УДК 655.4
DOI: 10.20535/2077-7264.3(85).2024.293209

© М. П. Горський, канд. фіз.-мат. наук, доц., М. О. Огірко, канд. техн. наук, асист., І. В. Солтис, канд. фіз.-мат. наук, доц., О. В. Дуболазов, д-р фіз.-мат. наук, проф., О. Г. Ушенко, д-р фіз.-мат. наук, проф., В. В. Морфлюк-Щур, канд. техн. наук, асист., Л. С. Слоцька, канд. техн. наук, доц., Чернівецький національний університет, м. Чернівці, Україна

СУЧАСНІ ФРЕЙМВОРКИ ДЛЯ СТВОРЕННЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ В ТЕХНОЛОГІЯХ ЕЛЕКТРОННИХ ВИДАНЬ

В роботі розглядаються сучасні фреймворки для створення користувацького інтерфейсу в технологіях електронних видань. Проаналізовано найрозповсюджені фреймворки з різними функціональними можливостями, підходами та, які спрямовані на різні типи веб-додатків.

Ключові слова: технології електронних видань; веб-сайти; програмні продукти; UI фреймворки; React; Angular; Vue.js; Svelte; Material-UI; Ant Design; Foundation; Bootstrap.

Постановка проблеми

Розробка веб-сайтів є значною частиною сучасних технологій електронних видань. Користувацький інтерфейс (UI) є одним з важливих компонентів сучасного сайту. Під час розробки UI важливо прискорити розробку, поліпшити якість та забезпечити консистентний інтерфейс [1, 2]. Зазвичай для цього треба розв'язати наступні задачі:

— Прискорення розробки: бажано зменшити час, потрібний для розробки, щоб не було потрібно писати кожен елемент з нуля.

— Консистентність: готові компоненти сайту повинні мати єдиний стиль та структуру, що робить інтерфейс більш зручним для користувачів.

— Адаптивний дизайн: оптимізація інтерфейсу для різних розмірів екранів та пристроїв.

— Підтримка кросбраузерності: оптимізація UI для різних веббраузерів, що допомагає забезпечити сумісність сайту з різними платформами.

— Безпека: захист сайту від різних атак, таких як інжекція SQL-запитів або між-сайтового скриптування (XSS).

— Оновлення та підтримка: можливість отримувати нові функції та виправлення помилок.

ISSN 2077-7264. Технологія і техніка друкарства. 2024. № 3(85)

ISSN 2077-7264. Технологія і техніка друкарства. 2024. № 3(85)

Наявність активних спільнот, які надають підтримку та ресурси для розробників.

— Відокремлення завдань: можливість розділити розробку інтерфейсу на менші завдання, що полегшує роботу в команді та забезпечує більшу організованість.

Саме на вирішення цих задач спрямована розробка UI фреймворків. На сьогоднішній день існує багато сучасних UI фреймворків [3, 4]. Ось декілька з них:

1. React: розроблений компанією Facebook, є одним з найпопулярніших фреймворків для створення веб-інтерфейсів. Він використовує концепцію компонентів для побудови динамічних та інтерактивних інтерфейсів.

2. Angular: розроблений Google, є повноцінним фреймворком для створення веб-додатків зі складними логікою та інтерфейсами. Він використовує TypeScript і має багато інструментів для розробки.

3. Vue.js: легкий та простий у використанні фреймворк, який набуває популярності. Vue.js дозволяє створювати реактивні інтерфейси та легко інтегрується з існуючими веб-додатками.

4. Svelte: відомий своєю простотою та високою продуктивністю, перетворює код в нативний JavaScript на етапі компіляції. Це дозволяє зменшити розмір бандлів та покращити продуктивність додатків.

5. Material-UI: фреймворк на основі дизайну Material Design від Google. Він надає готові компоненти та стилізацію для швидкого створення інтерфейсу у стилі Material Design.

6. Ant Design: фреймворк для розробки інтерфейсів, який надає готові компоненти та дизайн у стилі Ant Design.

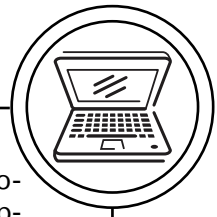
7. Foundation: фреймворк для респонсивного та мобільного веб-дизайну, який має набір готових компонентів та інструментів для розробки.

