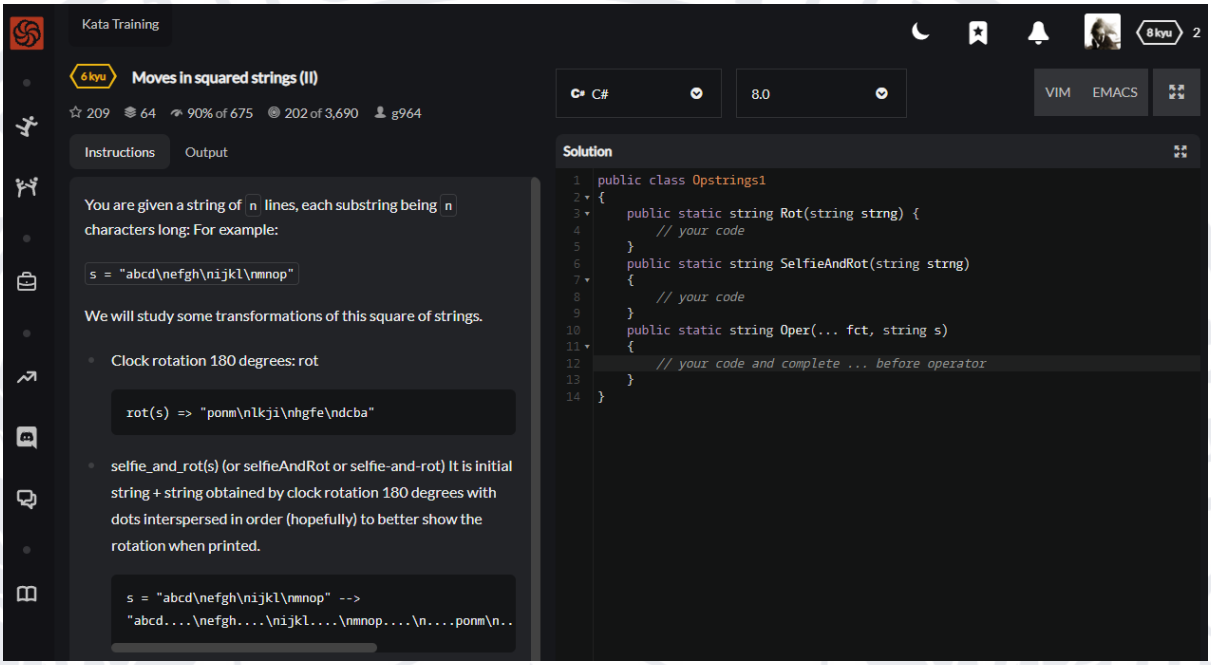


*Рис. 1.4 Веб-сервіс Codewars*

## 1.4 Висновки до першого розділу

В даному розділі було проаналізовано особливості сервісів з автоматичної перевірки коду. Онлайн компілятори вже давно користуються популярністю, адже це заощадження часу та ресурсів. Так як є високий попит на подібні сервіси, на просторах інтернету їх є дуже багато. Всі вони різноманітні, мають свої особливості, переваги, та недоліки.

Переваги онлайн сервісів з автоматичної перевірки коду:

- Велика різноманітність (є безліч рішень, для різних задач)

- Простота використання (на багатьох сервісах не потрібно проходити автентифікацію, аби розпочати роботу)

Недоліки онлайн сервісів з автоматичної перевірки коду:

- Відсутність дружнього до користувача дизайну (здебільшого такі сервіси мають важкий для сприйняття дизайн сторінок)
- Повільна робота сервісів (більшість рішень реалізовано за допомогою застарілих технологій, і не мають високої швидкодії)

# РОЗДІЛ 2. АНАЛІЗ ІНФОРМАЦІЙНИЙ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ РОЗРОБКИ

Як вже описувалось в 1 розділі, мета цього дипломного проєкту – створення такого веб-сервісу для автоматичної перевірки коду, який буде мати високу швидкість виконання необхідних операцій, буде мати дружній до клієнта інтерфейс, забезпечувати надійність роботи в такому середовищі.

Саме для цих задач будуть використані провідні технології в сфері фронтенд розробки. Веб-сервіс буде створений як SPA (Single Page Application), за допомогою бібліотеки React. В цьому розділі знаходиться опис тих технологій, які були використанні для розробки фронтенд частини системи автоматичної перевірки коду та інструментів, які будуть використовуватись в ході розробки.

## 2.1 Основні технології розробки

### 2.1.1 Javascipt

JavaScript (JS) — динамічна, об'єктно-орієнтована, прототипна мова програмування. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Програми цією мовою називаються скриптами. Вони можуть вбудовуватися в HTML і виконуватися автоматично під час завантаження веб-сторінки. Скрипти розповсюджуються і виконуються як простий текст. Їм не потрібна спеціальна підготовка чи компіляція для запуску.

Коли JavaScript створювався, він мав інше ім'я – «LiveScript». Однак, мова Java була дуже популярна в той час, і було вирішено, що позиціонування JavaScript як «молодшого брата» Java буде корисним. З часом JavaScript став повністю незалежною мовою зі своєю власною специфікацією, що називається ECMAScript, і зараз не має жодного відношення до Java.

Сьогодні JavaScript може виконуватися не тільки в браузері, але й на сервері або на будь-якому іншому пристрої, який має спеціальну програму, що називається "движком" JavaScript.

Сучасний JavaScript - це "безпечна" мова програмування. Він не надає низькорівневий доступ до пам'яті або процесора, тому що спочатку був створений для браузерів, які цього не вимагають.

Можливості JavaScript залежать від оточення, в якому він працює. Наприклад, Node.JS підтримує функції читання/запису довільних файлів, виконання мережевих запитів тощо.

У браузері для JavaScript доступне все, що пов'язане з маніпулюванням веб-сторінками, взаємодією з користувачем та веб-сервером.

### 2.1.2 React

Бібліотека React була вперше випущена компанією Facebook у 2013 році. Для того, щоб зрозуміти, наскільки популярною ця технологія стала за минулий час, звернімося до опитування розробників, проведеного сайтом StackOverflow цього року. Понад 50 000 розробників поділилися інформацією про свою роботу та професійні уподобання. Крім більш-менш передбачуваних результатів, які

описують стан IT-індустрії на сьогоднішній день, є також і щось цікаве щодо безпосередньо React. Ця бібліотека стала однією з найулюбленіших і найзатребуваніших технологій, а також найтрендовішою технологією на StackOverflow:

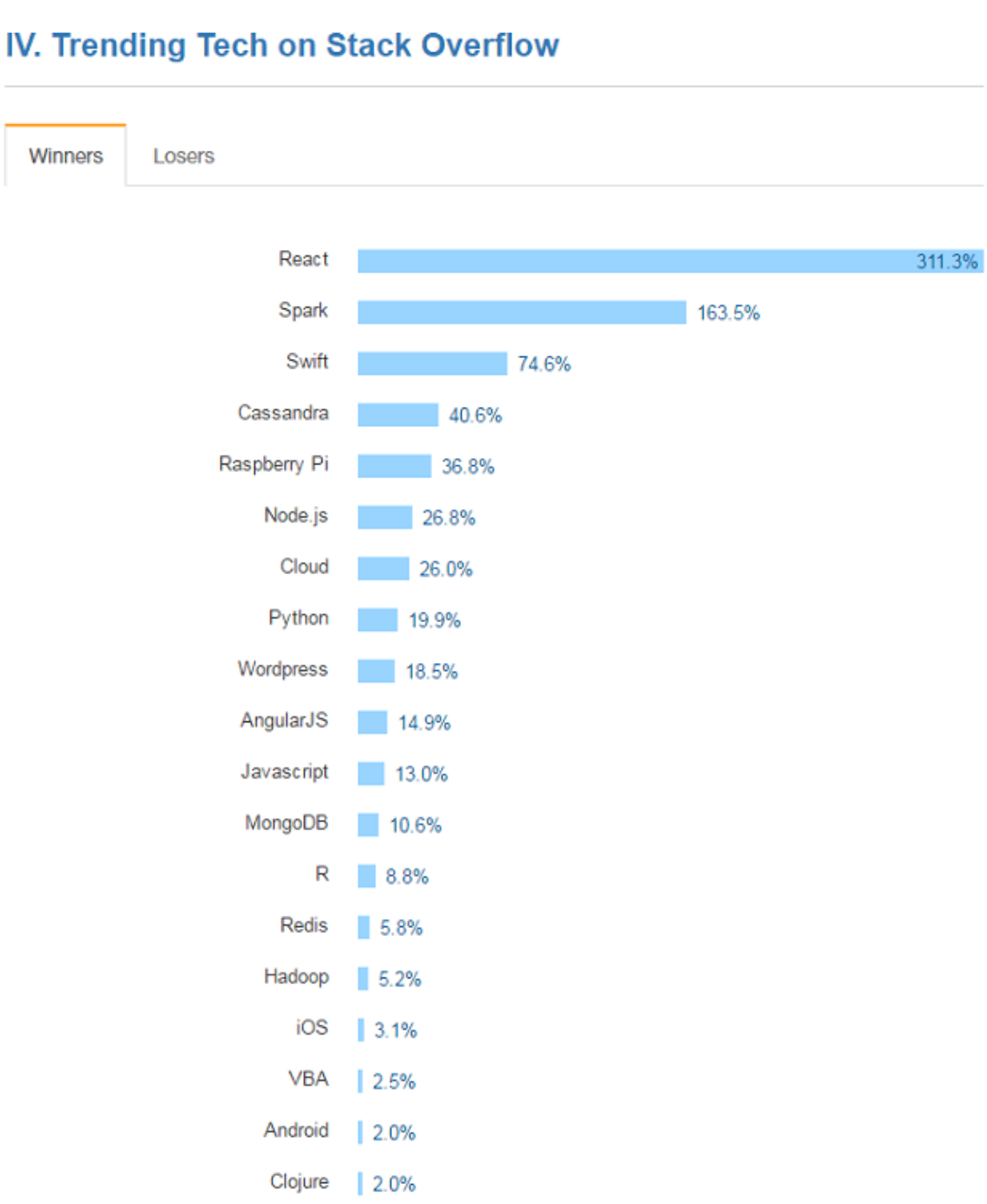


*Рис. 2.1 Опитування розробників зі StackOverflow*

Якщо розглядати React в розрізі фронтенд розробки, то однозначно можна сказати, що ця бібліотека є найбільш популярною серед інших, що можна побачити на графіку нижче. Підтримкою цієї бібліотеки займається компанія Meta, тому для розробників react є надійною технологією, яка активно підтримується провідною компанією в сфері IT технологій.

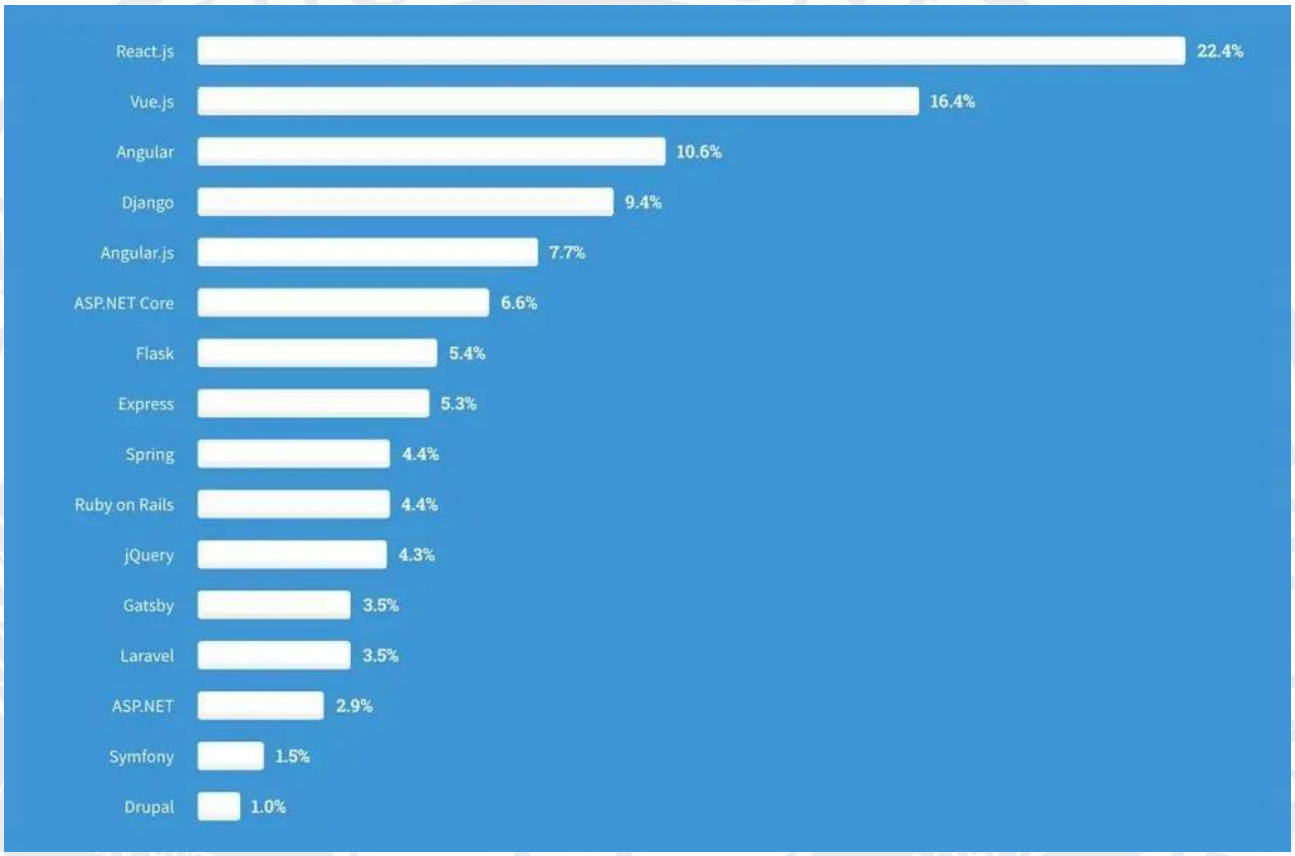


*Рис. 2.2 Опитування розробників зі StackOverflow*

React це бібліотека для створення інтерфейсів користувача. Однією з її відмінних рис є можливість використовувати JSX, мову програмування з близьким до HTML синтаксисом, який компілюється в JavaScript. Розробники можуть досягати високої продуктивності програм за допомогою Virtual DOM. З React можна створювати ізоморфні програми, які допоможуть позбутися неприємної ситуації, коли користувач з нетерпінням чекає, коли ж нарешті завершиться завантаження даних і на екрані його комп'ютера нарешті з'явиться щось, окрім анімації завантаження. Створені компоненти можуть бути легко

змінені і використані заново в нових проектах. Високий відсоток перевикористання коду підвищує покриття тестами, що, у свою чергу, призводить до більш високого рівня контролю якості.

Коли ми говоримо про ізоморфні програми або про ізоморфний JavaScript, ми маємо на увазі, що ми можемо використовувати один і той самий код як у серверній, так і в клієнтській частині програми. Коли користувач відкриває сайт у своєму браузері, вміст сторінки має бути завантажений із сервера. У випадку з SPA-додатками (Single Page Application) це може зайняти деякий час. Під час завантаження користувачі бачать порожню сторінку або анімацію завантаження. Враховуючи, що за сучасними стандартами очікування протягом більш ніж двох секунд може бути помітною незручністю для користувача, скорочення часу завантаження може виявитися вкрай важливим. А ще одна вагома проблема: пошукові машини не індексують такі сторінки так добре, як нам хотілося б. Виконання JavaScript коду на стороні сервера допомагає виправити такі проблеми. Якщо створювати ізоморфні програми, можна отримати помітну вигоду, виконуючи рендеринг на стороні сервера. Після завантаження сторінки все ще можна продовжувати рендеринг компонентів. Така можливість рендерингу сторінок як на сервері, так і на клієнті призводить до помітних переваг, таким як можливість кращого індексування сторінок пошуковими машинами та поліпшення досвіду користувача. Більше того, такий підхід дозволяє знизити час, що витрачається на розробку.

Document Object Model, або DOM, - це спосіб представлення та взаємодії з об'єктами в HTML, XHTML та XML документах. Відповідно до цієї моделі, кожен такий документ є ієрархічним деревом елементів, так званим DOM-деревом. Використовуючи спеціальні методи, можна отримати доступ до певних елементів нашого документа та змінювати їх. Коли ми створюємо динамічну інтерактивну веб-сторінку, ми хочемо, щоб DOM оновлювався так швидко, як це можливо після зміни стану певного елемента. Для цього деякі фреймворки

використовують прийом, який називається «dirty checking» і полягає в регулярному опитуванні стану документа і перевірці змін у структурі даних. Як ви можете здогадатися, подібне завдання може стати справжнісіньким головним болем у разі високонавантажених додатків. Virtual DOM, своєю чергою, зберігається у пам'яті. Саме тому в момент, коли «справжній» DOM змінюється, React може змінювати Virtual DOM миттєво. React «збирає» такі зміни порівнює їх зі станом DOM, а потім перемальовує компоненти, що змінилися.

При цьому підході ви не виконується регулярне оновлення DOM. Саме тому може бути досягнуто вищої продуктивності React додатків. Друге слідство випливає з ізоморфної природи React: можна робити рендеринг на стороні сервера так само як на стороні клієнта.

### 2.1.3 HTML5

HTML – це мова розмітки документів. Він застосовується у всьому світі. Браузер інтерпретує HTML-код для відображення його на комп'ютері, планшеті або телефоні. Мова HTML була розроблена британцем Тімом Бернерсом-Лі. Насамперед мова HTML призначалася для обміну науковими документами. Верстка документів відбувається за допомогою спеціальних дескрипторів (але найчастіше їх називають тегами). Якщо простіше відповісти на запитання: «Що таке HTML документ?» - це простий текст, який містить багато тегів, що утворює веб-сторінку.