8. Bootstrap: один з найпоширеніших фреймворків для розробки інтерфейсів, що має готові компоненти та стилізацію для швидкого старту.

Ці фреймворки мають різні функціональні можливості, підходи та спрямовані на різні типи веб-додатків.

Аналіз попередніх досліджень

Будь-який UI веб-сайт побудований на стороні клієнта (користувача) з використанням технологій HTML, CSS та JavaScript. HTML розроблено і випущено у 1993 р., за кілька років до JavaScript, тоді як CSS випущено у 1996 р., через рік після JavaScript. Ці три мови сформували технологічний стек, який називають «тріадою технологій, які повинні вивчити всі веб-розробники». HTML — це абревіатура від мови гіпертекстової розмітки, яка визначається World Wide Web Consortium як «основну мову Інтернету для створення вмісту» [5]. Гіпертекст відноситься до тексту, який містить посилання або гіперпосилання на інше текстові сегменти або текстові сторінки. Розмітка відноситься до тексту, що містить анотації, уточнюючі дані властивостей, що належать до певного тексту, за межами самого видимого текстового вмісту. Крім того, це мова описової



розмітки, що означає, що вона використовується для маркування тексту, а не для надання вказівки щодо обробки тексту. Зв'язок HTML та JavaScript сьогодні полягає в тому, що він може працювати або як доповнення до JavaScript, або як інтегрований HTML-подібний синтаксис, який скомпільовано в елементи HTML. Прикладом цього є синтаксис JSX, рекомендований для використання під час розробки з React, який не є ні суто JavaScript, ні HTML, а їх комбінацією, інтегруючи створення основних елементів і структурну функціональність HTML з динамічними можливостями JavaScript. CSS як технологія також часто використовується в тандемі з JavaScript. Одним із варіантів є реалізація властивостей CSS в окремих файлах .css, що є більш традиційний формат. Інший варіант, подібний до того, як поєднано HTML та JavaScript полягає в інтеграції CSS у самі фреймворки JavaScript за допомогою спеціалізованих бібліотек, таких як CSS-in-JS і styled-components [6, 7]. Такі бібліотеки дозволяють розробнику написати код JavaScript, який стилізує візуальні елементи за допомогою CSS-подібного синтаксису. Цей код стилю потім зазвичай компілюється в чистий CSS у браузері.

Термінологія, що порівнює фреймворки з бібліотеками, може бути розпливчастою: іноді React згадується як бібліотека, а в інших випадках як фреймворк, особливо в онлайн-обговореннях і статтях. Для ясності було б корисно відокремити фреймворки від бібліотек.

Бібліотека — це пасивна колекція енергонезалежних ресурсів, контроль над якою надається розробнику, про те як використовувати ресурси. Бібліотеки JavaScript відповідають цьому стандартному визначенню. Прикладом бібліотеки JavaScript є jQuery, яка використовується для таких функцій як маніпулювання DOM, обробка подій і функціональність AJAX. Бібліотеки, які дуже прості та виконують лише одне певне завдання можна класифікувати як інструменти, такі як JSLint, що використовується лише для перевірки синтаксису, або Mocha, використовується тільки для тестування. Фреймворки розроблені таким чином, щоб бути менш пасивними та змушувати розробника робити розробку певним чином, надаючи скелет для інтерфейсу та запроваджуючи керування потоком подій [8–10]. Наприклад, фреймворк веб-програми надає розробнику певний спосіб для розробки та налаштування цілої веб-програми, тоді як бібліотека веб-програм забезпечує простіші операції, такі як мережеві запити або операції стилю, над якими розробник має контроль.

Вибір між цими фреймворками може бути складним через кілька чинників:

— Кожен веб-додаток має свої унікальні потреби та характеристики, такі як розмір, комплексність, рівень відомостей розробників, дизайн та багато інших. Вибір має відповідати конкретним вимогам.

— Вибір вимагає знань та досвіду з використання конкретних технологій. Використання незнайомого фреймворку може призвести до додаткового часу та зусиль для вивчення.

ISSN 2077-7264. Технологія і техніка друкарства. 2024. № 3(85)

ISSN 2077-7264. Технологія і техніка друкарства. 2024. № 3(85)

— Популярні фреймворки мають великі спільноти та ресурси для вивчення, допомоги та розв'язання проблем. Менш популярні фреймворки можуть бути менш підтримуваними та більш обмеженими у ресурсах.