Було безліч версій HTML, на даний момент остання версія – HTML5. Перша бета-версія HTML5 з'явилася восени 2007 року. Для спрощення та зручності було запроваджено поняття «гіпертекст». Гіперпосилання (або просто посилання) є частиною гіпертексту, і вона посилається на інший HTML документ.

HTML був побудований так, що сторінки відображалися на всіх пристроях однаково. Згодом додали графічне оформлення (CSS).

Семантична верстка - підхід до розмітки, який спирається не на зміст сайту, а на значення кожного блоку і логічну структуру документа. Існують заголовки різних рівнів — це допомагає читачеві побудувати структуру документа.

Для чого потрібна семантика:

- *Щоб зробити веб-сайт доступним*. Зрячі користувачі можуть без проблем з першого погляду зрозуміти, де яка частина сторінки знаходиться, де заголовок, списки або зображення. Для незрячих чи частково незрячих все складніше. Основний інструмент для перегляду сайтів є не браузер, який малює сторінку, а скринрідер, який читає текст зі сторінки вголос.

  Цей інструмент «зачитує» вміст сторінки, і семантична структура допомагає йому краще визначати, який це блок, а користувачеві розуміти, про що йдеться. Таким чином, семантична розмітка допомагає більшій кількості користувачів взаємодіяти з вашим сайтом. Наприклад, наявність заголовків допомагає незрячим у навігації по сторінці. У скринридерів є функція навігації за заголовками, що прискорює знайомство з інформацією на сайті.

- *Щоб сайт був вищим у пошукових системах*. Компанії, які створюють пошукові системи, не розголошують правила ранжування, але відомо, що наявність семантичної розмітки сторінок допомагає пошуковим роботам краще розуміти, що знаходиться на сторінці, і в залежності від цього ранжувати сайти в пошуковій видачі.

### 2.1.4 CSS3

Каскадні таблиці стилів (CSS) - це мова, яка використовується для ілюстрації зовнішнього вигляду, стилю та формату документа, написаного будь-якою мовою розмітки. Простіше кажучи, він використовується для стилізації та організації макету веб-сторінок. CSS3 є останньою версією раніше CSS2.

Істотною зміною CSS3 в порівнянні з CSS2 є введення модулів. Перевага цієї функціональності полягає в тому, що вона дозволяє завершити специфікацію та прийняти її швидше, оскільки сегменти завершуються та приймаються порціями. Крім того, це дозволяє браузеру підтримувати сегменти специфікації.

Деякі з ключових модулів CSS3:

- Модель коробки
- Значення зображень та замінений контент
- Текстові ефекти
- Селектори
- Фони та межі
- Анімації
- Інтерфейс користувача (UI)
- Розташування кількох стовпців
- 2D/3D перетворення

CSS3 використовується з HTML для створення та форматування структури контенту. Він відповідає за кольори, характеристики шрифту, вирівнювання тексту, фонові зображення, графіку, таблиці і т. д. Він забезпечує позиціонування різних елементів з фіксованими, абсолютними та відносними значеннями.

Щоб допомогти у створенні інтерактивних веб-сторінок із високим ступенем інтерактивності, CSS3 дуже рекомендується, оскільки він надає ширші можливості для проектування. Рекламні продукти та послуги веб-сайту спочатку переглядається клієнтом, тому він повинен бути привабливим, і це може бути досягнуто за допомогою CSS3.

CSS3 дозволяє дизайнеру створювати веб-сайти, багаті на контент із низьким вмістом коду. Ця технологія надає деякі функції, які роблять сторінку

привабливою, дає можливість користувачу з легкістю пересуватись по ній, а також функціонує бездоганно.

Деякі особливості, такі як тіні, закруглені кути та градієнти, знаходять застосування практично на кожній веб-сторінці. Ці покращення дизайну можуть зробити сайт привабливим за умови правильного використання. Раніше, щоб використовувати ці методи, доводилося вдаватися до багатьох складних методів із великою кількістю елементів кодування та HTML. Але тепер CSS3 дозволяє включати набагато простіше реалізувати стилізацію веб-сторінок.

## 2.2 Допоміжні веб-технології

### 2.2.1 Axios

Багатьом веб-проектам на певному етапі їх розвитку потрібна взаємодія з REST API. Проєкт системи автоматичної перевірки коду не є виключенням. Для взаємодії з сервером, а саме отримання відповіді від бекенд частини сервісу, буде використано бібліотеку axios.

Axios це полегшений клієнт HTTP на базі сервісу $http з Angular.js v1.x, схожого на власний JavaScript Fetch API.

Axios заснований на промісах, що дає можливість використовувати можливості JavaScript async і await для отримання зручнішого асинхронного коду. За допомогою axios можна перехоплювати та скасовувати запити, також у клієнта є вбудований захист від підробки запитів між сайтами. Також використання axios дозволяє уникнути написання великих обсягів шаблонного коду і зробити код чистішим і зрозумілішим.

### 2.2.2 JSX

JSX дозволяє створювати HTML-елементи прямо в JavaScript і поміщати їх у DOM без використання таких методів, як createElement() або appendChild(). JSX перетворює HTML-теги на елементи React. React використовує JSX для шаблонування замість звичайного JavaScript. Використовувати JSX не обов'язково, однак він надає кілька переваг:

- Він швидший завдяки оптимізації під час компіляції коду JavaScript
- Він також є "типобезпечним", більшість помилок перехоплюються під час компіляції.
- Він дозволяє легше і швидше створювати шаблони

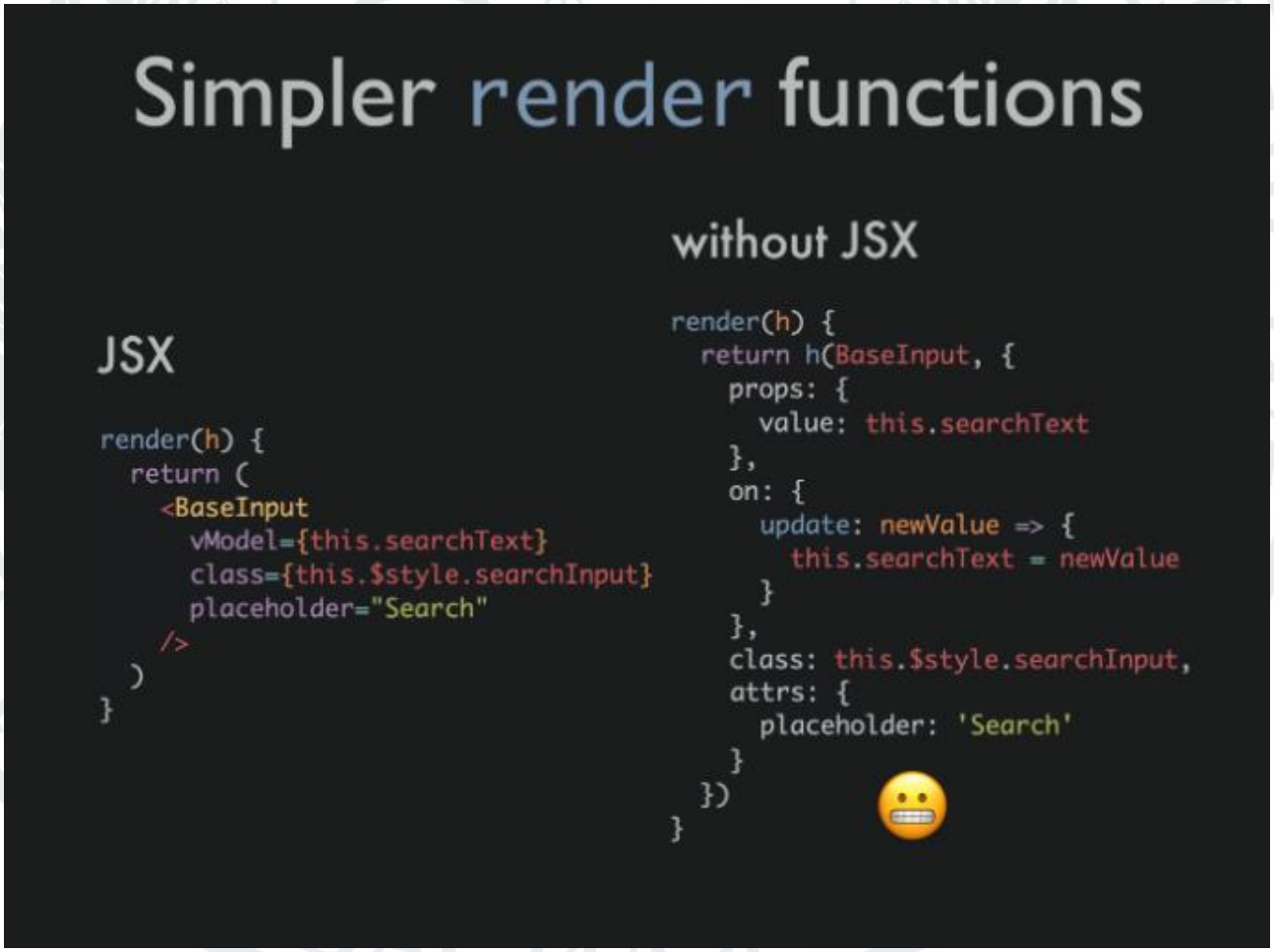


*Рис. 2.3 Приклад функції з використанням JSX*

### 2.2.3 JSON

Позначення об'єктів JavaScript (JSON – JavaScript Object Notation) – стандартний текстовий формат для представлення структурованих даних на основі синтаксису об'єкта JavaScript. Він зазвичай використовується для передачі даних у веб-додатках (наприклад, відправлення деяких даних із сервера клієнту, таким чином, щоб це могло відображатися на веб-сторінці або навпаки).

JSON - текстовий формат даних, який слідує за синтаксисом об'єкта JavaScript, який був популяризований Дугласом Крокфордом. Незважаючи на те, що він дуже схожий на буквений синтаксис об'єкта JavaScript, його можна використовувати незалежно від JavaScript, і багато програмних середовищ мають можливість читати (аналізувати) і генерувати JSON.

JSON існує як рядок, що необхідний при передачі даних через мережу. Він має бути перетворений на власний об'єкт JavaScript, якщо потрібно отримати доступ до даних. Але, це не велика проблема. JavaScript надає глобальний об'єкт JSON, який має методи перетворення між ними.

Об'єкт JSON може бути збережений у власному файлі, який найчастіше є текстовим файлом з розширенням .json та MIME type application/json.

Деякі властвості JSON:

- JSON – це чисто формат даних – він містить лише властивості, без методів.
- JSON вимагає подвійних лапок, які будуть використовуватися навколо рядків та імен властивостей. Одиночні лапки недійсні.
- Навіть одна недоречна кома або двокрапка можуть призвести до збою JSON-файлу і не працювати. Потрібно бути обережним, та перевірити будь-які дані, які будть використовуватись (хоча згенерований комп'ютером JSON з меншою ймовірністю включає помилки, якщо програма генератора працює правильно). Можна перевірити JSON за допомогою програми типу JSONLint.

- JSON може набувати форми будь-якого типу даних, допустимого для включення в JSON, а не тільки масивів або об'єктів. Наприклад, один рядок або номер буде дійсним об'єктом JSON.
- На відміну від коду JavaScript, в якому властивості об'єкта можуть не полягати в подвійні лапки, JSON як властивості можуть використовуватися тільки рядки укладені в подвійні лапки.

Також існує розширення JSON5, яке відповідає синтаксису ECMAScript 5, але також повністю сумісне з базовим форматом JSON. У ньому реалізована підтримка одно- та багаторядкових коментарів, одинарні та подвійні лапки для рядків, ключі записів без лапок та низка інших нововведень.

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

*Рис. 2.4 Приклад інформації в JSON форматі*

### 2.2.4 Препроцесор SCSS

Sass це розширення CSS, яке надає потужності та елегантності цій простій мові. Sass дає можливість використовувати змінні, вкладені правила, міксини, інлайнові імпорти та багато іншого, все з повністю сумісним із CSS синтаксисом. Sass допомагає зберігати великі таблиці стилів добре організованими, а невеликим стилям працювати швидко, особливо за допомогою бібліотеки стилів Compass.

Для Sass є два синтаксиси. Перший, відомий як SCSS (Sassy CSS) - це розширений синтаксис CSS. Це означає, що кожна валідна таблиця стилів CSS це валідний SCSS файл, що несе в собі ту ж саму логіку. Більш того, SCSS розуміє більшість хаків у CSS та вендорні синтаксиси, наприклад такий як синтаксис фільтра у старому IE. Цей синтаксис покращено Sass функціоналом. Файли, які використовують цей синтаксис мають розширення .scss.

Другий і старіший синтаксис, також відомий як краєний синтаксис або іноді просто Sass, дає більш стислу можливість роботи з CSS. Він використовує відступи замість дужок, що відокремлює вкладення селекторів і нові рядки замість точок з комою для поділу властивостей. Іноді люди знаходять такий спосіб простіше для розуміння та швидше для написання, ніж SCSS. За фактом, такий синтаксис має той самий функціонал, хоча деякі з них мають трохи інший підхід. Файли, що використовуються цей синтаксис, мають розширення .sass.

В даному проєкті буде використано синтаксис SCSS, так як цей підхід є більш сучасним та ефективним.

Можливості SCSS:

- *Вкладені правила:* можна вкладати CSS властивості, в кілька наборів дужок {}. Це зробить CSS чистішим і зрозумілішим.
- *Змінні*: у стандартному CSS теж є змінні, але змінні Scss значно потужніший інструмент. Наприклад, можна використовувати змінні в

циклах і генерувати значення властивостей динамічно. Також можна впроваджувати змінні в імена властивостей, наприклад: property-name-N { … }.

- *Краща реалізація операторів*: можна підсумовувати, віднімати, ділити та множити CSS значення. Scss реалізація інтуїтивніша, ніж стандартний функціонал CSS calc().
- *Функції*: Scss дозволяє багаторазово використовувати CSS стилі як функції.
- *Тригонометрія*: крім базових операцій (+, -, *, /), SCSS дозволяє писати власні функції. Наприклад, функції sin та cos можна написати, використовуючи лише синтаксис Sass/SCSS. Такі функції можуть знадобитися для створення анімації.
- *Зручний робочий процес*: можна писати CSS, використовуючи конструкції, знайомі з інших мов: for-цикли, while-цикли, if-else. Але потрібно мати на увазі, що це лише препроцесор, а не повноцінна мова, Scss контролює генерацію властивостей та значень, а на виході отримується стандартний CSS.
- *Міксини*: дозволяють один раз створити набір правил, щоб потім їх багаторазово або змішувати з іншими правилами. Наприклад, міксини використовують для створення окремих тем макету.

SCSS насправді не додає нічого нового в саму мову CSS. Це просто новий синтаксис, який здебільшого скорочує час написання стилів.

На рисунку нижче приведено приклад Scss коду, та показано його вигляд в Css, вже після компіляції.

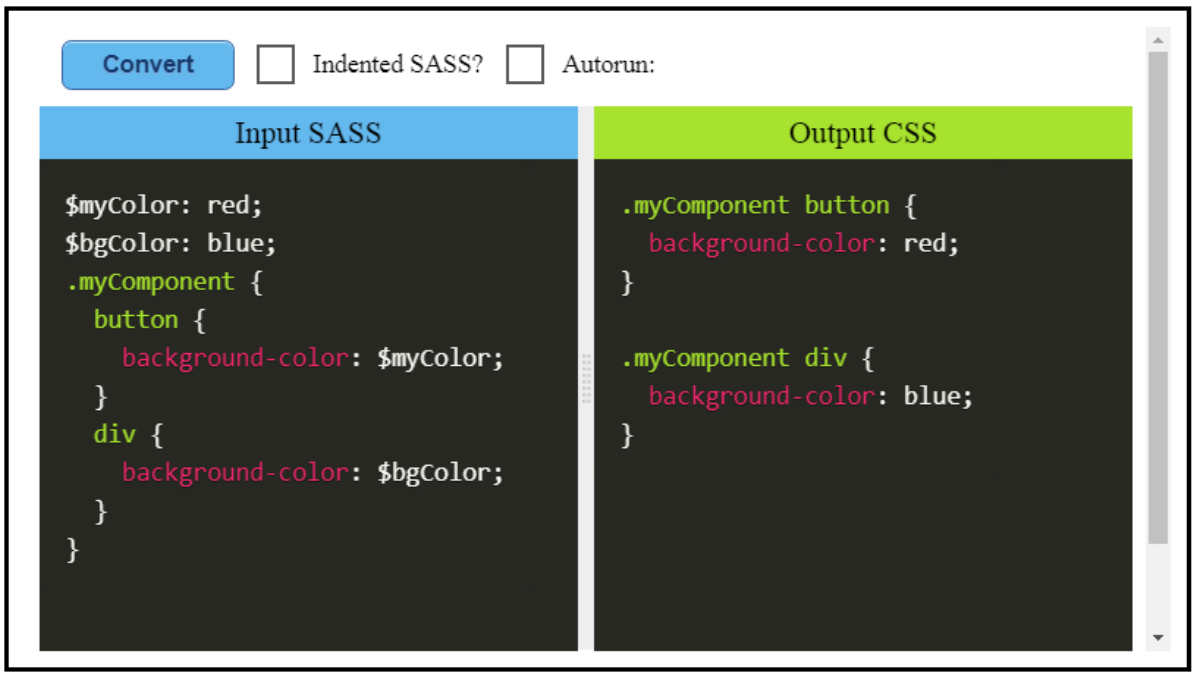


*Рис. 2.5 Код в Scss та Css відповідно*

### 2.2.5 BEM methodology

БЕМ - абревіатура від Блок, Елемент, Модифікатор. Це три ключові сутності, які використовуються для розробки компонентів інтерфейсу.

Історія БЕМ розпочалася у 2005 році. Тоді з погляду інтерфейсу звичайний проект Яндекса був набором статичних HTML-сторінок, які використовувалися як основа для створення шаблонів на XSL.