Мета роботи

Детальний аналіз та вибір фреймворку, який найкраще підходить для певного веб-додатку.

Результати проведених досліджень

Сучасні фреймворки можна розділити на дві великі групи:

— CSS фреймворки, що надають готові стилі, компоненти та сітку для швидкої побудови інтерфейсу. Вони зосереджені на забезпеченні швидкості розробки та стилізації. Зазвичай такі фреймворки надають набір готових компонентів (кнопки, форми, картки тощо) та стилів, які можна використовувати для швидкого створення інтерфейсу. Прикладами таких фреймворків можуть бути: Material-UI, Ant Design, Foundation та Bootstrap.

— Фреймворки призначені для побудови інтерактивних та динамічних веб-додатків (JavaScript фреймворки). Вони дозволяють створювати компоненти, які реагують на зміни даних та забезпечують швидку та ефективну реакцію на дії користувачів. Для цього використовуються динамічні компоненти, які можуть відреагувати на події, зміни даних та відображати їх на сторінці, також забезпечується більш високий рівень контролю над логікою додатку та можливість взаємодії із сервером. Прикладами таких фреймворків можуть бути: React, Angular, Vue.js та Svelte.

Слід зауважити що CSS фреймворки можуть використовуватися разом з JavaScript фреймворками для швидкої стилізації компонентів, стилів та сітки. Також, навпаки, JavaScript фреймворки можуть використовуватися разом з CSS фреймворками для реалізації динамічних компонентів та інтерфейсів з багатою функціональністю.

Важливими аспектами CSS фреймворків є: тип дизайну, набір компонент, гнучкість налаштування, адаптивний дизайн, оновлення та підтримка, легкість вивчення та використання. Порівняння популярних CSS фреймворків за цими критеріями наведено нижче [6, 7]:

1. *Тип дизайну*:

— Material-UI: заснований на дизайні Material Design від Google. Має сучасний та чистий вид, з анімаціями та ефектами, характерними для цього стилю.

— Ant Design: має сучасний дизайн, заснований на стандартах Ant Design Language, який акцентує на простоту та чіткість.

— Foundation: має базові стилі та компоненти, призначені для респонсивного та мобільного дизайну. Стиль менше наочний, але дозволяє більше кастомізації.

— Bootstrap: має класичний і впізнаваний стиль, який спрощує розробку з використанням готових класів та компонентів.

2. *Набір компонент*:

— Material-UI: має великий набір готових компонентів, з якими можна легко побудувати інтерфейс, включаючи такі як кнопки, картки, панелі, таблиці тощо.

— Ant Design: має розширений набір компонентів для широкого спектра функцій, включаючи вміст, форми, навігацію та багато іншого.

— Foundation: має основні компоненти та рішення для респонсивного дизайну, дозволяючи додати специфічні компоненти при необхідності.

— Bootstrap: має розгорнуту бібліотеку компонентів для створення різноманітних елементів інтерфейсу.

3. *Гнучкість налаштування*:

— Material-UI: можливості кастомізації дещо обмежені, але можна налаштувати палітру кольорів та деякі стилі.

— Ant Design: має гнучку систему налаштувань теми, яка дозволяє змінювати вид компонентів, колірні схеми та інші аспекти.

— Foundation: дозволяє змінювати стилі, але менше зосереджений на готових інструментах для кастомізації.

— Bootstrap: має великий набір налаштувань, що дозволяє кастомізувати стилі та складові за власними потребами.

4. *Адаптивний дизайн*:

— Material-UI: підтримує респонсивний дизайн, але менше акцентує на цьому аспекті.

— Ant Design: має розгорнуту підтримку респонсивного дизайну з окремими компонентами для різних розмірів екранів.

— Foundation: основна функціональність Foundation пов'язана з респонсивним дизайном.

— Bootstrap: відомий своєю респонсивністю та сіткою, яка допомагає легко адаптувати інтерфейс до різних пристроїв.

5. *Оновлення та підтримка*:

— Material-UI: активна спільнота та підтримка, оскільки є популярним вибором для додатків, що використовують React.

— Ant Design: також має велику спільноту та активну підтримку.