- Для кожної сторінки створювався окремий HTML-файл. У верстці використовувалися id та класи.
- Скрипти зберігалися в одному файлі для проекту — project.js. JavaScript використовувався як допоміжний інструмент пожвавлення сторінки, тому project.js був невеликим.
- Картинки складалися до окремої директорії. Необхідність підтримки IE 5 і відсутність CSS3 у браузерах змушувало використовувати картинки для реалізації будь-якого оформлення, навіть для закруглених куточків.

- Стилі та скрипти писалися у файлах: project.css, project.js. Для відокремлення стилів різних частин сторінки використовувалися коментарі із зазначенням початку та кінця

У 2006 році почалася робота над першими великими проектами. Ці проекти з десятками сторінок виявили основні недоліки поточного підходу до розробки:

- неможливо внести зміни до коду однієї сторінки, не торкаючись коду іншої;
- складно підбирати назви класам.

**Блок:**

Функціонально незалежний компонент сторінки, який можна повторно використати. HTML блоки представлені атрибутом class.

Назва блоку характеризує сенс *("що це?" - "Меню": menu, "кнопка": button), а не стан ("який, як виглядає?" - "червоний": red, "великий": big).*

- Блок не повинен впливати на своє оточення, тобто блоку не слід задавати зовнішню геометрію (у вигляді відступів, кордонів, що впливають на розміри) та позиціонування.
- У CSS БЕМ також не рекомендується використовувати селектори за тегами або id.

Таким чином, забезпечується незалежність, при якій можливе повторне використання або перенесення блоків з місця на місце.

**Елемент:**

Складова частина блоку, яка не може використовуватись у відриві від нього.

- Назва елемента характеризує сенс *("що це?" - "пункт": item, "текст": text), а не стан ("який, як виглядає?" - "червоний": red, "великий": big).*

- Структура повного імені елемента відповідає схемі: імя-блока__імя-элемента. Ім'я елемента відокремлюється від імені блоку двома підкресленнями (__).

**Модифікатор:**

Сутність, що визначає зовнішній вигляд, стан чи поведінку блоку чи елемента.

- Назва модифікатора характеризує зовнішній вигляд *("який розмір?", "яка тема?" і т. п. - "розмір": size_s, "тема": theme_islands)*, стан *("чим відрізняється від інших?" - "відключено"): disabled, "фокусований": focused)* і поведінка *("як поводиться?", "як взаємодіє з користувачем?" - "напрямок": directions_left-top).*
- Ім'я модифікатора відокремлюється від імені блоку або елемента одним підкресленням (_).

### 2.2.6 Адаптивний дизайн та верстка

Адаптивність - одна з ключових вимог до сучасних сайтів. Ресурс повинен однаково добре демонструватися на екрані комп'ютера, планшета, смартфона. Для цього створюється адаптивний дизайн та використовується адаптивна верстка. У результаті процес веб-розробки ускладнюється, так як потребує додаткових ресурсів. Але результат вартує вкладень: сайт відмінно позиціонується на всіх основних типах пристроїв, що гарантує повноцінне охоплення аудиторії.

Адаптивність - це здатність сайту "підлаштовуватися" під різні технічні умови (а саме, під розміри екрана користувача пристрою). Адаптивний сайт добре виглядає і на моніторі (звичайний ПК), і на планшетному комп'ютері, і на екрані смартфона. Причому якість відображення не залежить ні від діагоналі, ні від позиціонування екрана.

Адаптивна верстка — це спосіб створення веб-сторінок, при якому вони автоматично підлаштовуються під розміри та орієнтацію екрана пристрою, а їх дизайн змінюється в залежності від дій користувача.

Мета адаптивної верстки - домогтися того, щоб сайт залишався зручним та забезпечував конверсію під час завантаження на різних пристроях.

Сьогодні близько 50% користувачів відвідують сайти з гаджетів – смартфонів, планшетів. Це зручно, адже можна серфити по мережі, перебуваючи в будь-якій точці простору (де є інтернет) - лежачи в ліжку, на вулиці, у транспорті. Мобільна аудиторія постійно зростає, і ігнорувати її потреби не можна. Ось чому розробники адаптують веб-сайти під портативні пристрої.

Раніше коли частка мобільної аудиторії була порівняно невелика, адаптивна верстка не вважалася чимось вкрай необхідним. Тепер питання про адаптивність порушується в обов'язковому порядку - адаптивність сайту є однією з основних вимог при замовленні клієнтом

На графіку, приведеному нижче, демонструється, який відсоток населення користується тим чи іншим гаджетом, і звідси можна зробити висновок, що найбільший відсоток користується саме смартфоном. Тому, адаптивність вкрай важлива при розробці веб-сайту.

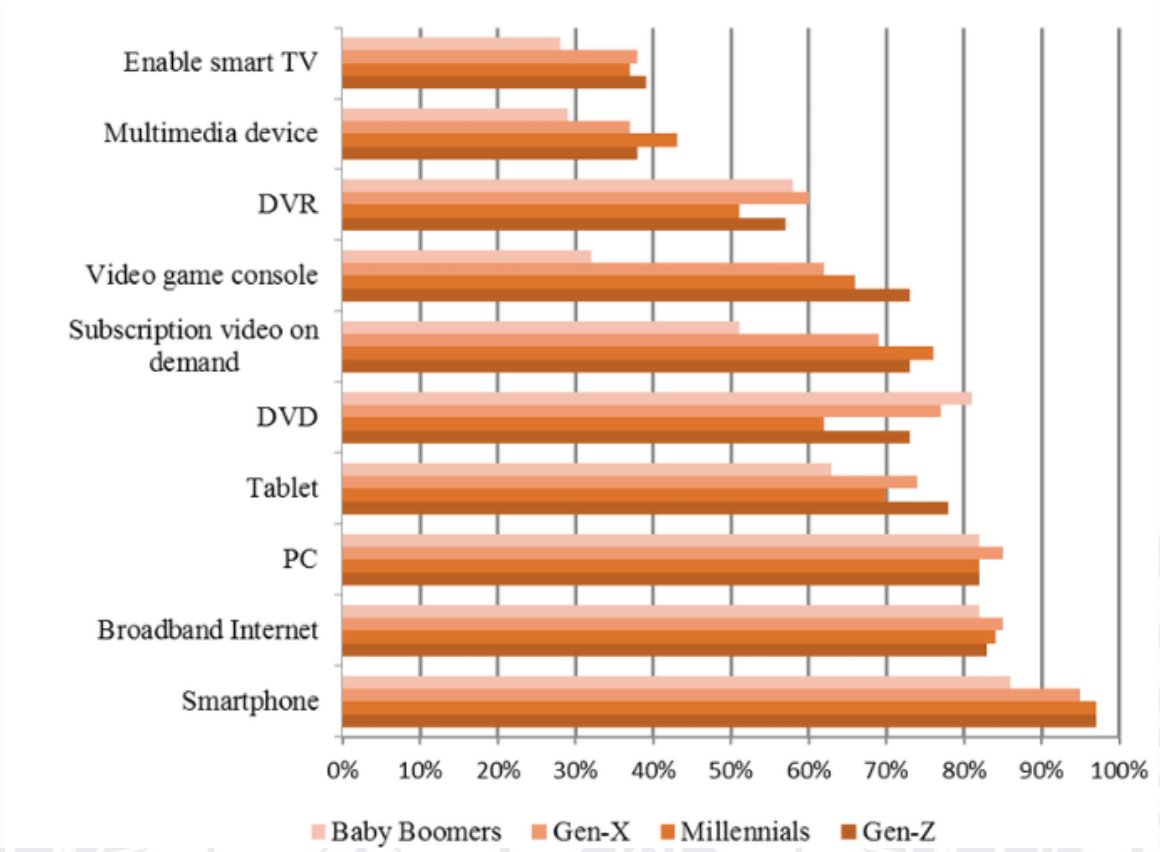


*Рис. 2.6 Графік використовування різних гаджетів*

## 2.3 Git

Git — розподілена система керування версіями файлів та спільної роботи. Проєкт створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

Git дозволяє повертати окремі файли і весь проект до попереднього стану, переглядати зміни, що відбуваються з часом. Визначати, хто останнім вносив зміни в модуль, що раптово перестав працювати, відстежити ланцюг подій, який

привів до помилок і багато іншого. Одним з найпопулярніших ресурсів для роботи з Git є GitHub. Для зручності роботи з ним існує графічний клієнт GitHub Desktop та консольний Git Shell.

Git дозволяє створювати кілька різних паралельних версій проекту – гілок, кожна з яких призначена для різних цілей. Допустимо, існує працюючий проект. Основна гілка проекту – master. У ньому зберігається остання стабільна версія. Для додавання до проекту нової функціональності необхідно створити окрему гілку, при цьому ці зміни ніяк не позначаться на основній версії. Можна переключатися між гілками, що призведе до зміни файлів локальної версії репозиторію.

Перед створенням нової гілки необхідно зафіксувати зміни в поточній гілці або видалити їх з робочого каталогу та області встановлення.

Щоб створити нову гілку та перейти до роботи над нею, виконується така команда:

*git checkout -b <ім'я нової гілки>.*

Батьківською гілкою буде та гілка, над якою працюєте на даний момент.

Є кілька базових команд, які потрібні для роботи з Git. Деякі з них:

*- git add - додавання файлів в область зберігання*

*- git commit -m "message" - фіксування файлів у локальному репозиторії*

*- git reset - зняти зміни*

*- git status - перевірити статус файлів проекту*

*- git log - показати журнал фіксацій*

*- git clone url folder - створює копію віддаленого git-репозиторію з url у локальну папку. Також привласнює віддаленому репозиторію ім'я "origin" і встановлює його як віддалений за замовчуванням*

*- git remote add name url - створює зв'язок між поточним репозиторієм та названим віддаленим репозиторієм*

## 2.4 WebStorm

Програмне забезпечення JetBrains WebStorm є інструментом для розробки веб-сайтів та редагування HTML, CSS та JavaScript коду. Рішення забезпечує швидку навігацію по файлах і генерує повідомлення про проблеми в коді в режимі реального часу. JetBrains WebStorm дозволяє додавати розмітку HTML-документів або елементів SQL безпосередньо до JavaScript. JetBrains WebStorm здійснює розгортання та синхронізацію проектів через протокол FTP.

Використовуючи можливості коду HTML/XHTML та XML, WebStorm забезпечує автоматичне завершення стилів, посилань, атрибутів та інших елементів коду. Під час роботи з CSS здійснюється завершення коду класів, HTML-номерів, ключових слів тощо. WebStorm пропонує автоматичне вирішення таких проблем, як вибір формату, властивостей, класів, посилань на файли та інших атрибутів CSS. Рішення дозволяє використовувати потужність інструменту Zen coding для верстки HTML, відображає дії тега на веб-сторінці. Продукт WebStorm здійснює завершення JavaScript для ключових слів, лейблів, змінних, параметрів і функцій DOM і підтримує специфічні особливості популярних браузерів. Реалізовані у вирішенні функції рефакторингу JavaScript дозволяють перетворювати структуру коду та файлів та .js.

WebStorm забезпечує налагодження JavaScript і надає широкий діапазон можливостей: знаходження точки зупинки в HTML і JavaScript, налаштування параметрів точки зупинки, тестування синтаксису коду в режимі реального часу і т. д. Продукт підтримує платформи JQuery, YUI, Prototype, DoJo, MooTools, Qooxdoo та Bindows. WebStorm передбачає інтегровану перевірку тексту на теги, послідовність коду, помилки в написанні і т. д. WebStorm дозволяє редагувати

файли та автоматично синхронізувати їх на вимогу при віддаленій роботі або зберіганні.

Продукт підтримує функцію контролю версій та попередніх варіантів коду та фіксує всі вироблені дії та зміни. Завдяки створенню історії WebStorm можна відновлювати кодові висловлювання, блоки і навіть цілі файли.

## 2.5 Висновки до другого розділу

Основною перевагою веб-сервісу, який розробляється в ході даної роботи являється використання найбільш провідних технологій в сфері фронтенд розробки. Саме це забезпечує швидкодію такого сервісу, простоту реалізації, читабельний програмний код, та можливість подальшої оптимізації. В другому розділі було коротко описано технології та інструменти, які були використанні для реалізації фронтенд частини системи автоматичної перевірки коду

# РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ

Як було розглянуто в першому розділі, багато аналогів даного веб сервісу мають суттєві недоліки. Найбільшою проблемою є повільність роботи. Більшість таких сервісів написані за допомогою застарілих технологій. Даний проєкт написаний за допомогою React, який направлений на забезпечення швидкодії веб сервісу.

## 3.1 Create React App

Create React App — це комфортний осередок для вивчення React, а також це найкращий шлях щоб почати будувати нові односторінкові додатки за допомогою React. Він встановлює осередок для розробки таким чином, щоб можна було використовувати найновіші можливості JavaScript, робить розробку комфортнішою, а також оптимізує додаток для продакшну.

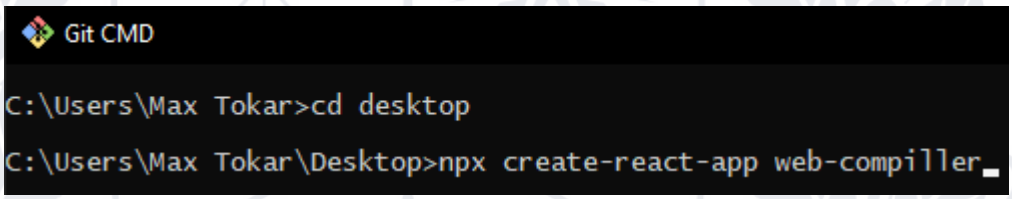


*Рис 3.1 Створення реакт застосунку*

Create React App не опрацьовує бекенд логіку чи логіку баз данних, а лише надає команди для побудови фронтенду, тому можна використовувати його з будь-яким бекендом. Під капотом він використовує Babel та webpack.

## 3.2 Структура веб сервісу

Одною з переваг бібліотеки React є компонентний підхід. Тому, сам веб сервіс складається з компонентів (файли з розширенням jsx), які можна використовувати повторно при необхідності, а також із допоміжних файлів з необхідними функціями (файли з розширенням js)

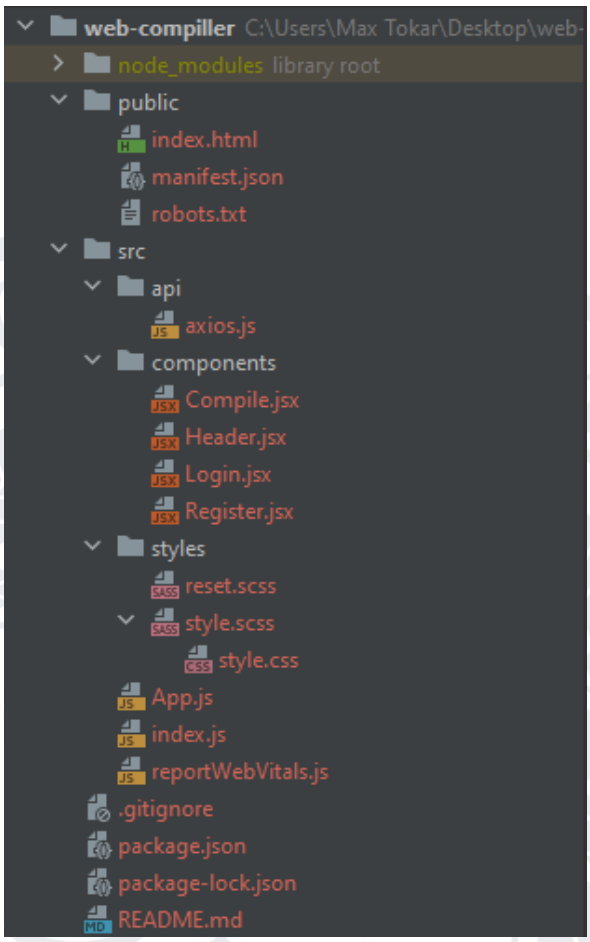


*Рис 3.2 Структура реакт застосунку*

У файлі App.js відбувається рендер всього застосунку, там викликаються компоненти, у відповідній черговості

App.js

```
import React from "react";
import Header from "./components/Header";
import Compile from "./components/Compile";
import 'normalize.css';


function App() {
  return (
    <>
      <Header />
      <Compile />
    </>
  );
}

export default App;
```

*Рис 3.3 Файл App.js*

### 3.3 Зовнішній вигляд веб-сервісу

Даний веб-сервіс складається з двох основник блоків

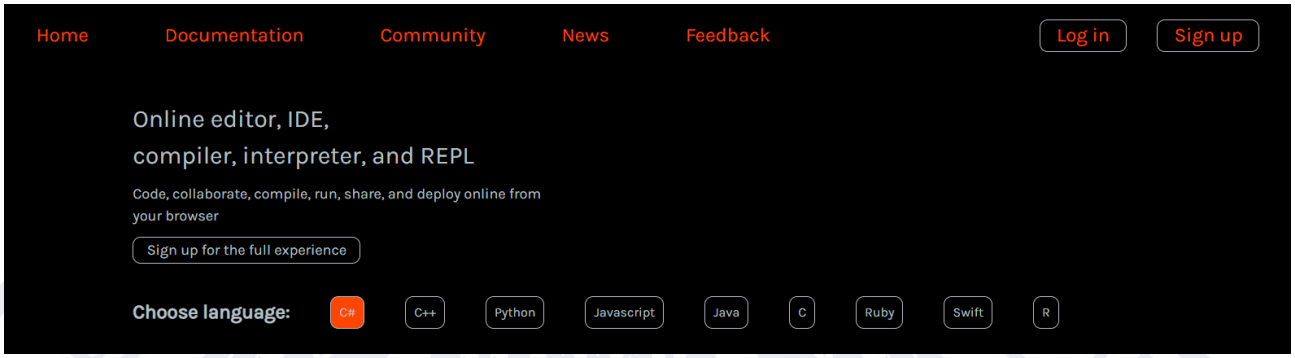


*Рис 3.4 Перший блок веб-сервісу*

Перший основний блок складається з шапки самого сайту, загального опису роботи даного веб-сервісу, та блоку з вибором мови програмування для компіляції.