— Foundation: підтримується командою ZURB, але може бути менше активною порівняно з іншими фреймворками.

— Bootstrap: має широку спільноту та активну підтримку, оскільки є одним з найпопулярніших фреймворків.

6. *Легкість вивчення та використання*:

— Material-UI: для вивчення Material-UI необхідно розуміння React та JSX. Може вимагати деякого часу для освоєння компонентів та рекомендацій Material Design. Після навчання Material-UI досить просто використовувати його компоненти, відповідно до документації. Він має зрозумілий та логічний API.

— Ant Design: потребує базового розуміння React та JSX. Має докладну документацію та приклади, що сприяє швидкому вивченню. Компоненти Ant Design легко інтегруються у веб-додатки за допомогою документації та прикладів. Загалом, використання вважається досить простим.

— Foundation: має власні класи та структуру, яку потрібно вивчити. Навчання може бути трохи складнішим для новачків порівняно з React-базовими фреймворками. Foundation дозволяє швидко розробляти респонсивні інтерфейси, але вимагає деякої підготовки та розуміння його особливостей.

— Bootstrap: вважається одним з найбільш простих для навчання, особливо для розробників, які знайомі з HTML та CSS. Має велику кількість документації та прикладів. Bootstrap відомий своєю простотою використання. Готові класи та компоненти дозволяють швидко додавати стилізовані елементи на сторінку.

ISSN 2077-7264. Технологія і техніка друкарства. 2024. № 3(85)

ISSN 2077-7264. Технологія і техніка друкарства. 2024. № 3(85)

Важливими аспектами Javascript фреймворків є: продуктивність та швидкодія, реактивність (збереження стану), розмір разом з бібліотеками, оновлення та підтримка, легкість вивчення та використання [3, 4]. Порівняння популярних CSS фреймворків за цими критеріями наведено нижче:

1. *Продуктивність та швидкодія*:

— React: добре оптимізований для віртуального DOM, що забезпечує високу швидкодію, але продуктивність може залежати від використання додаткових бібліотек стану.

— Angular: може бути трохи важким та вимагати більше ресурсів, але може використовувати механізми оптимізації та зниження навантаження.

— Vue.js: має добру продуктивність завдяки підходу до реактивності та оптимізації рендерингу компонентів.

— Svelte: спрямований на найкращу продуктивність компонентів на етапі компіляції, що дозволяє зменшити навантаження під час виконання.

2. *Реактивність (збереження стану)*:

— React: потребує використання бібліотеки стану, такої як Redux чи Mobx, для ефективного збереження та керування станом додатку.

— Angular: має вбудовану систему обробки стану через RxJS, що дозволяє реалізувати реактивну логіку.

— Vue.js: вбудована система обробки стану дозволяє легко відслідковувати та змінювати стан компонентів.

— Svelte: має вбудовану реактивність, що дозволяє автоматично оновлювати DOM на основі змін стану.

3. *Розмір разом з бібліотеками*:

— React: розмір додатку може зростати залежно від використання бібліотек стану та інших додаткових пакетів.

— Angular: має більший розмір, особливо для менших вебдодатків, через велику кількість вбудованих функцій.

— Vue.js: має середній розмір та добре оптимізовані бібліотеки завдяки розподіленню коду на компоненти.

— Svelte: створює найменший розмір бібліотек завдяки компіляції на етапі зборки.

4. *Оновлення та підтримка*:

— React: має часті оновлення та активну спільноту, але оновлення може вимагати зусиль через зміни API.

— Angular: потребує більше зусиль для оновлення через свої специфічні концепції та структуру.

— Vue.js: має стабільну оновлену версію та активну спільноту, оновлення зазвичай менш проблематичне.

— Svelte: має швидкі оновлення, які обмежуються перекомпіляцією.

5. *Легкість навчання та використання*:

— React: потребує розуміння JavaScript, JSX та основ React. Перший відгук може бути складним для новачків.

— Angular: потребує розуміння TypeScript, а також докладне вивчення фреймворку та його структури.

— Vue.js: досить легкий для вивчення, оскільки має схожий синтаксис з HTML та JavaScript. Ідеальний для новачків.

— Svelte: може бути вивчений швидше за інші фреймворки завдяки мінімальному API та компіляції.