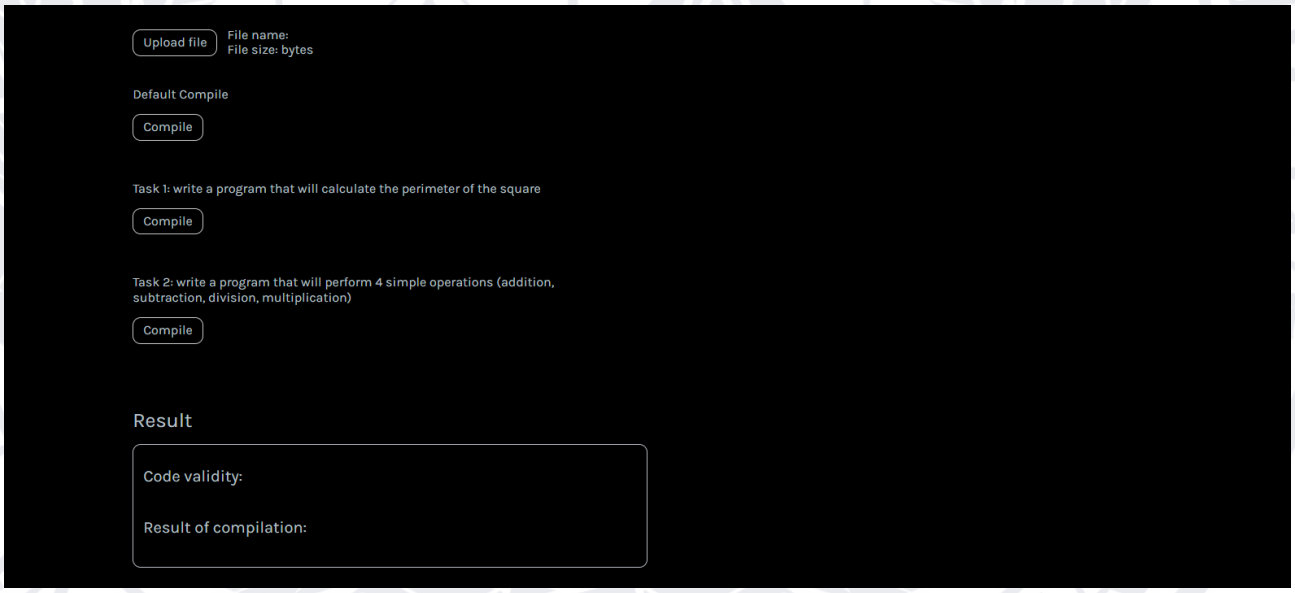


*Рис 3.5 Другий блок веб-сервісу*

В другому блоці реалізовано основний функціонал веб-сервісу, а саме

- Блок, в якому можна завантажити файл, який має компілюватись.
- Блок, в якому завантажений файл буде відправлятись до серверної частини сервісу.

- Блок, де буде висвітлюватись результат компіляції.

### 3.4 Стилізація

В даному проєкті було використано препроцесор scss, за допомогою якого відбувалась стилізація веб-сайту. Вище вже було розглянуто можливості цього препроцесора, тому деякі з них були використані при написанні стилів.

На рис. X зображено використання вкладень в scss, що дає змогу писати стилі набагато компактніше. На рис X зображено використання змінних в scss. В даному випадку, в змінні записуються кольори, які потім використовуються при стилізації. Це є дуже зручно, тому що в подальшому буде можливість замінити якийсь колір одразу у всіх місцях, просто помінявши його в цій змінній.

```
.header{
  width: 95%;
  margin-top: 20px;
  display: flex;
  justify-content: space-between;

  &__links{
    display: flex;
    width: 60%;
    justify-content: space-between;

    &_item{
      display: flex;
      align-items: center;
      font-size: 24px;
      color: $button-hover;
      text-decoration: none;
    }
  }
```

*Рис 3.6 Вкладеність стилів*

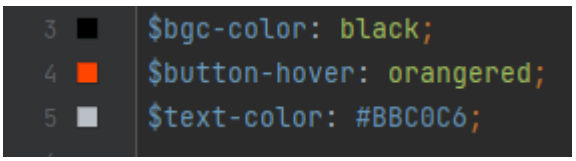


*Рис 3.7 Використання змінних в scss*

### 3.5 Основний функціонал

Основна ціль онлайн сервісу системи автоматичної перевірки коду – швидко і зручно, в режимі реального часу компілювати програмний код. Наразі реалізована компіляція завантажених файлів. Можна використовувати звичайну компіляцію, а можна перевіряти на правильність вирішені алгоритмічні задачі. В подальшому веб-сервіс має великий потенціал на розширення функціоналу.

#### 3.5.1 Завантаження файлів

В даному блоці відбувається завантаження файлу з ПК користувача.

Кнопка «Upload file» візуально має 2 стана. При наведенні мишкою на кнопку, вона підсвічується помаранчевим кольором. Це реалізовано за допомогою псевдокласу *hover*. При завантаженні файлу справа від кнопки з’являться основна інформація про цей файл, а саме його назва та розмір в байтах.


*Рис 3.8 Звичайний вигляд кнопки завантаження*


*Рис 3.9 Вигляд кнопки завантаження при наведенні*

Щоб мати змогу використовувати завантажений файл надалі, було створено функцію *handleUpload,* за допомогою якої файл зберігається в *state,* і доступ до цього файлу є в будь-якій частині коду. На саму кнопку навішано обробник подій *onChange*, в який передається дана функція

```
const handleUpload = (e) => {
  const reader = new FileReader()
  reader.onload = async (e : ProgressEvent<FileReader> ) => {
    const text = e.target.result
    setLoadText(text)
  }
  reader.readAsText(e.target.files[0])

  setSelectFile(e.target.files[0])
}
```

*Рис 3.10 Функція для роботи з файлами*

### 3.5.2 Робота з сервером

В даному блоці відбувається безпосередня взаємодія з бекенд частиною сервісу. Як і в блоці з завантаженням файлів, тут використовується псевдоклас для зміни фону кнопок, при наведенні мишкою на них.

Доступно декілька варіантів компіляції. *Default compile* використовується для компіляції будь-яких файлів з програмним кодом. Інші ж кнопки відповідають за перевірку поставлених задач. В подальшому на сервіс будуть додаватись різноманітні алгоритмічні задачі, таким чином можна буде вирішувати їх, та перевіряти.

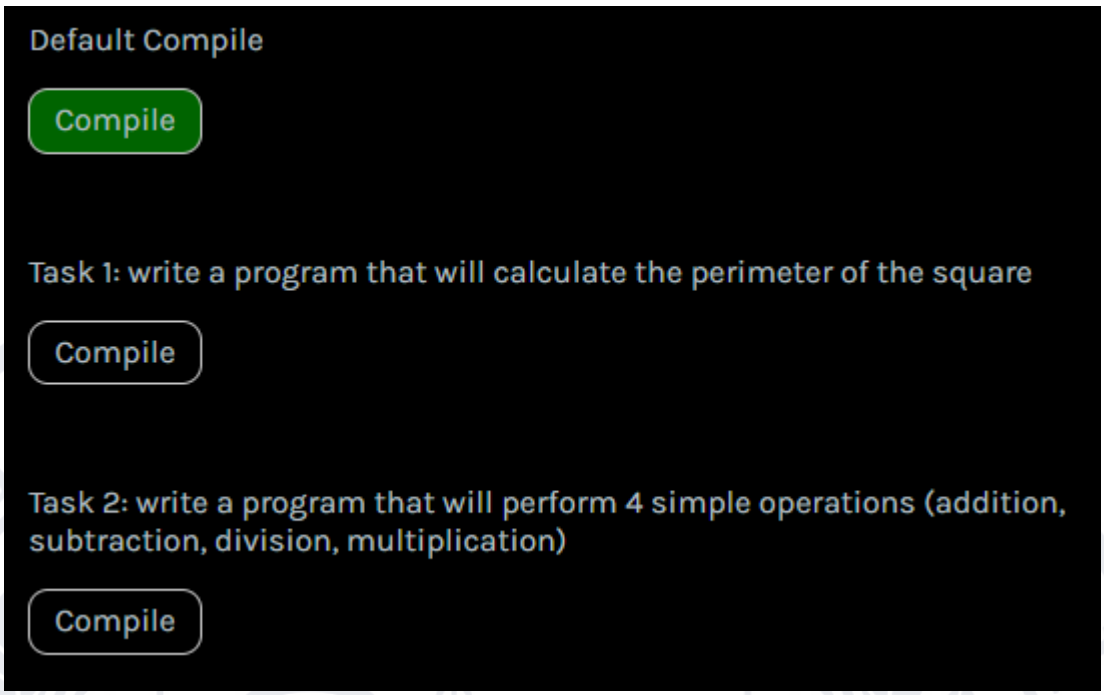


*Рис 3.11 Вигляд блоку, в якому відбувається робота з сервером*

Для відправки завантаженого файлу створено асинхронну функцію *compileFile*, за допомогою якої і здійснюється зв'язок з сервером. В цій функції використано конструкцію *try/catch*, щоб в разі помилки з'єднання було висвітлене відповідне повідомлення про збій. Відповідь сервера записується в *state*, щоб в подальшому мати доступ до результатів компіляції. На кнопку навішується обробник подій *onClick*, який після натискання на одну з кнопок для компіляції намагається відправити завантажений файл до серверу.

```
const compileFile = (e) => {
  try {
    axios.post( url: `/DefaultCompilator?code=${loadText}`)
      .then(res => {
        setRight(res.data.right.toString())
        setResult(res.data)
      })
  } catch (err){
    console.log(err)
  }
}
```

*Рис 3.12 Функція для надсилання запиту на сервер*

Як видно на *рис. 5*, в тілі запиту не використовується повна url адреса, на яку відправляється цей запит. Щоб код виглядав зрозуміло, було використано допоміжний модуль axios.js, в якому повна url адреса записується в константну змінну *BASE_URL*.

```
import axios from 'axios';
const BASE_URL = 'https://programcompilapi.azurewebsites.net/Compilation';

export default axios.create({
  baseURL: BASE_URL
});
```

*Рис 3.13 Допоміжний модуль axios.js*

### 3.5.3 Результати компіляції

В цьому блоці будуть висвітлюватись результати компіляції, та валідність програмного коду. Як було описано вище, результат компіляції, який повертає сервер зберігається в *state*, тому маючи доступ до цих результатів, можна з легкістю висвітлити їх на екрані одразу після завершення компіляції. Як результат отримується значення, яке отримується після компіляції, та валідність програмного коду, у вигляді *true* або *false*

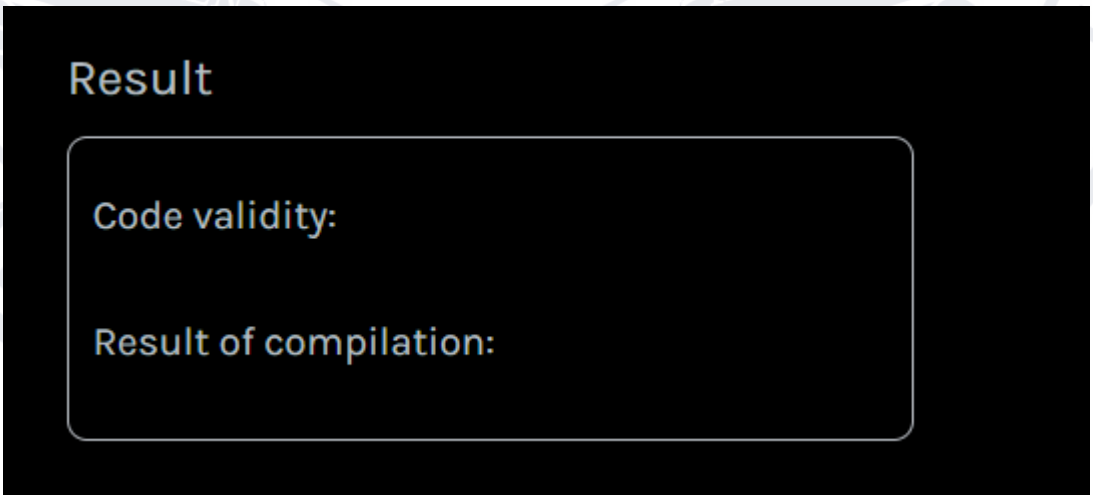


*Рис 3.14 Блок з результатами компіляції*

## 3.6 Висновки до третього розділу

В даному розділі було розглянуто реалізацію фронтенд частини автоматичної перевірки коду. Описано роботу з react, структуру проєкту, зовнішній вигляд веб-сервісу, реалізацію функцій (за допомогою яких відбувається завантаження файлів, та надсилання запитів на сервер), роботу з сервером (а саме надсилання запитів та отримання відповідей від нього) та обробку результатів.

# ВИСНОВКИ

В ході даної бакалаврської роботи було розроблено веб-сервіс для автоматичної перевірки програмного коду. Такий сервіс є досить актуальний в наш час, так як заощаджує час та ресурси для виконання поставлених задач. Було проаналізовано подібні рішення, визначені їх переваги та недоліки, що дало зрозуміти доцільність розробки такого сервісу.

За допомогою веб-сервісу можна здійснювати компіляцію програмного коду (на даному етапі розробки, тільки на мові C#), а також перевіряти код на валідність.

Для розробки веб-сервісу були використанні провідні технології фронтенд розробки, що надає йому високу швидкодію та надійність. В ході розробки було проаналізовано ефективність роботи з обраними технологіями, що доводить правильність вибору стеку технологій.

Під час написання коду було використано методологію DRY (don’t repeat yourself - принцип розробки програмного забезпечення, націлений зниження повторення інформації). Завдяки цьому код є доволі читабельним, що дозволяє в подальшому працювати над ним в команді. Також, весь програмний код опублікований, та є у вільному доступі на github.com. Код доступний за посиланням https://github.com/mxtokar/Web-Code-Runner.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ


1. Рейтинг онлайн компіляторів https://dev.ua/news/onlajn-kompylyatory

2. Онлайн компілятори та їх види https://gitjournal.tech/podborka-onlajn-kompiljatorov-chto-jeto-kak-oni-rabotajut-i-kakoj-vybrat/#i

3. Онлайн компілятор ideone https://ideone.com/

4. Онлайн компілятор codepad http://codepad.org/

5. Онлайн компілятор dotnetfiddle https://dotnetfiddle.net/

6. Онлайн компілятор та збірник задач codewars https://www.codewars.com/

7. Javascript https://uk.wikipedia.org/wiki/JavaScript

8. Довідник JS https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference

9. Посібник по JavaScript https://learn.javascript.ru/

10. Javascript для початківців https://ru.code-basics.com/languages/javascript

11. JavaScript Tutorial https://www.w3schools.com/js/

12. React https://uk.wikipedia.org/wiki/React

13. Документація React https://uk.reactjs.org/docs/getting-started.html

14. Початок роботи з react (базовий набір інструментів бібліотеки react) https://developer.mozilla.org/ru/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started

15. Create React App https://create-react-app.dev/

16. Вихідний код https://github.com/axios/axios

17. Документація Axios https://axios-http.com/

18. Вступ до JSX https://uk.reactjs.org/docs/introducing-jsx.html

19. Детально про JSX https://habr.com/ru/post/319270/

20. JSON https://uk.wikipedia.org/wiki/JSON

21. Документація JSON https://www.json.org/json-en.html

22. Sass https://uk.wikipedia.org/wiki/Sass

23. Основи sass/scss https://sass-scss.ru/guide/

24. Scss на практиці https://habr.com/ru/post/140612/

25. Scss 7-1 pattern https://gist.github.com/rveitch/84cea9650092119527bc

26. Документація БЕМ https://ru.bem.info/methodology/

27. Методологія БЕМ в дії https://avivi.pro/ua/blog/metodologiya-bem-v-deystvii/

28. Історія створення методології БЕМ, та способи застосування https://habr.com/ru/company/yandex/blog/276035/

29.Git https://uk.wikipedia.org/wiki/Git

30. Документація Git https://git-scm.com/doc

31. Git book https://git-scm.com/book/uk/v2

32. Шпаргалка по git https://training.github.com/downloads/ru/github-git-cheat-sheet/

33. Офіційний сайт WebStorm https://www.jetbrains.com/ru-ru/webstorm/

34. Огляд IDE https://itpro.ua/product/jetbrains-webstorm/?tab=description

35. Довідник HTML https://webref.ru/html

36. Fielding J. Beginning Responsive Web Design with HTML5 and CSS3 // Beginning Responsive Web Design with HTML5 and CSS3. 2014.

37. Freeman A. Pro JavaScript for Web Apps // Pro JavaScript for Web Apps. Apress, 2012.

38. Ambler T., Cloud N. JavaScript frameworks for modern web dev // JavaScript Frameworks for Modern Web Dev. Apress Media LLC, 2015. 1–485 p.

39. Gackenheimer C. Introduction to React // Introduction to React. 2015.

40. 26.Кириченко В, Хрусталев А. HTML5+CSS3. Основы современного Web-дизайна. -СПб.: "Наука и Техника", 2018. -352 с.

Декларація щодо унікальності текстів роботи
та невикористання матеріалів інших авторів без посилань

Токар Максим Олександрович
Прізвище, ім’я, по батькові

Факультет інформаційних і прикладних технологій
Факультет

122 Комп’ютерні науки
Шифр і назва спеціальності

Сучасні інформаційні технології та програмування
Освітня програма

**ДЕКЛАРАЦІЯ**

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «ТЕМА БАКАЛАВРСЬКОЇ РОБОТИ» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю,. що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

дата підпис