Висновки

Сучасні фреймворки забезпечують вирішення завдань, призначених прискорити розробку, поліпшити якість та забезпечити консистентний інтерфейс. Вибір між цими фреймворками залежить від вимог до веб-додатку, рівня досвіду, розміру та складності UI інтерфейсу та особистих вподобань.

Bootstrap та Ant Design можуть бути найпростішими для вивчення та використання, особливо для тих, хто вже знайомий з HTML та CSS. Material-UI та Foundation можуть вимагати трохи більше часу та зусиль для вивчення, але вони також надають більше можливостей для створення більш зручних та індивідуальних інтерфейсів.

React — ідеально підходить для складних та великих веб-додатків, має велику спільноту, але вимагає глибокого розуміння JavaScript та може бути незручним для новачків.

Angular — підходить для корпоративних та масштабних вебдодатків, забезпечує вбудовану підтримку реактивності, але потребує вивчення TypeScript та має більший поріг входження.

Vue.js — легкий для вивчення та використання, підходить для різних розмірів веб-додатків, має гнучку систему обробки стану, але може бути менш потужним для дуже великих веб-додатків.

Svelte — забезпечує найкращу продуктивність завдяки компіляції, підходить для велико об'ємних додатків, але є менш популярним та може вимагати нового підходу до розробки.

Список використаної літератури/References

1. Schramm, A. et al. (October 3–8, 2010). Rapid UI development for enterprise applications: Combining manual and model-driven techniques. *Proc. Model Driven Engineering Languages and Systems: 13th International Conference, MODELS 2010*, Part I 13, 271–285. Retrieved from https://www.semanticscholar.org/paper/Rapid-UI-development-for-enterprise-applications%3A-Schramm-Preu%C3%9Fner/73efe89eb8d502f42fa41af4fc7752c0fd8514e6.

2. Malik, S. et al. (2023). Reimagining Application User Interface (UI) Design using Deep Learning Methods: Challenges and Opportunities. *arXiv preprint arXiv:2303.13055*. Retrieved from https://www.researchgate.net/publication/369476884_Reimagining_Application_User_Interface_UI_Design_using_Deep_Learning_Methods_Challenges_and_Opportunities.

3. Levlin, M. (2020). *DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte*, 33–36. Retrieved from https://www.doria.fi/bitstream/handle/10024/177433/levlin_mattias.pdf.

4. Saks, E. (2019). *JavaScript Frameworks: Angular vs React vs Vue*, 101–106. Retrieved from https://www.theseus.fi/handle/10024/261970?show=full.

5. (1 Aug., 2023). *World Wide Web Consortium, 'HTML'*. Retrieved from https://www.w3.org/html/.

6. Elrom, E. (2021). Starter React Project and Friends. *React and Libraries: Your Complete Guide to the React Ecosystem*, 21–51. DOI: 10.1007/978-1-4842-6696-0.

ISSN 2077-7264. Технологія і техніка друкарства. 2024. № 3(85)

7. Nguyen, M. (2023). *Full-stack crud application: User management system*, 19–22. Retrieved from https://www.theseus.fi/handle/10024/792892.

8. Fourie, P. J. (2017). Normative media theory in the digital media landscape: from media ethics to ethical communication. *South African Journal for Communication Theory and Research*, 43(2), 109–127. DOI: 10.1080/02500167.2017.1331927.

9. Otto, C., et al. (2021). Predicting knowledge gain during web search based on multimedia resource consumption. *Proc. International Conference on Artificial Intelligence in Education*, 194–198. DOI: 10.48550/arXiv.2106.06244.

10. Peláez, C. A., et al. (July 26–31, 2019). Methodologies and trends in multimedia systems: a systematic literature review. In: *Social Computing and Social Media. Design, Human Behavior and Analytics: 11th International Conference, SCSM 2019, Held as Part of the 21st HCI International Conference, HCII 2019*, *Proc.*, Part I 21, 88–92. DOI: 10.1007/978-3-030-21902-4_9.

The paper examines modern frameworks for creating user interfaces in electronic publishing technologies. The most common frameworks with different functionalities, approaches, and which are aimed at different types of web applications are analyzed.

Keywords: electronic publishing technologies; web-sites; software products; UI frameworks; React; Angular; Vue.js; Svelte; Material-UI; Ant Design; Foundation; Bootstrap.

Надійшла до редакції 23.08.23